Open System Services Shell and Utilities Reference Manual

Abstract

This manual documents the HP NonStop Open System Services (OSS) shell and utilities. It is written for general users, programmers, system administrators, and operators.

Product Version

N.A.

Supported Release Version Updates (RVUs)

This manual supports J06.03 and all subsequent J-series RVUs, H06.08 and all subsequent H-series RVUs, and G06.29 and all subsequent G-series RVUs until otherwise indicated by its replacement publication.

Part Number	Published
527188-021	August 2013

Document History

Part Number	Product Version	Published
527188-017	N/A	February 2011
527188-018	N/A	August 2011
527188-019	N/A	February 2012
527188-020	N/A	August 2012
527188-021	N/A	August 2013

Contents

LIST OF TABLES

What is New in This Manual

This section describes changes made to the *Open System Services Shell and Utilities Reference Manual* since the last edition (527188-020).

Unless otherwise indicated in the text, discussions of native mode behavior, processes, and so forth apply to both the TNS/R code that runs on systems running G-series RVUs and to the TNS/E code that runs on systems running J-series RVUs or H-series RVUs. Discussions of TNS or accelerated code behavior in the OSS environment apply only to systems running G-series RVUs; systems running J-series RVUs or H-series RVUs do not support TNS or accelerated code execution in the OSS environment.

Unless otherwise indicated in the text, all text that applies to systems running H06.14 and later H-series RVUs also applies to systems running J06.03 and later J-series RVUs.

This manual contains information about some of the following G-series development tools. For servers running H-series RVUs, these tools are supported only in H06.05 and subsequent H-series RVUs:

- TNS/R native C compiler
- TNS/R native C++ compiler
- TNS/R native C++ runtime library version 2
- SQL/MP for TNS/R native C
- SQL/MP Compilation Agent for TNS/R programs
- NMCOBOL compiler and **nmcobol** frontend
- **ld**
- **nld**
- **noft**
- TNS/R native pTAL

If your server is running the H06.03 or H06.04 RVU, continue to use the HP Enterprise Toolkit—NonStop Edition or servers running G-series RVUs for development tasks that require these tools. If your server is running J06.03 or later J-series RVUs, these tools are supported.

Beginning with the J06.14 and H06.25 RVUs, the OSS Core Utilities product (T1202) provides support for additional Open Source utilities. Except for edits to include

NonStop-specific information, the OSS Core Utilities reference pages are passed through without changes to their Open Source original content. These reference pages are available in the **/usr/coreutils/share/man** directory; they are not available from HP in book form and not included in Open System Services reference manuals. For more information, see the *Open System Service Management and Operations Guide*, the *Open System Services User's Guide*, and the individual reference pages.

New Commands and Utilities

The following commands have been added to improve OSS usability:

- **useradd**
- **userdel**
- **usermod**
- **newusers**

Changed Commands

The following commands have been changed to correct errors:

- **cp**
- **gtacl**
- **man**
- **su**

About This Manual

HP NonStop Open System Services (OSS) is partially derived from the Open Software Foundation OSF/1 product version 1.2. The *Open System Services Shell and Utilities Reference Manual* contains reference pages for OSS user commands and utilities.

Unless otherwise indicated in the text, discussions of native mode behavior, processes, and so forth apply to both the TNS/R code that runs on systems running G-series RVUs and to the TNS/E code that runs on systems running J-series RVUs or H-series RVUs. Discussions of TNS or accelerated code behavior in the OSS environment apply only to systems running G-series RVUs; systems running J-series RVUs or H-series RVUs do not support TNS or accelerated code execution in the OSS environment.

Unless otherwise indicated in the text, all text that applies to systems running H06.14 and later H-series RVUs also applies to systems running J06.03 and later J-series RVUs.

Audience

The *Open System Services Shell and Utilities Reference Manual* is intended for general users of OSS as well as system and application programmers.

Purpose

The *Open System Services Shell and Utilities Reference Manual* provides a complete description of OSS commands. It is intended primarily as a reference and not as a tutorial.

Document Usage

This document contains 12 sections:

- Sections 1 through 10 contain reference pages for all user commands and utilities included in the basic OSS product set.
- Section 11 contains reference pages for file formats related to the commands and utilities.
- Section 12 contains miscellaneous reference pages, usually intended for administrator use.

The reference pages are organized alphabetically within each section.

If you are not sure of the name of the command you want, you can find help in the table of contents, index, and permuted index. The permuted index is created from the descriptions in the **NAME** section of each reference page (see **Reference Page Format**); use it by searching for any term that might appear in a brief description of the desired command's purpose.

Reference Page Format

The top area of each reference page in the *Open System Services Shell and Utilities Reference Manual* includes the name of the command described on that page. If more than one command is described, all command names are included.

Each reference page is organized into sections. The sections generally appear in the same order, but some appear in all reference pages and some are optional.

NAME	Provides a brief description of the purpose of the command or commands described.
SYNOPSIS	Summarizes the syntax of the command and elaborates on the brief description of its use and function found in the **NAME** section.
FLAGS	Lists and describes the command's required or optional flags, if any.
DESCRIPTION	Describes the command more fully than the **NAME** and **SYNOPSIS** sections.
SUBCOMMANDS	Describes in detail the command's subcommands, if any.
EXAMPLES	Provides examples of ways in which the command is typically used.
FILES	Lists any OSS system files that are read, employed, referred to, or written to by the command, or that are otherwise relevent to its use.
NOTES	Provides additional information about the command that is not of general interest.

CAUTIONS Cautions users about circumstances to be avoided when using the command, or about loss of data that might result if the command is used incorrectly.

DIAGNOSTICS Provides information useful for diagnosing errors that might result when the command is used.

EXIT VALUES Lists and describes exit values returned by the command.

RELATED INFORMATION Lists OSS commands, functions, file formats, and special files employed by the command, that have a purpose related to that of the command, or that are otherwise of interest within the context of the command. This section can also list related documents.

STANDARDS CONFORMANCE Summarizes features that are fully described in previous subsections and are flagged as implementation-defined or HP extensions to XPG4.

The POSIX standards leave some features to the implementing vendor to define. These features are flagged as implementation-defined. Features that HP has included that are not in an XPG4 specification are flagged as HP extensions to the XPG4 or XPG4 Version 2 specification.

Related Documents

For information about OSS commands and utilities, library calls, system calls, and guidelines for general usage, see these manuals:

- *Accelerator Manual*
- *Binder Manual*
- *C/C++ Programmer's Guide*
- *HP COBOL Manual for TNS/E Programs*
- *HP COBOL Manual for TNS and TNS/R Programs*
- *Common Run-Time Environment (CRE) Programmer's Guide*
- *eld Manual* (TNS/E systems only)
- *enoft Manual* (TNS/E systems only)
- *ld Manual*
- *Native Inspect Manual* (TNS/E systems only)
- *nld Manual*
- *noft Manual*
- *Object Code Accelerator (OCA) Manual*

- *Open System Services Porting Guide*
- *Open System Services Programmer's Guide*
- *Open System Services Library Calls Reference Manual*
- *Open System Services System Calls Reference Manual*
- *Open System Services User's Guide*
- *rld Manual*
- *Software Internationalization Guide*
- *TCP/IP and TCP/IPv6 Programming Manual*

If you are working in or with the Guardian environment, see the *Guardian Procedure Calls Reference Manual* and its related manuals.

If you are using the Guardian C run-time library, see the *Guardian Native C Library Calls Reference Manual*.

Reference Section Numbers

Some topics in the reference pages have more than one reference page file, and the reference section number indicates which set of information to display. For example, **iconv** has a reference page for the **iconv( )** function in section 3 and a reference page for the **iconv** command in section 1.

Reference section numbers are included under the **RELATED INFORMATION** heading and in the heading at the top of every reference page. The following table shows the correspondence between reference section numbers and OSS manuals.

Section	Content	Manual
(1)	User commands	*OSS Shell and Utilities Reference Manual*
(2)	System calls	*OSS System Calls Reference Manual*
(3)	Library calls	*OSS Library Calls Reference Manual* *OSS System Calls Reference Manual* (**SPT_*( )** functions only)
(4)	File formats and data structures	*OSS System Calls Reference Manual* *OSS Library Calls Reference Manual* *OSS Shell and Utilities Reference Manual*
(5)	Miscellaneous topics and environment variables	*OSS System Calls Reference Manual* *OSS Library Calls Reference Manual* *OSS Shell and Utilities Reference Manual*

(6)	Games	Not supplied by HP
(7)	Special files	*OSS System Calls Reference Manual*
(8)	Administrator commands	*OSS Shell and Utilities Reference Manual*

Synopsis Format and Conventions

The **SYNOPSIS** section of each reference page summarizes the ways a command is invoked. The following list describes the conventions used in these summaries.

- Command names and all flags, required and optional, are always shown in **bold** type.
- Arguments, to the command itself or to its flags, are always shown in *italic* type.
- Optional items, including both flags and arguments, appear in brackets: for example, [*file*]. Brackets are not always nested; therefore, an optional argument to an optional flag appears in its own pair of brackets, following the flag in its pair of brackets. For example, [**-a**] [*file*] indicates an optional flag **-a** with its optional argument *file*, as opposed to [**-a** *file*], which indicates an optional flag **-a** and its required argument *file*. The lack of nesting for brackets might incorrectly imply that the argument could be specified without the flag; when in doubt, consult the **FLAGS** section of the reference page.
- In general, flags that do not take arguments and are not mutually exclusive are grouped together (in a pare of brackets if they are optional). For example:

  -aj[k]v In gereral, flags that have related arguments are shown separately. For example:

  [**-cCdfFnqvV**] [**-b** *maxbits*]

- Beyond the preceding grouping requirements, flags appear in alphabetical order (U.S. English), with uppercase letters following lowercase letters—for example, **-aAjkKv**.
- Command arguments appear in the order required by the command, if any. Mandatory arguments appear before optional arguments unless the command requires otherwise.
- Operands of indeterminate number are indicated by an ellipsis following the flag name—for example, [**-a** *file* ...].
- Because some flags are separated from their operands by spaces, a diagram might be unclear as to whether an operand is an operand to the command or to a required flag. When in doubt, consult the **FLAGS** section of the reference page.
- When two or three flags or operands are mutually exclusive (that is, they cannot be used together) they are separated by vertical bars — for example, **-a** | **-j**, or [**-k** *file* | *directory*].
- When a greater number of items are mutually exclusive, or some other aspect of the command's use creates greater than normal complexity, more than one diagram is provided.

Obsolescent Flags

Obsolescent flags (that is, flags that have been replaced by new flags that reflect future trends in conformance) are documented. The new flags supersede the obsolescent flags, but the obsolescent flags are still supported.

General Typographic and Keying Conventions

This document uses several typographic conventions. (See also **Synopsis Format and Conventions**.)

Bold — **Bold** words or characters represent system elements that you must use literally, such as commands, flags, pathnames, and variable names. **Bold** type is always used to represent user input; anything to be typed or entered by the user is represented in **bold** type. (The **EXAMPLES** section is an exception; in that section, `constant width` type represents user input.)

Italic — *Italic* words or characters represent variable values. When *italic* type appears in user input examples, it represents a portion of the input that varies according to the situation or the user's choice.

`Constant width` — System output in command examples in the **EXAMPLES** section is represented by `constant width` typeface. In addition, `constant width` is used in descriptive text to represent system messages.

Keystrokes — Keystrokes are indicated by the name of the key to be pressed in angle brackets in bold type: for example, **<Return>** or **<Tab>**. Key names vary from one keyboard to the next; for example, the **<Esc>** key might appear on your keyboard as **<Escape>** and **<Ctrl>** as **<Control>**. If you are unable find some of the keys referred to on your keyboard, consult your hardware documentation or system administrator.

When two keys are to be pressed at the same time, they are shown together within the brackets, separated by a hyphen: for example, **<Ctrl-c>**, indicates that you should hold down the **<Ctrl>** key and press the **<c>** key. (Control characters in system output are represented as **^C**, **^X**, and so on.)

In general, the **<Delete>** and **<Erase>** keys generate different codes, but are functionally equivalent. While **<Delete>** is usually mapped to ASCII code 127, represented by **^?**, and **<Erase>** is usually mapped to the ASCII backspace character, **^H**, either can be mapped to another code on a particular keyboard or system. If these keys do not have the

results described, consult the **stty** and **tset** reference pages or your system administrator.

Standard Key Sequences

Some standard keystroke sequences are used for general purposes. For example, the Interrupt key sequence interrupts and cancels the current action, without proceeding further; you might use it to stop a command that is displaying output you do not want on your screen.

The actual keys used are the same on most systems, but they are *not* universal. For example, the Interrupt key sequence is usually **<Ctrl-c>**, but not always.

The reference pages in this manual refer to these key sequences by name. In some cases, the customary keystroke is given. For the two most common sequences, the customary keystrokes are not given. The End-of-File key sequence, used to end user input or otherwise complete an action, is usually **<Ctrl-d>**. The Interrupt sequence, used to interrupt or cancel the current action, is usually **<Ctrl-c>**.

Use these keystrokes for the End-of-File and Interrupt key sequences, and the keystrokes listed in the text for the others. If these keystrokes do not have the desired effect, consult the **stty** and **tset** reference pages or your system administrator.

Unsupported OSS Utilities

Unsupported OSS utilities are utilities that HP has released, but not tested. HP does not guarantee the behavior or performance of these utilities and is not obligated to fix problems associated with them.

The unsupported utilities are in **/bin/unsupported**. Any documentation for the unsupported utilities also would be in **/bin/unsupported**. To access reference pages for unsupported utilities with the **man** command, add the path **/bin/unsupported** to your **MANPATH** or specify it as an alternative path using the **-M** flag with the **man** command:

man -M /bin/unsupported -k *keyword*

A reference page for an unsupported utility includes text stating that the utility is unsupported. Not all of the unsupported utilities are documented by reference pages.

Section 1. User Commands (a - b)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the Letters **a** through **b**.

NAME

add_define - Creates one or more DEFINEs for the current OSS shell

SYNOPSIS

add_define
{*define_name1*}...
[**-like**=*define_name2*]
[*attribute_specs*]...

FLAGS

-like=*define_name2*

Creates a DEFINE with the attributes and values of the specified *define_name2* and modified by the specified *attribute_specs* clauses. If the **-like** flag is not specified, the DEFINE created has the attributes and values of the working attribute set modified by the specified *attribute_specs* clauses.

DESCRIPTION

The **add_define** command is specific to HP and an OSS shell built-in command. It creates DEFINES for the OSS shell and is similiar to the TACL ADD DEFINE command.

The **add_define** command accepts Guardian attributes. As a result, input must follow Guardian conventions.

Operands

define_name1

Specifies the name of the DEFINE to be created. The name can be 2 through 24 characters long. The first character must be an equal sign (=), and the second must be a letter. If neither the **-like** flag nor the *attribute_specs* parameter is specified, the DEFINE is created with the attributes and value of the working attribute set.

attribute_specs

Specifies the names of one or more valid DEFINE attributes and the values they are to have. If the **-like** flag is specified, a DEFINE is created with the attributes and values of the specified DEFINE and modified by the clauses specified by *attribute_specs*. If the **-like** flag is not specified, a DEFINE is created with the attributes and values of the working attribute set and modified by the specified *attribute_specs*. *attribute_specs* is defined as:

class={catalog | defaults | map | search | sort | spool | subsort | tape}
{*class_attributes*}...

Class Attributes

Certain characters are special in the OSS environment and must be preceded by an escape character or they will not be accepted by the **add_define** command. For a detailed description of the valid class attributes, refer to the ADD DEFINE and SET DEFINE commands in the *TACL Reference Manual*.

The **add_define** command accepts Guardian attributes for setting up the Guardian environment. As a result, input must follow Guardian conventions.

For **class=catalog** (a CATALOG DEFINE), you must use the escape character in *class-attributes* as follows:

subvol=\$a123

For **class=defaults** (a DEFAULTS DEFINE), you must use the escape character in *class-attributes*

as follows:

volume=\$oss.joe
swap=\$null
catalog=\$system.catalog

For **class=map** (a MAP DEFINE), you must use the escape character in *class-attributes* as follows:

file=\$volume.subvolume.file

For **class=search** (a SEARCH DEFINE), you must use the escape character in *class-attributes* as follows:

subvol0=\(a,b,c,d\)
relsubvol0=\\foxii.\$coral.i
subvol2=\(\$data.y2,y22\)

For **class=sort** (a SORT DEFINE), you must use the escape character in *class-attributes* as follows:

scratch=\\foxii.\$oss.joe.scratch
swap=\\foxii.\$oss.joe.swap
program=\\foxii.\$oss.joe.suprsort
cpus=\(1,2\)
notcpus=\(0,3\)
subsorts=\(=subsort1,-subsort2\)

For **class=spool** (a SPOOL DEFINE), you must use the escape character in *class-attributes* as follows:

loc=\\kt22.\$s.#a

For **class=subsort** (a SUBSORT DEFINE), you must use the escape character in *class-attributes* as follows:

scratch=\\foxii.\$oss.joe.scratch
swap=\\foxii.\$oss.joe.swap
program=\\foxii.\$oss.joe.suprsort

For **class=tape** (a TAPE DEFINE), you must use the escape character in *class-attributes* as follows:

device=\$device
volume=\(v1,v2\)
system=\\foxii

EXAMPLES

1. To make the name **=PLUTO** represent the file name **\\FAR.\$OFF.WORLDS.PLUTO** enter:

   add_define =PLUTO file=\\FAR.\$OFF.WORLDS.PLUTO

2. To set up a TAPE DEFINE named **=S2** that describes a tape file on the IBM standard, enter:

   add_define =S2 class=tape labels=ibm fileid=\$TAPE

3. To create a new DEFINE named **PLUTO2** that has the characteristics of the DEFINE named **=PLUTO**, enter:

   add_define =PLUTO2 -like ==PLUTO

EXIT VALUES

The following exit values are returned:

0	DEFINEs were created successfully.
>0	An error occurred.

NOTES

The **add_define** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **del_define(1)**, **info_define(1)**, **reset_define(1)**, **set_define(1)**, **show_define(1)**.

STANDARDS CONFORMANCE

The **add_define** command is an HP extension to the XPG4 Version 2 specification.

NAME

alias - Defines and lists aliases

SYNOPSIS

alias [**-tx**] [*name*[=*value* ...]]

FLAGS

-t Sets or lists tracked aliases.

-x Sets or prints exported aliases.

DESCRIPTION

The **alias** command with no arguments prints the list of aliases in the form *name=value* on standard output, where *name* is the name of an alias and *value* is the current definition of that alias.

If a name and value of the form *name=value* are specified an alias is defined for each name whose value is given.

A trailing space in *value* causes the next word to be checked for alias substitution.

The **-t** flag is used to set and list tracked aliases. The value of a tracked alias is the full pathname corresponding to the given name. The value becomes undefined when the value of **PATH** is reset but the aliases remained tracked. If the **-t** flag is not specified, the name and value of the alias is printed for each *name* in the argument list for which no value is given.

The **-x** flag is used to set or print exported aliases. An exported alias is defined for scripts invoked by name.

EXAMPLES

1. The following command lists all of the aliases in the current shell.

   alias

EXIT VALUES

Exit status is nonzero if a name is given without a value and no alias was defined.

NOTES

The **alias** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1), **unalias**(1).

NAME

apropos - Locates reference pages by keyword

SYNOPSIS

apropos *keyword* ...

The **apropos** command shows which reference pages contain instances of any of the given keywords in their purpose lines.

DESCRIPTION

In looking for keywords, **apropos** considers each word separately and ignores the case of letters. Words that are part of other words are also considered; thus, when looking for the word **compile**, **apropos** will also find all instances of **compiler**. The *keyword* argument can also be a regular expression. For more information, see the **grep** reference page.

If the output of the **apropos** command begins with a name and reference section number, you can enter **man** *section name*. For example, if the output of the **apropos** command is **printf(3)**, you can enter **man 3 printf** to obtain the reference page on the **printf( )** function.

The **apropos** command works just like the **man** command with the **-k** flag.

EXAMPLES

To find reference pages with the keyword **password** in the purpose line, enter:

apropos password

FILES

/usr/share/man/whatis Keyword database.

RELATED INFORMATION

Commands: **grep(1)**, **man(1)**, **whatis(1)**.

NAME

ar - Creates and maintains archive files and libraries

SYNOPSIS

ar -d [**-v**] [**-l**] *archive file ...*

ar -m [**-aAbilv**] [*position_name*] *archive file ...*

ar -p [**-v**] [**-s**] *archive* [*file ...*]

ar -q [**-clv**] *archive* [*file ...*]

ar -r [**-cuv**] [**-abil**] [*position_name*] *archive file ...*

ar -t [**-v**] [**-s**] [**-Wfiletype**] *archive* [*file ...*]

ar -Wobey *obey_file*

ar -x [**-v**] [**-sCT**] *archive* [*file ...*]

FLAGS

In an **ar** command, you can list selected flags together in one group on the command line, with no spaces between them. You can precede this single flag list with a **-** (dash), but it is not required.

You must specify at least one flag from the required-flag set **dmpqrtx**. You can also specify any number of optional flags from the set **abcilsuvCT**. If you select a positioning flag (**a**, **b**, or **i**), you must also specify the name of a file within the library (*position_name*) immediately following the flag list and separated from it by a space.

-a Positions new files in the archive after the file identified by the *position_name* operand.

-A Suppresses warning messages about optional access control list (ACL) entries. The **ar** utility does not archive optional ACL entries. If this flag is not set, a warning message is issued for each file that has optional ACL entries. For more information about ACLs, see the **acl(5)** reference page.

-b Positions new files in the archive before the file identified by the *position_name* operand.

-c Suppresses the diagnostic message written to the standard error file by default when the archive file *archive* is created.

-C Prevents extracted files from replacing like-named files in the file system. This flag is useful with the **-T** flag to prevent truncated filenames from replacing files with the same prefix.

-d Deletes the named files from the archive.

-i Positions new files in the archive before the file identified by the *position_name* operand. (This flag is equivalent to the **-b** flag.)

-l Creates temporary files in the local current working directory, rather than in the directory specified by the environment variable **TMPDIR**.

-m Moves the named files to some other position in the archive. By default, this flag moves the named files to the end of the archive. Use the positioning flags (**a**, **b**, or **i**) to specify a position other than the default position.

-p Writes the contents of the named files from *archive* to the standard output file. If no files are specified, the contents of all files in the archive are written in the order they occur in the archive.

-q Quickly appends the named files to the end of the archive file. The **ar** command does not check whether the appended files already exist in the archive. **ar** is useful if you want to bypass the search process that is otherwise done when a large archive is created piece by piece.

-r Replaces or adds files to an archive. If the archive named in the *archive* operand does not exist, a new archive file is created and a diagnostic message is written to the standard error file (unless the **-c** flag is specified). If no files are specified and the archive exists, no changes are made to the archive. Files that replace existing files do not change the order of the archive. Files that do not replace existing files are appended to the end of the archive.

-s Forces the regeneration of the archive symbol table even if **ar** is not invoked with a flag that modifies the archive file contents.

-t Writes a table of contents for the library to the standard output file. The files specified by the *file* operands are included in the written list. If no *file* operands are specified, all files in the archive given by the *archive* operand are included in the order of the archive.

-T Allows truncation of names of files whose archive names are longer than 255 characters. By default, extracting a file with a name that is too long is an error: a diagnostic message is written, and the file is not extracted.

-u Updates older files by copying to the archive only those files that have changed since they were last entered into the archive. When used with the **-r** flag, **-u** allows files in the archive to be replaced only by files having more recent modification times.

-v Writes to the standard output file a verbose description of a library. When the **-v** flag is used with flags **-d**, **-r**, or **-x**, it writes a file-by-file description of the archive operations as they are performed.

When **-v** is used with the **-p** flag, it writes the name of the file to the standard output file before it writes the file to the standard output file.

When **-v** is used with the **-t** flag, it writes a detailed list of information about each file, including access, ownership, size, and time, to the standard output file.

-Wfiletype Displays the file type. When specified with **-tv** flags, it displays **[elf-non PIC]** for TNS/R native object files that are not position-independent code (non-PIC), **[elf PIC]** for TNS/R PIC native linkfiles or native load files, **[tnse 32-bit]** for TNS/E native object files of 32-bit data model, **[tnse 64-bit]** for TNS/E native object files of 64-bit data model, **[tnse neutral]** for TNS/E native object files of neutral data model, **[tns]** for TNS and accelerated object files, or nothing for other file types such as text files. Use this flag when **ar** cannot generate the symbol table because native linkfiles or loadfiles are mixed with TNS or accelerated object files, or when TNS/E native object files of 32-bit and 64-bit data models are mixed.

-Wobey *obey_file*

Indicates that a flag and a list of files to be processed should be read from the file *obey_file* rather than from the command line. The **-Wobey** flag cannot be used on the command line when any other flag is used on the command line. Use the **-Wobey** flag to speed up execution of **ar** when more than one file must be processed.

The file *obey_file* must be either a Guardian EDIT file or an OSS text file. In the file *obey_file*, you must specify one and only one flag from the required-flag set **dmpqrtx**. You can also specify any number of optional flags from the set **abcilsuvCT**. If you select one of the positioning flags (**a**, **b**, or **i**), you must also specify the name of a file within the library (*position_name*) immediately following the flag list and separated from it by a space.

-x Extracts the files identified in the *file* operands from the archive specified in the *archive* operand. The contents of the archive file are not changed. If no *file* operands are specified, all files in the archive are extracted. If the filename of a file to be extracted from the archive is longer than the name length supported for the target directory, an error results and, unless **-T** has also been specified, the file is not extracted. If **-T** has also been specified, the file is extracted and given a truncated filename. The modification time of each extracted file is set to the time the file is extracted from the archive.

DESCRIPTION

The **ar** command creates and maintains groups of one or more named files as a single archive (library) file written in **ar** archive format. After an archive file has been created, new files can be added to the archive file, and existing files can be extracted from it, deleted, or replaced.

The **ar** command accepts the following as archive members:

- All OSS files
- Guardian TNS code files
- Accelerated object files
- Guardian C text files (file code 180 files)
- TNS/R (PIC and non-PIC) native object files
- TNS/E native linkfiles or loadfiles

You can mix one or more object file formats in one archive file. However, such an archive file cannot be used by the Binder or the **nld**, **ld**, or **eld** utility.

If an archive file contains only files in a single object file format, an archive symbol table is created as the first file member of the archive file. The format of this table depends upon the linker utility for the corresponding object file format. This table is maintained by **ar** and used by the Binder (for TNS and accelerated object files), the **nld** utility (for TNS/R native non-PIC object files), the **ld** utility (for TNS/R native PIC object files), and the **eld** utility (for TNS/E native PIC files) to search the archive file and extract archive members from it.

When the **ar** utility is used to create or update the content of such an archive, the symbol table is rebuilt automatically. The **-s** flag forces the symbol table to be rebuilt.

An archive file embedded as a member of another archive file is not used by the Binder or the **nld** utility, the **ld** utility, or the **eld** utility.

All *file* operands can be pathnames. However, files within archive files are given a filename, which is the last component of the pathname used when the file was entered into the archive file.

Multiple files within the archive file can be identically named. In these cases, each *file* and *position_name* operand matches only the first archive file having a name that is the same as the last component of the *file* or *position_name* operand.

An archive file can be created for the OSS, Guardian, or a target-independent execution environment. An OSS archive file is made up of OSS and target-independent object files. A Guardian archive file is made up of Guardian and target-independent object files. A target-independent archive file is made up of target-independent object files.

Text files are always target-environment independent.

If **ar** detects mixing of OSS and Guardian environment object files, it issues a warning message but does not prevent you from creating an archive file of mixed environments. It is your responsibility to ensure that procedures used for resolving references work in the target environment; otherwise, program execution results in runtime errors.

If **ar** detects mixing of TNS/E native object files of 32-bit and TNS/E 64-bit data models, it issues a warning message, but does not prevent you from creating an archive file.

The file structure of archive files is defined in the **ar.h** header file.

Operands

ar supports the following operands:

archive The pathname of the archive file to be created or modified.

The maximum size of an archive file is 1,024,491,520 bytes. If operations on the archive file cause the file to exceed that size limit, the **ar** command returns an error message and the archive file becomes corrupted because **ar** cannot create the symbol table information.

file The pathname of a file to be processed. Only the last component of a pathname is used in comparing the filename with names of files in the archive file. If two or more *file* operands have the same last component in their pathname (basename), they are all archived as separate members with the same name. **ar** does not truncate valid filenames of files added to or replaced in the archive file.

position_name The name of a file within the archive file; used for relative positioning. See flags **-m** and **-r**.

Environment Variables

The following environment variable affects the execution of **ar**.

TMPDIR Determines the pathname that overrides the default directory for temporary files, if any.

This utility supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

Input Files

The input file identified in the *archive* operand must be a file in the format created by the command **ar -r**.

Standard Output

Standard output depends on the flags used with **ar**.

- If the **-d** flag is used with the **-v** flag, the standard output format is:

  `"d - %s \n",` *file*

  where *file* is the *file* operand specified on the command line.

- If the **-p** flag is used with the **-v** flag, **ar** precedes the contents of each file with:

  `"\n%s\n\n",` *file*

  where *file* is the *file* operand specified on the command line if *file* operands were specified, or the name of the file in the archive if the *file* operands were not specified.

- If the **-r** flag is used with the **-v** flag and the file specified in the *file* operand is already in the archive, the standard output is:

  `"r - %s\n",` *file*

  where *file* is the *file* operand specified on the command line.

- If the file specified in the *file* operand is being added to the archive and the **-r** flag is used, the standard output is:

  `"a - %s\n",` *file*

  where *file* is the *file* operand specified on the command line if *file* operands were specified, or the name of the file within the archive file if the *file* operands were not specified.

- If the **-t** flag is used, **ar** writes the names of the files to the standard output file in the format:

  `"%s\n",` *file*

  where *file* is the *file* operand specified on the command line if *file* operands were specified, or the name of the file within the archive file if the *file* operands were not specified.

- If the **-t** flag is used with the **-v** flag the standard output is:

  `"%s %u/%u %u %s %d %d:%d %d %s\n",` *member-node*, *user-ID*, *group-ID*, *number-of-bytes-in-member*, *abbreviated-month*, *day-of-the-month*, *hour*, *minute*, *year*, *file*

  Where:

  file is the *file* operand specified on the command line if *file* operands were specified, or the name of the file within the archive file if the *file* operands were not specified.

  member-node
  is formatted the same as the *file-node* string in the output of the **ls** command, except for the first character, the *entry-type*, which is not used. The string represents the file mode of the archive member at the time it was added to, or replaced within, the archive file.

  user-ID is the OSS user ID (UID) associated with the file at the time it was added to, or replaced within, the archive file.

  group-ID
  is the group ID (GID) associated with the file at the time it was added to, or replaced within, the archive file.

  number-of-bytes-in-member
  is the size of the file in bytes at the time it was added to, or replaced within, the archive file.

  The following output entries record the last modification time of a file when the file was most recently added or replaced in the archive file.

abbreviated-month
is equivalent to the `%b` format in the output of the **date** command.

day-of-the-month
is equivalent to the `%e` format in the **date** command.

hour is equivalent to the `%H` format in the **date** command.

minute is equivalent to the `%M` format in the **date** command.

year is equivalent to the `%Y` format in the **date** command.

- If the **-x** flag is used with the **-v** flag, the standard output format is:

  `"x - %s\n",` *file*

  where *file* is the *file* operand specified on the command line if *file* operands were specified, or the name of the file within the archive file if the *file* operands were not specified.

Standard Error

Messages from the **ar** command are only diagnostic messages. The diagnostic message about creating a new archive file when the **-c** flag is not specified does not modify the exit status.

Output Files

Output is the normal archive file name with string **"!arch\n"** at the beginning.

EXAMPLES

1. The command

   ```
   ar -rcv newlib a1.o a2.o a3.o a4.o
   ```

   creates an archive file named `newlib` (if an archive file by that name does not already exist) with the files `a1.o, a2.o, a3.o,` and `a4.o` as its members. If `newlib` already exists and some of the files named `a1.o, a2.o, a3.o,` and `a4.o` are in the archive, these files are replaced by the new files. Files that are not current members of the archive are added to the end of the archive file. The **-v** flag causes a file-by-file replacement or addition message to be displayed on the terminal when each operation is initiated.

2. The command

   ```
   ar -tv newlib
   ```

   displays a detailed table of contents of the archive `newlib`.

3. The command

   ```
   ar -x oldlib x.o y.o
   ```

   extracts member files `x.o` and `y.o` from the archive `oldlib.`

4. The command

   ```
   ar -rb abc.o oldlib x.o y.o
   ```

   adds new member files `x.o` and `y.o` to the archive file `oldlib.` The new files are inserted in front of the existing member `abc.o`.

5. The command

   ```
   ar -dv libmylib f1.0 f2.0
   ```

   deletes files `f1.0` and `f2.0` from the archive `libmylib`.

FILES

ar.h Describes the file structure of archive files.

DIAGNOSTICS

`ar: creating archive` *archive*`.`
Informative message. The archive file specified in the command did not exist and has been created.

`ar:` *archive* `contains a mix of 32-bit and 64-bit data models`
Warning message. The **ar** utility detected a mixing of TNS/E native object files of 32-bit and TNS/E 64-bit data models, but did not prevent you from creating an archive file.

`ar:` *archive*`: Guardian or User Defined Error 43`
The **ar** utility cannot obtain disk space for a file extent.

`ar:` *archive*`: Guardian or User Defined Error 45`
The resulting file size exceeds 1,024,491,520 bytes and the file is not a valid archive file.

EXIT VALUES

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

RELATED INFORMATION

Commands: **c89(1)**, **c99(1)**, **eld(1)**, **ld(1)**, **make(1)**, **nld(1)**, **nm(1)**, **strip(1)**.

Files: **ar(4)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions and UNIX extensions. The **-W** flags are HP extensions to the specification.

NAME

at - Runs commands at a user-specified later time

SYNOPSIS

at [**-c** | **-s** | **-k**]
[**-f** *file*] [**-q** *queuename*]
when [*date*] [**+***increment*]
[*command* | *file*] ...

at [**-c** | **-s** | **-k**]
[**-f** *file*] [**-q** *queuename*]
-t *time*

at -l [**-o**] [**-q** *queuename*] [*user* ...]

at -l [*job_number* ...]

at -r [[**-Fi**] *job_number* ...] | [**-u** *user*]

at -n [*user*]

FLAGS

-c Requests that **csh** be used for executing this job. **csh** is not currently supported. In the current implementation, **ksh** will be used.

-f *file* Specifies the file to be used as input instead of the standard input file.

-F Suppresses delete verification.

-i Specifies interactive delete.

-k Requests that **ksh** be used for executing this job.

-l [*job_number* ...]
Reports the scheduled jobs for the invoking user. If the *job_number* argument is specified, **at** reports only the information for the specified jobs.

If a user with appropriate privileges issues the command with this flag, all of the queued **at** commands are listed with the name of the user who issued each one. The user with appropriate privileges can also request a report of scheduled jobs for only one specified user.

-n [*user*] Requests the number of files in the queue for the current user. A user with appropriate privileges can specify a different user with the *user* argument.

-o Lists jobs in scheduled order. This flag is useful only when used with the **-l** flag.

-q *queuename*
Specifies the queue you want to use. When used with the **-l** flag, this flag limits the search to the specified queue. A queue name can be specified by **a, b**, or **e**, as described in **DESCRIPTION**. Queue **a** is the default queue.

-r *job_number* ...
Removes a job previously scheduled by **at** or **batch**, where *job_number* is the number assigned by **at** or **batch**. If you do not have appropriate privileges, you can remove only your own jobs. The **atrm** command is available to users with appropriate privileges to remove jobs issued by other users or all jobs issued by a specific user. This flag can also be used in combination with the **-i**, **-f**, and **-u** flags.

-s Requests that the Bourne shell be used for executing this job. In the current implementation, **ksh** will be used.

-t *time* Submits the job to be run at the specified time. The *time* argument must be in the format described for the **touch** command: [[*cc*]*yy*]*MMddhhmm*[**.***ss*]. (For more information, refer to the **touch(1)** reference page.)

-u *user* Deletes all jobs for the specified user. This flag must be used with the **-r** flag as follows:

at -r -u *user*

DESCRIPTION

The **at** command reads from the standard input file or accepts as arguments the names of commands to be run at a later time. The **at** command allows you to specify when the commands are to be run.

If a file specified on an **at** command line is executable (that is, has the **x** permission for the user executing the command), **at** assumes that it is a command and that the job consists only of this command. If the file is not executable, **at** assumes that you want its contents to be the instructions for the job (same as BSD **at**). If **at** cannot find the file at all, the specification is passed to the date parser. If the specification is not recognized by the date parser, the user receives the error message `Unknown word`.

Variables in the shell environment, the current directory, **umask**, and **ulimit** are retained when the commands are run. Open file descriptors, traps, and priority are lost.

You can use **at** if your login name appears in the **/usr/lib/cron/at.allow** file. If that file does not exist, **at** checks the **/usr/lib/cron/at.deny** file to determine if your login name is denied access to **at**. The **at.allow** and **at.deny** files contain one login name per line.

If neither file exists, only a user with appropriate privileges can submit a job. If the **at.allow** file does exist, the login name of a user with appropriate privileges must be included in it for that user to be able to use the command.

Operands

when The required *when* operand can be one of the following:

- The **at** command recognizes a number followed by an optional suffix. **at** interprets 1-digit and 2-digit numbers as hours. It interprets 4-digit numbers as hours and minutes.

  The **LC_TIME** environment variable specifies the order of hours and minutes. The default order is the hour followed by the minute. You can also separate hours and minutes with a : (colon). The default order is *hour*:*minute*.

  In addition, you can specify a suffix of **am**, **pm**, or **utc**. If you do not specify **am** or **pm**, **at** uses a 24-hour clock. The suffix **utc** indicates that the time is UTC (Coordinated Universal Time).

- The **at** command also recognizes the following keywords as special times: **noon**, **midnight**, **now**, **A** for a.m., **P** for p.m., **N** for noon, and **M** for midnight. The **LC_TIME** environment variable controls the additional keywords that **at** recognizes.

date You can specify the *date* operand as either a month name and a day number (and possibly a year number preceded by a comma), or a day of the week.

The **LC_TIME** environment variable specifies the order of the month name and day number (by default, month followed by day). **at** recognizes two special days, **today** and **tomorrow**, by default. **today** is the default date if the specified time is later than the current hour; **tomorrow** is the default date if the specified time is earlier than the current hour.

If the specified month is less than the current month (and a year is not given), next year is the default year.

+increment The optional *increment* operand can be one of the following:

- A + (plus sign) followed by a number and one of the following words: **minute[s]**, **hour[s**], **day[s**], **week[s**], **month[s**], or **year[s**] (or their nonEnglish equivalents).
- The special word **next** followed by one of the following words: **minute[s]**, **hour[s]**, **day[s]**, **week[s]**, **month[s]**, or **year[s]** (or their nonEnglish equivalents).

job_number Job numbers are specified as follows:

*user***.***xxxxxxxxx***.***y*

user Identifies the user who scheduled the job.

xxxxxxxxx Is a 9-digit number indicating the encoded time for the job.

y Indicates the job type or queue name as follows:

a	**at** job
b	**batch** job
e	**ksh** job

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **LC_TIME**, and **NLSPATH** environment variables.

EXAMPLES

1. To schedule a command from a terminal, enter a command similar to one of the following:

   ```
   at 5 pm Friday runme
   at now next week runme
   at now + 2 days runme
   ```

   Note that the preceding commands can be scheduled as shown only if **runme** is in the current directory.

2. To run **cal** at 3:00 in the afternoon on January 24, enter any one of the following commands:

   ```
   echo cal | at 3:00 pm January 24
   echo cal | at 3pm Jan 24
   echo cal | at 1500 jan 24
   ```

3. To list the jobs you have sent to be run later, enter:

   at -l

4. To cancel a job, enter:

   at -r super.super.586748399.a

   This cancels job **super.super.586748399.a**.

FILES

/var/adm/cron	Main **cron** directory.
/usr/lib/cron/at.allow	List of allowed users.
/usr/lib/cron/at.deny	List of denied users.
/var/spool/cron/atjobs	Queue.

NOTES

The **at** utility does not accept jobs submitted from processes whose login user ID is different from the real user ID.

CAUTIONS

It is recommended that you do not use unspecified queues (queues other than **a**, **b**, and **e**). The results of such use are unspecified.

EXIT VALUES

The **at** command returns the following exit values:

0 (zero)	The **at** utility successfully submitted, removed, or listed one or more jobs.
>0	An error occurred.

RELATED INFORMATION

Commands: **atq(1)**, **atrm(1)**, **batch(1)**, **cron(8)**, **kill(1)**, **ps(1)**, **sh(1)**, **touch(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions, except for the following features:

- The **-m** flag is not supported.
- The **SHELL** and **TZ** environment variables are not used.

The following features are HP extensions to the XPG4 Version 2 specification:

- The **-c**, **-F**, **-i**, **-k**, **-s**, and **-u** flags are supported.

NAME

atq - Prints the queue of jobs waiting to be run

SYNOPSIS

atq [**-c** | **-n**] [**-q** *queuename*] [*user* ...]

FLAGS

-c Sorts the queue by the time that the **at** command was issued.

-n Prints only the number of files currently in the queue.

-q *queuename*
Specifies the queue you want to use.

DESCRIPTION

The **atq** command prints the queue of jobs waiting to be run at a later date. These jobs were created with the **at** command.

If no flags are specified in the **atq** command, the queue is sorted in the order that the jobs will be executed. If the user has appropriate privileges and one or more user names are provided, only jobs belonging to those users are displayed. If no user names are provided, then a list of all jobs submitted is displayed.

If flags are specified, the list of jobs belonging to the user who invoked the **atq** command is displayed.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

To look at the queue created by the **at** command, enter:

```
atq
```

If there are jobs in the queue, a message similar to the following is displayed for each job:

```
super.super.62169200.a          Tue Sep 12 11:00:00 1990
```

The **.a** extension specifies an **at** job.

FILES

/var/adm/cron	Main **cron** directory.
/usr/lib/cron/at.allow	List of allowed users.
/usr/lib/cron/at.deny	List of denied users.
/var/spool/cron/atjobs	Queue.

EXIT VALUES

The **atq** command returns the following values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **at(1)**, **atrm(1)**, **batch(1)**, **cron(8)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

atrm - Removes jobs queued by the **at** command

SYNOPSIS

atrm [**-f** | **-i**] **-a** [| *job_number* ... | *user* ...]

FLAGS

-a Removes all jobs belonging to the user invoking **atrm**. If this flag is specified by a user with appropriate privileges, all jobs on the queue are removed.

-f Suppresses the printing of information about the jobs being removed.

-i Prompts the user before a job is removed; a response of **y**, or the locale's equivalent of **y**, causes the job to be removed.

DESCRIPTION

The **atrm** command removes jobs that were put in a queue by the **at** command. If one or more job numbers are specified, **atrm** attempts to remove only those jobs.

If one or more user names are specified, all jobs belonging to those users are removed. This form of invoking **atrm** is can be used only if you have appropriate privileges.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

To remove job number **super.super.62169200.a**, created by user **super.super**, from the queue created by the **at** command, enter:

```
atrm super.super.62169200.a
```

Note that **.a** specifies an **at** job for **sh**. (The **.e** extension specifies an **at** job for **ksh**.)

FILES

/var/adm/cron	Main **cron** directory.
/usr/lib/cron/at.allow	List of allowed users.
/usr/lib/cron/at.deny	List of denied users.
/var/spool/cron/atjobs	Queue.

EXIT VALUES

The **atrm** command returns the following values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **at(1)**, **atq(1)**, **batch(1)**, **cron(8)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

awk - Manipulates text and matches patterns in files

SYNOPSIS

awk -f *program* [**-F***character*] [*file* ...]

awk [**-F***character*] *statement* ... [*file* ...]

FLAGS

-F*character*

Uses *character* as the field separator character (a space by default).

-f *program*

Searches for the patterns and performs the actions found in the file *program*.

DESCRIPTION

The **awk** command provides a flexible text-manipulation language suitable for simple report generation. It is a more powerful tool for text manipulation than either **sed** or **grep**.

The **awk** command:

- Performs convenient numeric processing.
- Allows variables within actions.
- Allows general selection of patterns.
- Allows control flow in the actions.
- Does not require any compiling of programs.

Pattern-matching and action *statements* can be specified either on the command line or in a program file. In either case, **awk** first reads all matching and action statements, then reads a line of input and compares it to each specified pattern. If the line matches a specified pattern, **awk** performs the specified actions and writes the result to standard output. When it has compared the current input line to all patterns, it reads the next line.

The **awk** command reads input files in the order stated on the command line. If you specify a filename as a **-** (dash) or do not specify a filename, **awk** reads standard input.

Enclose pattern-action statements on the command line in **'** (single quotes) to protect them from interpretation by the shell. Consecutive pattern-action statements on the same command line must be separated by a **;** (semicolon), within one set of quotes. Consecutive pattern-action statements in an **awk** program file must appear on separate lines.

You can assign values to variables on the **awk** command line as follows:

variable=value

The **awk** command treats input lines as fields separated by spaces, tabs, or a field separator you set with the **FS** variable. (Consecutive spaces are recognized as a single separator.) Fields are referenced as **$1**, **$2**, and so on. **$0** refers to the entire line.

Pattern-Action Statements

Pattern-action statements follow the form:

pattern {*action*}

If a *pattern* lacks a corresponding *action*, **awk** writes the entire line that contains the pattern to standard output. If an *action* lacks a corresponding *pattern*, **awk** applies the action to every line.

Actions

An action is a sequence of statements that follow C language syntax. These statements can include:

if (*expression***)** *statement* [**else** *statement*]

while (*expression***)** *statement*

for (*expression***;***expression***;***expression***)** *statement*

for (*variable* **in** *array***)** *statement*

break

continue

{ [*statement ...*] **}**

variable=expression

print [*expression_list*] [>*file*] [| *command*]

printf *format*[**,***expression_list*] [>*file* | >>*file* | | *command*]

next

exit [*expression*]

delete *array* [*expression*]

Statements can end with a semicolon, a newline character, or the right brace enclosing the action.

Expressions can have string or numeric values and are built using the operators **+**, **-**, , /, **%**, and ^ (exponentiation), a space for string concatenation, and the C operators **++**, **--**, **+=**, **-=**, , /=, **%=**, ^=, ***=**, >, >=, <, <=, ==, **!=**, and **?:**.

Because the actions process fields, input white space is not preserved in the output.

The *file* and *command* arguments can be literal names or expressions enclosed in parentheses. Identical string values in different statements refer to the same open file.

The **print** statement writes its arguments to standard output (or to a *file* if > *file* or >> *file* is present), separated by the current output field separator and terminated by the current output record separator.

The **printf** statement writes its arguments to standard output (or to a file if >*file* or >>*file* is present, or to a pipe if | *command* is present), separated by the current output field separator, and terminated by the output record separator. *file* and *command* can be literal names or parenthesized expressions. Identical string values in different statements denote the same open file. You can redirect the output into a file using the **print ...** > *file* or **printf (...)** > *file* statements. The **printf** statement formats its expression list according to the format of the **printf( )** subroutine.

Variables

Variables can be scalars, array elements (denoted **x[i])**, or fields.

Variable names can consist of uppercase and lowercase alphabetic letters, the underscore character, the digits (**0** to **9**), and extended characters. Variable names cannot begin with a digit.

Variables are initialized to the null string. Array subscripts can be any string; they do not have to be numeric. This approach allows for a form of associative memory. Enclose string constants in expressions in **""** (double quotes). Multiple subscripts such as [*i*,*j*,*k*] are permitted; the constituents are concatenated and separated by the value of **SUBSEP** (see the description in the following list).

There are several variables with special meaning to **awk**. They include:

ARGC Argument count, assignable.

ARGV Argument array, assignable; nonnull members are interpreted as filenames.

FS Input field separator (default is a space). If it is a space, then any number of spaces and tabs can separate fields.

NF The number of fields in the current input line (record), with a limit of 99.

NR The number of the current input line (record).

FNR The number of the current input line (record) in the current file.

FILENAME
The name of the current input file.

RS Input record separator (default is a newline character).

OFS The output field separator (default is a space).

ORS The output record separator (default is a newline character).

OFMT The output format for numbers (default **% .6g**).

SUBSEP
Separates multiple subscripts (default is 031).

Functions

Functions are defined at the position of a pattern-action statement, as follows:

function foo(a, b, c) { ... ; return x }

Arguments are passed by value if scalar and by reference if array name; functions can be called recursively. Arguments are local to the function; all other variables are global.

There are several built-in functions that can be used in **awk** actions. (For information about regular expressions as referred to in this subsection, see the **grep(1)** refeernce page.)

length(*argument***)**
Returns the length, in characters, of *argument*, or of the entire line if there is no *argument*.

blength(*argument***)**
Returns the length, in bytes, of *argument*, or of the entire line if there is no *argument*.

close(*argument***)**
Closes the file or pipe expression. Note that you must enclose a filename in double quotes when redirecting output with the **awk** command; otherwise, it is treated as an **awk** variable. For example:

```
print "Hello" > "/tmp/junk"
close ("/tmp/junk")
```

exp(*number***)**
Takes the exponential of its argument.

rand Returns a random number on (0, 1).

srand(*number***)**
Sets seed for **rand**. The default is the time of day.

log(*number***)**
Takes the base e logarithm of its argument.

sqrt(*number***)**
Takes the square root of its argument.

int(*number***)**
Takes the integer part of its argument.

substr(*string***,***position***,***number***)**
Returns the substring *number* characters long of *string*, beginning at *position*.

index(*string***,***string2***)**
Returns the position in *string* where *string2* occurs, or **0** (zero) if it does not occur.

match(*string***,***regular_expression***)**
Returns the position in *string* where *regular_expression* occurs, or **0** (zero) if it does not occur. The **RSTART** and **RLENGTH** built-in variables are set to the position and length, in bytes, of the matched string.

split(*string***,***a***,**[*regular_expression*]**)**
Splits *string* into array elements *a*[**1**], *a*[**2**], . . ., *a*[*number*], and returns *number*. The separation is done with the specified regular expression or with the **FS** field separator if *regular_expression* is not given.

sub(*regular_expression***,***string2***,**[*string*]**)**
Substitutes *string2* for the first occurrence of the regular expression *regular_expression* in *string*. If *string* is not given, the entire line is used.

gsub(*regular_expression***,***string2***,**[*string*]**)**
Same as **sub** except that all occurrences of the regular expression are replaced; both **sub** and **gsub** return the number of replacements.

sprintf(*fmt***,***expression1***,**
*expression2***, ...)**" Formats the expressions according to the **printf** format string *fmt* and returns the resulting string.

system(*command***)**
Executes *command* and returns its exit status.

The **getline** function sets **$0** to the next input record from the current input file; **getline** < *file* sets **$0** to the next record from *file*. **getline x** sets variable *x* instead. Finally, *command* **|** **getline** pipes the output of *command* into **getline**. Each call of **getline** returns the next line of output

from *command*. In all cases, *getline* returns **1** for a successful input, **0** (zero) for End-of-File, and **-1** for an error.

Patterns

Patterns are arbitrary Boolean combinations of patterns and relational expressions (the **!**, |, and **&** operators and parentheses for grouping). You must start and end regular expressions with slashes. You can use regular expressions as described for the **grep** command, including the following special characters:

+ One or more occurrences of the pattern.

? Zero or one occurrence of the pattern.

| Either of two statements.

( ) Grouping of expressions.

Isolated regular expressions in a pattern apply to the entire line. Regular expressions can occur in relational expressions. Any string (constant or variable) can be used as a regular expression, except in the position of an isolated regular expression in a pattern.

If two patterns are separated by a comma, the action is performed on all lines between an occurrence of the first pattern and the next occurrence of the second.

Regular expressions can contain extended (multibyte) characters with one exception: range constructs in character class specifications using brackets cannot contain multibyte extended characters. Individual instances of extended (multibyte) characters can appear within brackets; however, extended characters are treated as separate one-byte characters.

Inclusion of extended characters in ranges is determined by the collating sequence as defined by the current locale. The wild-card characters , **+**, and **?** match characters and character strings, not bytes.

There are two types of relational expressions you can use. The first type has the form:

expression match_operator pattern

where *match_operator* is either: **~** (for *contains*) or **!~** (for *does not contain*).

The second type has the form:

expression relational_operator expression

where *relational_operator* is any of the six C relational operators: <, >, <=, >=, ==, and **!=**. A conditional can be an arithmetic expression, a relational expression, or a Boolean combination of these expressions.

You can use the **BEGIN** and **END** special patterns to capture control before the first and after the last input line is read, respectively. **BEGIN** must be the first pattern; **END** must be the last. **BEGIN** and **END** do not combine with other patterns.

You have two ways to designate a character other than white space to separate fields. You can use the **-F***character* flag on the command line, or you can start *program* with the following sequence:

BEGIN { FS = *c* }

Either action changes the field separator to *c*.

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, add **0** (zero) to it. To force it to be treated as a string, append a null string ("").

EXAMPLES

1. To display the lines of a file longer than 72 bytes, enter:

   awk 'length >72' chapter1

   This command selects each line of the file **chapter1** that is longer than **72** bytes. **awk** then writes these lines to standard output because no action is specified.

2. To display all lines between the words **start** and **stop**, enter:

   awk '/start/,/stop/' chapter1

3. To run an **awk** program (**sum2.awk**) that processes a file (**chapter1**), enter:

   awk -f sum2.awk chapter1

4. To print the first two fields of a file named filename in reverse order, enter:

   awk '{ print $2, $1 }' filename

5. The following **awk** program prints the first two fields of the input file in reverse order, with input fields separated by a comma and a space, then adds up the first column and prints the sum and average:

```
BEGIN   { FS = ",[ ]*|[ ]+" }
        { print $2, $1}
        { s += $1 }
END     { print "sum is", s, "average is", s/NR }
```

RELATED INFORMATION

Commands: **grep(1)**, **nawk(1)**, **sed(1)**.

Functions: **printf(3)**.

Files: **locale(4)**.

NAME

banner - Creates a large banner

SYNOPSIS

banner *message*

DESCRIPTION

The **banner** command prints the specified *message* in large letters on the standard output file.

Each line in the banner can be up to 10 uppercase or lowercase characters long. On output, all characters appear in uppercase, with the lowercase input characters appearing smaller than the uppercase input characters. The **banner** command displays only ASCII characters.

EXAMPLES

1. To display a one-word banner, enter:

   banner SMILE

2. To display more than one word on a line, enclose the text in quotes, for example:

   banner "Out to" Lunch

   This command displays **Out to** on one line and **Lunch** on the next line.

RELATED INFORMATION

Commands: **echo(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with the following exceptions:

- Only one message string is allowed.
- The format of the output is specified.

NAME

basename - Returns specified parts of pathnames

SYNOPSIS

basename *string* [*suffix*]

DESCRIPTION

The **basename** command reads the string specified on the command line, deletes the portion from the beginning to the last / (slash), and writes the base filename to standard output. If *suffix* is specified on the command line and *suffix* appears in *string*, the string is returned with the suffix removed.

The **basename** command is generally used inside command substitutions within a shell procedure to specify an output filename that is some variation of a specified input filename.

EXAMPLES

1. To display the base filename of a shell variable, enter:

   basename $WORKFILE

   This command displays the base filename of the value assigned to the **WORKFILE** shell variable. If **WORKFILE** is set to **/u/gabe/program.c**, then **program.c** is displayed.

2. To construct, in a shell script, a filename that is the same as another filename except for its suffix, enter:

   OFILE=‘basename $1 .c‘.o

   This command assigns to **OFILE** the value of the first positional parameter (**$1**), but with its **.c** suffix changed to **.o**. If **$1** is **/u/jim/program.c**, then **OFILE** becomes **program.o**.

RELATED INFORMATION

Commands: **dirname**(1), **sh**(1).

NAME

batch - Runs commands at a system-determined later time

SYNOPSIS

batch

DESCRIPTION

The **batch** command reads from the standard input file the names of commands to be run at a later time. The **batch** command runs these jobs when the system load level permits.

The **batch** command is equivalent to the following **at** command:

at -q b now

Queue **b** is an **at** queue for batch jobs.

The user redirects the errors and output from these jobs.

Variables in the shell environment, the current directory, **umask**, and **ulimit** are retained when the commands are run. Open file descriptors, traps, and priority are lost.

You can use **batch** if your login name appears in the **/usr/lib/cron/at.allow** file. If that file does not exist, **batch** checks the **/usr/lib/cron/at.deny** file to determine if your login name is denied access to **batch**. The **at.allow** and **at.deny** files contain one login name per line.

If neither file exists, only a user with appropriate privileges can submit a job. If the **at.allow** file does exist, the login name of a user with appropriate privileges must be included in it for that user to be able to use the command.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **LC_TIME**, and **NLSPATH** environment variables.

EXAMPLES

To execute a command when the system load level permits, enter:

```
batch
cat infile > outfile
<EOF>
```

where **<EOF>** is the End-of-File character.

FILES

/var/adm/cron	Main **cron** directory.
/usr/lib/cron/at.allow	List of allowed users.
/usr/lib/cron/at.deny	List of denied users.
/var/spool/cron/atjobs	Queue.

NOTES

The **batch** utility does not accept jobs submitted from processes whose login user ID is different from the real user ID.

EXIT VALUES

The **batch** command returns the following exit values:

0 (zero)	The **batch** utility successfully finished its processing.
>0	An error occurred. The job will not be scheduled.

RELATED INFORMATION

Commands: **at(1)**, **atq(1)**, **atrm(1)**, **cron(8)**, **kill(1)**, **ps(1)**, **sh(1)**.

STANDARDS CONFORMANCE

This commmand conforms to the XPG4 Version 2 specification with extensions, except for the following features:

- Mail notification does not occur.
- The **SHELL** and **TZ** environment variables are not used.

NAME

bc - Arbitrary-precision arithmetic language processor

SYNOPSIS

bc [**-cl**] [*file* ...]

The **bc** command is an interactive program that provides unlimited precision arithmetic. It is a preprocessor for the **dc** command.

FLAGS

-c Compiles *file*, but does not invoke **dc**.

-l Includes a library of mathematical functions and initializes **scale** to 20, instead of the default of 0 (zero).

DESCRIPTION

The **bc** command invokes **dc** automatically, unless the **-c** (compile only) flag is specified. If the **-c** flag is specified, the output from **bc** goes to the standard output.

The **bc** command lets you specify an input and output base in decimal, octal, or hexadecimal with the **ibase** and **obase** keywords (the default is decimal). The command also has a scaling provision for decimal point notation through the **scale** keyword. The syntax for **bc** is similar to that of the C language.

The **bc** command takes input first from the specified file. This input file can contain additional functions besides internal functions such as **sqrt** and **length**, and functions included in the math library. When **bc** reaches the end of the input file, it reads standard input.

The following are valid items that can be used in the input file and for standard input. In the following description of syntax for **bc**, *letter* means one of the ASCII letters **a**-**z**.

The combination of a \ (backslash) character immediately followed by a newline character delimits lexical tokens with the following exceptions:

- When it is interpreted as a literal newline character in **STRING** tokens
- When it is ignored as part of a multiline **NUMBER** token

Comments

Comments are enclosed in **/*** and ***/**.

Names

Simple variables: *letter*

Array elements: *letter*[*expression*]

The words **ibase**, **obase**, and **scale**

Other Operands

Arbitrarily long numbers with optional sign and decimal point.

(*expression*)

sqrt (*expression*)

length (*expression*) Number of significant decimal digits

scale (*expression*) Number of digits to right of decimal point

letter (*expression***,...,***expression*)

Operators

+ - * / % ^ (% is remainder; ^ is power)

++ -- (prefix and suffix; apply to names)

== <= >= != <>

= =+ =- =* =/ =% ^=

+= -= *= /= %=

Statements

expression

{*statement***;**...**;***statement*}

if (*expression*) *statement*

while (*expression*) *statement*

for (*expression***;***expression***;***expression*) *statement*

(null statement)

break

quit

The statement following a **for** or **while** statement must begin on the same line.

Function Definitions

define *letter* **(***letter***,...,***letter***) {**

    auto *letter***,...,***letter*
    *statement***;...***statement*
    return (*expression* **)**

}

Functions in -l Math Library

s(x)	sine
c(x)	cosine
e(x)	exponential
l(x)	log
a(x)	arctangent
j(n,x)	Bessel function

General Syntax

All function parameters are passed by value.

The value of a statement that is an expression is displayed unless the main operator is an assignment. A semicolon or newline character separates statements. Assignments to **scale** control the number of decimal places printed on output and maintained during multiplication, division, and exponentiation. Assignments to **ibase** or **obase** set the input and output number radix, respectively.

The same letter may refer to an array, a function, and a simple variable simultaneously. Automatic variables are pushed down during function calls. All other variables are global to the program. When you use arrays as function parameters, or define them as automatic variables, empty brackets must follow the array name.

All **for** statements must have all three *expression*s.

The **quit** statement is interpreted immediately, not when **bc** is evaluating statements.

EXAMPLES

When you enter **bc** expressions directly from the keyboard, press the End-of-File key sequence to end the **bc** session and return to the shell command line.

1. To use **bc** as a calculator, proceed as follows:

   Enter:

   bc
   1/4

   The system responds as follows:

   ```
   0
   ```

   Enter:

   scale = 1 /* Keep 1 decimal place */
   1/4

   The system responds as follows:

   ```
   0.2
   ```

   Enter:

   scale = 3 /* Keep 3 decimal places */
   1/4

   The system responds as follows:

   ```
   0.250
   ```

   Enter:

   16+63/5

   The system responds as follows:

   ```
   28.600
   ```

   Enter:

   (16+63)/5

   The system responds as follows:

   ```
   15.800
   ```

   Enter:

   71/6

   The system responds as follows:

   ```
   11.833
   ```

Enter:

1/6

The system responds as follows:

```
0.166
```

You may type the comments (enclosed in **/* */**), but they are provided only for your information. The **bc** command displays the value of each expression when you press <**Return**>, except for assignments. Exit by typing quit followed by <**return**>

2. To convert numbers from one base to another, proceed as follows:

   Enter:

   ```
   bc
   obase = 16          /* Display numbers in Hexadecimal */
   ibase = 8 /* Input numbers in Octal */
   12
   ```

   The system responds as follows:

   ```
   A
   ```

   Enter:

   123

   The system responds as follows:

   ```
   53
   ```

   Enter:

   123456

   The system responds as follows:

   ```
   A72E
   ```

3. To write and run C-like programs, proceed as follows:

   Create the following file **prog.bc**:

   ```
   /* compute the factorial of n */

   define f(n) {
           auto i, r;

           r = 1;
           for (i=2; i<=n; i++) r =* i;
           return (r);
   }
   ```

   Note that the statement following a **for** or **while** statement must begin on the same line.

Enter:

bc -l prog.bc

This statement interprets the **bc** program saved in **prog.bc**, then reads more **bc** command statements from standard input (the keyboard). Starting the **bc** command with the **-l** flag makes the math library available. This example uses the **e** (exponential) function from the math library, and **f** is defined in the program **prog.bc**.

Enter:

e(2) /* e squared */

The system responds as follows:

```
7.38905609893065022723
```

Enter:

f(5) /* 5 factorial */

The system responds as follows:

```
120
```

Enter:

f(10) /* 10 factorial */

The system responds as follows:

```
3628800
```

4. To convert an infix expression to Reverse Polish Notation (RPN), enter:

   Enter:

   bc -c
   (a * b) % (3 + 4 * c)

   The system responds as follows:

   ```
   lalb* 3 4lc*+%ps.
   ```

   This statement compiles the **bc** infix-notation expression into an expression that the **dc** command can interpret. **dc** evaluates extended RPN expressions. In the compiled output, the lowercase **l** before each variable name is the **dc** subcommand to load the value of the variable onto the stack. The **p** displays the value on top of the stack, and the **s.** discards the top value by storing it in register . (dot). You can save the RPN expression in a file for **dc** to evaluate later by redirecting the standard output of this command.

FILES

/usr/lib/lib.b	Mathematical library.
/bin/dc	Desk calculator proper; uses **bc** as preprocessor.

RELATED INFORMATION

Commands: **dc**(1).

NAME

bg - Causes processes to run in the background

SYNOPSIS

bg [*job* ...]

DESCRIPTION

The **bg** command causes stopped processes specified as *job* to run in the background. If no process is specified as *job*, the most recently stopped process is restarted as a background process. (See **Jobs** for a description of the format of *job*.)

EXAMPLES

1. The following command restarts, as a background process, the previously stopped job whose job number is 149.

   bg %149

NOTES

The **bg** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **jobs**(1), **sh**(1).

NAME

break - Exits from **for, while, until,** or **select** loop

SYNOPSIS

break [*n*]

DESCRIPTION

Exits from the enclosing **for, while, until,** or **select** loop, if any. If *n* is specified, breaks at the *n*th enclosing level.

EXAMPLES

1. The following shell script demonstrates the use of the **break** command to exit from a loop:

```
for x in 1 2 3 4 5
do
if [ $x != 3 ]
then
print $x
else
break
fi
done
```

EXIT VALUES

If an invalid argument is specified, the exit value is greater than 0 (zero).

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **break** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

Section 2. User Commands (c)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letter **c**.

NAME

c89 - Compiles C and C++ programs using the native compilers

SYNOPSIS

c89

[**-c** | **-Wnolink**]
[[**-D** *name*[**="***value***"**]] ...] [**-E**] [**-g**]
[**-I** *directory* ...]
[**-L** *directory* ...] [**-o** *outfile*] [**-O**] [**-s**]
[**-U** *name*]
[**-Wallow_cplusplus_comments**]
[**-Wallow_extern_explicit_instantiation**]
[**-Wansistreams**]
[**-Wbasename**]
[**-WBdllsonly** | **-WBdynamic** | **-WBstatic**]
[**-Wbitfield_container=***value*]
[**-Wbuild_neutral_library**]
[**-WC**]
[**-Wc99lite**]
[**-Wcall_shared** | **-Wnon_shared** | **-Wshared**]
[**-Wcodecov**]
[**-Wcolumns=***c*]
[**-Wcplusplus**]
[**-WD***name*[**="***value***"**]]
[**-Wdryrun**]
[**-Weld=***args*]
[**-Weld_obey=***file*]
[**-Wenv=***env*]
[**-Werrors=***e*]
[**-W**[**no**]**extensions**]
[**-Wextern_data=**{**no_gp** | **gp_ok**}]
[**-Wfieldalign=***align*]
[**-Wforce_static_typeinfo**]
[**-Wforce_static_vtbl**]
[**-Wforce_vtbl** | **-Wsuppress_vtbl**]
[**-Wglobalized**]
[**-WH**]
[**-Wheap=***n*[**b** | **w** | **p**]
[**-Whelp** | **-Wusage**]
[**-Whighpin=**{**on** | **off**}]
[**-Whighrequesters=**{**on** | **off**}]
[**-WIEEE_float** | **-WTandem_float**]
[**-Wilp32** | **-Wlp64**]
[**-W**[**no**]**include_whole**]
[**-W**[**no**]**inline**]
[**-Winline_compiler_generated_functions**]
[**-Winline_limit=***n*]
[**-Winline_string_literals**]
[**-W**[**no**]**innerlist**]
[**-W**[**no**]**inspect**]
[**-Wkr**]
[**-Wld="***args***"**]
[**-Wld_obey="***file***"**]
[**-Wlines=***l*]

[**-W**[**no**]**list**]
[**-WM**]
[**-W**[**no**]**map**]
[**-Wmigration_check**]
[**-Wmigration_check=32to64**]
[**-WmoduleCatalog=**"*catalog_spec*"]
[**-WmoduleGroup**[="[*group_spec*"]]]
[**-WmoduleSchema=**"*schema_spec*"]
[**-WmoduleTableSet**[="[*tableset_spec*"]]]
[**-WmoduleVersion**[="[*version_spec*"]]]
[**-Wmultibyte_char**]
[**-Wmxcmp**[="*args*"]]
[**-Wmxcmp_add=**"*args*"]
[**-Wmxcmp_files=**"*file*"[**,**...]]
[**-Wmxcmp_querydefault=**"*attr_name=attr_value*"[**,**...]]
[**-Wnld=**"*args*"]
[**-Wnld_obey**="*file*"]
[**-Wnoexceptions**]
[**-Wnomain**]
[**-Wnostdinc**]
[**-Wnostdlib**]
[**-WOlimit=***value*]
[**-Woptfile=**"*filename*"]
[**-Woptimize**="*opt*"]
[**-W**[**no**]**optional_lib**]
[**-W**[**no**]**overflow_traps**]
[**-WP**]
[**-Wpool_string_literals**]
[**-Wprofdir=***name*]
[**-Wprofgen**]
[**-Wprofuse**[=*filename*]]
[**-Wr**]
[**-W**[**no**]**reexport**]
[**-Wrefalign=***ref*]
[**-WRefMemFuncsOnly**]
[**-W**[**no**]**remarks**]
[**-Wrunnamed**]
[**-WRVU=**{*g-series-rvu* | *h-series-rvu*}]
[**-W**[**no**]**saveabend**]
[**-Wsavetemps**]
[**-Wsql**[="*args*"]]
[**-Wsqlcomp**[="*args*"]]
[**-Wsqlmx**[="*args*"]]
[**-Wsqlmxadd**[="*args*"]]
[**-Wsrl**]
[**-W**[**no**]**stdfiles**]
[**-W**[**no**]**suppress**]
[**-Wsyntax**]
[**-Wsystype=**{**guardian** | **oss**}]
[**-Wtarget=***platform*]
[**-Wtimestamp=***value*]
[**-Wu=**"*symbol_name*"]
[**-Wv**]

[**-Wverbose**]
[**-Wversion1** | **-Wversion2** | **-Wversion3**]
[**-Ww**]
[**-W**[**no**]**warn**[=*w* [**,***w*] ...]]
[**-Wx**]
operand ...

FLAGS

-c | **-Wnolink**

Compiles the specified C or C++ source files but suppresses linking, even if another flag specifies linking.

You cannot specify the **-c** flag if you use the **-Wshared** flag.

-D *name*[**="***value***"**]

Defines the preprocessor symbol *name* as *value*. It is equivalent to a **#define** directive in the source. If no *value* is given, *name* is defined as 1. The **-D** flag has lower precedence than the **-U** flag. Thus, if *name* is specified in both a **-U** and a **-D** flag, *name* is undefined regardless of the order of the flags.

Use this flag to define compiler feature-test macros.

When the NonStop SQL/MX preprocessor is invoked, all **-D** specifications are automatically passed to the preprocessor as the preprocessor's **-d** options.

-E Preprocesses the specified source files. No compilation or linking is performed. Output is sent to the standard output file and contains **#line** directives.

If the **-Wsql** flag is specified, embedded NonStop SQL/MP statements are processed. If the **-Wsqlmx** flag is specified, embedded NonStop SQL/MX statements are processed.

The **-WH** and **-WM** flags override the **-E** flag.

-g Produces in the object or executable files information (symbol tables) used for symbolic debugging.

-I *directory*

Adds *directory* to the list of directories searched to locate **#include** files with relative pathnames. (Relative pathnames do not begin with a slash, '/'). **#include** filenames enclosed in double quotes are searched for first in the directory of the file with the **#include** directive, then in directories named with **-I** flags, and last in the standard include directories. **#include** filenames enclosed in angle brackets (<>) are searched for first in directories named with **-I** flags and then in the standard include directories. Refer to the **Standard Include Directories** subsection for details.

-L *directory*

Adds *directory* to the list of directories searched to locate libraries specified by operands of the form **-l** *library*. See the **Operands** subsection for details.

-o *outfile* Uses the pathname *outfile* instead of the default pathname **a.out** for the name of the output object file.

If only one source file is specified and the **-c** flag is specified, the generated output is placed into *outfile*. Only one **-o** flag can be specified. The file specified cannot be an SQL preprocessing output file.

If a single source file is compiled and linked in one invocation of **c89**, and if the *outfile* is the same name as that of the input object file, **c89** issues a warning message and places the output in a temporary file.

-O Sets the compiler to optimization level 2. This flag is equivalent to a **-Woptimize=2** flag.

-s Strips symbolic information not required for proper execution from object and executable files. The resulting object file cannot be debugged using a symbolic debugger. This flag is ignored if the **-Wr** flag is also specified.

-U *name* Removes any initial definition of the preprocessor symbol *name*. The **-U** flag has higher precedence than the **-D** flag. If *name* is specified in both a **-U** and a **-D** flag, *name* is undefined regardless of the order of the flags.

-Wallow_cplusplus_comments
Directs the compiler to allow comments that use the C++ comment style in C source files.

-Wallow_extern_explicit_instantiation
Allows an **extern** storage attribute to be applied to an explicit template instantiation. This flag suppresses the instantiation of the template. If this flag is omitted, the template is instantiated.

-Wansistreams
Generates a Guardian program that opens text files as type 180 instead of type 101 if **-Wsystype=guardian** is specified. (By default Guardian programs open text files as type 101.) This flag is ignored when **-Wsystype=oss** is used. OSS programs can open text files only as type 180.

-Wbasename
Directs the compiler to place only the last part of the source file name, known as the basename, into the dynamic information (DYN) file when the instrumented process runs.

If you use the **-Wbasename** option and the **-Wprofgen** option to compile a source file, you also must use the **-Wbasename** option when you use the **-Wprofuse** option to compile this source file. When you use the **-Wbasename** option, the source file is not required to be in the same location as it was when you compiled the file using the **-Wprofgen** option, but the basename must be the same.

If you did not use both the **-Wbasename** and the **-Wprofgen** options when you compiled this source file, do not specify the **-Wbasename** flag when you compile this source file with the **-Wprofuse** flag. If you do not use the **-Wbasename** flag, when you compile the source file and specify the **-Wprofuse** flag, the source file must be in the same location as it was when you used the **-Wprofgen** flag to compile the file.

This flag is valid only for TNS/E-targeted compilations. For more information about profile-guided optimization and the rules for using the **-Wbasename** flag, see the *Code Profiling Utilities Manual*.

-WBdllsonly
Tells the **ld** or **eld** linker to limit searches to position-independent code (PIC) files that are dynamic-link libraries (DLLs) when resolving the file names specified for the **-l** operands and **-L** flags.

If a file name is qualified, the linker searches for a DLL with that name.

If a filename is unqualified, in each search path, the linker first searches for a DLL with the file name as specified in the **-l** operand or **-L** flag. If the linker cannot find a DLL, the file name is unqualified, and the search path is not in the Guardian file system (/G), then the linker prefixes **lib** and suffixes **.so** to the file name and searches again. If the linker still cannot find the DLL, it searches the path again with the same prefix but with

.srl as the suffix. For more information on search paths, see the **Finding Libraries** subsection of the **ld(1)** or **eld(1)** reference page under **DESCRIPTION**.

When a DLL cannot be found, the linker issues an error message unless its **-allow_missing_libs** flag is specified.

The **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags are search control toggles. Multiple flags can be specified in a single linker invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** operands and **-L** flags and search for just archive files for others. The default library search control is **-WBdynamic**.

-WBdynamic

Specifies that the linker utility uses dynamic linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Dynamic linking is in effect until a **-WBstatic** flag is specified. **-WBdynamic** is the default setting. Refer to the **Dynamic and Static Linking** subsection for details.

-WBstatic

Specifies that the linker utility uses static linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Static linking is in effect until a **-WBdynamic** flag is specified. **-WBdynamic**, not **-WBstatic**, is the default setting. Refer to the **Dynamic and Static Linking** subsection for details.

-Wbitfield_container=*value*

Directs the compiler to accept larger and more flexible bit fields where *value* is one of the following:

int Directs the compiler to pack bit fields into 32-bit **int**s. In this mode, the compiler will not accept bit fields larger than 32 bits. The compiler returns an error for any bit field declared to be of type **long long** (unless the **-Wextensions** flag is also specified). The compiler returns warnings for other non-standard integer types.

long Directs the compiler to pack bit fields whose base type is larger than 32-bits into 64-bit **int**s. All other bit fields will be packed into 32-bit **int**s. In this mode, the compiler accepts the **long long** and **long** bit-field types and up to 64 bits in length. The compiler returns warnings for other non-standard integer types.

all Directs the compiler to pack all bit fields into **int**s defined by their base type. The compiler will accept any integer type for a bit field. This mode provides additional compatibility with the various methods other compilers may use to pack bit fields.

The default *value* is **int** except when the **-Wlp64** flag is specified, in which case the default becomes **long**.

Note: The above rules apply to bit fields declared in **auto** or **platform** (the default) **struct**s. Any bit field declared in a **shared2 struct** will continue to follow **shared2** rules. Bit fields in **shared2 struct**s can never be larger than 32-bits. Bit fields declared in **shared8 struct**s can be larger than 32-bits (if the value is **long** or **long**), but the compiler will emit a warning. Bit fields in **shared8 struct**s will be packed as before except that bit fields larger than 32-bits will be packed into 64-bit containers.
This flag is supported on systems running J06.13 or later J-series RVUs or H06.24 or later H-series RVUs only.

-Wbuild_neutral_library
Specifies that the compiler should issue an error message when it encounters any exported or imported interface in a DLL that depends on types marked as being incompatible with the neutral C++ dialect.

This flag is valid only for TNS/E-targeted C++ compilations and only when the **-Wversion2** or **-Wversion3** flag is also used.

-WC Retains comments when preprocessing files. Comments are removed from preprocessor output by default.

-Wc99lite
Enables selected c99 features. This flag:

- Must not be used with the **-Wsql** or the **Wkr** flag.
- Has an effect only when you compile C source code. It has no effect when you compile C++ source code and no diagnostic message is issued when you use this flag when you compile C++ source code.
- Has an effect only if you also specify either the **-Wtarget=ipf** flag or the **-Wtarget=tns/e** flag. If you specify the **-Wtarget=mips** or the **-Wtarget=tns/r** flag, the compiler issues a warning message.

If you want to use c99 features not offered through the **-Wc99lite** option, you can use the **c99** utility. For more information about **c99** utility, see the **c99(1)** reference page.

-Wcall_shared | **-Wnon_shared** | **-Wshared**
Directs the compiler to create a specific type of object file:

-Wcall_shared Specifies that the object file should be a PIC file; the **ld** or **eld** linker is invoked. If the **-c** flag is also specified, the file is a linkfile. Otherwise, the file is an executable object file (loadfile).

This is the default behavior for a TNS/E-targeted compilation.

You cannot use this flag if you use the **-Wsrl** flag.

-Wnon_shared
Specifies that the object file should not be a PIC file; the **nld** linker is invoked. You can use this flag only for a TNS/R-targeted compilation; this flag is ignored when specified for a TNS/E-targeted compilation.

This is the default behavior for a TNS/R-targeted compilation.

-Wshared Specifies that the file should be a PIC DLL; the **ld** or **eld** linker is invoked. You cannot use this flag if you use the **-c** or **-Wsrl** flag.

-Wcodecov
Directs the compiler to create an instrumented object file and to create or add to an existing SPI file. This flag has an effect only if you also specify either the **-Wtarget=ipf** flag or the **-Wtarget=tns/e** flag.

The first time the **-Wcodecov** flag is used to compile a program, the compiler creates a Static Profiling Information (SPI) file. This file is one of the input files for the Code Coverage tool. If the program is compiled in an OSS directory:

- The default name for the SPI file is **pgopti.spi**.

- If the default file is not write-accessible, the name of the SPI file created is **tpopti.spi**.
- A lock file called **pgopti.spl**. When compilation is complete, the compiler deletes this file.

If the program is compiled in a Guardian directory:

- The default name for the SPI file is **pgospi**.
- If the default file is not write-accessible, the name of the SPI file created is **tpgospi**.
- A lock file called **pgospl**. When compilation is complete, the compiler deletes this file.

If the SPI file already exists when the program is compiled with the **-Wcodecov** flag, the compiler updates or adds information to the existing SPI file. If more than one SPI file exists for the same program, you must concatenate the files manually before you can use the resulting file as input to the Code Coverage Tool.

For more information about the Code Coverage Tool, see the *Code Profiling Utilities Manual*.

-Wcolumns=*c*

Specifies the maximum number of columns in input source files to process. *c* is in the range 20 through 32767. Text in columns beyond column *c* is ignored.

-Wcplusplus

Directs **c89** to assume that files with a **.c** or **.i** suffix contain C++ source code, and defines the feature-test macro **__cplusplus**. If linking occurs, this flag directs the linker utility to search the C++ standard run-time library.

If this flag is omitted and none of the operand filenames end in **.C**, **.cpp**, **.cc**, or **.cxx**, then source files are compiled as C files only and, if **-c** is not specified, are only linked with the C standard library. See **Standard Libraries** for details.

-WD*name*[="*value*"]

Specifies a macro that is defined only during the NonStop SQL/MX preprocessing step. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about its **-d** *flag*=[*value*] option. This flag can be specified more than once.

Note that all **-D** values that are supplied to **c89** are automatically passed as **-d** options to the NonStop SQL/MX preprocessor.

This flag is ignored unless the **-Wsqlmx** flag is also specified.

-Wdryrun

Verifies the syntax and semantics of flags and operands specified to **c89** and enables the **-Wv** flag. No compilation system components are run.

-Weld="*args***"**

Passes the arguments specified in *args* to the **eld** utility after any other arguments are passed. This flag is valid only for TNS/E-targeted compilations.

Use this flag to pass arguments to **eld** when creating a PIC file. **c89** does not check the validity of **eld** arguments.

You can only use this flag when you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when the command does not initiate linking.

-Weld_obey="*file***"**
Directs the **eld** utility to read additional command-line arguments from the command file specified in the *file* argument. The arguments are processed as if they had been passed directly to **eld** in place of *file*. **c89** does not verify the existence or readability of *file*.

The **-Weld_obey** flag is valid only for TNS/E-targeted compilations. You can only use this flag if you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when the command does not initiate linking.

-Wenv=*env*
Specifies the run-time environment. *env* can be one of **common**, **embedded**, **library**, or **libspace**. The default value is **common**.

-Werrors=*e*
Stops compiling when *e* errors have been encountered.

-W[**no**]**extensions**
Enables [disables] HP extensions. If **-Wextensions** is specified, **c89** also defines the **_TANDEM_SOURCE** feature-test macro. The default value is **-Wnoextensions**.

-Wextern_data={**no_gp** | **gp_ok**}
Determines the addressing method for external data references (objects declared **extern**). The default value is **no_gp**. This flag applies to TNS/R-targeted compilations only.

Setting **gp_ok** specifies that external data references use GP-relative addressing. GP-relative addressing can increase program performance. **gp_ok** cannot be specified for native user libraries or when the **-Wcall_shared** or **-Wshared** flag is used.

-Wfieldalign=*align*
Specifies the field alignment for structures. *align* can be one of **auto**, **cshared2**, **shared2**, **shared8**, or **platform**. The default value is **auto**. You cannot specify a **struct** tag with this flag.

-Wforce_static_typeinfo
Specifies that the **typeinfo** variables are to be static in the object file. This flag applies only to variables that are not part of an exported or imported class.

-Wforce_static_vtbl
Specifies that the virtual function tables that are created by the compiler are to be static in the object file and are not exported. This flag applies only to variables that are not part of an exported or imported class.

-Wforce_vtbl | **-Wsuppress_vtbl**
Controls whether virtual function tables are created in cases where the compiler cannot determine the need for the tables.

The virtual function table for a class is defined in a compilation if the compilation contains a definition of the first noninline, nonpure virtual function of the class. For classes that contain no such function, the default behavior is to define the virtual function table (but to define it as a local static entity).

The flag **-Wsuppress_vtbl** suppresses the definition of the virtual function tables for such classes, and the flag **-Wforce_vtbl** forces the definition of the virtual function table for such classes. The **-Wsuppress_vtbl** flag is valid only for C++ compilations.

The **-Wforce_vtbl** flag forces definition of virtual function tables in cases where the heuristic used by the compiler to decide on definition of virtual function tables provides no guidance. The **-Wforce_vtbl** flag differs from the default behavior in that it does not force the definition to be local. The **-Wforce_vtbl** flag is valid only for C++ compilations.

-Wglobalized
Specifies that the code generated by the compiler is preemptable. By default, compilers generate code that is not preemptable. Preemptable code allows named references in a DLL to resolve to externally-defined code and data items instead of to resolve to its own internally-defined code and data items. Preemptable code is less efficient than code that is not preemptable, and is only needed in a few instances when creating a DLL.

This flag has an effect only if you also specify either the **-Wtarget=ipf** flag or the **-Wtarget=tns/e** flag.

-WH Preprocesses the specified source files and prints the names of header files, as opened, to the standard error file. No compilation or linking is performed. Unlike the **-WP** flag, no preprocessed files with **.i** (for C) or **.ii** (for C++) suffixes are produced.

The **-WH** flag overrides the **-E** and **-WP** flags.

-Wheap=*n*[**b** | **w** | **p**]
Specifies the value that the linker should use for the **HEAP_MAX** attribute of the output file. *n* can be any positive value that gives a size valid for the NonStop server node on which the file is used.

The size can be specified in units of:

b	Bytes; this is the default unit
w	Words
p	Pages

-Whelp | **-Wusage**
Displays help information on how to run **c89**. No compilation system components are run.

-Whighpin={**on** | **off** }
Directs the linker utility to set the **HIGHPIN** attribute to **on** or **off** in the output object files. This attribute specifies whether the object file will run at a high PIN or a low PIN. If **-Wsystype=guardian** is used, the default setting is **-Whighpin=off**. If **-Wsystype=oss** is used, the default setting is **-Whighpin=on**. This flag is set only if an executable object file is produced.

-Whighrequesters={**on** | **off** }
Directs the linker utility to set the **HIGHREQUESTERS** attribute to **on** or **off** in the output object file. This attribute specifies whether the object file supports requests from requesters running at a high PIN. The object file must contain the **main( )** function. If **-Wsystype=guardian** is used, the default setting is **-Whighrequesters=off**. If **-Wsystype=oss** is used, the default setting is **-Whighrequesters=on**. This flag is set only if an executable object file is produced.

-WIEEE_float | **-WTandem_float**

Specifies the floating-point format to be used by the compiler for values of type **float** or type **double**. The differences between the two formats are summarized in the **float(4)** reference page.

IEEE floating-point values can include NaN and infinity, and the sign of 0.0 (zero) can be either positive or negative. Refer to the **fp_class(3)** reference page for a description of IEEE value classes.

Guardian functions are available to convert between floating-point formats. For a discussion of floating-point conversions, see to the *Guardian Programmer's Guide*.

On systems with processors that support IEEE Std 754-1985 floating-point format data, the compiler uses that format when **-WIEEE_float** is specified. Specifying **-WTandem_float** selects HP's proprietary Tandem floating-point format.

On systems without processors that support IEEE Std 754-1985 floating-point format data, the **-WIEEE_float** flag is not available. Use of the **-WIEEE_float** flag on such systems produces an error diagnostic.

The **-WIEEE_float** flag cannot be used when the **-Wsql** or **-Wsqlcomp** flag is specified.

The default setting is **-WTandem_float** for TNS/R-targeted compilations and **-WIEEE_float** for TNS/E-targeted compilations.

-Wilp32 | **-Wlp64**

Specifies the data model to be used: 32-bit (ilp32) or 64-bit (lp64). The default data model is ilp32. The following c89 options are not allowed with **-Wlp64**: **-Wsql**, **-Wsystype=guardian**, or **-Wversion2**. For more information about data models, see the *C/C++ Programmer's Guide*.

-W[**no**]**include_whole**

Tells the **ld** or **eld** linker whether to include in the loadfile all linkable archive members of all archive libraries encountered after this flag is specified.

Specifying **-Winclude_whole** begins this linking action. When **-Wnoinclude_whole** behavior is in effect, archive searches are controlled by the existence of undefined symbols. Archives are searched in the order specified on the command line. Symbols are marked as undefined by compilers or by the user through the **-Wu** flag or the **ld** linker **-u** flag. When an archive member is found that resolves an undefined symbol, the member´s symbols are merged into the external symbol table for the loadfile being created. After the merge, the undefined symbol that triggered the merge is resolved (marked as defined). The same merge might resolve other undefined symbols or result in more undefined symbols.

You can stop the linking action of **-Winclude_whole** by specifying the **-Wnoinclude_whole** flag later in the command line or an obey file.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

The default setting is **-Wnoinclude_whole**.

-W[**no**]**inline**

Enables [disables] the generation of inline code for C++ functions declared inline and for C++ member functions declared within their class. This flag does not affect C code nor does the compiler generate inline functions for other reasons. The default setting is **-Winline**.

-Winline_compiler_generated_functions
Allows all compiler-generated functions to be inline. Specifying this flag does not guarantee that a function can be inlined. If this flag is omitted, compiler-generated functions are not inlined and are exported.

-Winline_limit=*n*
Specifies the maximum number of lines that the compiler can inline, where *n* is an integer in the range 0 through 2147483647. Specifying the value 0 (zero) means there is no limit.

The **-Winline_limit** flag is valid only for TNS/R-targeted C++ compilations.

-Winline_string_literals
Allows the compiler to generate an inline function when a function takes the address of a string literal. Specifying this flag does not guarantee that a function can be inlined. If a function is inlined by this specification, its program will not conform to section 7.1.2 of the 1998 ISO C++ standard.

-W[**no**]**innerlist**
Enables [disables] the generation of instruction code mnemonics in the listing text immediately following each corresponding statement. This flag is ignored unless **-Wnosuppress** is specified. The default setting is **-Wnoinnerlist**.

-W[**no**]**inspect**
Designates [does not designate] the symbolic debugger as the default debugger for the output object file. Use this flag with the **-g** flag. The default setting is **-Wnoinspect**. This flag is set only if an executable object file is produced.

-Wkr Directs the C compiler to process C source files according to the traditional Kernighan and Ritchie C or Common C rules, instead of according to ISO/ANSI Standard C.

-Wld="*args***"**
Passes the arguments specified in *args* to the **ld** utility after any other arguments are passed. This flag is valid only for TNS/R-targeted compilations.

Use this flag to pass arguments to **ld** when creating a PIC file. **c89** does not check the validity of **ld** arguments.

You can only use this flag when you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when the command does not initiate linking.

-Wld_obey="*file***"**
Directs the **ld** utility to read additional command-line arguments from the command file specified in the *file* argument. The arguments are processed as if they had been passed directly to **ld** in place of *file*. **c89** does not verify the existence or readability of *file*.

This flag is valid only for TNS/R-targeted compilations. This flag does not invoke **ld**. If the **ld** utility is not invoked, this flag is ignored. Use this flag to pass arguments to **ld** when creating a PIC TNS/R native program or dynamic-link library (DLL).

You can only use this flag if you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when the command does not initiate linking.

-Wlines=*l*

Specifies the maximum number of lines on a listing page, if a listing is generated. *l* must be in the range 10 through 32767.

-W[**no**]**list**

Temporarily enables [disables] the generation of listing text. Both the **-Wlist** and **-Wnolist** flags are ignored unless **-Wnosuppress** is specified. The default setting is **-Wlist**.

-WM Preprocesses the specified source files and prints a list of files that the specified source files depend on to the standard output file. The list can be used with the **make** utility. No compilation or linking is performed. Unlike the **-WP** flag, no preprocessed files with **.i** (for C) or **.ii** (for C++) suffixes are produced.

-W[**no**]**map**

Enables [disables] the generation of identifier maps in the listing. This flag is ignored unless the **-Wnosuppress** flag is specified. The default setting is **-Wnomap**.

-Wmigration_check

Directs the compiler to perform a migration check on C++ version 2 source files specified on the command line. The check uses the compiler and header files provided in Release Version Update (RVU) G06.20 and later to issue warnings where classes or member functions are used that changed or became obsolete for C++ version 3.

This flag is only valid for C++ version 2 compiles; therefore, when this flag is specified, the **-Wversion2** flag must also be specified. When this flag is used, no other compiler warning messages are output and no object file is produced.

If listings are not enabled, migration warnings are output only to the standard error file. If listings are enabled, migration warnings are sent to both the standard error file and the listing file.

-Wmigration_check=32to64

Directs the compiler to emit additional warnings that detect valid C/C++ code that potentially may behave in an unexpected manner when code designed for ilp32 is compiled using the lp64 data model. The **-Wmigration_check=32to64** flag does not require the **-Wlp64** flag.

-WmoduleCatalog="*catalog_spec***"**

Specifies a NonStop SQL/MX module catalog name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a catalog name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor release 2.0 and newer.

-WmoduleGroup[**="**[*group_spec***"**]]

Specifies a string for a NonStop SQL/MX module group specification to use as a prefix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a group name. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-WmoduleSchema="*schema_spec***"**

Specifies a NonStop SQL/MX module schema name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a schema name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor release 2.0 and newer.

-WmoduleTableSet[**="**[*tableset_spec*]**"**]
Specifies a string for a NonStop SQL/MX tableset specification to use as the first suffix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a tableset name. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-WmoduleVersion[**="**[*version_spec*]**"**]
Specifies a string for a NonStop SQL/MX tableset specification to use as the second suffix to the externally qualified module name that is written to the module file. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-Wmultibyte_char
Directs the compiler to allow multibyte character sequences in comments, string literals, and character constants.

-Wmxcmp[**="***args***"**]
Invokes the NonStop SQL/MX compiler to process any file operands of the form *file***.m** and any module definition files produced when the NonStop SQL/MX preprocessor was invoked. If the C or C++ compilation detects any errors in the source code, the NonStop SQL/MX compiler is not invoked.

If a value is supplied for *args*, it must be one of the following:

replace Directs the NonStop SQL/MX compiler to replace the existing module or create a new one. The default action does not replace an existing module.

warn Directs the NonStop SQL/MX compiler to generate a warning rather than an error if a table does not exist at compilation time.

verbose Directs the NonStop SQL/MX compiler to display summary information as well as error and warning messages.

If more than one value is specified for *args*, the values must be separated by commas without white space.

If the **-Wmxcmp** flag is specified more than once, only the last occurrence has an effect. If the **-Wmxcmp** flag is specified with any of the options that prevent compilation (**-E**, **-WH**, **-WM**, **-WP**, or **-Wsyntax**), the **-Wmxcmp** flag is ignored.

If the **-Wmxcmp** flag is specified, the **-Wsql** and **-Wsqlcomp** flags cannot be used.

-Wmxcmp_add="*args***"**
Specifies a string to pass to the NonStop SQL/MX compiler without validation or change.

-Wmxcmp_files="*file***"**[**,**...]
Passes MDF files specified to **mxcmp** in release 1 compilation mode. Passes all specified files without the **.m** extension to **mxCompileUserModule** in release 2 compilation mode.

-Wmxcmp_querydefault="*attr_name=attr_value***"**[**,**...]
Specifies attribute settings (CONTROL QUERY DEFAULT settings) to pass to the NonStop SQL/MX compiler. These attribute settings override any corresponding entries in the **SYSTEM_DEFAULTS** table.

-Wnld="*args***"**

Passes the arguments specified in *args* to the **nld** utility after any other arguments are passed. This flag is valid only for TNS/R-targeted compilations.

This flag does not invoke **nld**. If the **nld** utility is not invoked, this flag is ignored. Use this flag to pass arguments to **nld** when creating a TNS/R native non-PIC program or user library. **c89** does not check the validity of **nld** arguments.

You cannot use this flag if you use the following flags:

-Wcall_shared or **-Wshared**

-Wnld_obey="*file***"**

Directs the **nld** utility to read additional command-line arguments from the command file specified in the *file* argument. The arguments are processed as if they had been passed directly to **nld** in place of *file*. **c89** does not verify the existence or readability of *file*.

This flag is valid only for TNS/R-targeted compilations. This flag does not invoke **nld**. If the **nld** utility is not invoked, this flag is ignored. Use this flag to pass arguments to **nld** when creating a TNS/R native non-PIC program or user library.

You cannot use this flag if you use any of the following flags:

-Wcall_shared or **-Wshared**

-Wnoexceptions

Disables support for exceptions and exception handling. This flag can improve application performance by removing unneeded processing steps when an application does not use exceptions or perform exception handling.

This flag affects only C++ programs compiled with the version 2 or version 3 dialect of C++; it is ignored for C programs and C++ programs compiled using the version 1 dialect.

-Wnomain

Specifies that the object file should be linked without a **main( )** function. This flag prevents the compiler from specifying to the linker those modules and libraries that provide customary run-time support for C or C++ programs. The resulting file has no _MAIN function and no standard run-time libraries unless those are specified separately in a file identified by the:

- **-Weld_obey=***file* flag or in an obey file used by the **eld** utility
- **-Wld_obey=***file* flag or in an obey file used by the **ld** utility
- **-Wnld_obey=***file* flag or in an obey file used by the **nld** utility

-Wnostdinc

Suppresses the searching of the standard include directories to locate included files. Refer to the **Standard Include Directories** subsection for details.

-Wnostdlib

Suppresses the searching of the standard library directories to locate libraries. Refer to the **Standard Library Directories** subsection for details.

-WOlimit=*value*

Specifies the maximum decimal number of basic blocks of a routine that the global optimizer will optimize. When a routine has more basic blocks than this number, it is not optimized and a warning message is printed.

When the **-WOlimit** flag is specified, either the **-O** or **-Woptimize=2** flags must also be specified.

When the **-WOlimit** flag is not specified, an optimized routine can contain at most 2500 basic blocks. When a routine has more basic blocks than this number, it is not optimized but a warning message is not printed.

The **-WOlimit** flag is only valid for TNS/R-targeted compilations.

-Woptfile="*filename***"**

Specifies an optimizer file, which contains a list of functions that are to be optimized at the level specified in the file. The optimizer file can raise or lower the optimize level for the given functions.

Functions in the module that are not listed in the optimizer file are compiled at the level given in the **-Woptimize** flag, or, if no **-Woptimize** flag is specified, at the default optimize level.

Each line of the optimizer file can contain only one function name and the optimize level (0, 1, or 2) that you want for that function. The function name must be the internal name used for linking; for C++ programs, the mangled name must be used.

-Woptimize=*opt*

Specifies the optimization level. *opt* must be **0**, **1**, or **2**. The default value is **1**. **-O** is equivalent to **-Woptimize=2**.

-W[no]optional_lib

Indicates whether a library specified in the command stream should be considered optional when the **ld** or **eld** linker creates a loadfile.

When **-Wnooptional_lib** behavior is in effect, any library specified in a **-l** or **-lib** flag is included in the **.liblist** section of the loadfile being created. When **-Woptional_lib** behavior is in effect, a specified library can be omitted from the **.liblist** section of the loadfile being created if omitting it would not affect how symbolic references are resolved.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

If a library is specified more than once, and at least one specification occurs when **-Wnooptional_lib** is in effect, the library is included in the **.liblist** section of the loadfile being created.

The default behavior is **-Wnooptional_lib**.

-W[no]overflow_traps

Enables [disables] overflow traps. The default setting is **-Wnooverflow_traps**.

-WP Preprocesses the specified source files. No compilation or linking is performed. Output is placed in corresponding files with **.i** (for C) or **.ii** (for C++) suffixes in the current working directory.

If the **-Wsql** flag is specified, embedded NonStop SQL/MP statements are processed. If the **-Wsqlmx** flag is specified, embedded NonStop SQL/MX statements are processed.

The **-E**, **-WH**, and **-WM** flags override the **-WP** flag.

-Wpool_string_literals

Specifies that, within a compilation unit, multiple occurences of the same string literal should occupy the same storage space. This flag applies to C++ compilations only; it is ignored when C++ is not used.

The default assignment for multiple occurrences of a string literal gives them separate storage space.

-Wprofdir=*name*

Specifies the location in which to create the dynamic information (DYN) file when the **-Wprofgen** flag directs the compiler to generate instrumented code. If the application is to run in the Guardian environment, *name* must be a string that specifies a valid Guardian subvolume. Otherwise, *name* must be a string that specifies a valid OSS directory. If an invalid name is specified, no profiling information will saved.

If this flag is not specified, the DYN file is created in the default Guardian subvolume or the current OSS working directory for the process.

If object files that were compiled with different **profdir** locations are linked together, when the application is run, the DYN file is created in the location specified by one of **profdir** flags. However, it is not possible to predict which **profdir** location will be used.

This flag is valid only for TNS/E-targeted compilations. For more information about profile-guided optimization, see the *Code Profiling Utilities Manual*.

-Wprofgen

Directs the compiler to generate instrumented code, used for profile-guided optimization. All or part of an application can be instrumented by turning this flag on or off for individual compilations of object files. These object files can be linked into programs or DLLs.

Instrumented code references symbols defined in the public DLL named **zpgodll**. When you link any program or DLL that contains instrumented code, the **zpgodll** DLL must be specified at link time. The **zpgodll** DLL is automatically linked when you specify the **-Wcodecov** or **-Wprofgen** flags.

When you use the **-Wprofgen** flag and you use the compiler to automatically invoke the **eld** linker to build a program or DLL, the compiler passes the **-l pgo** option to **eld**.

HP recommends that you do not combine code that has been compiled with the **-Wcodecov** flag with code that has been compiled with the **-Wprofgen** flag in the same application.

This flag is valid only for TNS/E-targeted compilations. For more information about profile-guided optimization, see the *Code Profiling Utilities Manual*.

-Wprofuse[=*filename*]

Directs the compiler to generate optimized code based on information in a dynamic profiling information (DPI) file. This flag cannot be specified with either the **-Wcodecov** or the **-Wprofgen** flags.

The DPI file is always in the current Guardian subvolume or OSS directory. You can specify the name of the DPI file using the *filename* variable. If you do not specify a *filename*, the name of the DPI file defaults to:

- *pgopti.dpi* if the compilation is done in an OSS directory that is not a Guardian subvolume.

- *pgodpi* if the current OSS working directory is a Guardian subvolume.

This flag is valid only for TNS/E-targeted compilations. For more information about profile-guided optimization, see the *Code Profiling Utilities Manual*.

-Wr Passes the **-r** option to the linker, which directs the linker to create a linkable object file instead of an executable object file (the default).

-W[**no**]**reexport**
Tells the **ld** or **eld** linker whether to mark any library specified in a **-l** operand or **-L** flag after this flag for reexport in its **libList** entry in the loadfile being created. Specifying **-Wnoreexport** leaves the library unmarked; specifying **-Wreexport** marks the library. Reexport is a run-time attribute that is used by the **rld** loader to decide what DLLs it needs to load.

-Wnoreexport is the default action.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

-Wrefalign=*ref*
Specifies the global reference alignment for pointers. *ref* can be either **2** or **8**. The default value is **8**.

-WRefMemFuncsOnly
Specifies that the compiler provide debug information for referenced member functions only. If this flag is not used, the compiler provides debug information for all member functions in a class. This flag can be used to reduce the size of the debug region. You must also specify the **-g** flag for this flag to have an effect.

-W[**no**]**remarks**
Enables [disables] compiler remark messages. Remark messages are informative diagnostics that are less severe than warnings and errors. The default setting is **-Wnoremarks**.

-Wrunnamed
Directs the linker utility to set the RUNNAMED ON attribute in the output object file. This attribute specifies that the object file runs as a named process. The default attribute setting is RUNNAMED OFF. The RUNNAMED ON attribute is set only if an executable object file is produced.

-WRVU={*g-series-rvu*|*h-series-rvu*}
Sets the value of the _H_SERIES_RVU or the _G_SERIES_RVU feature test macro. These feature test macros are used in HP NonStop standard header files to determine whether declarations that depend on a specific RVU are available. No checking is performed to determine whether the specified RVU actually exists. The default value for this flag is the G-series RVU or the H-series RVU in which the compiler was last released. If you are running the G-series version of the **c89( )** command, specify the equivalent H-series RVU. If you are running the G-series version of the **c89( )** command, you can specify a G-series RVU for the **-WRVU** flag only. If you are running the J-series or H-series version of the **c89( )** command, you can specify either a G-series RVU or an H-series RVU for the **-WRVU** flag.
If you specify a G-series RVU:

- You must specify the value in the form **G06.***nn*—for example **-WRVU=G06.28**.

- For a C module compilation, this option causes the compiler to issue an error, instead of a warning, for implicitly declared functions.

If you specify an H-series RVU:

- You must specify the value in the form **H06.***nn*—for example **-WRVU=H06.05**.
- The **-Wtarget** flag is not required, but if you do specify it, you must specify TNS/E for the value. If you specify TNS/E for the value of the **-Wtarget** flag, but you specify a G-series RVU for the value of the **-WRVU** flag, the **-WRVU** flag is ignored. If you specify TNS/R for the value of the **-Wtarget** flag, but you specify an H-series RVU for the value of the **-WRVU** flag, the **-WRVU** flag is ignored.
- For a C module compilation, this option causes the compiler to issue an error, instead of a warning, for implicitly declared functions.

-W[**no**]**saveabend**
Specifies that a saveabend (process snapshot) file is [is not] created if the program terminates abnormally. The default setting is **-Wnosaveabend**. This flag is meaningful only if an executable object file is produced.

-Wsavetemps
Saves all temporary and intermediate files created by compilation system components. Use the **-Wv** flag to display the filenames.

-Wsql[="*args*"]
Enables NonStop SQL/MP support when processing C source files. It has no effect on C++ source files. The C source files are processed by the C SQL processor (**sqlcfe**). Arguments specified in *args* are passed to the processor without being checked for validity. This flag sets the **-Wextensions** flag. If this is a TNS/E-targeted compilation, specifying this flag also implicitly sets **-Wtandem_float**.

The **-Wsql** flag cannot be used when the **-WIEEE_float**, **-Wmxcmp**, or **-Wsqlmx** flag is specified.

-Wsqlcomp[="*args*"]
Invokes the NonStop SQL/MP compiler (**sqlcomp**) if not suppressed by another flag. A file that has already been linked can be processed by the NonStop SQL/MP compiler; for example:

```
c89 -Wsqlcomp -c exefile
```

Arguments specified in *args* are passed to the NonStop SQL/MP compiler without being checked for validity. NonStop SQL/MP compiler error messages are sent to the standard error file. Note that C++ does not support embedded SQL.

-Wsqlmx[="*args*"]
Invokes the NonStop SQL/MX **mxsqlc** preprocessor before compilation for any file operands of the form *file***.sql**, *file***.ec**, *file***.eC**, *file***.ecpp**, *file***.ecxx**, or *file***.ec++**. If an argument is specified, it must be one of the following:

listing Directs the preprocessor to write its diagnostic messages to a file named *file***.eL** in addition to the standard error file, where *file* is the name of the primary source file.

noansi_varchars
Directs the preprocessor to turn off generation of ANSI **varchar** data.

This option is valid only for preprocessor release 1.8 and newer.

noline Directs the preprocessor to suppress generation of **#line** directives in the preprocessed output source file that it creates.

null_terminate
Directs the preprocessor to terminate host variable strings with a NULL before fetch operations into them.

This option is valid only for preprocessor release 1.8 and newer.

preprocess_only
Directs the preprocessor to suppress all steps after preprocessing.

This option is valid only for preprocessor release 2.0 and newer.

process_includes
Directs the SQL/MX preprocessor to process one level of **include** files. This setting applies only to **include** files for SQL definitions; it does not apply to **include** file processing performed by the C/C++ compiler.

refrain_r2 Directs the SQL/MX preprocessor to use only the rules and features that apply to preprocessors prior to release 2.0. The default action is to use only the rules and features that apply to preprocessors beginning with release 2.0.

This option is valid only for preprocessor release 2.0 and newer.

IEEE_float Directs the SQL/MX preprocessor to use IEEE floating-point format instead of Tandem floating-point format. This option is valid only for preprocessor release 2.0 and newer.

For preprocessors before release 2.0, the default floating-point format is the same as that of the **c89** compiler (Tandem format on G-series systems, IEEE format on H-series systems). For preprocessors beginning with release 2.0, the default floating-point format is IEEE format.

If more than one argument is specified, the arguments must be separated by commas without any white space.

The **-Wsqlmx** flag cannot be specified when the **-Wsql** or **-Wsqlcomp** flag is specified.

-Wsqlmxadd[="*args*"]
Specifies a string to pass to the SQL/MX preprocessor without validation or change.

-Wsrl Directs the compiler to generate a TNS/R native user library file (a special shared run-time library, or SRL, file), instead of an executable file. If the **-Wsrl** flag is specified to create a native user library, a **-Wnld=-ul** flag is also required.

This flag is valid only for TNS/R-targeted compilations. You cannot specify this flag if you specify the **-Wcall_shared** or **-Wshared** flag.

-W[**no**]**stdfiles**
Generates a Guardian program that opens [does not open] the standard input, output, and error files by default. You can specify this flag only if **-Wsystype=guardian** is also specified. The default setting is **-Wstdfiles**.

-W[**no**]**suppress**
Disables [enables] the generation of listings. The listing is placed in a file in the current working directory with the same name as the source, but with a suffix of **.L**. The default setting is **-Wsuppress**.

-Wsyntax
Performs only a syntax check. No code is generated.

-Wsystype={**guardian** | **oss** }
Specifies the target execution environment. This flag selects definitions used during compilation, program startup code, default libraries, and system routines used during linking. The default setting is **-Wsystype=oss**. (To run files compiled for a Guardian TNS/R target execution environment, you must set the file code to 700 with a `FUP ALTER` *filename*`, CODE 700` command from a TACL prompt. To run files compiled for a Guardian TNS/E target execution environment, you must set the file code to 800 with a `FUP ALTER` *filename*`, CODE 800` command from a TACL prompt.)

-Wtarget=*platform*
Specifies the system architecture for which code should be generated. The possible values are:

-Wtarget=tns/r
Generate native mode code for a G-series (MIPS RISC) server. This is the default specification on G-series nodes.

-Wtarget=tns/e
Generate native mode code for an H-series (Itanium EPIC) server. This is the default specification on H-series nodes.

The **-Wtarget** flag is supported for systems running H-series RVUs only.

-Wtimestamp=*value*
Provides a creation timestamp for the NonStop SQL/MX preprocessor that is written to the two output files created by the preprocessor. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about the form for the *value* allowed for the timestamp. If this option is specified more than once, only the last occurrence has an effect. Note that **c89** does not check that *value* is valid; it relies on the NonStop SQL/MX preprocessor to validate this argument.

This flag is ignored unless the **-Wsqlmx** flag is also specified.

-Wu="*symbol_name*"
Tells the **ld** or **eld** linker to add *symbol_name* as an undefined symbol. This causes the linker to search for this symbol in any archive libraries that are specified after this flag on the command line or in an obey file.

The search constraint specified by the **-Wu** flag is overridden by use of the **-Winclude_whole** flag.

-Wv Echoes the command line to the standard error file as each component of the compilation system is run.

-Wverbose
Echoes the command line to the standard error file as each component of the compilation system is run and causes additional output and listings from the SQL compiler to be sent to the standard output file. SQL compiler error messages are sent to the standard error file.

-Wversion1 | **-Wversion2** | **-Wversion3**
Specifies which C++ dialect to compile.

-Wversion1 specifies the original version, released with D40. This is the default for systems with a Release Version Update (RVU) prior to G06.00. This value is only valid for TNS/R-targeted C++ compilations.

-Wversion2 specifies the dialect released with D45. This version supports such features as the **bool** type, namespaces, and exceptions.

-Wversion3 specifies the dialect released with G06.20. This version supports an ANSI/ISO Standard C++ Library corresponding to ISO/IEC IS 14882. This is the default for TNS/R systems beginning with RVU G06.00 and for TNS/E systems beginning with RVU H06.01.

These three dialects are not compatible.

All modules of a C++ program must be compiled and linked using the same dialect.

-Ww Suppresses the printing of compiler warning messages. This flag overrides any **-Wwarn** or **-Wnowarn** flags.

-W[**no**]**warn**[=*w* [,*w*] ...]
For each *w* value that appears, this flag enables [disables] the compiler warning message specified by *w*.

Declaring a *w* value enables [disables] the specified message. Specifying **-Wwarn** [**-Wnowarn**] by itself enables [disables] all compiler warning messages.

If **-Wwarn=***w* is specified, then **-Wnowarn** must also be specified or the **-Wwarn=***w* flag is ignored. If **-Wnowarn=***w* is specified, then **-Wwarn** need not be specified.

If white space is present after the commas, the list of warning message numbers should be enclosed in quotation marks.

-Wwarn is the default specification for this flag. **-Ww** overrides the **-W**[**no**]**warn** flag.

-Wx Strips part of the symbol table from the output object file but keeps information necessary for the object file to be used as input to a linker utility again. This flag is typically used with **-Wr**.

Multiple instances of the **-D**, **-I**, and **-U** flags and of the **-l** operands can be specified.

The position of **-l** *library* operands within a list of flags affects the order in which the libraries are searched.

The order of specifying the **-I** and **-L** flags is significant.

Quotation marks around string values in flags are optional but recommended to avoid errors caused by shell substitutions or deletions.

Refer to the *C/C++ Programmer's Guide* for details.

DESCRIPTION

c89 is a driver program for the native C and C++ language compilation system. This reference page describes using **c89** in the OSS environment.

c89 performs simple validation of the flags and operands from the arguments on its command line and, depending on those arguments, invokes components of the language compilation system. **c89** does not verify the existence of files it passes to compilation system components. It does verify that the operand suffix identifies a valid operand to pass to compilation system components. **c89** and the components it runs issue messages to the standard error file.

c89 performs the following steps:

1. If the corresponding **-W** flag is specified, invokes an SQL preprocessor to process any embedded SQL statements in C or C++ source files, creating C only, C++ only, or module definition files as appropriate.
2. Compiles any specified C and C++ source files or source files produced by Step 1 into object files.
3. If the **-Wmxcmp** flag is specified, invokes the NonStop SQL/MX compiler to process files created by Step 1 or specified as module definition files in the command.
4. Links the object files together with any libraries specified on the command line. (This occurs if no flags that prevent linking are specified and the source files are compiled without errors.)
5. If the **-Wsqlcomp** flag is specified, invokes the NonStop SQL/MP compiler to process files created by Step 1 or specified in the command.
6. Writes an executable object file or library to the file specified by a **-o** flag (if present) or to the file **a.out**.

Libraries can be:

- Archives, with a suffix of **.a**
- DLLs, with a suffix of **.so**
- TNS/R native user libraries
- TNS/R native SRLs, with a suffix of **.srl**

The default executable file in the Guardian file system is **aout** in the subvolume from which **c89** is invoked.

If only a single source file is given and no flags that suppress linking are specified, then the file is compiled into an object file and linked into an executable object file. If the executable file is created successfully, the object file is removed.

c89 places object files (loadfiles) in the current working directory with the same base name as the corresponding source file, but with a suffix of **.o**.

c89 also names several temporary or intermediate files that are created during the compilation process. Like the output object file, **c89** places these files in the current working directory. **c89** removes these temporary or intermediate files unless the **-Wsavetemps** flag is specified.

If **-Wsystype=oss** is set, the C and C++ compilers define the predefined feature-test macros **_OSS_TARGET** and **_XOPEN_SOURCE**. If **-Wsystype=guardian** is set, the C and C++ compilers define the predefined feature-test macros **_GUARDIAN_TARGET** and **_TANDEM_SOURCE**. These macros are used in the standard header files to determine the execution environment of a program. The feature-test macros can also be defined with a **-D** flag. Because these macros are defined internally, not by **c89**, the macros are not defined when **-Wdryrun** or **-Wv** are used.

Dynamic and Static Linking

The **-WBdllsonly** and **-WBdynamic** flags specify dynamic linking. The **-WBstatic** flag specifies static linking. In dynamic linking:

- The **nld** utility first searches for a shared run-time library (SRL). If an SRL cannot be found, **nld** then searches for an archive file. If neither of these files are found, an error is issued. In static linking, **nld** searches for an archive file but does not search for an SRL.
- The **ld** or **eld** utility first searches for a dynamic-link library (DLL). If a DLL cannot be found, the linker then searches for an archive file. If neither of these files are found, an error is issued. In static linking, the linker searches for an archive file but does not search for a DLL.

If the archive file cannot be found, an error is issued.

Dynamic and static linking are not exact opposites. Dynamic linking accepts either an SRL or DDL or an archive, but static linking accepts only an archive.

Unlike other **c89** flags, multiple **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags can be specified in a single **c89** invocation. Thus, it is possible to perform dynamic linking for some **-l** operands and static linking for others.

The **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags specified to **c89** affect linking arguments specified in **-Weld**, **-Weld_obey**, **-Wld**, **-Wnld**, **-Wld_obey**, or **-Wnld_obey** flags. Each specification remains in effect until another is encountered. To change how a linker performs linking for such arguments, you can specify **eld**, **ld**, or **nld** flags that control linking within the argument list. All linkers perform dynamic linking by default. Refer to the **eld(1)**, **ld(1)**, or **nld(1)** reference page for more information.

Handling of Files in the Guardian File System

Files in the Guardian file system can be accessed using OSS pathname syntax (/**G**/*volume*/*subvol*/*fileID*).

c89 requires that files in the Guardian file system be identified with a suffix as is done in the OSS file system. Because Guardian filenames do not allow the **.***suffix* format, the period is dropped and the suffix becomes the last character of the filename. However, the **.***suffix* format must be used when specifying the file to **c89**.

Thus, the Guardian file system file **$VOL.SUBVOL.FILEC**, which identifies a C source file, is specified to **c89** as **/G/VOL/SUBVOL/FILE.c**. Likewise, **c89** generates an object file **$VOL.SUBVOL.FILEO** that can be specified to **c89** again as **/G/VOL/SUBVOL/FILE.o**.

The default executable object file when the current working directory is in the Guardian file system is **aout**.

Predefined Preprocessor Symbols and Macros

c89 defines the following preprocessor symbols and feature-test macros:

__cplusplus
: Directs the preprocessor to process the source text as C++ source code. **c89** defines this symbol if the **-Wcplusplus** flag is specified or the name of an OSS source file ends in a C++ suffix (**.ii**, **.C**, **.cpp**, **.c++**, **.cxx**, **.cc**, **.eC**, **.ecpp**, **.ec++**, **.ecc**, or **.ecxx**).

_TANDEM_SOURCE
: Makes visible to the preprocessor identifiers required or permitted by extensions made by HP. **c89** defines this feature-test macro if the **-Wextensions** flag is specified.

_XOPEN_SOURCE
Makes visible to the preprocessor identifiers required or permitted by extensions made by the XPG4 specification. **c89** defines this feature-test macro by default unless the **-Wsystype=guardian** flag is specified.

There are other feature-test macros defined by the compiler itself, not by **c89**. These feature-test macros do not appear in the output of **-Wv** and **-Wdryrun** flags. Refer to the *C/C++ Programmer's Guide* for further information on feature-test macros.

Operands

An *operand* is in the form of:

- A pathname
- **-l** *library*
- **-WBdllsonly**
- **-WBdynamic**
- **-WBstatic**

At least one operand of the pathname form must be specified. The following operands are supported:

file A file that has been linked but has not been processed by the NonStop SQL/MP compiler

*file***.a** An archive library of object files typically produced by the **ar** command and passed directly to a linker utility

*file***.c** A C language source file to be preprocessed, compiled, and optionally linked; embedded NonStop SQL/MP information might be present

*file***.C** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.cc** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.cpp** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.cxx** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.c++** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.ec** A C language source file containing embedded NonStop SQL/MP information or NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.eC** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.ecc** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.ecpp** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.ecxx** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.i** A preprocessed C source file to be compiled and optionally linked

*file***.ii** A preprocessed C++ source file to be compiled and optionally linked

*file***.m** A module definition file (MDF) containing NonStop SQL/MX information for a corresponding C source file

*file***.o** An object file passed directly to a linker utility

*file***.so** A dynamic-link library (DLL) containing position-independent code (PIC) for use by the **eld**, **ld**, or **rld** utility

*file***.sql** A C language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.srl** A shared run-time library passed directly to the **nld** utility

-l *library* A file to be searched by a linker utility to resolve current unresolved external references; **ld** or **eld** searches for files named **lib***library***.so** and **lib***library***.a**, **nld** searches for files named **lib***library***.srl** and **lib***library***.a**

A library is searched when its name is encountered, so the placement of **-l** is significant. See the **Standard Libraries** subsection for more details.

-WBdllsonly
Specifies that the linker utility uses dynamic linking when searching for dynamic-link libraries specified in subsequent **-l** operands; placement of this operand is significant (refer to the **Dynamic and Static Linking** subsection for details)

-WBdynamic
Specifies that the linker utility uses dynamic linking when searching for libraries specified in subsequent **-l** operands; placement of this operand is significant (refer to the **Dynamic and Static Linking** subsection for details)

-WBstatic
Specifies that the linker utility uses static linking when searching for libraries specified in subsequent **-l** operands. Placement of this operand is significant. Refer to the **Dynamic and Static Linking** subsection for details.

Input Files

An input file is one of the following:

- A text file containing a C language or C++ language source program
- An object file in the format produced by the command **c89 -c**
- A library of object files in the format produced by archiving zero or more object files using the **ar** command
- A linkfile or loadfile produced by the **eld** or **ld** utility
- An executable file produced by the **nld** utility

- A module definition file created by the NonStop SQL/MX preprocessor.

When the **-Wsql** flag is specified, C source files that contain embedded NonStop SQL/MP information can have names suffixed with **.c** or **.ec**. When the **-Wsqlmx** flag is specified, **c89** uses the source file filename extension to determine the language mode (C or C++) and the names of the source files created by the NonStop SQL/MX preprocessor.

The name of a created source file is the name of the primary source file plus one of the following extension transformations:

- If the primary source file suffix is **.ec** or **.sql** and the **-Wcpluplus** flag is not specified, the created source file has the suffix **.c** and uses the C language mode.
- If the primary source file suffix is **.ec** or **.sql** and the **-Wcplusplus** flag is also specified, the created source file has the suffix **.cpp** and uses the C++ language mode.
- If the primary source file suffix is **.eC**, **.ecpp**, **.ec++**, **.ecc**, or **.ecxx**, the created source file has the suffix **.C**, **.cpp**, **.c++**, **.cc**, or **.cxx**, resapectively, and uses the C++ language mode.

When **c89** is passed a file name suffixed with **.C**, **.cpp**, **.c++**, **.cc**, or **.cxx**, that file is not passed to the NonStop SQL/MX preprocessor. Such files are assumed to contain only C or C++ statements without embedded NonStop SQL/MX information.

Output Files

An output file can be a preprocessed source file, an object file, or an executable file.

Standard Output File

The standard output file is empty unless a **-E**, **-WM**, or **-WP** flag is specified. If one of these flags is specified, preprocessed source code is sent to the standard output file. When **-WH** is used, the standard output file contains a line indicating which file is currently being operated upon.

Standard Error File

The standard error file contains diagnostic and informational messages from **c89** and the compilation components it calls. If more than one source file operand is specified, then for each such file the format `"%s: \n",`*file* is used to print the name of the source file before it is processed.

Standard Libraries

The **c89** utility recognizes the following **-l** operands for standard libraries.

-l c Contains all library functions provided by HP that are specified in the XPG4 Version 2 (X/Open UNIX) specification, including those functions listed as residing in the **math.h** header file.

-l C Contains the correct C++ run-time libraries, based on the value of the **-Wversion***n* flag. If the **-Wversion***n* flag is omitted, the default version C++ library is used.

When you specify the **-Wcplusplus** and **-Wversion***n* flags, you need not specify **-l C**; all needed libraries are automatically linked. For TNS/R programs, this includes the **cppinit**[*n*]**.o** and main function (_MAIN). When you specify the **-Wversion2** flag, **cppinit.o** is linked for a nonPIC program or **cppinit2.o** is linked for a PIC program. When you specify the **-Wversion3** flag, **cppinit3.o** is linked for a nonPIC program or **cppinit4.o** is linked for a PIC program.

-l l Contains all functions required by the C language output of the **lex** utility that are not made available through the **-l c** operand.

-l m Contains all functions referenced in the **math.h** header file.

-l y Contains all functions required by the C language output of the **yacc** utility that are not made available through the **-l c** operand.

In the absence of flags that inhibit invocation of a linker utility, such as **-c** and **-E**, **c89** directs the linker to search the standard C library after all other object files and libraries are searched.

If a C++ source file operand or a **-Wcpluscplus** flag is specified, **c89** directs the linker to search the C++ run-time library before it searches the standard C library. If you want the libraries to be searched in a specific order or linking options to be processed in a specified order, you should start the appropriate linker (**eld**, **ld**, or **nld**) directly from the OSS shell and not use the **c89** command to do the linking.

Libraries residing in the Guardian file system cannot be specified as **-l** operands because of the naming convention. They can be specified in the desired order with the **-Weld**, **-Wld**, or **-Wnld** flag.

Standard Include Directories

The standard include directory contains the standard C and C++ header files. **c89** passes a **-I** flag naming this directory as the last **-I** flag when processing source files. In the OSS environment, the directory is **/usr/include**.

Standard Library Directories

The standard library directories contain the TNS/R native shared run-time libraries (SRLs) used by the **nld** utility or the dynamic-link libraries (DLLs) used by the **eld** or **ld** utility to resolve external references.

In the OSS environment, a linker first searches the directory that contains the current version of the operating system image (the active **/G/system/sys***nn* directory). The linker then searches the **/lib**, **/usr/lib**, and **/usr/local/lib** directories.

The value of the **COMP_ROOT** environment variable is added to the beginning of **/lib**, **/usr/lib**, and **/user/local/lib**. By default, the value of **COMP_ROOT** is null in the OSS environment.

See the **eld(1)** or **ld(1)** reference page for more information about controlling the search order for a PIC file linker.

Default Flags

If no flags are specified, **c89** behaves as if the following flags were specified:

```
-Wsystype=oss -o a.out -Wenv=common -Wfieldalign=auto
-Wrefalign=8 -Woptimize=1 -Wansistreams -Winline
-WTandem_float (TNS/R-targeted compilations) or
-WIEEE_float (TNS/E-targeted compilations)
-Wlist -Wnoinnerlist -Wsuppress -Wstdfiles -Wnomap
-Wnoextensions -Wnooverflow_traps -Wnoremarks
-Wnoinspect -Wnosaveabend -WBdynamic
-Wnon_shared (TNS/R-targeted compilations) or
-Wcall_shared (TNS/E-targeted compilations)
-Wnoincludewhole -Wnoreexport -Wnooptionl_lib
-Wversion3
-Wwarn
-Wextern_data=no_gp (TNS/R-targeted compilations)
-D_XOPEN_SOURCE -I/usr/include -L/lib -L/usr/lib
-L/usr/local/lib -lc
```

Environment Variables

The following environment variables affect the execution of **c89**.

AS1 Determines the pathname of the **as1** assembler component of the C and C++ compilers. **/usr/lib/as1** is the default location for the OSS environment.

This environment variable is used for TNS/R-targeted compilations only.

CCOMBE

Determines the pathname of the **ccombe** component of the C and C++ compilers. **/usr/cmplr/ccombe** is the default location for the OSS environment.

This environment variable is used for TNS/E-targeted compilations only.

CFE Determines the pathname of the **cfe** language preprocessor component of the C and C++ compilers. **/usr/lib/cfe** is the default location for the OSS environment.

This environment variable is used for TNS/R-targeted compilations only.

COMP_ROOT

Changes the default pathnames for:

- The **c89** compilation system components
- The standard include directory
- The standard library directories

In the OSS environment, the string specified in **COMP_ROOT** is added to the beginning of the default pathnames.

If a component's environment variable is set explicitly, the **COMP_ROOT** environment variable does not modify that component's environment variable.

ELD Determines the pathname of the **eld** utility invoked by **c89**. **/usr/bin/eld** is the default location for the OSS environment.

This environment variable is only used for TNS/E-targeted compilations.

LD Determines the pathname of the **ld** utility invoked by **c89**. **/usr/bin/ld** is the default location for the OSS environment.

This environment variable is only used for TNS/R-targeted compilations.

MXCMP

Determines the pathname of the NonStop SQL/MX release 1 compiler. **/G/system/system/mxcmp** is the default.

MXCMPUM

Determines the pathname of the NonStop SQL/MX release 2 compiler. **/usr/tandem/sqlmx/bin/mxCompileUserModule** is the default.

MXSQLC

Determines the pathname of the C/C++ NonStop SQL/MX preprocessor, **mxsqlc**. **/usr/tandem/sqlmx/bin/mxsqlc** is the default.

NLD Determines the pathname of the **nld** utility invoked by **c89**. **/usr/bin/nld** is the default location for the OSS environment.

This environment variable is only used for TNS/R-targeted compilations.

SQLCFE
Determines the pathname of the native C NonStop SQL/MP processor, **sqlcfe**. **/usr/lib/sqlcfe** is the default location for the OSS environment.

This environment variable is used for TNS/R-targeted compilations only.

SQLCOMP
Determines the pathname of the native NonStop SQL/MP compiler, **sqlcomp**. **$SYSTEM.SYSTEM.SQLCOMP** is the default location for the OSS environment. The value of **SQLCOMP** must be a Guardian filename.

SQLMX_PREPROCESSOR_VERSION
Indicates the preprocessor rules and features to be used. Specifying the value 800 causes rules and features associated with release 1.8 to be used; the **mxcmp** compiler is used and only MDF files and annotated source files are produced, while rules and features associated with release 2.0 and later are ignored. Specifying a value of 1200 or larger or not specifying a value causes rules and features associated with release 2.0 and later to be used; the **mxCompileUserModule** compiler is used and annotated source files that contain embedded module definitions are produced instead of MDF files, while restrictions associated with release 1.8 or earlier are ignored.

TMPDIR
Determines the pathname that overrides the default directory for temporary files created by **c89** and the components it invokes. By default, temporary files are stored in the **/tmp** directory. If **TMPDIR** is set to a directory that does not exist or is not writeable, **c89** uses the default directory as described on the **tempnam(3)** reference page.

UGEN Determines the pathname of the **ugen** assembler component of the C and C++ compilers. **/usr/lib/ugen** is the default location for the OSS environment.

This environment variable is used for TNS/R-targeted compilations only.

UOPT Determines the pathname of the **uopt** optimizer component of the C and C++ compilers. **/usr/lib/uopt** is the default location for the OSS environment.

This environment variable is used for TNS/R-targeted compilations only.

Use the **COMP_ROOT** environment variable instead of specifying each compilation system component's environment variable, if possible.

EXAMPLES

1. The command

   ```
   c89 test1.c
   ```

   compiles the source file **test1.c** and links the object file into an executable file **a.out** in the current working directory.

2. The command

   ```
   c89 -Wnowarn -Wwarn=262 test1.c
   ```

   compiles the source file **test1.c** and links the object file into an executable file **a.out** in the current working directory. All compiler warning messages except message number 262 are disabled.

3. The command

   ```
   c89 -c /home/me/app/test1.c
   ```

   compiles the source file **/home/me/app/test1.c** into the object file **test1.o** in the current working directory.

4. The command

   ```
   c89 -g -o test2 x.c y.c z.c
       -Wnostdinc
       -I/dev/product/app/src
       -I/new/usr/include
       -lclient -lserver
       -L/dev/product/lib
       -L/new/usr/lib
   ```

   compiles the source files **x.c**, **y.c**, and **z.c** and links their respective object files **x.o**, **y.o**, and **z.o** into the executable file **test2**. Symbolic information is generated by the compiler and retained by the linker utility for debugging.

   Included files are searched for in the directories **/dev/product/app/src** and **/new/usr/include**; **/usr/include** is not searched. **nld** searches for the libraries **libclient.srl** and **libserver.srl** in the directories **/dev/product/lib** and **/new/usr/lib** before searching in the directories **/G/system/sys***nn*, **/lib**, **/usr/lib**, and **/usr/local/lib**.

5. The command

   ```
   c89 -g -o test2 x.c y.c z.c
       -Wcall_shared
       -Wnostdinc
       -I/dev/product/app/src
       -I/new/usr/include
       -lclient -lserver
       -L/dev/product/lib
       -L/new/usr/lib
   ```

   compiles the source files **x.c**, **y.c**, and **z.c** and links their respective object files **x.o**, **y.o**, and **z.o** into the loadfile **test2**. Symbolic information is generated by the compiler and retained by the linker utility for debugging.

   Included files are searched for in the directories **/dev/product/app/src** and **/new/usr/include**; **/usr/include** is not searched. **ld** searches for the libraries **libclient.so** and **libserver.so** in the directories **/dev/product/lib** and **/new/usr/lib** before searching in the directories **/G/system/sys***nn*, **/lib**, **/usr/lib**, and **/usr/local/lib**.

6. The command

   ```
   c89 -o test3 -O -DTYPE=3
       -I/usr/friend
       -I/usr/myself/headers
       foo.c bar.o baz.c
   ```

   compiles the source files **foo.c** and **baz.c** and links their respective object files with **bar.o** into the object file **test3.o**. The preprocessor symbol **TYPE** is defined to 3, and full optimization is performed by the compiler. The compiler looks for included files in the directory **/usr/friend**, then in **/usr/myself/headers**, then in **/usr/include**.

7. The command

   ```
   c89 -Wsql=release2,sqlmap -c file.c
   ```

   compiles **file.c** with NonStop SQL/MP support enabled. The listing includes an SQL map. The NonStop SQL/MP processor is run, expecting NonStop SQL/MP release 2 features. The NonStop SQL/MP compiler is not run. A **-Wsqlcomp** flag would run the NonStop SQL/MP compiler. Note that there is no white space after the comma in the **-Wsql** flag.

8. The command

   ```
   c89 -Wsqlmx -Wmxcmp -o sqlprog.exe sqlprog.ec
   ```

   causes the following steps to be performed:

   a. **c89** invokes the NonStop SQL/MX preprocessor, **mxsqlc**. **mxsqlc** takes the file **sqlprog.ec** (consisting of a single C module with embedded NonStop SQL/MX information) as input and produces two files: **sqlprog.c** and **sqlprog.m**. **sqlprog.c** is the C-only equivalent of **sqlprog.ec**; that is, NonStop SQL/MX statements are translated to the appropriate C code. **sqlprog.m** is the corresponding MDF.

   b. If there are no errors in Step a, **c89** invokes the C compiler to compile **sqlprog.c**, creating the object file **sqlprog.o**.

   c. If there are no errors in Step b, **c89** invokes the NonStop SQL/MX compiler to process the MDF file **sqlprog.m**.

   d. If there are no errors in Step c, **c89** invokes **nld** to link **sqlprog.o** with the C standard library and produce the executable file **sqlprog.exe**.

9. The command

   ```
   c89 -c -Wsqlmx file1.eC file2.ecc file3.ec++
   ```

   uses the **mxsqlc** preprocessor on several C++ source files and also compiles them, but does not link the results. If no errors are detected in either the preprocessing or compilation steps, the following files are created: **file1.m**, **file1.C**, **file2.m**, **file2.cc**, **file3.m**, **file3.c++**, **file1.o**, **file2.o**, **file3.o**.

10. The command

    ```
    c89 -Wsqlmx -E file.ec > file-cpp.c
    ```

    uses the **mxsqlc** preprocessor to expand embedded SQL statements and invokes the NonStop SQL/MX preprocessor to create a single source file containing only C statements.

11. The command

    ```
    c89 -c -Wsqlmx file1.C file2.ecc file3.ec++ file4.cpp
    ```

    illustrates mixing C++ source files, with and without NonStop SQL/MX information, on a single command line. Only source files that have names with one of the SQL extension suffixes invoke the **mxsqlc** preprocessor. However, all files are compiled but not linked. If no errors are detected in either the preprocessing or compilation steps, the following files are created: **file2.m**, **file2.cc**, **file3.m**, **file3.c++**, **file1.o**, **file2.o**, **file3.o**, **file4.o**.

12. The command

    ```
    c89 -Wmxcmp -Wmxcmp_files=test1.m,test1.o
    ```

    SQL-compiles the MDF file **test1.m** using the NonStop SQL/MX **mxcmp** compiler and processes the file **test1.o** using the NonStop SQL/MX **mxCompileUserModule** without also linking it.

13. The following command on a TNS/R system

    ```
    c89 -Wsrl -Wnld=-ul -o mylib mylib.c
    ```

    compiles the source file **mylib.c** and links the object file to create a native user library, a special shared run-time (SRL) library. The file is named **mylib** in the current working directory.

14. The command

    ```
    c89 file.c -lc -WBstatic -l archive -WBdynamic -l native
    ```

    compiles the source file **file.c** and links the object file into an executable file **a.out** in the current working directory. The linker performs dynamic linking by searching first for the file **libc.srl** and then **libc.a**. The linker then performs static linking by searching for the file **libarchive.a**. The linker then performs dynamic linking by searching first for the file **libnative.srl** and then **libnative.a**.

15. The command

    ```
    c89 file.c -lc -Wcall_shared -WBstatic
        -l archive -WBdynamic
        -l native
    ```

    compiles the source file **file.c** and links the object file into loadfile **a.out** in the current working directory. **eld** or **ld** performs dynamic linking by searching first for the file **libc.so** and then **libc.a**. The linker then performs static linking by searching for the file **libarchive.a**. The linker then performs dynamic linking by searching first for the file **libnative.so** and then **libnative.a**.

16. The command

    ```
    c89 -Wsqlcomp -c exefile
    ```

    invokes the NonStop SQL/MP compiler to process the already linked file **exefile**.

17. The command

    ```
    c89 -Wmxcmp -c module.m
    ```

    invokes the NonStop SQL/MX compiler to process the module definition file **module.m**.

FILES

/usr/bin/c89
Native **c89** in the OSS environment.

/G/system/sys*nn*/zcppcdll
C++ run-time library function object code for J-series and H-series processes; linked automatically when you compile and link C++ surce files or when you specify the **-Wcplusplus** flag.

/G/system/sys*nn*/zcpp2dll
C++ run-time library function object code for J-series and H-series processes; linked automatically when you compile and link C++ surce files or when you specify the **-Wcplusplus** flag.

/G/system/sys*nn*/zcpp3dll
C++ run-time library function object code for J-series and H-series processes; linked automatically when you compile and link C++ surce files or when you specify the **-Wcplusplus** flag.

/G/system/sys*nn*/zcresrl
Common Run-Time Environment (CRE) function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zcredll
Common Run-Time Environment (CRE) function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zcrtlsrl
C run-time library function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zcrtldll
C run-time library function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zicnvsrl
Function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zicnvdll
Function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zi18nsrl
Internationalization function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zi18ndll
Internationalization function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zosscsrl
Function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zosscdll
Function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zossesrl
Function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zossedll
Function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zossfsrl
Function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zossfdll
Function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zossksrl
Function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zosskdll
Function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zpgodll
Symbols referenced by instrumented code for J-series and H-series processes; must be linked when you link a program or DLL that contains instrumented code; linked automatically when you specify the **-Wcodecov** or **-Wprofgen** flags.

/G/system/sys*nn*/zsecsrl
Security function object code for G-series processes; linked automatically.

/G/system/sys*nn*/zsecdll
Security function object code for H-series processes; linked automatically.

/G/system/sys*nn*/zstfnsrl
Function object code for G-series processes; linked automatically.

DIAGNOSTICS

If **c89** encounters a compilation error that prevents an object file from being created, it writes a diagnostic message to the standard error file and continues to compile other source code operands. However, it does not perform program linking and returns a nonzero exit status.

If the linking is unsuccessful, **c89** writes a diagnostic message to the standard error file and returns a nonzero exit status.

EXIT VALUES

The following exit values are returned:

0 (zero)	Successful completion.
>0	An error occurred.

RELATED INFORMATION

Commands: **ar(1)**, **c99(1)**, **eld(1)**, **ld(1)**, **nld(1)**, **strip(1)**.

Functions: **fp_class(3)**, **tempnam(3)**.

Files: **float(4)**.

STANDARDS CONFORMANCE

All **-W** options are HP extensions to the POSIX and XOPEN standards.

NAME

c99 - Compiles C99-compliant C and C++ programs using the TNS/E native compilers

SYNOPSIS

c99

[**-c** | **-Wnolink**]
[[**-D** *name*[**="***value***"**]] ...] [**-E**] [**-g**]
[**-I** *directory* ...]
[**-L** *directory* ...] [**-o** *outfile*] [**-O***optlevel*] [**-s**]
[**-U** *name*]
[**-Wallow_extern_explicit_instantiation**]
[**-Wansistreams**]
[**-Wbasename**]
[**-WBdllsonly** | **-WBdynamic** | **-WBstatic**]
[**-Wbitfield_container=***value*]
[**-Wbuild_neutral_library**]
[**-WC**]
[**-Wcall_shared** | **-Wshared**]
[**-Wcodecov**]
[**-Wcolumns=***c*]
[**-Wcplusplus**]
[**-WD***name*[**="***value***"**]]
[**-Wdryrun**]
[**-Weld=***args*]
[**-Weld_obey=***file*]
[**-Wenv=***env*]
[**-Werrors=***e*]
[**-W**[**no**]**extensions**]
[**-Wfieldalign=***align*]
[**-Wforce_static_typeinfo**]
[**-Wforce_static_vtbl**]
[**-Wforce_vtbl** | **-Wsuppress_vtbl**]
[**-Wglobalized**]
[**-WH**]
[**-Wheap=***n*[**b** | **w** | **p**]
[**-Whelp** | **-Wusage**]
[**-Whighpin=**{**on** | **off**}]
[**-Whighrequesters=**{**on** | **off**}]
[**-WIEEE_float** | **-WTandem_float**]
[**-Wilp32** | **-Wlp64**]
[**-W**[**no**]**include_whole**]
[**-W**[**no**]**inline**]
[**-Winline_compiler_generated_functions**]
[**-Winline_string_literals**]
[**-W**[**no**]**innerlist**]
[**-W**[**no**]**inspect**]
[**-Wlines=***l*]
[**-W**[**no**]**list**]
[**-WM**]
[**-W**[**no**]**map**]
[**-Wmigration_check=32to64**]
[**-WmoduleCatalog="***catalog_spec***"**]
[**-WmoduleGroup**[**="**[*group_spec***"**]]]
[**-WmoduleSchema="***schema_spec***"**]

[**-WmoduleTableSet**[="[*tableset_spec*"]]]
[**-WmoduleVersion**[="[*version_spec*"]]]
[**-Wmultibyte_char**]
[**-Wmxcmp**[="*args*"]]
[**-Wmxcmp_add=**"*args*"]
[**-Wmxcmp_files=**"*file*"[**,**...]]
[**-Wmxcmp_querydefault=**"*attr_name=attr_value*"[**,**...]]
[**-Wnoexceptions**]
[**-Wnomain**]
[**-Wnostdinc**]
[**-Wnostdlib**]
[**-Woptfile=**"*filename*"]
[**-W**[**no**]**optional_lib**]
[**-W**[**no**]**overflow_traps**]
[**-WP**]
[**-Wpool_string_literals**]
[**-Wprofdir=***name*]
[**-Wprofgen**]
[**-Wprofuse**[=*filename*]]
[**-Wr**]
[**-W**[**no**]**reexport**]
[**-Wrefalign=***ref*]
[**-WRefMemFuncsOnly**]
[**-W**[**no**]**remarks**]
[**-Wrunnamed**]
[**-WRVU=**{*h-series-rvu*}]
[**-W**[**no**]**saveabend**]
[**-Wsavetemps**]
[**-Wsqlmx**[="*args*"]]
[**-Wsqlmxadd**[="*args*"]]
[**-W**[**no**]**stdfiles**]
[**-W**[**no**]**suppress**]
[**-Wsyntax**]
[**-Wsystype=**{**guardian** | **oss**}]
[**-Wtarget=***platform*]
[**-Wtimestamp=***value*]
[**-Wu=**"*symbol_name*"]
[**-Wv**]
[**-Wverbose**]
[**-Wversion3**]
[**-Ww**]
[**-W**[**no**]**warn**[=*w* [**,***w*] ...]]
[**-Wx**]
operand ...

FLAGS

-c | **-Wnolink**

Compiles the specified C or C++ source files but suppresses linking, even if another flag specifies linking.

You cannot specify the **-c** flag if you use the **-Wshared** flag.

-D *name*[="*value*"]

Defines the preprocessor symbol *name* as *value*. It is equivalent to a **#define** directive in the source. If no *value* is given, *name* is defined as 1. The **-D** flag has lower precedence than the **-U** flag. Thus, if *name* is specified in both a **-U** and a **-D** flag, *name* is undefined regardless of the order of the flags.

Use this flag to define compiler feature-test macros.

When the NonStop SQL/MX preprocessor is invoked, all **-D** specifications are automatically passed to the preprocessor as the preprocessor's **-d** options.

-E Preprocesses the specified source files. No compilation or linking is performed. Output is sent to the standard output file and contains **#line** directives.

If the **-Wsqlmx** flag is specified, embedded NonStop SQL/MX statements are processed.

The **-WH** and **-WM** flags override the **-E** flag.

-g Produces in the object or executable files information (symbol tables) used for symbolic debugging.

-I *directory*

Adds *directory* to the list of directories searched to locate **#include** files with relative pathnames. (Relative pathnames do not begin with a slash, '/'). **#include** filenames enclosed in double quotes are searched for first in the directory of the file with the **#include** directive, then in directories named with **-I** flags, and last in the standard include directories. **#include** filenames enclosed in angle brackets (<>) are searched for first in directories named with **-I** flags and then in the standard include directories. Refer to the **Standard Include Directories** subsection for details.

-L *directory*

Adds *directory* to the list of directories searched to locate libraries specified by operands of the form **-l** *library*. See the **Operands** subsection for details.

-o *outfile* Uses the pathname *outfile* instead of the default pathname **a.out** for the name of the output object file.

If only one source file is specified and the **-c** flag is specified, the generated output is placed into *outfile*. Only one **-o** flag can be specified. The file specified cannot be an SQL preprocessing output file.

If a single source file is compiled and linked in one invocation of **c99**, and if the *outfile* is the same name as that of the input object file, **c99** issues a warning message and places the output in a temporary file.

-O*optlevel*

Sets the compiler to the optimization level specified by *optlevel*, where *optlevel* is 0 (zero), 1, or 2. If this flag is not set, the application is compiled at optimization level 1. If **-O** is specified without a value for *optlevel*, the application is compiled at optimization level 2. This option is equivalent to the **c89 -Woptimize=***n* option.

-s Strips symbolic information not required for proper execution from object and executable files. The resulting object file cannot be debugged using a symbolic debugger. This flag is ignored if the **-Wr** flag is also specified.

-U *name* Removes any initial definition of the preprocessor symbol *name*. The **-U** flag has higher precedence than the **-D** flag. If *name* is specified in both a **-U** and a **-D** flag, *name* is undefined regardless of the order of the flags.

-Wallow_extern_explicit_instantiation
Allows an **extern** storage attribute to be applied to an explicit template instantiation. This flag suppresses the instantiation of the template. If this flag is omitted, the template is instantiated.

-Wansistreams
Generates a Guardian program that opens text files as type 180 instead of type 101 if **-Wsystype=guardian** is specified. (By default Guardian programs open text files as type 101.) This flag is ignored when **-Wsystype=oss** is used. OSS programs can open text files only as type 180.

-Wbasename
Directs the compiler to place only the last part of the source file name, known as the basename, into the dynamic information (DYN) file when the instrumented process runs.

If you use the **-Wbasename** option and the **-Wprofgen** option to compile a source file, you also must use the **-Wbasename** option when you use the **-Wprofuse** option to compile this source file. When you use the **-Wbasename** option, the source file is not required to be in the same location as it was when you compiled the file using the **-Wprofgen** option, but the basename must be the same.

If you did not use both the **-Wbasename** and the **-Wprofgen** options when you compiled this source file, do not specify the **-Wbasename** flag when you compile this source file with the **-Wprofuse** flag. If you do not use the **-Wbasename** flag, when you compile the source file and specify the **-Wprofuse** flag, the source file must be in the same location as it was when you used the **-Wprofgen** flag to compile the file.

For more information about profile-guided optimization and the rules for using the **-Wbasename** flag, see the *Code Profiling Utilities Manual*.

-WBdllsonly
Tells the **eld** linker to limit searches to position-independent code (PIC) files that are dynamic-link libraries (DLLs) when resolving the file names specified for the **-l** operands and **-L** flags.

If a file name is qualified, the linker searches for a DLL with that name.

If a filename is unqualified, in each search path, the linker first searches for a DLL with the file name as specified in the **-l** operand or **-L** flag. If the linker cannot find a DLL, the file name is unqualified, and the search path is not in the Guardian file system (/G), then the linker prefixes **lib** and suffixes **.so** to the file name and searches again. If the linker still cannot find the DLL, it searches the path again with the same prefix but with **.srl** as the suffix. For more information on search paths, see the **Finding Libraries** subsection of the **eld(1)** reference page under **DESCRIPTION**.

When a DLL cannot be found, the linker issues an error message unless its **-allow_missing_libs** flag is specified.

The **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags are search control toggles. Multiple flags can be specified in a single linker invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** operands and **-L** flags and search for just archive files for others. The default library search control is **-WBdynamic**.

-WBdynamic

Specifies that the linker utility uses dynamic linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Dynamic linking is in effect until a **-WBstatic** flag is specified. **-WBdynamic** is the default setting. Refer to the **Dynamic and Static Linking** subsection for details.

-WBstatic

Specifies that the linker utility uses static linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Static linking is in effect until a **-WBdynamic** flag is specified. **-WBdynamic**, not **-WBstatic**, is the default setting. Refer to the **Dynamic and Static Linking** subsection for details.

-Wbitfield_container=*value*

Directs the compiler to accept larger and more flexible bit fields where *value* is one of the following:

int Directs the compiler to pack bit fields into 32-bit **int**s. In this mode, the compiler will not accept bit fields larger than 32 bits. The compiler returns an error for any bit field declared to be of type **long long** (unless the **-Wextensions** flag is also specified). The compiler returns warnings for other non-standard integer types.

long Directs the compiler to pack bit fields whose base type is larger than 32-bits into 64-bit **int**s. All other bit fields will be packed into 32-bit **int**s. In this mode, the compiler accepts the **long long** and **long** bit-field types and up to 64 bits in length. The compiler returns warnings for other non-standard integer types.

all Directs the compiler to pack all bit fields into **int**s defined by their base type. The compiler will accept any integer type for a bit field. This mode provides additional compatibility with the various methods other compilers may use to pack bit fields.

The default *value* is **int** except when the **-Wlp64** flag is specified, in which case the default becomes **long**.

Note: The above rules apply to bit fields declared in **auto** or **platform** (the default) **struct**s. Any bit field declared in a **shared2 struct** will continue to follow **shared2** rules. Bit fields in **shared2 struct**s can never be larger than 32-bits. Bit fields declared in **shared8 struct**s can be larger than 32-bits (if the value is **long** or **long**), but the compiler will emit a warning. Bit fields in **shared8 struct**s will be packed as before except that bit fields larger than 32-bits will be packed into 64-bit containers. This flag is supported on systems running J06.13 or later J-series RVUs or H06.24 or later H-series RVUs only.

-Wbuild_neutral_library

Specifies that the compiler should issue an error message when it encounters any exported or imported interface in a DLL that depends on types marked as being incompatible with the neutral C++ dialect.

-WC Retains comments when preprocessing files. Comments are removed from preprocessor output by default.

-Wcall_shared | **-Wshared**

Directs the compiler to create a specific type of object file:

-Wcall_shared Specifies that the object file should be a PIC file; the **eld** linker is invoked. If the **-c** flag is also specified, the file is a linkfile. Otherwise, the file is an executable object file (loadfile).

This is the default behavior.

-Wshared Specifies that the file should be a PIC DLL; the **eld** linker is invoked. You cannot use this flag if you use the **-c** flag.

-Wcodecov

Directs the compiler to create an instrumented object file and to create or add to an existing SPI file. This flag has an effect only if you also specify either the **-Wtarget=ipf** flag or the **-Wtarget=tns/e** flag.

The first time the **-Wcodecov** flag is used to compile a program, the compiler creates a Static Profiling Information (SPI) file. This file is one of the input files for the Code Coverage tool. If the program is compiled in an OSS directory:

- The default name for the SPI file is **pgopti.spi**.
- If the default file is not write-accessible, the name of the SPI file created is **tpopti.spi**.
- A lock file called **pgopti.spl**. When compilation is complete, the compiler deletes this file.

If the program is compiled in a Guardian directory:

- The default name for the SPI file is **pgospi**.
- If the default file is not write-accessible, the name of the SPI file created is **tpgospi**.
- A lock file called **pgospl**. When compilation is complete, the compiler deletes this file.

If the SPI file already exists when the program is compiled with the **-Wcodecov** flag, the compiler updates or adds information to the existing SPI file. If more than one SPI file exists for the same program, you must concatenate the files manually before you can use the resulting file as input to the Code Coverage Tool.

For more information about the Code Coverage Tool, see the *Code Profiling Utilities Manual*.

-Wcolumns=*c*

Specifies the maximum number of columns in input source files to process. *c* is in the range 20 through 32767. Text in columns beyond column *c* is ignored.

-Wcplusplus

Directs **c99** to assume that files with a **.c** or **.i** suffix contain C++ source code, and defines the feature-test macro **__cplusplus**. If linking occurs, this flag directs the linker utility to search the C++ standard run-time library.

If this flag is omitted and none of the operand filenames end in **.C**, **.cpp**, **.cc**, or **.cxx**, then source files are compiled as C files only and, if **-c** is not specified, are only linked with the C standard library. See **Standard Libraries** for details.

-WD*name*[="*value*"]
Specifies a macro that is defined only during the NonStop SQL/MX preprocessing step. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about its **-d** *flag*=[*value*] option. This flag can be specified more than once.

Note that all **-D** values that are supplied to **c99** are automatically passed as **-d** options to the NonStop SQL/MX preprocessor.

This flag is ignored unless the **-Wsqlmx** flag is also specified.

-Wdryrun
Verifies the syntax and semantics of flags and operands specified to **c99** and enables the **-Wv** flag. No compilation system components are run.

-Weld="*args*"
Passes the arguments specified in *args* to the **eld** utility after any other arguments are passed.

Use this flag to pass arguments to **eld** when creating a PIC file. **c99** does not check the validity of **eld** arguments.

You can only use this flag when you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when the command does not initiate linking.

-Weld_obey="*file*"
Directs the **eld** utility to read additional command-line arguments from the command file specified in the *file* argument. The arguments are processed as if they had been passed directly to **eld** in place of *file*. **c99** does not verify the existence or readability of *file*.

You can only use this flag if you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when the command does not initiate linking.

-Wenv=*env*
Specifies the run-time environment. *env* can be one of **common**, **embedded**, **library**, or **libspace**. The default value is **common**.

-Werrors=*e*
Stops compiling when *e* errors have been encountered.

-W[**no**]**extensions**
Enables [disables] HP extensions. If **-Wextensions** is specified, **c99** also defines the **_TANDEM_SOURCE** feature-test macro. The default value is **-Wnoextensions**.

-Wfieldalign=*align*
Specifies the field alignment for structures. *align* can be one of **auto**, **cshared2**, **shared2**, **shared8**, or **platform**. The default value is **auto**. You cannot specify a **struct** tag with this flag.

-Wforce_static_typeinfo
Specifies that the **typeinfo** variables are to be static in the object file. This flag applies only to variables that are not part of an exported or imported class.

-Wforce_static_vtbl
Specifies that the virtual function tables that are created by the compiler are to be static in the object file and are not exported. This flag applies only to variables that are not part of an exported or imported class.

The **-Wforce_static_vtbl** flag is valid only for TNS/R-targeted C++ compilations.

-Wforce_vtbl | **-Wsuppress_vtbl**
Controls whether virtual function tables are created in cases where the compiler cannot determine the need for the tables.

The virtual function table for a class is defined in a compilation if the compilation contains a definition of the first noninline, nonpure virtual function of the class. For classes that contain no such function, the default behavior is to define the virtual function table (but to define it as a local static entity).

The flag **-Wsuppress_vtbl** suppresses the definition of the virtual function tables for such classes, and the flag **-Wforce_vtbl** forces the definition of the virtual function table for such classes. The **-Wsuppress_vtbl** flag is valid only for C++ compilations.

The **-Wforce_vtbl** flag forces definition of virtual function tables in cases where the heuristic used by the compiler to decide on definition of virtual function tables provides no guidance. The **-Wforce_vtbl** flag differs from the default behavior in that it does not force the definition to be local. The **-Wforce_vtbl** flag is valid only for C++ compilations.

-Wglobalized
Specifies that the code generated by the compiler is preemptable. By default, compilers generate code that is not preemptable. Preemptable code allows named references in a DLL to resolve to externally-defined code and data items instead of to resolve to its own internally-defined code and data items. Preemptable code is less efficient than code that is not preemptable, and is only needed in a few instances when creating a DLL.

-WH Preprocesses the specified source files and prints the names of header files, as opened, to the standard error file. No compilation or linking is performed. Unlike the **-WP** flag, no preprocessed files with **.i** (for C) or **.ii** (for C++) suffixes are produced.

The **-WH** flag overrides the **-E** and **-WP** flags.

-Wheap=*n*[**b** | **w** | **p**]
Specifies the value that the linker should use for the **HEAP_MAX** attribute of the output file. *n* can be any positive value that gives a size valid for the NonStop server node on which the file is used.

The size can be specified in units of:

b	Bytes; this is the default unit
w	Words
p	Pages

-Whelp | **-Wusage**
Displays help information on how to run **c99**. No compilation system components are run.

-Whighpin={on | off }

Directs the linker utility to set the **HIGHPIN** attribute to **on** or **off** in the output object files. This attribute specifies whether the object file will run at a high PIN or a low PIN. If **-Wsystype=guardian** is used, the default setting is **-Whighpin=off**. If **-Wsystype=oss** is used, the default setting is **-Whighpin=on**. This flag is set only if an executable object file is produced.

-Whighrequesters={on | off }

Directs the linker utility to set the **HIGHREQUESTERS** attribute to **on** or **off** in the output object file. This attribute specifies whether the object file supports requests from requesters running at a high PIN. The object file must contain the **main( )** function. If **-Wsystype=guardian** is used, the default setting is **-Whighrequesters=off**. If **-Wsystype=oss** is used, the default setting is **-Whighrequesters=on**. This flag is set only if an executable object file is produced.

-WIEEE_float | -WTandem_float

Specifies the floating-point format to be used by the compiler for values of type **float** or type **double**. The differences between the two formats are summarized in the **float(4)** reference page.

IEEE floating-point values can include NaN and infinity, and the sign of 0.0 (zero) can be either positive or negative. Refer to the **fp_class(3)** reference page for a description of IEEE value classes.

Guardian functions are available to convert between floating-point formats. For a discussion of floating-point conversions, see to the *Guardian Programmer's Guide*.

On systems with processors that support IEEE Std 754-1985 floating-point format data, the compiler uses that format when **-WIEEE_float** is specified. Specifying **-WTandem_float** selects HP's proprietary Tandem floating-point format.

On systems without processors that support IEEE Std 754-1985 floating-point format data, the **-WIEEE_float** flag is not available. Use of the **-WIEEE_float** flag on such systems produces an error diagnostic.

The default setting is **-WIEEE_float**.

-Wilp32 | -Wlp64

Specifies the data model to be used: 32-bit (ilp32) or 64-bit (lp64). The default data model is ilp32. The following c89 option is not allowed with **-Wlp64**: **-Wsystype=guardian**. For more information about data models, see the *C/C++ Programmer's Guide*.

-W[no]include_whole

Tells the linker whether to include in the loadfile all linkable archive members of all archive libraries encountered after this flag is specified.

Specifying **-Winclude_whole** begins this linking action. When **-Wnoinclude_whole** behavior is in effect, archive searches are controlled by the existence of undefined symbols. Archives are searched in the order specified on the command line. Symbols are marked as undefined by compilers or by the user through the **-Wu** flag. When an archive member is found that resolves an undefined symbol, the member´s symbols are merged into the external symbol table for the loadfile being created. After the merge, the undefined symbol that triggered the merge is resolved (marked as defined). The same merge might resolve other undefined symbols or result in more undefined symbols.

You can stop the linking action of **-Winclude_whole** by specifying the **-Wnoinclude_whole** flag later in the command line or an obey file.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

The default setting is **-Wnoinclude_whole**.

-W[**no**]**inline**
Enables [disables] the generation of inline code for C++ functions declared inline and for C++ member functions declared within their class. This flag does not affect C code nor does the compiler generate inline functions for other reasons. The default setting is **-Winline**.

-Winline_compiler_generated_functions
Allows all compiler-generated functions to be inline. Specifying this flag does not guarantee that a function can be inlined. If this flag is omitted, compiler-generated functions are not inlined and are exported.

-Winline_string_literals
Allows the compiler to generate an inline function when a function takes the address of a string literal. Specifying this flag does not guarantee that a function can be inlined. If a function is inlined by this specification, its program will not conform to section 7.1.2 of the 1998 ISO C++ standard.

-W[**no**]**innerlist**
Enables [disables] the generation of instruction code mnemonics in the listing text immediately following each corresponding statement. This flag is ignored unless **-Wnosuppress** is specified. The default setting is **-Wnoinnerlist**.

-W[**no**]**inspect**
Designates [does not designate] the symbolic debugger as the default debugger for the output object file. Use this flag with the **-g** flag. The default setting is **-Wnoinspect**. This flag is set only if an executable object file is produced.

-Wlines=*l*
Specifies the maximum number of lines on a listing page, if a listing is generated. *l* must be in the range 10 through 32767.

-W[**no**]**list**
Temporarily enables [disables] the generation of listing text. Both the **-Wlist** and **-Wnolist** flags are ignored unless **-Wnosuppress** is specified. The default setting is **-Wlist**.

-WM Preprocesses the specified source files and prints a list of files that the specified source files depend on to the standard output file. The list can be used with the **make** utility. No compilation or linking is performed. Unlike the **-WP** flag, no preprocessed files with **.i** (for C) or **.ii** (for C++) suffixes are produced.

-W[**no**]**map**
Enables [disables] the generation of identifier maps in the listing. This flag is ignored unless the **-Wnosuppress** flag is specified. The default setting is **-Wnomap**.

-Wmigration_check=32to64
Directs the compiler to emit additional warnings that detect valid C/C++ code that potentially may behave in an unexpected manner when code designed for ilp32 is compiled using the lp64 data model. The **-Wmigration_check=32to64** flag does not require the **-Wlp64** flag.

-WmoduleCatalog="*catalog_spec***"**
Specifies a NonStop SQL/MX module catalog name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a catalog name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor release 2.0 and newer.

-WmoduleGroup[**="**[*group_spec***"**]]
Specifies a string for a NonStop SQL/MX module group specification to use as a prefix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a group name. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-WmoduleSchema="*schema_spec***"**
Specifies a NonStop SQL/MX module schema name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a schema name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor release 2.0 and newer.

-WmoduleTableSet[**="**[*tableset_spec***"**]]
Specifies a string for a NonStop SQL/MX tableset specification to use as the first suffix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a tableset name. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-WmoduleVersion[**="**[*version_spec***"**]]
Specifies a string for a NonStop SQL/MX tableset specification to use as the second suffix to the externally qualified module name that is written to the module file. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-Wmultibyte_char
Directs the compiler to allow multibyte character sequences in comments, string literals, and character constants.

-Wmxcmp[**="***args***"**]
Invokes the NonStop SQL/MX compiler to process any file operands of the form *file***.m** and any module definition files produced when the NonStop SQL/MX preprocessor was invoked. If the C or C++ compilation detects any errors in the source code, the NonStop SQL/MX compiler is not invoked.

If a value is supplied for *args*, it must be one of the following:

replace Directs the NonStop SQL/MX compiler to replace the existing module or create a new one. The default action does not replace an existing module.

warn Directs the NonStop SQL/MX compiler to generate a warning rather than an error if a table does not exist at compilation time.

verbose Directs the NonStop SQL/MX compiler to display summary information as well as error and warning messages.

If more than one value is specified for *args*, the values must be separated by commas without white space.

If the **-Wmxcmp** flag is specified more than once, only the last occurrence has an effect. If the **-Wmxcmp** flag is specified with any of the options that prevent compilation (**-E**, **-WH**, **-WM**, **-WP**, or **-Wsyntax**), the **-Wmxcmp** flag is ignored.

-Wmxcmp_add="*args***"**
Specifies a string to pass to the NonStop SQL/MX compiler without validation or change.

-Wmxcmp_files="*file***"[,...]**
Passes MDF files specified to **mxcmp** in release 1 compilation mode. Passes all specified files without the **.m** extension to **mxCompileUserModule** in release 2 compilation mode.

-Wmxcmp_querydefault="*attr_name=attr_value***"[,...]**
Specifies attribute settings (CONTROL QUERY DEFAULT settings) to pass to the NonStop SQL/MX compiler. These attribute settings override any corresponding entries in the **SYSTEM_DEFAULTS** table.

-Wnoexceptions
Disables support for exceptions and exception handling. This flag can improve application performance by removing unneeded processing steps when an application does not use exceptions or perform exception handling.

-Wnomain
Specifies that the object file should be linked without a **main( )** function. This flag prevents the compiler from specifying to the linker those modules and libraries that provide customary run-time support for C or C++ programs. The resulting file has no _MAIN function and no standard run-time libraries unless those are specified separately in a file identified by the **-Weld_obey=***file* flag.

-Wnostdinc
Suppresses the searching of the standard include directories to locate included files. Refer to the **Standard Include Directories** subsection for details.

-Wnostdlib
Suppresses the searching of the standard library directories to locate libraries. Refer to the **Standard Library Directories** subsection for details.

-Woptfile="*filename***"**
Specifies an optimizer file, which contains a list of functions that are to be optimized at the level specified in the file. The optimizer file can raise or lower the optimize level for the given functions.

Functions in the module that are not listed in the optimizer file are compiled at the default optimize level.

Each line of the optimizer file can contain only one function name and the optimize level (0, 1, or 2) that you want for that function. The function name must be the internal name used for linking; for C++ programs, the mangled name must be used.

-W[**no**]**optional_lib**

Indicates whether a library specified in the command stream should be considered optional when the linker creates a loadfile.

When **-Wnooptional_lib** behavior is in effect, any library specified in a **-l** or **-lib** flag is included in the **.liblist** section of the loadfile being created. When **-Woptional_lib** behavior is in effect, a specified library can be omitted from the **.liblist** section of the loadfile being created if omitting it would not affect how symbolic references are resolved.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

If a library is specified more than once, and at least one specification occurs when **-Wnooptional_lib** is in effect, the library is included in the **.liblist** section of the loadfile being created.

The default behavior is **-Wnooptional_lib**.

-W[**no**]**overflow_traps**

Enables [disables] overflow traps. The default setting is **-Wnooverflow_traps**.

-WP Preprocesses the specified source files. No compilation or linking is performed. Output is placed in corresponding files with **.i** (for C) or **.ii** (for C++) suffixes in the current working directory.

If the **-Wsqlmx** flag is specified, embedded NonStop SQL/MX statements are processed.

The **-E**, **-WH**, and **-WM** flags override the **-WP** flag.

-Wpool_string_literals

Specifies that, within a compilation unit, multiple occurences of the same string literal should occupy the same storage space. This flag applies to C++ compilations only; it is ignored when C++ is not used.

The default assignment for multiple occurrences of a string literal gives them separate storage space.

-Wprofdir=*name*

Specifies the location in which to create the dynamic information (DYN) file when the **-Wprofgen** flag directs the compiler to generate instrumented code. If the application is to run in the Guardian environment, *name* must be a string that specifies a valid Guardian subvolume. Otherwise, *name* must be a string that specifies a valid OSS directory. If an invalid name is specified, no profiling information will saved.

If this flag is not specified, the DYN file is created in the default Guardian subvolume or the current OSS working directory for the process.

If object files that were compiled with different **profdir** locations are linked together, when the application is run, the DYN file is created in the location specified by one of **profdir** flags. However, it is not possible to predict which **profdir** location will be used.

For more information about profile-guided optimization, see the *Code Profiling Utilities Manual*.

-Wprofgen

Directs the compiler to generate instrumented code, used for profile-guided optimization. All or part of an application can be instrumented by turning this flag on or off for individual compilations of object files. These object files can be linked into programs or DLLs.

Instrumented code references symbols defined in the public DLL named **zpgodll**. When you link any program or DLL that contains instrumented code, the **zpgodll** DLL must be specified at link time. The **zpgodll** DLL is automatically linked when you specify the **-Wcodecov** or **-Wprofgen** flags.

When you use the **-Wprofgen** flag and you use the compiler to automatically invoke the **eld** linker to build a program or DLL, the compiler passes the **-l pgo** option to **eld**.

HP recommends that you do not combine code that has been compiled with the **-Wcodecov** flag with code that has been compiled with the **-Wprofgen** flag in the same application.

For more information about profile-guided optimization, see the *Code Profiling Utilities Manual*.

-Wprofuse[=*filename*]

Directs the compiler to generate optimized code based on information in a dynamic profiling information (DPI) file. This flag cannot be specified with either the **-Wcodecov** or the **-Wprofgen** flags.

The DPI file is always in the current Guardian subvolume or OSS directory. You can specify the name of the DPI file using the *filename* variable. If you do not specify a *filename*, the name of the DPI file defaults to:

- *pgopti.dpi* if the compilation is done in an OSS directory that is not a Guardian subvolume.
- *pgodpi* if the current OSS working directory is a Guardian subvolume.

For more information about profile-guided optimization, see the *Code Profiling Utilities Manual*.

-Wr Passes the **-r** option to the linker, which directs the linker to create a linkable object file instead of an executable object file (the default).

-W[**no**]**reexport**

Tells the linker whether to mark any library specified in a **-l** operand or **-L** flag after this flag for reexport in its **libList** entry in the loadfile being created. Specifying **-Wnoreexport** leaves the library unmarked; specifying **-Wreexport** marks the library. Reexport is a run-time attribute that is used by the **rld** loader to decide what DLLs it needs to load.

-Wnoreexport is the default action.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

-Wrefalign=*ref*

Specifies the global reference alignment for pointers. *ref* can be either **2** or **8**. The default value is **8**.

-WRefMemFuncsOnly
Specifies that the compiler provide debug information for referenced member functions only. If this flag is not used, the compiler provides debug information for all member functions in a class. This flag can be used to reduce the size of the debug region. You must also specify the **-g** flag for this flag to have an effect.

-W[**no**]**remarks**
Enables [disables] compiler remark messages. Remark messages are informative diagnostics that are less severe than warnings and errors. The default setting is **-Wnoremarks**.

-Wrunnamed
Directs the linker utility to set the RUNNAMED ON attribute in the output object file. This attribute specifies that the object file runs as a named process. The default attribute setting is RUNNAMED OFF. The RUNNAMED ON attribute is set only if an executable object file is produced.

-WRVU={*fh-series-rvu*}
Sets the value of the _H_SERIES_RVU feature test macro. These feature test macros are used in HP NonStop standard header files to determine whether declarations that depend on a specific RVU are available. No checking is performed to determine whether the specified RVU actually exists. The default value for this flag is the H-series RVU in which the compiler was last released. To specify a specific J-series RVU, specify the equivalent H-series RVU.

If you specify an H-series RVU:

- You must specify the value in the form **H06.***nn*—for example **-WRVU=H06.05**.
- The **-Wtarget** flag is not required, but if you do specify it, you must specify **tns/e** for the value.
- For a C module compilation, this option causes the compiler to issue an error, instead of a warning, for implicitly declared functions.

-W[**no**]**saveabend**
Specifies that a saveabend (process snapshot) file is [is not] created if the program terminates abnormally. The default setting is **-Wnosaveabend**. This flag is meaningful only if an executable object file is produced.

-Wsavetemps
Saves all temporary and intermediate files created by compilation system components. Use the **-Wv** flag to display the filenames.

-Wsqlmx[="*args*"]
Invokes the NonStop SQL/MX **mxsqlc** preprocessor before compilation for any file operands of the form *file***.sql**, *file***.ec**, *file***.eC**, *file***.ecpp**, *file***.ecxx**, or *file***.ec++**. If an argument is specified, it must be one of the following:

listing Directs the preprocessor to write its diagnostic messages to a file named *file***.eL** in addition to the standard error file, where *file* is the name of the primary source file.

noansi_varchars
Directs the preprocessor to turn off generation of ANSI **varchar** data.

This option is valid only for preprocessor release 1.8 and newer.

noline Directs the preprocessor to suppress generation of **#line** directives in the preprocessed output source file that it creates.

null_terminate
Directs the preprocessor to terminate host variable strings with a NULL before fetch operations into them.

This option is valid only for preprocessor release 1.8 and newer.

preprocess_only
Directs the preprocessor to suppress all steps after preprocessing.

This option is valid only for preprocessor release 2.0 and newer.

process_includes
Directs the SQL/MX preprocessor to process one level of **include** files. This setting applies only to **include** files for SQL definitions; it does not apply to **include** file processing performed by the C/C++ compiler.

refrain_r2 Directs the SQL/MX preprocessor to use only the rules and features that apply to preprocessors prior to release 2.0. The default action is to use only the rules and features that apply to preprocessors beginning with release 2.0.

This option is valid only for preprocessor release 2.0 and newer.

IEEE_float Directs the SQL/MX preprocessor to use IEEE floating-point format instead of Tandem floating-point format. This option is valid only for preprocessor release 2.0 and newer.

For preprocessors before release 2.0, the default floating-point format is the same as that of the **c99** compiler (IEEE format). For preprocessors beginning with release 2.0, the default floating-point format is IEEE format.

If more than one argument is specified, the arguments must be separated by commas without any white space.

-Wsqlmxadd[="*args*"]
Specifies a string to pass to the SQL/MX preprocessor without validation or change.

-W[**no**]**stdfiles**
Generates a Guardian program that opens [does not open] the standard input, output, and error files by default. You can specify this flag only if **-Wsystype=guardian** is also specified. The default setting is **-Wstdfiles**.

-W[**no**]**suppress**
Disables [enables] the generation of listings. The listing is placed in a file in the current working directory with the same name as the source, but with a suffix of **.L**. The default setting is **-Wsuppress**.

-Wsyntax
Performs only a syntax check. No code is generated.

-Wsystype={**guardian** | **oss** }
Specifies the target execution environment. This flag selects definitions used during compilation, program startup code, default libraries, and system routines used during linking. The default setting is **-Wsystype=oss**. To run files compiled for a Guardian TNS/E target execution environment, you must set the file code to 800 with a `FUP ALTER` *filename*`, CODE 800` command from a TACL prompt.)

-Wtarget=*platform*
Specifies the system architecture for which code should be generated. The supported values are **-Wtarget=ipf** and **-Wtarget=tns/e**.

-Wtimestamp=*value*
Provides a creation timestamp for the NonStop SQL/MX preprocessor that is written to the two output files created by the preprocessor. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about the form for the *value* allowed for the timestamp. If this option is specified more than once, only the last occurrence has an effect. Note that **c99** does not check that *value* is valid; it relies on the NonStop SQL/MX preprocessor to validate this argument.

This flag is ignored unless the **-Wsqlmx** flag is also specified.

-Wu="*symbol_name***"**
Tells the linker to add *symbol_name* as an undefined symbol. This causes the linker to search for this symbol in any archive libraries that are specified after this flag on the command line or in an obey file.

The search constraint specified by the **-Wu** flag is overridden by use of the **-Winclude_whole** flag.

-Wv Echoes the command line to the standard error file as each component of the compilation system is run.

-Wverbose
Echoes the command line to the standard error file as each component of the compilation system is run and causes additional output and listings from the SQL compiler to be sent to the standard output file. SQL compiler error messages are sent to the standard error file.

-Wversion3
Specifies which C++ dialect to compile.

-Wversion3 specifies the dialect released with G06.20. This version supports an ANSI/ISO Standard C++ Library corresponding to ISO/IEC IS 14882. This is the default. The **c99** utility does not support earlier dialects.

All modules of a C++ program must be compiled and linked using the same dialect.

-Ww Suppresses the printing of compiler warning messages. This flag overrides any **-Wwarn** or **-Wnowarn** flags.

-W[**no**]**warn**[=*w* [,*w*] ...]
For each *w* value that appears, this flag enables [disables] the compiler warning message specified by *w*.

Declaring a *w* value enables [disables] the specified message. Specifying **-Wwarn** [**-Wnowarn**] by itself enables [disables] all compiler warning messages.

If **-Wwarn=***w* is specified, then **-Wnowarn** must also be specified or the **-Wwarn=***w* flag is ignored. If **-Wnowarn=***w* is specified, then **-Wwarn** need not be specified.

If white space is present after the commas, the list of warning message numbers should be enclosed in quotation marks.

-Wwarn is the default specification for this flag. **-Ww** overrides the **-W**[**no**]**warn** flag.

-Wx Strips part of the symbol table from the output object file but keeps information necessary for the object file to be used as input to a linker utility again. This flag is typically used with **-Wr**.

Multiple instances of the **-D**, **-I**, and **-U** flags and of the **-l** operands can be specified.

The position of **-l** *library* operands within a list of flags affects the order in which the libraries are searched.

The order of specifying the **-I** and **-L** flags is significant.

Quotation marks around string values in flags are optional but recommended to avoid errors caused by shell substitutions or deletions.

Refer to the *C/C++ Programmer's Guide* for details.

DESCRIPTION

c99 is a driver program for the native C and C++ language compilation system. The **c99** utility compiles C and C++ programs using the native TNS/E compilers. Source files must comply with the C99 (ISO/IEC 9899:1999) standard, but are permitted to use HP extensions if you specify the **-Wextensions** flag. For source files that comply with the C89 standard or for programs that require the compiler to process C source files traditional Kernighan and Ritchie C or Common C rules, instead of according to ISO/ANSI Standard C, use the **c89** utility with the **-Wkr** flag.

This reference page describes using **c99** in the OSS environment.

c99 performs simple validation of the flags and operands from the arguments on its command line and, depending on those arguments, invokes components of the language compilation system. **c99** does not verify the existence of files it passes to compilation system components. It does verify that the operand suffix identifies a valid operand to pass to compilation system components. **c99** and the components it runs issue messages to the standard error file.

c99 performs the following steps:

1. If the corresponding **-W** flag is specified, invokes the SQL/MX preprocessor to process any embedded SQL statements in C or C++ source files, creating C only, C++ only, or module definition files as appropriate.
2. Compiles any specified C and C++ source files or source files produced by Step 1 into object files.
3. If the **-Wmxcmp** flag is specified, invokes the NonStop SQL/MX compiler to process files created by Step 1 or specified as module definition files in the command.
4. Links the object files together with any libraries specified on the command line. (This occurs if no flags that prevent linking are specified and the source files are compiled without errors.)
5. Writes an executable object file or library to the file specified by a **-o** flag (if present) or to the file **a.out**.

Libraries can be:

- Archives, with a suffix of **.a**
- DLLs, with a suffix of **.so**

The default executable file in the Guardian file system is **aout** in the subvolume from which **c99** is invoked.

If only a single source file is given and no flags that suppress linking are specified, then the file is compiled into an object file and linked into an executable object file. If the executable file is created successfully, the object file is removed.

c99 places object files (loadfiles) in the current working directory with the same base name as the corresponding source file, but with a suffix of **.o**.

c99 also names several temporary or intermediate files that are created during the compilation process. Like the output object file, **c99** places these files in the current working directory. **c99** removes these temporary or intermediate files unless the **-Wsavetemps** flag is specified.

If **-Wsystype=oss** is set, the C and C++ compilers define the predefined feature-test macros **_OSS_TARGET** and **_XOPEN_SOURCE**. If **-Wsystype=guardian** is set, the C and C++ compilers define the predefined feature-test macros **_GUARDIAN_TARGET** and **_TANDEM_SOURCE**. These macros are used in the standard header files to determine the execution environment of a program. The feature-test macros can also be defined with a **-D** flag. Because these macros are defined internally, not by **c99**, the macros are not defined when **-Wdryrun** or **-Wv** are used.

Dynamic and Static Linking

The **-WBdllsonly** and **-WBdynamic** flags specify dynamic linking. The **-WBstatic** flag specifies static linking. In dynamic linking, the **eld** utility first searches for a dynamic-link library (DLL). If a DLL cannot be found, the linker then searches for an archive file. If neither of these files are found, an error is issued. In static linking, the linker searches for an archive file but does not search for a DLL.

If the archive file cannot be found, an error is issued.

Dynamic and static linking are not exact opposites. Dynamic linking accepts either a DDL or an archive, but static linking accepts only an archive.

Unlike other **c99** flags, multiple **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags can be specified in a single **c99** invocation. Thus, it is possible to perform dynamic linking for some **-l** operands and static linking for others.

The **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags specified to **c99** affect linking arguments specified in **-Weld**, or **-Weld_obey** flags. Each specification remains in effect until another is encountered. To change how a linker performs linking for such arguments, you can specify **eld** flags that control linking within the argument list. All linkers perform dynamic linking by default. Refer to the **eld(1)** reference page for more information.

Handling of Files in the Guardian File System

Files in the Guardian file system can be accessed using OSS pathname syntax (/**G**/*volume*/*subvol*/*fileID*).

c99 requires that files in the Guardian file system be identified with a suffix as is done in the OSS file system. Because Guardian filenames do not allow the **.***suffix* format, the period is dropped and the suffix becomes the last character of the filename. However, the **.***suffix* format must be used when specifying the file to **c99**.

Thus, the Guardian file system file **$VOL.SUBVOL.FILEC**, which identifies a C source file, is specified to **c99** as **/G/VOL/SUBVOL/FILE.c**. Likewise, **c99** generates an object file

$VOL.SUBVOL.FILEO that can be specified to **c99** again as **/G/VOL/SUBVOL/FILE.o**.

The default executable object file when the current working directory is in the Guardian file system is **aout**.

Predefined Preprocessor Symbols and Macros

c99 defines the following preprocessor symbols and feature-test macros:

__cplusplus
Directs the preprocessor to process the source text as C++ source code. **c99** defines this symbol if the **-Wcplusplus** flag is specified or the name of an OSS source file ends in a C++ suffix (**.ii**, **.C**, **.cpp**, **.c++**, **.cxx**, **.cc**, **.eC**, **.ecpp**, **.ec++**, **.ecc**, or **.ecxx**).

_TANDEM_SOURCE
Makes visible to the preprocessor identifiers required or permitted by extensions made by HP. **c99** defines this feature-test macro if the **-Wextensions** flag is specified.

_XOPEN_SOURCE
Makes visible to the preprocessor identifiers required or permitted by extensions made by the XPG4 specification. **c99** defines this feature-test macro by default unless the **-Wsystype=guardian** flag is specified.

There are other feature-test macros defined by the compiler itself, not by **c99**. These feature-test macros do not appear in the output of **-Wv** and **-Wdryrun** flags. Refer to the *C/C++ Programmer's Guide* for further information on feature-test macros.

Operands

An *operand* is in the form of:

- A pathname
- **-l** *library*
- **-WBdllsonly**
- **-WBdynamic**
- **-WBstatic**

At least one operand of the pathname form must be specified. The following operands are supported:

*file***.a** An archive library of object files typically produced by the **ar** command and passed directly to a linker utility

*file***.c** A C language source file to be preprocessed, compiled, and optionally linked

*file***.C** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.cc** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.cpp** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.cxx** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.c++** A C++ language source file to be preprocessed, compiled, and optionally linked

*file***.ec** A C language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.eC** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.ecc** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.ecpp** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.ecxx** A C++ language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

*file***.i** A preprocessed C source file to be compiled and optionally linked

*file***.ii** A preprocessed C++ source file to be compiled and optionally linked

*file***.m** A module definition file (MDF) containing NonStop SQL/MX information for a corresponding C source file

*file***.o** An object file passed directly to a linker utility

*file***.so** A dynamic-link library (DLL) containing position-independent code (PIC) for use by the **eld** or **rld** utility

*file***.sql** A C language source file containing embedded NonStop SQL/MX information to be preprocessed, compiled, and optionally linked

-l *library* A file to be searched by a linker utility to resolve current unresolved external references; **eld** searches for files named **lib***library***.so** and **lib***library***.a**,

A library is searched when its name is encountered, so the placement of **-l** is significant. See the **Standard Libraries** subsection for more details.

-WBdllsonly

Specifies that the linker utility uses dynamic linking when searching for dynamic-link libraries specified in subsequent **-l** operands; placement of this operand is significant (refer to the **Dynamic and Static Linking** subsection for details)

-WBdynamic

Specifies that the linker utility uses dynamic linking when searching for libraries specified in subsequent **-l** operands; placement of this operand is significant (refer to the **Dynamic and Static Linking** subsection for details)

-WBstatic

Specifies that the linker utility uses static linking when searching for libraries specified in subsequent **-l** operands. Placement of this operand is significant. Refer to the **Dynamic and Static Linking** subsection for details.

Input Files

An input file is one of the following:

- A text file containing a C language or C++ language source program
- An object file in the format produced by the command **c99 -c**

- A library of object files in the format produced by archiving zero or more object files using the **ar** command
- A linkfile or loadfile produced by the **eld** utility
- A module definition file created by the NonStop SQL/MX preprocessor.

When the **-Wsqlmx** flag is specified, **c99** uses the source file filename extension to determine the language mode (C or C++) and the names of the source files created by the NonStop SQL/MX preprocessor.

The name of a created source file is the name of the primary source file plus one of the following extension transformations:

- If the primary source file suffix is **.ec** or **.sql** and the **-Wcpluplus** flag is not specified, the created source file has the suffix **.c** and uses the C language mode.
- If the primary source file suffix is **.ec** or **.sql** and the **-Wcplusplus** flag is also specified, the created source file has the suffix **.cpp** and uses the C++ language mode.
- If the primary source file suffix is **.eC**, **.ecpp**, **.ec++**, **.ecc**, or **.ecxx**, the created source file has the suffix **.C**, **.cpp**, **.c++**, **.cc**, or **.cxx**, resapectively, and uses the C++ language mode.

When **c99** is passed a file name suffixed with **.C**, **.cpp**, **.c++**, **.cc**, or **.cxx**, that file is not passed to the NonStop SQL/MX preprocessor. Such files are assumed to contain only C or C++ statements without embedded NonStop SQL/MX information.

Output Files

An output file can be a preprocessed source file, an object file, or an executable file.

Standard Output File

The standard output file is empty unless a **-E**, **-WM**, or **-WP** flag is specified. If one of these flags is specified, preprocessed source code is sent to the standard output file. When **-WH** is used, the standard output file contains a line indicating which file is currently being operated upon.

Standard Error File

The standard error file contains diagnostic and informational messages from **c99** and the compilation components it calls. If more than one source file operand is specified, then for each such file the format `"%s: \n",`*file* is used to print the name of the source file before it is processed.

Standard Libraries

The **c99** utility recognizes the following **-l** operands for standard libraries.

-l c Contains all library functions provided by HP that are specified in the XPG4 Version 2 (X/Open UNIX) specification, including those functions listed as residing in the **math.h** header file.

-l C Contains the correct C++ run-time libraries, based on the value of the **-Wversion***n* flag. If the **-Wversion***n* flag is omitted, the default version C++ library is used.

When you specify the **-Wcplusplus** and **-Wversion***n* flags, you need not specify **-l C**; all needed libraries are automatically linked. When you specify the **-Wversion3** flag, **cppinit3.o** is linked for a nonPIC program or **cppinit4.o** is linked for a PIC program.

-l l Contains all functions required by the C language output of the **lex** utility that are not made available through the **-l c** operand.

-l m Contains all functions referenced in the **math.h** header file.

-l y Contains all functions required by the C language output of the **yacc** utility that are not made available through the **-l c** operand.

In the absence of flags that inhibit invocation of a linker utility, such as **-c** and **-E**, **c99** directs the linker to search the standard C library after all other object files and libraries are searched.

If a C++ source file operand or a **-Wcpluscplus** flag is specified, **c99** directs the linker to search the C++ run-time library before it searches the standard C library. If you want the libraries to be searched in a specific order or linking options to be processed in a specified order, you should start the appropriate linker (**eld**) directly from the OSS shell and not use the **c99** command to do the linking.

Libraries residing in the Guardian file system cannot be specified as **-l** operands because of the naming convention. They can be specified in the desired order with the **-Weld** flag.

Standard Include Directories

The standard include directory contains the standard C and C++ header files. **c99** passes a **-I** flag naming this directory as the last **-I** flag when processing source files. In the OSS environment, the directory is **/usr/include**.

Standard Library Directories

The standard library directories contain the dynamic-link libraries (DLLs) used by the **eld** utility to resolve external references.

In the OSS environment, a linker first searches the directory that contains the current version of the operating system image (the active **/G/system/sys***nn* directory). The linker then searches the **/lib**, **/usr/lib**, and **/usr/local/lib** directories.

The value of the **COMP_ROOT** environment variable is added to the beginning of **/lib**, **/usr/lib**, and **/user/local/lib**. By default, the value of **COMP_ROOT** is null in the OSS environment.

See the **eld(1)** reference page for more information about controlling the search order for a PIC file linker.

Default Flags

If no flags are specified, **c99** behaves as if the following flags were specified:

```
-Wsystype=oss -o a.out -Wenv=common -Wfieldalign=auto
-Wrefalign=8 -Wansistreams -Winline
-WIEEE_float
-Wlist -Wnoinnerlist -Wsuppress -Wstdfiles -Wnomap
-Wnoextensions -Wnooverflow_traps -Wnoremarks
-Wnoinspect -Wnosaveabend -WBdynamic
-Wcall_shared
-Wnoincludewhole -Wnoreexport -Wnooptionl_lib
-Wversion3
-Wwarn
-D_XOPEN_SOURCE -I/usr/include -L/lib -L/usr/lib
-L/usr/local/lib -lc
```

Environment Variables

The following environment variables affect the execution of **c99**.

CCOMBE

Determines the pathname of the **ccombe** component of the C and C++ compilers. **/usr/cmplr/ccombe** is the default location for the OSS environment.

COMP_ROOT

Changes the default pathnames for:

- The **c99** compilation system components
- The standard include directory
- The standard library directories

In the OSS environment, the string specified in **COMP_ROOT** is added to the beginning of the default pathnames.

If a component's environment variable is set explicitly, the **COMP_ROOT** environment variable does not modify that component's environment variable.

ELD Determines the pathname of the **eld** utility invoked by **c99**. **/usr/bin/eld** is the default location for the OSS environment.

MXCMP

Determines the pathname of the NonStop SQL/MX release 1 compiler. **/G/system/system/mxcmp** is the default.

MXCMPUM

Determines the pathname of the NonStop SQL/MX release 2 compiler. **/usr/tandem/sqlmx/bin/mxCompileUserModule** is the default.

MXSQLC

Determines the pathname of the C/C++ NonStop SQL/MX preprocessor, **mxsqlc**. **/usr/tandem/sqlmx/bin/mxsqlc** is the default.

SQLMX_PREPROCESSOR_VERSION

Indicates the preprocessor rules and features to be used. Specifying the value 800 causes rules and features associated with release 1.8 to be used; the **mxcmp** compiler is used and only MDF files and annotated source files are produced, while rules and features associated with release 2.0 and later are ignored. Specifying a value of 1200 or larger or not specifying a value causes rules and features associated with release 2.0 and later to be used; the **mxCompileUserModule** compiler is used and annotated source files that contain embedded module definitions are produced instead of MDF files, while restrictions associated with release 1.8 or earlier are ignored.

TMPDIR

Determines the pathname that overrides the default directory for temporary files created by **c99** and the components it invokes. By default, temporary files are stored in the **/tmp** directory. If **TMPDIR** is set to a directory that does not exist or is not writeable, **c99** uses the default directory as described on the **tempnam(3)** reference page.

EXAMPLES

1. The command

   ```
   c99 test1.c
   ```

   compiles the source file **test1.c** and links the object file into an executable file **a.out** in the current working directory.

2. The command

   ```
   c99 -Wnowarn -Wwarn=262 test1.c
   ```

   compiles the source file **test1.c** and links the object file into an executable file **a.out** in the current working directory. All compiler warning messages except message number 262 are disabled.

3. The command

   ```
   c99 -c /home/me/app/test1.c
   ```

   compiles the source file **/home/me/app/test1.c** into the object file **test1.o** in the current working directory.

4. The command

   ```
   c99 -g -o test2 x.c y.c z.c
       -Wnostdinc
       -I/dev/product/app/src
       -I/new/usr/include
       -lclient -lserver
       -L/dev/product/lib
       -L/new/usr/lib
   ```

   compiles the source files **x.c**, **y.c**, and **z.c** and links their respective object files **x.o**, **y.o**, and **z.o** into the executable file **test2**. Symbolic information is generated by the compiler and retained by the linker utility for debugging.

   Included files are searched for in the directories **/dev/product/app/src** and **/new/usr/include**; **/usr/include** is not searched. The linker searches for the libraries **libclient.srl** and **libserver.srl** in the directories **/dev/product/lib** and **/new/usr/lib** before searching in the directories **/G/system/sys***nn*, **/lib**, **/usr/lib**, and **/usr/local/lib**.

5. The command

   ```
   c99 -g -o test2 x.c y.c z.c
       -Wcall_shared
       -Wnostdinc
       -I/dev/product/app/src
       -I/new/usr/include
       -lclient -lserver
       -L/dev/product/lib
       -L/new/usr/lib
   ```

   compiles the source files **x.c**, **y.c**, and **z.c** and links their respective object files **x.o**, **y.o**, and **z.o** into the loadfile **test2**. Symbolic information is generated by the compiler and retained by the linker utility for debugging.

   Included files are searched for in the directories **/dev/product/app/src** and **/new/usr/include**; **/usr/include** is not searched. The linker searches for the libraries **libclient.so** and **libserver.so** in the directories **/dev/product/lib** and **/new/usr/lib** before searching in the directories **/G/system/sys***nn*, **/lib**, **/usr/lib**, and **/usr/local/lib**.

6. The command

   ```
   c99 -o test3 -O -DTYPE=3
       -I/usr/friend
       -I/usr/myself/headers
       foo.c bar.o baz.c
   ```

compiles the source files **foo.c** and **baz.c** and links their respective object files with **bar.o** into the object file **test3.o**. The preprocessor symbol **TYPE** is defined to 3, and full optimization is performed by the compiler. The compiler looks for included files in the directory **/usr/friend**, then in **/usr/myself/headers**, then in **/usr/include**.

7. The command

   ```
   c99 -c -Wsqlmx file1.eC file2.ecc file3.ec++
   ```

   uses the **mxsqlc** preprocessor on several C++ source files and also compiles them, but does not link the results. If no errors are detected in either the preprocessing or compilation steps, the following files are created: **file1.m**, **file1.C**, **file2.m**, **file2.cc**, **file3.m**, **file3.c++**, **file1.o**, **file2.o**, **file3.o**.

8. The command

   ```
   c99 -Wsqlmx -E file.ec > file-cpp.c
   ```

   uses the **mxsqlc** preprocessor to expand embedded SQL statements and invokes the NonStop SQL/MX preprocessor to create a single source file containing only C statements.

9. The command

   ```
   c99 -c -Wsqlmx file1.C file2.ecc file3.ec++ file4.cpp
   ```

   illustrates mixing C++ source files, with and without NonStop SQL/MX information, on a single command line. Only source files that have names with one of the SQL extension suffixes invoke the **mxsqlc** preprocessor. However, all files are compiled but not linked. If no errors are detected in either the preprocessing or compilation steps, the following files are created: **file2.m**, **file2.cc**, **file3.m**, **file3.c++**, **file1.o**, **file2.o**, **file3.o**, **file4.o**.

10. The command

    ```
    c99 -Wmxcmp -Wmxcmp_files=test1.m,test1.o
    ```

    SQL-compiles the MDF file **test1.m** using the NonStop SQL/MX **mxcmp** compiler and processes the file **test1.o** using the NonStop SQL/MX **mxCompileUserModule** without also linking it.

11. The command

    ```
    c99 file.c -lc -WBstatic -l archive -WBdynamic -l native
    ```

    compiles the source file **file.c** and links the object file into an executable file **a.out** in the current working directory. The linker performs dynamic linking by searching first for the file **libc.srl** and then **libc.a**. The linker then performs static linking by searching for the file **libarchive.a**. The linker then performs dynamic linking by searching first for the file **libnative.srl** and then **libnative.a**.

12. The command

    ```
    c99 file.c -lc -Wcall_shared -WBstatic
        -l archive -WBdynamic
        -l native
    ```

    compiles the source file **file.c** and links the object file into loadfile **a.out** in the current working directory. The linker performs dynamic linking by searching first for the file **libc.so** and then **libc.a**. The linker then performs static linking by searching for the file **libarchive.a**. The linker then performs dynamic

linking by searching first for the file **libnative.so** and then **libnative.a**.

13. The command

    ```
    c99 -wmxcmp -c module.m
    ```

    invokes the NonStop SQL/MX compiler to process the module definition file **module.m**.

FILES

/usr/bin/c99
Native **c99** in the OSS environment.

/G/system/sys*nn*/zcppcdll
C++ run-time library function object code for J-series and H-series processes; linked automatically when you compile and link C++ surce files or when you specify the **-Wcplusplus** flag.

/G/system/sys*nn*/zcpp3dll
C++ run-time library function object code for J-series and H-series processes; linked automatically when you compile and link C++ surce files or when you specify the **-Wcplusplus** flag.

/G/system/sys*nn*/zcredll
Common Run-Time Environment (CRE) function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zcrtldll
C run-time library function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zicnvdll
Function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zi18ndll
Internationalization function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zosscdll
Function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zossedll
Function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zossfdll
Function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zosskdll
Function object code for J-series and H-series processes; linked automatically.

/G/system/sys*nn*/zpgodll
Symbols referenced by instrumented code for J-series and H-series processes; must be linked when you link a program or DLL that contains instrumented code; linked automatically when you specify the **-Wcodecov** or **-Wprofgen** flags.

/G/system/sys*nn*/zsecdll
Security function object code for J-series and H-series processes; linked automatically.

DIAGNOSTICS

If **c99** encounters a compilation error that prevents an object file from being created, it writes a diagnostic message to the standard error file and continues to compile other source code operands. However, it does not perform program linking and returns a nonzero exit status.

If the linking is unsuccessful, **c99** writes a diagnostic message to the standard error file and returns a nonzero exit status.

EXIT VALUES

The following exit values are returned:

0 (zero)	Successful completion.
>0	An error occurred.

RELATED INFORMATION

Commands: **ar(1)**, **c89(1)**, **eld(1)**, **strip(1)**.

Functions: **fp_class(3)**, **tempnam(3)**.

Files: **float(4)**.

STANDARDS CONFORMANCE

All **-W** options are HP extensions to the POSIX and XOPEN standards.

NAME

cal - Displays a calendar

SYNOPSIS

cal [[*month*] *year*]

DESCRIPTION

The **cal** command writes to the standard output file a Gregorian calendar for the specified month or year.

If you provide two operands, **cal** assumes the first to be *month* and the second to be *year*. The *month* operand specifies the month for which you want the calendar, and it must be a number in the range 1 through 12 for January through December, respectively. The *year* operand specifies the year for which you want the calendar, and it must be a number in the range 1 through 9999. Because **cal** can display a calendar for any year in this range, enter a full 4-digit *year* (for example, **1993**) rather than just the last two digits.

If you provide only one operand, **cal** assumes that operand to be *year* (even if it is in the range 1 through 12) and displays a calendar for all 12 months of that year.

If you do not provide any operands, **cal** displays a calendar for the current month of the current year.

Environment Variables

The **cal** command checks the **LC_TIME** environment variable and uses the correct headers for the current locale. If **LC_TIME** is not set, **cal** checks the value of **LANG**. If neither variable is set, you receive English headers.

EXAMPLES

1. To display a calendar for February 1999, enter:

   cal 2 1999

2. To print a calendar for the entire year of 1999, enter:

   cal 1999 | print

3. To display a calendar for the year 84 of the Common Era, enter:

   cal 84

EXIT VALUES

The **cal** command returns the following exit values:

0 (zero)	The **cal** utility successfully finished its processing.
>0	An error occurred.

RELATED INFORMATION

Files: **locale(4)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions, except for the following features:

- The **SHELL** and **TZ** environment variables are not used.

NAME

cancel - Removes job requests from the line printer spooling queue

SYNOPSIS

cancel *[request-ID ...] location ...*

The **cancel** command removes one or more requests from a printer's spool queue.

DESCRIPTION

Because the spooling directory is protected from users, using the **cancel** command is normally the way a user can remove a request.

Users can cancel jobs they initiated. Superusers with special privileges can cancel jobs initiated by other users.

You can remove an individual request from a queue by specifying its request ID. (You can obtain the request ID by using the **lp** command.)

The argument *location* specifies a spooler location where the job request to be removed resides. If no ID is specified, all the jobs initiated by the user that reside in the specified location are canceled. If *location* is not specified, the system default location is used.

The **cancel** command does not return a message if there are no requests in the queue that match the request list.

The **cancel** command kills an active daemon, if necessary, before removing any spooling files. If a daemon is killed, a new one is automatically restarted upon completion of file removals.

You cannot use **cancel** to remove another user's job unless you have superuser privileges.

cancel can remove only jobs in the READY or HOLD state (see the reference page for the **lpstat** command). When a job request is under the control of another process, such as a **PRINT** process, **cancel** has no effect.

Environment Variables

LANG Provides a default value for the internationalization variables that are unset or null. If the **LANG** variable is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the **cancel** command behaves as if none of the variables have been defined.

LC_ALL

When set with a nonempty string, overrides the values of all other internationalization variables.

LC_CTYPE

Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multibyte characters in arguments and input files).

LC_MESSAGES

Determines the locale to be used to affect the format and contents of diagnostic messages written to the standard error file and informative messages written to the standard output file.

NLSPATH

Determines the location of message catalogs for processing the **LC_MESSAGES** variable.

EXAMPLES

1. To remove a job whose request ID is **123** in the default print queue, enter:

   cancel 123

2. To remove a job whose request ID is **123** in the print queue for printer1, enter:

   cancel 123 -Pprinter1

NOTES

Because race conditions are possible in the update of the lock file, the currently active request may be incorrectly identified.

EXIT VALUES

The following exit values are returned by the **cancel** command:

0	Completion was successful.
>0	An error occurred.

RELATED INFORMATION

Commands: **lp(1), lpstat(1).**

NAME

cat - Concatenates or displays files

SYNOPSIS

cat [**-benrstuv**] [- | *file*] ...

The **cat** command reads each specified file in sequence and writes it to standard output.

FLAGS

-b Omits line numbers from blank lines when **-n** is specified. If you specify the **-b** flag, the **-n** flag is automatically invoked with it.

-e Same as the **-v** flag with a **$** (dollar sign) character displayed at the end of each line.

-n Displays output lines preceded by line numbers, numbered sequentially from 1.

-r Replaces multiple consecutive empty lines with one empty line, so there is never more than one empty line between lines containing characters.

-s Does not display a message if **cat** cannot find an input file. (Silent option.)

-t Same as the **-v** flag, with the tab character printed as **<Ctrl-i>** (**^I**).

-u Does not buffer output. Writes bytes from the input file to standard output without delay as each is read.

-v Displays nonprinting characters so they are visible.

DESCRIPTION

The **cat** command is frequently used with **>** (redirection symbol) to concatenate the specified files and write them to the specified destination. (See **CAUTIONS**.) **cat** is also used with **>>** to append a file to another file.

If you do not specify a file or if you specify **-** (dash) instead of *file*, **cat** reads from standard input. The **cat** command accepts multiple occurrences of **-** (dash) as a file argument.

EXAMPLES

1. To display the file **notes**, enter:

   cat notes

   If the file is longer than one screenful, it scrolls by too quickly to read. To display a file one page at a time, use the **more** command.

2. To concatenate several files, enter:

   cat section1.1 section1.2 section1.3 > section1

   This creates a file named **section1** that is a copy of **section1.1** followed by **section1.2** and **section1.3**.

3. To suppress error messages about files that do not exist, enter:

   cat -s section2.1 section2.2 section2.3 > section2

   If **section2.1** does not exist, this command concatenates **section2.2** and **section2.3**. Note that the message goes to standard error, so it does not appear in the output file. The result is the same if you do not use the **-s** flag except that **cat** displays the error message:

   ```
   cat: cannot open section2.1
   ```

You may want to suppress this message with the **-s** flag when you use the **cat** command in shell procedures.

4. To append one file to the end of another, enter:

   cat section1.4 >> section1

   The >> in this command specifies that a copy of **section1.4** be added to the end of **section1**. If you want to replace the file, use a single > symbol.

5. To add text to the end of a file, enter:

   cat >> notes
   Get milk on the way home
   <Ctrl-y>

   Get milk on the way home is added to the end of **notes**. When you use this syntax, the **cat** command waits for you to enter text. Press the End-of-File key sequence (**<Ctrl-y>** above) to indicate you are finished.

6. To concatenate several files with text entered from the keyboard, enter:

   cat section3.1 - section3.3 > section3

   This concatenates **section3.1**, text from the keyboard, and **section3.3** to create the file **section3**.

7. To concatenate several files with output from another command, enter:

   ls | cat section4.1 - > section4

   This command copies **section4.1** and then the output of the **ls** command to the file **section4**.

8. To get two pieces of input from the terminal (when standard input is a terminal) with a single command invocation, enter:

   cat start - middle - end > file1

   If standard input is a regular file, however, the preceding command is equivalent to the following:

   cat start - middle /dev/null end > file1

   The commands are equivalent because the entire contents of the file would be consumed by **cat** the first time **-** (dash) was used as a file argument. An End-of-File condition would then be detected immediately when **-** appeared the second time.

CAUTIONS

Do not redirect output to one of the input files using the > (redirection symbol). If you do this, you lose the original data in the input file because the shell truncates it before **cat** can read it. (See also the **sh** command.)

RELATED INFORMATION

Commands: **more**(1), **pr**(1), **sh**(1).

NAME

cd - Changes the current directory

SYNOPSIS

cd *argument*

DESCRIPTION

The **cd** command changes the current directory to *argument*, where *argument* is a pathname.

If *argument* is a - (dash), the directory is changed to the previous directory.

The **HOME** shell parameter is the default *argument*. The **PWD** parameter is set to the current directory. The **CDPATH** shell parameter defines the search path for the directory containing *argument*.

Separate alternative directory names with a : (colon). The default path is a null string, specifying the home directory. Note that the current directory is specified by a full pathname.

If *argument* begins with a / (slash), the search path is not used. Otherwise, each directory in the path is searched for *argument*.

EXAMPLES

1. The following command changes to the directory one level above the current directory.

   cd ..

2. The following command changes to a directory named **mywork** that is one level beneath the user's home directory.

   cd mywork

NOTES

The **cd** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **pwd**(1), **sh**(1).

NAME

chgrp - Changes the group ownership of a file or directory

SYNOPSIS

chgrp [**-W NOG**] [**-W NOE**] [**-fhR**] *group file ...*

The **chgrp** command changes the group associated with the specified file or directory to *group*.

FLAGS

-f Suppresses all error reporting.

-h Changes the group ownership of a symbolic link, instead of the file to which the symbolic link points. When you use this flag, the **chgrp** command does not affect the file pointed to by the symbolic link. If you use the **-R** flag with this flag, recursion does not take place.

-R Causes **chgrp** to descend recursively through its directory arguments, setting the specified group ID. If **chgrp** fails to change the group ID of a particular file in the hierarchy, it continues to process the remaining files. If **chgrp** cannot read or process a directory in the hierarchy, it continues to process the other parts of the hierarchy. When symbolic links are encountered and the **-h** flag is not used, the group ownership of the parent file or directory changes, but the group ownership of linked files or directories does not change. If you use the **-h** flag with this flag, recursion does not take place.

HP Extensions

-W NOG Specifies that the **/G** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not **/**, **/E**, or **/E/***system* or when recursion does not occur.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not root or when recursion does not occur.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

The effective user of the process must match the owner of the file.

Users can change the group of a file to a group that they belong to (their effective group or one of their supplementary groups). If you do not own the file and do not belong to the new group, you must have superuser authority to change the group name or group ID.

The *group* argument must be either a valid group name or a valid group ID that exists in the group database. If a numeric group operand exists in the group database as a group name, the group ID number associated with that group name is used as the group ID.

Access Control Lists (ACLs)

A user can allow or deny specific individuals and groups access to a file by using the access control list (ACL) for the file. When using the **chgrp( )** function in conjunction with ACLs, if the new owner and/or group of a file have optional ACL entries corresponding to **user:***uid***:***perm* or **group:***gid***:***perm* in the ACL for a file, those entries remain in the ACL but no longer have any effect because they are superseded by the **user::***perm* or **group::***perm* entries in the ACL.

Access control lists (ACLs) are not supported for symbolic links.

For more information about ACLs, see the **acl(5)** reference page.

Environment Variables

The following environment variables affect the execution of the **chgrp** command:

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root and a recursive operation occurs in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE Omit the **/E** directory.

NOG Omit the **/G** directory.

NOG:NOE Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

EXAMPLES

1. To change the group ownership of the file or directory named **proposals** to **staff**, enter:

   ```
   chgrp staff proposals
   ```

   The group access permissions for **proposals** now apply to **staff**. See the **chmod(1)** reference page for details.

2. To recursively change the group ownership of all OSS files on the local node to the SUPER group, enter:

   ```
   chgrp -W NOG -W NOE -R SUPER /
   ```

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOE** and **-W NOG** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

RELATED INFORMATION

Commands: **chmod(1)**, **chown(1)**, **ls(1)**.

Functions: **chmod(2)**, **chown(2)**.

STANDARDS CONFORMANCE

The **-W NOE** and **-W NOG** flags and the **UTILSGE** environment variable are HP extensions to the XPG4 Version 2 specification.

NAME

chmod - Changes permissions and other file mode settings

SYNOPSIS

chmod [**-fR**] *absolute_mode file ...*

chmod [**-W NOG**] [**-W NOE**] [**-fhR**] [*who*] *+permission ...* | *-permission ...* | *=permission ... file ...*

FLAGS

-f Does not report an error if the **chmod** command fails to change the mode on a file.

-h Changes the mode of a symbolic link, instead of the file to which the symbolic link points. When you use this flag, the **chmod** command does not affect the file pointed to by the symbolic link. If the symbolic link refers to a directory and the both the -R and -h flags are used, the ownership of the symbolic link is changed, but ownership of the directory to which the symbolic link refers remains unchanged and recursion into that directory does not occur.

-R Causes **chmod** to descend recursively through its directory arguments, setting the mode for each file as described in **Symbolic Mode** and **Absolute Mode** in the **DESCRIPTION** section. If **chmod** is unable to change the mode of a particular file, or unable to read or search a particular directory, it continues processing through the hierarchy. When the **chmod** command encounters a symbolic link:

- If the symbolic link refers to a file, the owner of the file is changed.
- If the symbolic link refers to a directory, the owner of the directory is changed, but recursion into the directory does not occur.
- If the symbolic link refers to a directory and the both the -R and -h flags are used, the ownership of the symbolic link is changed, but ownership of the directory to which the symbolic link refers remains unchanged and recursion into that directory does not occur.

HP Extensions

-W NOG Specifies that the **/G** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not **/**, **/E**, or **/E/***system* or when recursion does not occur.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not root or when recursion does not occur.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

You can use either symbolic mode or absolute mode to specify the desired permission settings. Symbolic mode is more portable but does not offer all of the options of absolute mode.

To change the file access permissions of a file or directory, the effective user ID of the process must match the super ID or the owner of the file, or the effective user ID of the process or one of its group affiliations must qualify it for membership in the Safeguard SECURITY-OSS-

ADMINISTRATOR group.

If **chmod** is invoked by a process whose effective user ID does not equal the super ID or file owner, the set-user-ID and set-group-ID bits of the file mode (04000 and 02000, respectively) are cleared.

Access Control Lists (ACLs)

When you execute the **chmod** command, you can change the effective permissions granted by optional entries in the ACL for a file. In particular, using the **chmod** command to remove read, write, and execute permissions from a file owner, owning group, and all others works as expected, because the **chmod** command affects the **class** entry in the ACL, limiting any access that can be granted to additional users or groups through optional ACL entries. To verify the effect, use **getacl** command on the file after the **chmod** command completes and note that all optional (nondefault) ACL entries with nonzero permissions also have the comment **# effective:---**.

To set the permission bits of access control list entries, use the **setacl** command instead of the **chmod** command.

ACLs are not supported for symbolic links.

Symbolic Mode

Symbolic mode has the form:

[*who*] *operation permission*[**,** *operation permission* ...]

The *who* argument specifies whether you are defining permissions for a user, group, or all others, or any combination of these. The *operation* argument specifies whether the permission is being added, removed, or assigned absolutely. The *permission* argument identifies the operation that the specified users can perform on *file*.

Valid options for the *who* argument are as follows:

a User, group, and all others (same effect as the combination **ugo**)

g Group

o All others

u User (owner)

If the *who* argument is omitted, the default value is **a**, but the setting of the file creation mask, **umask** (see the reference page for **sh(1)**), is applied.

Valid options for the *operation* argument are as follows:

- Removes specified permissions.

+ Adds specified permissions.

= Clears the selected permission field and sets it to the specified code. If you do not specify a permission code following **=**, the **chmod** command removes all permissions from the selected field.

Valid options for the *permission* argument are as follows:

r Read permission.

w Write permission.

x Execute permission for files, search permission for directories.

X Execute permission only if *file* is a directory or if at least one execute bit (**S_IXUSR**, **S_IXGRP**, or **S_IXOTH**) is set.

s Set-user-ID or set-group-ID permission.

This permission bit sets the effective user ID or group ID to that of the owner or group owner of *file* whenever the *file* is run. Use this permission setting with the **u** or **g** option to allow temporary or restricted access to files not normally accessible to other users. An **s** appears in the user or group execute position of a long listing (see the reference page for the **ls** command) to show that the file runs with set-user-ID or set-group-ID permission.

Note that the command **chmod o+s** has no effect (the set-user-ID-on-execution and set-group-ID-on-execution bits are not modified).

t Save text permission.

In some versions of the UNIX system, setting this permission bit causes the text segment of a program to remain in virtual memory after its first use. Such systems therefore do not transfer the program code of frequently accessed programs into the paging area.

You can specify this permission for OSS files, but it has no effect. The letter **t** appears in the execute position of the **all others** option to indicate that the file has this bit (the *sticky* bit) set.

If a directory has this bit set, then deletion in it is restricted. An entry in a sticky directory can be removed or renamed by a user only if the user has write permission for the directory and the user is the owner of the file, the owner of the directory, or has appropriate permissions.

The **u**, **g**, and **o** options indicate that *permission* is to be taken from the current mode. Omitting *permission* is useful only with **=** to remove all permissions. For example, entering the following command clears all permission fields for the user and resets them all to those of the group for **file1**:

```
u=g file1
```

All permission bits not explicitly specified are cleared.

You can specify multiple symbolic modes, separated with commas. Do not separate items in this list with spaces. Operations are performed in the order they appear from left to right.

Absolute Mode

Absolute mode lets you use octal notation to set each bit in the permission code. The **chmod** command sets the permissions to the *permission_code* you provide. *permission_code* is constructed by combining with logical OR the following values:

01000000

Sets the trust bit for a TNS/E native loadfile regardless of whether an I/O buffer is in a shared memory segment (the **S_TRUSTSHARED** bit). On a server running an H-series RVU, only a user with appropriate privileges (the super ID) can use this setting. This bit is ignored on a server running a G-series RVU.

00400000
Sets the trust bit for a TNS/E native load file for cases where an I/O buffer is not in a shared memory segment (the **S_TRUST** bit). On a server running an H-series RVU, only a user with appropriate privileges (the super ID) can use this setting. This bit is ignored on a server running a G-series RVU.

00004000
Sets user ID on execution (the **S_ISUID** bit).

00002000
Sets group ID on execution (the **S_ISGID** bit).

00001000
Sets sticky bit:

- Retains memory image after execution (executable file).
- Restricts file removal (directory).

00000400
Permits read by owner (the **S_IRUSR** bit).

00000200
Permits write by owner (the **S_IWUSR** bit).

00000100
Permits execute or search by owner (the **S_IXUSR** bit).

00000040
Permits read by group (the **S_IRGRP** bit).

00000020
Permits write by group (the **S_IWGRP** bit).

00000010
Permits execute or search by group (the **S_IXGRP** bit).

00000004
Permits read by others (the **S_IROTH** bit).

00000002
Permits write by others (the **S_IWOTH** bit).

00000001
Permits execute or search by others (the **S_IXOTH** bit).

Use on Guardian Objects

The **chmod** command does not work on files in the **/G** directory. Once a file has been created in **/G**, you cannot change its permissions with the **chmod** command.

To avoid errors when using the **chmod** command from the root directory (/), use the **-W NOG** flag to prevent an attempt to access files in **/G**.

Environment Variables

The following environment variables affect the execution of the **chmod** command:

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root and a recursive operation occurs in an OSS shell command. Application programs that test this variable

might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE	Omit the **/E** directory.
NOG	Omit the **/G** directory.
NOG:NOE	Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

EXAMPLES

1. To add a type of permission to several files, enter:

   chmod g+w chap1 chap2

   This command adds write permission for group members to the files **chap1** and **chap2**.

2. To make several permission changes at once, enter:

   chmod go-w+x mydir

   This command denies group members and others the permission to create or delete files in the directory **mydir** (**go-w**) and allows them to search **mydir** or use it in a pathname (**go+x**). This command is equivalent to the following command sequence:

   ```
   chmod g-w mydir
   chmod o-w mydir
   chmod g+x mydir
   chmod o+x mydir
   ```

3. To permit only the owner to use a shell procedure as a command, enter:

   chmod u=rwx,go= cmd

   This command gives read, write, and execute permission to the user who owns the file (**u=rwx**). It also denies the group and others the permission to access **cmd** in any way (**go=**).

4. To use set-ID modes, enter:

   chmod ug+s cmd

   When the file **cmd** is executed, this command causes the effective user and group IDs to be set to those that own the file **cmd**. Only the effective IDs associated with the subprocess that runs **cmd** are changed. The effective IDs of the shell session remain unchanged.

   This feature allows you to permit restricted access to important files. Suppose the file **cmd** has the set-user-ID mode enabled and is owned by a user called **dbms**. **dbms** is not actually a person but might be associated with a database management system. The user **betty** does not have permission to access any of **dbms**'s data files. However, she does have permission to execute **cmd**. When she does so, her effective user ID is temporarily changed to **dbms**, so that the **cmd** program can access the data files owned by **dbms**.

This way **betty** can use **cmd** to access the data files, but she cannot accidentally damage them with the standard shell commands.

5. To use the absolute mode form of the **chmod** command, enter:

   chmod 644 text

   This command sets read and write permission for the owner, and it sets read-only mode for the group and all others.

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

RELATED INFORMATION

Commands: **chgrp(1)**, **chown(1)**, **getacl(1)**, **ls(1)**, **setacl(1)**, **sh(1)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

HP extensions to the XPG4 Version 2 specification are:

- To change the file access permissions of a file or directory, the effective user ID of the process must match the super ID or the owner of the file, or the effective user ID or one of the group affiliations for the process must qualify the process for membership in the Safeguard SECURITY-OSS-ADMINISTRATOR group.
- The **-W NOG** and **-W NOE** flags and the **UTILSGE** environment variable are used.

NAME

chown - Changes the owner of files or directories

SYNOPSIS

chown [**-W NOG**] [**-W NOE**] [**-fhR**] *owner*[**:***group*] *file ...*

FLAGS

-f Turns off error reporting.

-h Changes the ownership of a symbolic link instead of the file to which the symbolic link points. When you use this flag, the **chown** command does not affect the file pointed to by the symbolic link. If the symbolic link refers to a directory and the both the -R and -h flags are used, the ownership of the symbolic link is changed, but ownership of the directory to which the symbolic link refers remains unchanged and recursion into that directory does not occur.

-R Causes **chown** to descend recursively through its directory arguments, setting the specified owner (and group, if specified). If **chown** fails to change the owner or group of a particular file, or cannot read or search a particular directory, it continues processing through the hierarchy. When the **chmod** command encounters a symbolic link:

- If the symbolic link refers to a file, the owner of the file is changed.
- If the symbolic link refers to a directory, the owner of the directory is changed, but recursion into the directory does not occur.
- If the symbolic link refers to a directory and the -h option is used, the ownership of the symbolic link is changed, but ownership of the directory to which the symbolic link refers remains unchanged and recursion into the directory does not occur.

HP Extensions

-W NOG Specifies that the **/G** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not **/**, **/E**, or **/E/***system* or when recursion does not occur.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not root or when recursion does not occur.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

The **chown** command changes the owner of the specified files or directories to the specified username or user ID.

A user with super ID priviledges can use the **chown** command to change the owner of a file.

The *owner* argument must be a valid username or a valid numerical user ID. The optional *group* argument must be a valid group name or a valid numerical group ID. Only a process running with an effective user ID equal to the super ID or with a user ID or group affiliation qualifying for

membership in the Safeguard SECURITY-OSS-ADMINISTRATOR group can use the **chown** command to change the owner of a file.

Only a process that has an effective user ID equal to the super ID or to the file owner, or that has an effective user ID or group affiliation qualifying for membership in the Safeguard SECURITY-OSS-ADMINISTRATOR group can use the **chown** command to change the group of a file. However, processes that have an effective user ID equal to the file owner can only change the group of a file to a group to which they belong (their effective group or one of their supplementary groups).

If the **chown** command is invoked by a process whose effective user ID does not equal the super ID, the set-user-ID and set-group-ID bits of the file mode (04000 and 02000, respectively) are cleared.

Access Control Lists (ACLs)

A user can use the ACL for a file to allow or deny specific individuals and groups access to a file. When you use the **chown( )** function with ACLs, if the new owner and/or group of a file have optional ACL entries corresponding to **user:***uid***:***perm* or **group:***gid***:***perm* in the ACL for a file, those entries remain in the ACL but no longer have any effect because they are superseded by the **user::***perm* or **group::***perm* entries in the ACL.

ACLs are not supported for symbolic links.

For more information about ACLs, see the **acl(5)** reference page.

Environment Variables

The following environment variables affect the execution of the **chown** command:

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root and a recursive operation occurs in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE Omit the **/E** directory.

NOG Omit the **/G** directory.

NOG:NOE Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

EXAMPLES

1. To change the owner of the file **program.c**, to **steffan** enter:

   chown steffan program.c

   The user access permissions for **program.c** now apply to **steffan**. As the owner, **steffan** can use the **chmod** command to permit or deny the other users access to **program.c**. (See the **chmod(1)** reference page for details.)

2. To recursively change the owner of all OSS files on the local node to the username GROUP1.USER1 without affecting local Guardian files, enter:

   chown -W NOG -W NOE -R GROUP1.USER1 /

   or

```
export UTILSGE=NOG:NOE
chown -R GROUP1.USER1 /
```

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

RELATED INFORMATION

Commands: **chgrp(1)**, **chmod(1)**, **ls(1)**.

Functions: **chmod(2)**, **setacl(2)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

HP extensions to the XPG4 Version 2 specification are:

- To change the file access permissions of a file or directory, the effective user ID of the process must match the super ID or the owner of the file, or the effective user ID or one of the group affiliations for the process must qualify the process for membership in the Safeguard SECURITY-OSS-ADMINISTRATOR group.
- The **-W NOG** and **-W NOE** flags and the **UTILSGE** environment variable are used.

NAME

cksum - Displays the checksum and byte count of a file

SYNOPSIS

cksum [*file* ...]

DESCRIPTION

The **cksum** command reads the files specified by the *file* argument and calculates a 32-bit checksum Cyclic Redundancy Check (CRC) and the byte count for each file. If no files are specified, the standard input file is read. The checksum, number of bytes, and filename are written to the standard output file. If standard input is used, no pathname is printed.

The **cksum** command can be used to compare a suspect file copied or communicated over noisy transmission lines against an exact copy of a trusted file. The comparison made by the **cksum** command may not be cryptographically secure; however, it is unlikely that an accidentally damaged file will produce the same checksum as the original file.

The checksum of a program can change. The first time a program is executed after the system is cold loaded, external references are resolved. This changes the contents of the program file and hence its contents.

The **cksum** command uses a different algorithm than the **sum** command to calculate the 32-bit checksum CRC. The **cksum** command uses a CRC algorithm based on the Ethernet standard frame check. In addition, the sum block count is an octet count in **cksum**.

The CRC checksum is obtained in the following way:

The encoding is defined by the generating polynomial:

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Mathematically, the CRC value corresponding to a given file is defined by the following procedure:

1. The n bits to be evaluated are considered to be the coefficients of a mod 2 polynomial $M(x)$ of degree n-1. These n bits are the bits from the file, with the most significant bit being the most significant bit of the first octet of the file and the last bit being the least significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral number of octets, followed by one or more octets representing the length of the file as a binary value, least significant octet first. The smallest number of octets capable of representing this integer is used.
2. $M(x)$ is multiplied by x^{32} (that is, shifted left 32 bits) and divided by $G(x)$ using mod 2 division, producing a remainder $R(x)$ of degree less than or equal to 31.
3. The coefficients of $R(x)$ are considered to be a 32-bit sequence.
4. The bit sequence is complemented and the result is the CRC.

EXAMPLES

To display the checksum and the size, in bytes, of **file1** and **file2**, enter:

cksum file1 file2

```
3995432187      1390    file1
3266927833      20912   file2
```

This output shows that the checksum of the **file1** file is 3995432187 and it contains 1390 bytes, and that the checksum of the **file2** file is 3266927833 and it contains 20912 bytes.

RELATED INFORMATION

Commands: **wc(1).**

NAME

clear - Clears terminal screen

SYNOPSIS

clear

DESCRIPTION

The **clear** command clears your terminal screen, if possible. It checks the **ENV** file for the terminal type and then uses the **termcap** database to determine how to perform this operation.

FILES

/etc/termcap

RELATED INFORMATION

Files: **termcap(4)**.

NAME

cmp - Compares two files

SYNOPSIS

cmp [**-l** | **-s**] *file1 file2*

The **cmp** command compares two files.

FLAGS

-l Prints the byte number (in decimal) and the differing bytes (in octal) for each difference.

-s Does not print data for differing files; returns only an exit value.

DESCRIPTION

If the *file1* or *file2* argument is specified as a **-** (dash), the standard input file is used. By default, the **cmp** command prints no information if the files are the same. If the files differ, **cmp** prints the byte and line number where the difference occurred. The **cmp** command also specifies whether one file is an initial subsequence of the other (that is, if the **cmp** command reads an End-of-File character in one file before finding any differences). Normally, you use the **cmp** command to compare nontext files and the **diff** command to compare text files.

Note that bytes and lines reported by **cmp** are numbered from 1. The first differing byte number is from *file1*, and the second is from *file2*.

EXAMPLES

1. To determine whether two files are identical, enter:

   cmp prog.o.bak prog.o

   The preceding command compares the files **prog.o.bak** and **prog.o**. If the files are identical, a message is not displayed. If the files differ, the location of the first difference is displayed as follows:

   prog.o.bak prog.o differ: char 5, line 1

   If the message `cmp: EOF on prog.o.bak` is displayed, the first part of **prog.o** is identical to **prog.o.bak**, but there is additional data in **prog.o**. If the message `cmp: EOF on prog.o` is displayed, **prog.o.bak** that is identical to **prog.o** but that also contains additional data.

2. To display each pair of bytes that differ, enter:

   cmp -l prog.o.bak prog.o

   This command compares the files and then displays the byte number (in decimal) and the differing bytes (in octal) for each difference. For example, if the fifth byte is octal 101 in file **prog.o.bak** and 141 in file **prog.o**, then the **cmp** command displays:

   ```
   5 101 141
   .
   .
   ```

EXIT VALUES

Exit value 0 (zero) is returned for identical files, 1 for differing files, and 2 for an inaccessible file, a missing argument, or some other error.

RELATED INFORMATION

Commands: **comm(1), diff(1).**

NAME

cobol - Compiles COBOL85 TNS programs

SYNOPSIS

cobol [**-c**] [**-g**] [**-L** *directory*] ... [**-O**[*optlevel*]]
[**-o** *outfile*] [**-s**]
[**-Waxcel[**="*args*"]]
[**-WBdynamic**]
[**-WBstatic**]
[**-Wbind**[="*args*"]]
[**-Wcobol**="*args*"]
[**-Wcopylib**=*pathname*]
[**-Wnobind**]
[**-Wrunlib**=*pathname*]
[**-Wsql[**="*args*"]]
[**-Wverbose**]
operand ...

FLAGS

-c Performs compilation of the specified source files but supresses the binding phase. This flag does not delete any object files that are produced.

-g Produces symbols information for symbolic debugging in the object or executable files. This is equivalent to specifying the SYMBOLS and INSPECT directives to the COBOL85 compiler.

-L *directory*
Changes the algorithm for searching the libraries named in the **-l** arguments to look in the directory named by the *directory* pathname before looking in the default directories **/lib, /nonnative/usr/lib, /usr/lib,** and **/usr/local/lib**. Directories named in **-L** options are searched in the order specified. At least ten instances of this option are supported in a single **cobol** command line. If a directory specified by **-L** contains files named **libc.a, libm.a, libl.a,** or **liby.a**, these files are used as libraries instead of the libraries in the default directories.

-o *outfile* Uses the pathname *outfile* instead of the default pathname **a.out** for the executable file produced. If **-o** is used with the **-c** flag, *outfile* is used to name the object file instead of the standard naming convention.

-O [*optlevel*]
Specifies the optimization level to be used for the program file. A **0** *optlevel* argument specifies an `OPTIMIZE 0` COBOL85 compiler directive. A **1** *optlevel* argument specifies an `OPTIMIZE 1` COBOL85 compiler directive. A **2** *optlevel* argument or no argument specifies an `OPTIMIZE 2` COBOL85 compiler directive and processes the program file with the Accelerator, **axcel**. If a **-O** flag is not specified, an `OPTIMIZE 1` COBOL85 compiler directive is specified.

-s Strips symbolic and other information not required for proper execution from object and executable files. If both the **-g** and **-s** flags are used, symbolic information is kept in the object files but is stripped from the executable file. Do not specify the **-s** and
-Wsql flags in the same **cobol** invocation because the NonStop SQL/MP compiler requires the symbols region to be present. Strip the file in a separate **cobol** invocation after the SQL compilation.

-Waxcel[="*args*"]
Invokes the Accelerator, **axcel**, and passes to it the argument string enclosed in quotation marks. Refer to the *Accelerator Manual* for a description of the arguments that can be passed to the Accelerator.

-WBdynamic
Specifies that dynamic binding is performed. In dynamic binding, the Binder resolves references to library functions using a shared runtime library, such as **lib.so**, but does not bind the functions into the program. Final resolution of references is performed at run time. Dynamic binding is the default action.

-WBstatic
Specifies that static binding is performed. In static binding, the Binder resolves references to library functions using a user library, such as **libc.a**, and binds the functions into the program. Dynamic binding is the default action.

-Wbind[="*args*"]
Invokes the Binder and passes to it the argument string enclosed in quotation marks. Dynamic binding is the default action. Refer to the *Binder Manual* for a description of the arguments that can be passed to the Binder.

-Wcobol="*args*"
Passes to the COBOL85 compiler the directives in the argument string enclosed in quotation marks. This string follows any directives generated by other flags.

-Wcopylib=*pathname*
Specifies *pathname* as the source file to use as the default COPY library for any COPY statement in the source program that does not specify a library.

-Wnobind
Suppresses invocation of the Binder. This flag is used when the Accelerator or NonStop SQL/MP compiler is to be invoked without having to rebind the program file.

-Wrunlib = *pathname*
Specifies a user library to be used at run time. The user library must be specified as a member of the Guardian file system in an OSS pathname of the form */**G**/volume/subvolume/file-identifier*.

-Wsql[="*args*"]
Invokes the NonStop SQL/MP compiler, **sqlcomp**, and passes to it the argument string enclosed in quotation marks. Refer to the *NonStop SQL/MP Programming Manual for COBOL85* for a description of the arguments that can be passed to the NonStop SQL/MP compiler. Do not specify the **-s** and **-Wsql** flags in the same **cobol** invocation because the NonStop SQL/MP compiler requires the symbols region to be present. Strip the file in a separate **cobol** invocation after the SQL compilation.

-Wverbose
Displays more detailed information during the program generation process from the COBOL85 compiler, Binder, Accelerator, or NonStop SQL/MP compiler.

Multiple instances of the **-L** and **-l** flags can be specified.

Do not include a space before or after the "=" (equal sign).

The position of **-l** *library* arguments within a list of flags affects the order in which libraries are searched.

The order of specifying the **-l** and **-L** flags is significant.

DESCRIPTION

The **cobol** utility is the interface to the COBOL85 compilation system; it accepts source code conforming to the ISO COBOL85 standard. The system conceptually consists of a COBOL85 compiler and Binder, with additional program components supporting NonStop SQL/MP compilation (**sqlcomp**), and object code acceleration (**axcel**). The files specified in the *operand* list are operated on by the appropriate program components of the compilation system, depending on the command line flags and the type of file operands.

If the **-c** flag is specified, for all pathname operands of the form *file***.cbl**, the files *$(basename pathname.c).o* are created as the result of successful compilation.

If **-c** is not specified, the object files created after successful compilation are combined into a program file by the bind operation. Object files created are not deleted after successful generation of the executable program file.

If no flags are present to prevent program binding (such as **-c** or **-Wnobind**, and if all file operands compile and link without error, the resulting executable file is written according to the **-o** *outfile* flag (if present) or to the file **a.out**.

The executable file is created according to OSS file creation rules, except that the file permissions are set to **S_IRWXO | S_IRWXG | S_IRWXU** and the bits specified by the umask of the process are cleared.

HP Extensions

The **-W** flags are HP-specific flags supporting the HP compilation environment. The argument strings following these flags are passed to the program components unchanged, along with default argument strings and argument strings corresponding to **cobol** command line flags meaningful to the program components. Do not specify conflicting instructions in **-W** flag argument strings or **cobol** command line flags. The results of conflicting instructions are undefined.

Operands

An *operand* is either a pathname or in the form **-l** *library*. At least one operand of the pathname form must be specified. The following operands are supported:

*file***.cbl** A COBOL85 language source file to be compiled and optionally linked.

*file***.cob** A COBOL85 language source file to be compiled and optionally linked.

*file***.a** A library of object files typically produced by the **ar** command, and passed directly to the Binder.

*file***.o** An object file produced by the command **cobol -c** and passed directly to the Binder.

*file***.so** A shared runtime library produced by the Binder. The shared runtime library is used by the Binder to resolve external references.

-l *library* In the static binding mode, search for the library named **lib***library***.a**. In the dynamic binding mode, search for the library named **lib***library***.so**. If lib*library***.so** is not found, **lib***library***.a** is used.

A library is searched when its name is encountered, so the placement of **-l** is significant.

Input Files

Input files are one of the following:

- A text file containing a COBOL85 language source program
- An object file in the format produced by the command **cobol -c**
- A library of object files in the format produced by archiving zero or more object files using the **ar** command
- A library of object files produced by the Binder
- An executable file produced by the Binder

Output Files

Output files are object files or executable files (or both).

Standard Output

The standard output is a text file that contains the compiler listing, if generated.

Standard Error

Standard error is used for diagnostic and informational messages. If more than one file operand is specified, for each such file, `"%s: \n",<file>` might be written. These messages precede the processing of each input file.

Environment Variables

The environment variables listed below affect the execution of **cobol**. The **cobol** utility and its program components do not support locale variables.

AXCEL Determines the pathname of the Accelerator invoked by **cobol**. By default, the program **axcel** in the directory **/G/system/system** is used.

BIND Determines the pathname of the Binder invoked by **cobol**. By default, the program **bind** in the directory **/G/system/system** is used.

SQLCOMP

Determines the pathname of the NonStop SQL/MP compiler invoked by **cobol**. By default, the program **sqlcomp** in the directory **/G/system/system** is used.

Operations

cobol is the driver program of the COBOL85 language compilation system. It accepts COBOL85 source files and generates binary files executable in the OSS environment. Depending on user-specified flags, input file operands, and completion code at each stage, **cobol** invokes one or more components of the compilation system. A program-generation process can involve the following steps:

1. Each COBOL85 source module is compiled into an object module by the COBOL85 compiler.

2. Object modules are bound together, with additional library routines if necessary, by the Binder into a single program file, unless flag **-c** or **-Wnobind** is specified. In this case, processing stops after compilation of source modules.
3. If the program is to be run on TNS/R systems, the user can accelerate the program with the Accelerator to obtain maximum performance.
4. If the program contains embedded SQL statements, it needs to be compiled by the NonStop SQL/MP compiler as the final step.

With the exception of the **cobol** utility, all components are invoked as Guardian processes. **cobol** provides terminal emulation (input from and output to) the controlling terminal for these processes, if necessary. For Guardian processes that do not use **cobol** for terminal emulation or are not interactive, the stop/continue type of OSS job control is not supported.

The Binder is invoked with a command file containing the following commands:

```
MODE NOUPSHIFT
SELECT CHECK PARAMETER STRONG
SELECT FILESYS OSS
SET SYSTYPE OSS
ADD * FROM <object created by COBOL85 compiler or input object
file>
SELECT SEARCH <library specified by -l flag>
SELECT IMPORT LIBRARY <shared runtime library if found in
dynamic binding mode>
SET HIGHPIN ON
SET HIGHREQUESTERS ON
SELECT LIST * OFF
SELECT RUNNABLE OBJECT ON
<SET INSPECT ON if -g flag>
SET SAVEABEND ON
SET HEAP 32 PAGES
<argument string from -Wbind options>
SELECT SEARCH <standard C library, libc.a or libc.so>
BUILD <filename from -o option or a.out>!
<STRIP filename if -s option>
```

Standard Libraries

The following libraries are available for COBOL85 programs that call OSS functions.

-l c Contains all library functions specified in the POSIX.1 specification, except for those functions listed as residing in the **math.h** header file. The presence of this operand is not required to cause a search of this library if the target execution environment is the OSS environment.

-l C Contains the C++ runtime library. Bind this library before the standard C runtime library.

-l m Contains all functions referenced in the **math.h** header file.

-l l Contains all functions required by the C language output of **lex** utility that are not made available through the **-l c** operand.

-l y Contains all functions required by the C language output of **yacc** utility that are not made available through the **-l c** operand.

-l gwc Contains the C runtime library for the Guardian environment, using the wide data-model.

In the absence of flags that inhibit invocation of the Binder, such as **-c**, **-E**, or **Wno-bind**, **cobol** passes an **-l c** operand to the Binder as the last **-l** operand, causing the standard C library to be searched after all other object files and libraries are loaded.

Handling of Guardian Libraries

The **cobol** command tries to determine the type of a Guardian file from its content. Text input files are assumed to be COBOL85 language source files.

The object file for a Guardian source file is named according to the following convention:

- The object file uses the same filename with the letter "**o**" appended to the end. For example, the object file `xyzo` is generated from the source file `xyz`.
- If a valid filename cannot be generated using the rule above, a filename returned from **tmpnam** is used, and a message informs the user of the new filename.

Libraries residing in Guardian directories cannot be specified as **-l** operands because of the naming convention. They can be specified in the desired order with **-Wbind** flags.

The default executable file in a Guardian file system is **aout** in the directory from which **cobol** is invoked.

Differences Between Static and Dynamic Binding

In static binding, the Binder resolves references to library functions by binding into the program the functions from a user library, such as **libc.a**. In dynamic binding, the Binder resolves references to library functions by using information found in a shared runtime library, such as **libc.so**. Final resolution of references are not done until run time. Dynamic binding produces smaller program files and uses fewer system resources than static binding. By default, **cobol** performs dynamic binding.

A shared library must have a file extension of **.so**. Such a file can either be an actual shared library file created by the Binder or an OSS text file containing only one line of text in the form "**SRL=***srlpath*" where *srlpath* is the OSS pathname of the actual shared library file.

In dynamic binding, if a library is specified by an operand of the form **l***library*, **cobol** searches for a file **lib***library***.so** in a directory in the library search paths. If the file is found in the directory, **cobol** issues the Binder command `SET IMPORT LIBRARY` *srlpath*, where *srlpath* is the pathname of the actual shared library file. If the file is not found in the directory, **cobol** searches for a file **lib***library***.a** in the same directory. If a file is found, **cobol** issues the Binder command `SELECT SEARCH` *libpath* to perform a static binding of the library. If neither file is found, **cobol** repeats the search for the library in the next directory in the library search paths.

In static binding, **cobol** only searches for **lib***library***.a** from directories in the library search paths.

Static libraries can be used for dynamic binding. The Binder resolves external references using all the specified static libraries before using the shared runtime library. A program can only have one shared runtime library. **cobol** issues warnings if more than one shared runtime library is specified.

The standard C runtime library, **libc.a** or **libc.so** is used in dynamic binding only if no other

shared runtime libraries are used.

By default, the shared runtime library used during binding (the model library) is the shared runtime library used during execution (the runtime library), unless a different library is specified by the **-Wrunlib** flag.

The shared runtime library must be in the Guardian file system name space. If a shared runtime library is in the OSS file system name space, you must supply a library in the Guardian name space either using the **-Wrunlib** flag during program generation or through the LIB option of the TACL RUN command during program execution.

Refer to the *Binder Manual* for more details on using shared runtime libraries.

Using the c89 and cobol Utilities

OSS COBOL85 programs can contain COBOL85 modules and C modules. Compile COBOL85 modules using the **cobol** utility and C modules using the **c89** utility. To produce a program containing COBOL85 and C modules, first compile all the modules written in either COBOL85 or C. You can also bind these modules together or with other libraries at this time, but do not accelerate or SQL-compile the modules. After you have compiled all the modules of one language, compile the modules written in the other language, specifying any necessary binding, accelerating, or SQL-compiling options.

For example, to produce an executable object file made up of COBOL85 modules `cobol1.cbl` and `cobol2.cbl` and C modules `c1.c` and `c2.c`, you can first run the C compiler using the **c89** utility with:

```
c89 -c -o cprog.o c1.c c2.c
```

This directs **c89** to compile the two modules but not to bind them. The output object file is `cprog.o`.

You can then invoke the **cobol** utility to compile the two COBOL85 modules, bind the COBOL85 compiler output with the previously produced C object file and the standard C library, and run the Accelerator to produce the executable object `myprog` with:

```
cobol -o myprog -O cprog.o cobol1.cbl cobol2.cbl
```

Refer to the *C/C++ Programmer's Guide* and the *Open System Services Programmer's Guide* for details on writing and compiling C programs in the OSS environment.

EXAMPLES

1. The command

   ```
   cobol test1.cbl
   ```

   compiles the source file `test1.cbl` and binds the object file into a program file `a.out`.

2. The command

   ```
   cobol -c test1.cbl
   ```

   compiles the source file `test1.cbl` into an object file `test1.o`.

3. The command

   ```
   cobol -g -o test2 x.cbl y.cbl z.cbl
   ```

   compiles source files `x.cbl, y.cbl,` and `z.cbl` and binds the object files into a program file `test2.` Symbolic information is generated by the compiler and retained by the Binder for debugging.

4. The command

   ```
   cobol -o test3 -O 2 -WBstatic x1.cbl x2.o x3.cbl -l mylib
   ```

   compiles source files `x1.cbl` and `x3.cbl` and binds the object files together with `x2.o` into program file `test3`. Static binding has been specified, so the Binder tries to resolve references using the library `mylib.a` before using the standard library **libc.a**. The **-O 2** flag causes optimization during compilation and invocation of the Accelerator on the program file.

5. The command

   ```
   cobol -o xyz -Wbind="set heap 64" -Wsql x.o y.o z.o
   ```

   binds the object files `x.o, y.o,` and `z.o` into a program file `xyz`. The Binder sets the heap size of the program to 64 pages. The NonStop SQL/MP compiler, **sqlcomp**, is then invoked on `xyz` to compile `xyz`.

6. The command

   ```
   cobol -Wnobind -Waxcel xyz
   ```

   invokes the Accelerator, **axcel**, on object or program file `xyz` without going through the binding process.

7. The command

   ```
   cobol -Wnobind -Wsql="catalog \$abc.def" xyz
   ```

   invokes the NonStop SQL/MP compiler, **sqlcomp**, on program file `xyz` without going through the binding process. In addition to the input filename `xyz`, the catalog option is passed to the NonStop SQL/MP compiler.

8. The command

   ```
   cobol -o testprog -L .  -L /usr/test/lib testprog.cbl -l tdm
   ```

   compiles the COBOL85 language source program `testprog.cbl` and binds the object file with the library specified in the **-l** operand. It also binds the object file with a shared runtime library, if found. If a shared runtime library is not found, it uses the standard C runtime library. The Binder produces a program file named `testprog`. By default, dynamic linking is selected. **cobol** searches for the library specified by the **-l** flags and the **-l tdm** operand in the following order and selects the first one found:

libtdm.so	in the current directory (**-L .**)
libtdm.a	in the current directory (**-L .**)
libtdm.so	in **/usr/test/lib** (**-L /usr/test/lib**)
libtdm.a	in **/usr/test/lib** (**-L /usr/test/lib**)
libtdm.so	in **/lib** (by default)
libtdm.a	in **/lib** (by default)
libtdm.so	in **/nonnative/usr/lib** (by default)
libtdm.a	in **/nonnative/usr/lib** (by default)
libtdm.so	in **/usr/lib** (by default)

libtdm.a	in **/usr/lib** (by default)
libtdm.so	in **/usr/local/lib** (by default)
libtdm.a	in **/usr/local/lib** (by default)

EXIT VALUES

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

DIAGNOSTICS

If **cobol** encounters a compilation error that causes an object file to not be created, it writes a diagnostic message to standard error and continues to compile other source code operands. However, it does not perform program binding and returns a nonzero exit status. If the binding is unsuccessful, **cobol** writes a diagnostic message to standard error and returns a nonzero exit status.

RELATED INFORMATION

Commands: **ar(1)**, **c89(1)**, **ecobol(1)**, **nmcobol(1)**, **strip(1)**.

STANDARDS CONFORMANCE

The **-W** flags are HP extensions.

NAME

comm - Compares two sorted files

SYNOPSIS

comm [**-123**] *file1 file2*

The **comm** command reads *file1* and *file2* and writes three columns to the standard output file, showing which lines are common to both files and which are unique to each file.

FLAGS

-1 Suppresses output of the first column (lines only in *file1*).

-2 Suppresses output of the second column (lines only in *file2*).

-3 Suppresses output of the third column (lines common to *file1* and *file2*).

The command **comm -123** produces no output.

DESCRIPTION

The leftmost column of the standard output file includes lines that are only in *file1*. The middle column includes lines that are only in *file2*. The rightmost column includes lines that are in both *file1* and *file2*. Columns are separated by tab characters.

If you specify **-** (dash) in place of one of the filenames, the **comm** command reads the standard input file.

The *file1* and *file2* arguments should be sorted according to the collating sequence specified by the **LC_COLLATE** environment variable (see the reference page for the **sort** command), or the **comm** command fails.

EXAMPLES

In the following examples, file **file1** contains the following sorted list of North American cities:

```
Anaheim
Baltimore
Boston
Chicago
Cleveland
Dallas
Detroit
Kansas City
Milwaukee
Minneapolis
New York
Oakland
Seattle
Toronto
```

File **file2** contains this sorted list:

```
Atlanta
Chicago
Cincinnati
Denver
Houston
Los Angeles
Miami
Montreal
New York
Philadelphia
Pittsburgh
San Diego
San Francisco
St. Louis
```

1. To display the lines unique to each file and common to the two files, enter:

   comm file1 file2

   This command results in the following output:

```
Anaheim
        Atlanta
Baltimore
Boston
                Chicago
        Cincinnati
Cleveland
Dallas
        Denver
Detroit
        Houston
Kansas City
        Los Angeles
        Miami
Milwaukee
Minneapolis
        Montreal
                New York
Oakland
        Philadelphia
        Pittsburgh
        San Diego
        San Francisco
Seattle
        St. Louis
Toronto
```

   The leftmost column contains lines only in **file1**, the middle column contains lines only in **file2**, and the rightmost column contains lines common to both files.

2. To display any one or two of the three output columns, include the appropriate flags to suppress the columns you do not want. For example, the following command displays lines that appear only in **file1** and only in **file2**:

   comm -3 file1 file2

```
Anaheim
        Atlanta
Baltimore
Boston
        Cincinnati
Cleveland
Dallas
        Denver
Detroit
        Houston
Kansas City
        Los Angeles
        Miami
Milwaukee
Minneapolis
        Montreal
Oakland
        Philadelphia
        Pittsburgh
        San Diego
        San Francisco
Seattle
        St. Louis
Toronto
```

3. The following command displays lines that appear only in **file2**:

   comm -13 file1 file2

```
Atlanta
Cincinnati
Denver
Houston
Los Angeles
Miami
Montreal
Philadelphia
Pittsburgh
San Diego
San Francisco
St. Louis
```

4. The following command displays only those lines that appear in both **file1** and **file2**:

   comm -12 file1 file2

```
Chicago
New York
```

RELATED INFORMATION

Commands: **cmp(1), diff(1), sort(1), uniq(1).**

Files: locale(4).

NAME

command - Treats command arguments as a simple command

SYNOPSIS

command [**-p**] *command_name* [*argument* ...]

The **command** command causes the shell to treat the arguments to **command** as a simple command and suppresses the default shell function lookup.

FLAGS

-p Performs the command search using a default value for the path that is guaranteed to find all of the standard utilities.

DESCRIPTION

The **command** command allows you to run the following commands:

- User-defined commands whose names correspond to shell built-in commands.
- System commands whose names correspond to shell built-in commands.

The *command_name* argument specifies the name of a utility. The one or more optional arguments to *command_name* specify strings treated as arguments to the specified utility.

EXAMPLES

To ensure execution of the simple command **pwd** instead of the **pwd** shell built-in command, enter:

```
command -p pwd
```

The preceding command displays the full pathname of the current directory and does not perpetuate a display of the current directory location created by links, as the shell built-in command might do.

EXIT VALUES

127 An error occurred in the **command** command, or the utility specified by the *command_name* argument could not be invoked.

If no error occurs, the exit status of **command** is that of the command invoked by the arguments to **command**.

NAME

compress - Compresses or decompresses data

SYNOPSIS

compress [**-CdfFnqvV**] [**-b** *maxbits*] [*file* ...]

compress [**-cCdfFnqvV**] [**-b** *maxbits*] [*file*]

FLAGS

-b *maxbits*

Specifies the maximum number of bits to use to replace common substrings in the file. The default value for the *maxbits* argument is 16, with values of 9 through 16 acceptable. First, the algorithm uses 9-bit codes 257 through 512. Then it uses 10-bit codes, continuing until the *maxbits* limit is reached. (This action is not permitted with the **uncompress** command.)

After the *maxbits* limit is reached, the **compress** command periodically checks the compression ratio. If the ratio increases, **compress** continues to use the existing code dictionary. However, if the compression ratio decreases, **compress** discards the table of substrings and rebuilds it from the beginning. This strategy allows the algorithm to adapt to the next block of the file.

-c Makes the **compress** command write to the standard output file; no input files are changed and no **.Z** file is created. The nondestructive behavior of the **zcat** command is identical to that of **compress -cd**.

-C Produces output compatible with BSD **compress** revision 2.0.

-d Specifies decompression (uncompressing) the specified file.

-f or **-F** For both compressing and decompressing files (except when this command is run in the background under the **/usr/bin/sh** file), this flag suppresses the prompt as to whether an existing file given by the *file* operand should be overwritten. For compressing files only, this flag forces the compression of *file* even if the file does not actually get smaller or the corresponding *file***.Z** already exists.

-n Specifies that no header is added or expected. This flag might be useful for decompressing old files.

-q Specifies a quiet operation. This flag is the default action; diagnostics are displayed only if the **-v** flag is specified.

-v For a compression operation, prints the percentage reduction of each file. For a decompression operation, this flag prints the percentage expansion of each file.

-V Displays the version number and any options with which the program was compiled.

DESCRIPTION

The **compress** command reduces the size of the specified files using adaptive Lempel-Ziv coding.

For **compress**, whenever possible, each file is replaced by one with the extension **.Z** that has the same ownership modes, access, and modification times. Note that the filename extension used by **compress** is different from that used by **pack** command compression (**.z** extension). If no files

are specified or if the *file* operand is a **-** (dash), the standard input file is compressed and written to the standard output file. If appending the **.Z** extension to the filename would make the name exceed **NAME_MAX** bytes, the **compress** command fails.

Compressed files can be restored to their original form using the **uncompress** or **zcat** command or using the **compress** command with the **-d** flag.

The amount of compression obtained depends on the size of the input file, the number of bits per code, and the distribution of common substrings. Typically, files containing source code or plain text are reduced by 50 to 60 percent.

If the file has an access control list (ACL), the ACL is preserved when the file is compressed. For more information about ACLs, see the **acl(5)** reference page.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

To compress the file **folder** and print the percentage reduction, enter:

```
compress -v folder
```

The system responds with a message like:

```
folder: Compression: 43.94% -- replaced with folder.Z
```

EXIT VALUES

The **compress** command returns the following exit values:

0 (zero)	The **compress** utility successfully finished its processing.
1	An error occurred. The input file remains unmodified.
2	One or more files were not compressed because they would have increased in size (and the **-f** flag was not specified).
>2	An error occurred. The input file remains unmodified.

RELATED INFORMATION

Commands: **uncompress(1)**, **zcat(1)**.

STANDARDS CONFORMANCE

The following features are HP extensions to the XPG4 Version 2 specification:

- The **-C**, **-F**, **-q**, and **-V** flags are supported.

NAME

continue - Resumes a **for, while, until,** or **select** loop

SYNOPSIS

continue [*n*]

DESCRIPTION

The **continue** command resumes the next iteration of the enclosing **for, while, until,** or **select** loop. If *n* is specified, resumes at the *n*th enclosing level.

EXAMPLES

1. The following shell script demonstrates the use of the **continue** command to resume a loop:

```
for x in 1 2 3 4 5
do
if [ $x != 3 ]
then
print $x
else
continue
fi
done
```

EXIT VALUES

If an invalid argument is specified, the exit value is greater than 0 (zero).

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **continue** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **break**(1), **sh**(1)..

NAME

cp - Copies files

SYNOPSIS

cp [**-fip**] [**-W clobber**] *source_file target_file*

cp [**-fip**] [**-W clobber**] *source_file ... destination_directory*

cp [**-fLip**] [**-r** | **-R**] [**-W clobber**] [**-W NOG**] [**-W NOE**] [*source_file* | *source_directory*] *... destination_directory*

FLAGS

-f Tries to unlink the destination file and proceed if a file descriptor for a destination file cannot be obtained.

-L Forces **cp** to follow symbolic links; useful with the **-R** flag, which does not follow symbolic links by default.

-i Requests confirmation when the copy operation requires a destination file to be overwritten by a source file. An answer beginning with **y**, or the locale's equivalent of **y**, causes the **cp** command to overwrite the destination file and continue. Any other answer prevents overwriting of the file.

-p Preserves for the copy the modification time, access time, file mode, user ID, and group ID of the source, as allowed by permissions. If the user ID and group ID cannot be preserved, no error message is displayed and the exit value is not altered. If the original is set-user ID or set-group ID, and either the user ID or the group ID cannot be preserved, the set-user-ID and set-group-ID bits are not preserved in the copy's permissions.

The **-p** flag cannot preserve owner ID or group ID information when copying a file to a remote system.

-r Copies the source directory and all of the subdirectories within it. This flag is identical to the **-R** flag except that special files are not treated differently from regular files (that is, **-r** follows symbolic links).

If you are using the **-r** flag to copy the contents of one directory to another, and if the source directory contains subdirectories that do not exist in the destination directory, the subdirectories are created. Created directories have the same mode as the corresponding source directory, unmodified by the process's file mode creation mask (**umask**).

This flag has been proposed for obsolescence in a future revision of the Single UNIX standard. Use the **-R** flag instead to ensure portability.

-R Copies the source directory and all of the subdirectories within it. Special file types, such as symbolic links and block and character devices, are re-created instead of copied.

If you are using the **-R** flag to copy the contents of one directory to another, and if the source directory contains subdirectories that do not exist in the destination directory, the subdirectories are created. Created directories have the same mode as the corresponding source directory, unmodified by the process's file mode creation mask (**umask**).

HP Extensions

-L Overrides the default behavior of the **-R** flag so that **cp** follows symbolic links.

-W clobber Allows the existing target Guardian file to be overwritten using the data from the source file. This operation retains the following attributes of the target file:

- The file code
- The type of unstructured file (odd unstructured or even unstructured)
- The buffering attribute (BUFFERED or NOBUFFERED)
- The security attributes (RWEP)

You must set any other file attributes, as needed.

If the target file is protected by Safeguard and you use the **-W clobber** flag, the Safeguard protection might be lost or changed. The **cp** command issues a warning when used with the **-W clobber** flag on a file protected by Safeguard.

When the **-W clobber** flag is used, the **-p**, **-W NOG**, and **-W NOE** flags and the **UTILSGE** environment variable are ignored.

For more information about using this utility on Guardian files, see "Use on Guardian Objects."

-W NOG Specifies that the **/G** directory should be omitted when the initial directory is root (/) and a recursive flag (**-R** or **-r**) is used. This flag is ignored when the initial directory is not **/**, **/E**, or **/E**/*system* or when recursion does not occur.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root (/) and a recursive flag (**-R** or **-r**) is used. This flag is ignored when the initial directory is not root (/) or when recursion does not occur.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

The **cp** command copies a source file or directory to a destination file or directory

In the first form given in **SYNOPSIS**, the **cp** command copies the contents of *source_file* into *target_file*. If *target_file* exists, its contents are overwritten provided the user has write permission on *target_file*'s parent directory.

In the second form, two or more files are copied to the destination directory.

In the third form, source directories, including all subdirectories and files within them, are copied to the destination.

A destination directory must exist in order for a source directory or source file to be copied to it.

Appropriate permissions are always required for file creation or overwriting.

If the destination directory exists, the source directory or files are copied into the destination with their original names.

If the destination exists, and both the source and destination are files, the source file overwrites

the destination file, permissions allowing.

If the destination does not exist and the source is a file, the destination is considered a file, and the source is copied to it.

If the source and destination are either both directories or both files and they have the same name, an error occurs and the copy fails.

Use With Access Control Lists (ACLs)

If the file being copied has an ACL, typically the new file created by the **cp** command retains the ACL. However, if the **cp** command is executed remotely from a system that does not support OSS ACLs, then the ACL for the file is not copied to the destination file. If destination fileset supports OSS ACLs, the destination file might inherit ACL entries from the parent directory of the destination file. If the destination fileset does not support OSS ACLs, the destination files do not have ACLs. For more information about ACLs, see the **acl(5)** reference page.

For G-series RVUs, H06.19 and earlier H-series RVUs, or J06.08 and earlier J-series RVUs, the OSS Network File System (NFS) cannot access OSS objects that have OSS ACLs that contain optional ACL entries.

For J06.09 and later J-series RVUs and H06.20 and later H-series RVUs, access by the OSS Network File System (NFS) to OSS objects that have OSS ACLs that contain optional ACL entries can be allowed, depending upon the NFSPERMMAP attribute value for the fileset that contains the object, however:

- The **cp** command does not copy any ACLs associated with the object.
- The permissions used when an object is created by the **cp** command depend on the value of the NFSPERMMAP attribute for the fileset on the system that contained the original file.

For more information about NFS and ACLs, see the **acl(5)** reference page.

This table describes the impact of ACLs on the permissions used when a new file or directory is created by the **cp** command and the **cp** command is not issued from NFS. Cases not included in the table represent impossible situations.

cp Command Supports ACLs	Source Fileset Supports ACLs	Source File/Dir Has Opt. ACLs	Dest. Filesset Supports ACLs	Impact of ACLs on Permissions of New File/Dir
Y/N	Y/N	N	N	None
Y/N	Y/N	N	Y	See Note 2
Y/N	Y	Y	N	See Notes 1 and 3
N	Y	Y	Y	See Notes 1 and 2
Y	Y	Y	Y	See Note 4

Note 1: The optional ACLs for the source file or directory are not copied to the destination file or directory.

Note 2: If the destination parent directory has default ACL entries, those default ACL entries are inherited by the new file or directory (see the **acl(5)** reference page).

Note 3: If you use the **-p** flag, the file permissions are copied to the destination file or directory and the class entry permissions in the ACL are used for the destination file or directory group permissions. Otherwise, the permissions for the destination file or directory are set using the non-ACL

descriptions.

Note 4: If you use the **-p** flag, all ACL entries for the source file or directory are copied to the destination file or directory. Otherwise, the permissions for the destination file or directory are set using the non-ACL descriptions.

Use on Guardian Objects

Specify Guardian files with the **/G** pathname convention.

Only unstructured Guardian files are supported. If both the source and destination are Guardian files, the file attributes specific to Guardian (such as extent sizes, file code, and file type) are preserved. Thus if a type 101 EDIT Guardian file is copied within the Guardian volume, the target file is also a type 101 EDIT file, with all the line number information preserved. In addition, if the **-p** flag is specified, other Guardian file attributes (such as user ID, file security, and timestamps) are preserved in the same manner they are preserved with the TACL command FUP DUP.

If you are copying a Guardian file to the OSS file hierarchy, only the content of the Guardian file is copied: the Guardian file attributes are not preserved. Likewise, if you are copying an OSS file to the Guardian file hierarchy, the target file is created as an unstructured Guardian file. Thus if you copy a Guardian type 101 EDIT file to the OSS file hierarchy and then copy it back to the Guardian file hierarchy, it will no longer be a Guardian type 101 EDIT file.

If a source file and destination file are determined to be the same, a diagnostic message is written to the standard error file.

Because of the differences between the Guardian and OSS file systems, the following anomalies can occur when OSS files are copied to a Guardian destination.

- A destination pathname can contain illegal **/G** filename characters, even after it has been transformed into a **/G** pathname. As a result, the destination file cannot be created on the Guardian destination, and the copy operation fails.
- A destination pathname might be transformed into a **/G** pathname that is quite different from its original pathname. For example, the OSS filename **abcde.fghi** is converted into the **/G** filename **ABCDEFGH**. In this example, the copy operation succeeds but the name of the newly created destination file might cause confusion if it is not anticipated.
- OSS filenames that are similar to each other might be converted into the same **/G** filename when copied to the Guardian file system.
- If a source directory contains more than two levels of directories (the maximum that the Guardian file system currently supports), the entire source subtree cannot be copied completely to the Guardian target; only the directories at supported levels are copied.

Environment Variables

The following environment variables affect the execution of the **cp** command:

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root (/) and a recursive option (**-r** or **-R**) is used in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE Omit the **/E** directory.

NOG Omit the **/G** directory.

NOG:NOE Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

EXAMPLES

a. To copy one file to another, enter:

cp file1 file2

If the **file2** file exists (and is writable), its contents are replaced by the contents of the **file1** file.

b. To copy files to a directory, enter:

cp file1 file2 dir1

The directory **dir1** must exist.

c. To copy all files in a directory and preserve their modification times, enter:

cp -p dir1/* dir2

d. To copy a directory tree to another directory, enter:

cp -R dir1 dir2

The **dir1** directory tree is created in the directory **dir2**.

e. To copy all files with a name ending in the letter **c** from the current OSS directory to Guardian $VOL.SUBVOL, enter:

cp . /*c /G/vol/subvol

The Guardian filenames resulting from this copy are truncated to the first eight characters. Characters in the OSS filenames that are illegal to the Guardian file system are skipped during the renaming process. The specified target volume must exist in the Guardian environment.

f. To copy a Guardian file to an OSS directory, enter:

cp /G/vol/subvol/file /usr/bin

g. To copy the files in an OSS directory named **/usr/u/rose** to the Guardian volume **/G/vol**, enter:

cp -r /usr/u/rose /G/vol

If **/usr/u/rose** contains **file1** (a plain text file) and also **rose1** (a subdirectory), the **cp** command creates the subvolume **rose** in **/G/vol** and copies only the file **file1**. The subtree **rose1** is not copied. The destination volume must exist in the Guardian environment.

h. To copy all OSS files on remote node **node1** to remote node **node2**, enter:

cp -r -W NOG /E/node1 /E/node2

This command creates the directory tree named **node1** within the root directory of **node2**.

i. To copy all OSS files on the local node to the remote node **node1**, enter:

cp -r -W NOG -W NOE / /E/node1

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

RELATED INFORMATION

Commands: **mv(1)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The **-L** and **-W** flags and the **UTILSGE** environment variable are HP extensions to the XPG4 Version 2 specification.

NAME

cpio - Copies files to and from archive storage

SYNOPSIS

cpio -o[**aABcv**]

cpio -i[**ABcdfmrtuv**] [*pattern ...*]

cpio -p[**aAdlmuv**] *directory*

FLAGS

The **-i**, **-o**, and **-p** flags are described in the **DESCRIPTION** section of this reference page.

-a Resets the access times of copied files to the current time. (When the **-l** flag is also specified, the access times of the linked files are not reset.)

-A Suppresses warning messages about optional access control list (ACL) entries. The **cpio** utility does not archive optional ACL entries. If this flag is not set, a warning message is issued for each file that has optional ACL entries. For more information about ACLs, see the **acl(5)** reference page.

-B Performs block input/output, 5120 bytes to a record.

-c Writes header information in ASCII character form. Specify this flag when POSIX compliance is required and when you are creating or restoring archives for or from another system. Archives written with **-c** must also be read with **-c**. Use this flag to read archives written by **pax**.

-d Creates directories as needed.

-f Copies all files except those matching *pattern* (**cpio -i** only).

-l Links files rather than copying them, whenever possible. This flag can be used only with **cpio -p**.

-m Retains previous file modification time. This flag is ineffective when copying directories.

-r Causes **cpio** to ask whether you want to rename each file before copying it. If you do not want to change the filename, enter the current filename or press only Return. In this last case, **cpio** does not copy the file.

-t Prints a table of contents of the input. Printing the table of contents does not copy any files.

-u Copies unconditionally. Otherwise, a file from the archive with the same name as an existing file in the file system is copied only if the archived file is the newer one.

-v Lists filenames. If you use this with the **-t** flag, the output looks similar to that of the **ls -l** command.

DESCRIPTION

The **cpio** command is used to save and restore data from traditional-format **cpio** archives.

cpio -o (Copy Out)

This command reads file pathnames from the standard input file and copies these files to the standard output file along with pathnames and status information. Output is padded to a 512-byte boundary.

cpio -i (Copy In)

This command reads an archive file created by the **cpio -o** command from the standard input file and copies from it the files with names that match *pattern*. These files are copied into the current directory tree. Permissions of the new files are the same as the permissions associated with the files copied using **cpio -o**. The owner and group of the new files are those of the current user, unless that user has appropriate privileges; if the user has appropriate privileges, **cpio** retains the owner and group of the files copied using **cpio -o**. Only a user with appropriate privileges can extract block special or character special files from an archive.

You can list more than one *pattern* operand using the filename notation described in the **sh(1)** reference page. Note, however, that in the **cpio** command the special characters ***** (asterisk), **?** (question mark), and **[ ]** (brackets) match the **/** (slash) in pathnames, in addition to their use as described for the **sh** command. The default *pattern* is *****, which selects all files in the archive. In an expression such as **[a-z]**, the dash means "through" according to the current collating sequence. The collating sequence is determined by the **LC_COLLATE** environment variable.

cpio -p (Directory Copy)

This command reads file pathnames from the standard input file and copies these files into the directory named by *directory*. The specified directory must already exist. If these pathnames include directory names and if these directories do not already exist, you must use the **-d** flag to cause the directories to be created.

Note that you can copy special files only if you have appropriate privileges. Pathnames cannot exceed 128 bytes. Avoid giving **cpio** pathnames made up of many uniquely linked files, because **cpio** might not have enough memory to keep track of them and could lose linking information.

For filesets that support OSS access control lists (ACLs), this command also copies any ACL entries associated with the file, so that the copied file has the same ACL entries as the source file.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, **LC_TIME**, **NLSPATH**, and **TZ** environment variables.

CAUTIONS

When redirecting the output from **cpio** to a special file (device), redirect it to the raw device and not the block device. Because writing to a block device is done asynchronously, there is no way to know whether the end of the device has been reached.

EXIT VALUES

The exit values for **cpio** are as follows:

0 (zero)	The command executed successfully.
>0	An error occurred. If a file or directory cannot be created or overwritten, **cpio** continues with the next file in the archive or the next file to be added to the archive.

RELATED INFORMATION

Commands: **ar(1)**, **find(1)**, **ls(1)**, **pax(1)**, **sh(1)**.

Files: **cpio(4)**, **locale(4)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

crontab - Submits a schedule of commands to **cron**

SYNOPSIS

crontab [*file*]

crontab -l | **-v** | **-e**

crontab -r

The **crontab** command copies the specified file, or the standard input file if you do not specify a file, into a directory that holds all users' **crontab** files. The **cron** command runs commands according to the instructions in the **crontab** files.

FLAGS

-e Edits a copy of your **crontab** entry. If the **crontab** entry does not exist, **crontab** creates an empty entry to edit. The **-e** flag invokes the editor specified by the **EDITOR** environment variable, or it uses **/bin/vi** by default. **crontab** installs the new entry when editing is complete.

-l Lists the contents of your **crontab** file.

-r Removes the **crontab** file from the **crontab** directory.

-v Displays the name of your **crontab** file and the date and time at which you submitted it using **crontab**.

DESCRIPTION

The **crontab** command replaces your **crontab** file, which is contained in the **/var/spool/cron/crontabs** system directory, with the **crontab** file you specify. In your **crontab** file you have to redirect the output to the standard output or standard error file.

You can use the **crontab** command if your login name appears in the **/var/adm/cron/cron.allow** file. If that file does not exist, the **crontab** command checks the **/var/adm/cron/cron.deny** file to determine if your login name should be denied access to **crontab**. The **cron.allow** and **cron.deny** files contain one login name per line. If neither file exists, you can submit a job only if you are operating with appropriate privileges.

Each **crontab** file entry consists of a line with six fields, separated by spaces and tabs, that contain, respectively:

a. The minute (0 to 59) at which the command sequence executes.

b. The hour (0 to 23) of command execution.

c. The day of the month (1 to 31) of command execution.

d. The month of the year (1 to 12) of command execution.

e. The day of the week (0 to 6 for Sunday to Saturday) of command execution.

f. The shell command to be executed.

Each of these fields can contain:

- A number in the specified range.

- Two numbers separated by a dash to indicate an inclusive range.
- A list of numbers, separated by commas, which selects all numbers in the list.
- An asterisk, meaning all legal values.

Days can be specified by two fields (day of the month and day of the week). If you specify both as a list of elements, both are adhered to. For example, the following entry:

0 0 1,15 * 1 *command*

would run *command* at midnight on the first and fifteenth days of each month, as well as every Monday. To specify days by only one field, the other field should contain an ***** (asterisk).

The **cron** program runs the command named in the sixth field at the selected date and time. If you include a **%** (percent sign) in the sixth field, **cron** treats everything that precedes it (in that field) as the command invocation, and makes all that follows it available to the standard input file, unless you escape the percent sign (**\%**) or double quote it (**"%"**). A **%** (percent sign) in the sixth field is translated to a newline character.

The shell runs only the first line of the command field (up to a **%** or End-of-Line). All other lines are made available to the command as the standard input file.

Blank lines and lines whose first nonblank character is **#** (pound sign) are treated as comments and ignored by **cron**.

The **cron** program invokes a subshell from your **$HOME** directory. It will not run your **.profile** file. If you schedule a command to run when you are not logged in and you want to have commands in your **.profile** run, you must explicitly do so in the **crontab** file. (For a more detailed discussion of how you can invoke **sh**, see the **sh** command.)

Environment Variables

The **cron** program supplies a default environment for every shell, defining **HOME**, **LOGNAME**, **SHELL** (=**/bin/sh**), and **PATH** (=**:/bin:/bin/unsupported:/usr/bin**).

This command supports the use of the **EDITOR**, **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

The following examples show valid **crontab** file entries.

a. To write the time to the file every hour on the hour, enter:

```
0 * * * * echo The hour is ‘date‘. >/datefile
```

This example uses command substitution. (For more information, see the **sh** command.)

b. To run **cal** at 6:30 a.m. every Monday, Wednesday, and Friday, enter:

```
30 6 * * 1,3,5 cal > /calfile
```

c. To define text for the standard input file to a command, enter:

```
0 16 10-31 12 5 wall %HAPPY HOLIDAYS%  Drive safely%
```

This writes a message at 4:00 p.m. each Friday between December 10 and 31 to all users logged in.

The text following the first **%** (percent sign) defines the standard input file to the

wall command as follows:

HAPPY HOLIDAYS
Drive safely

FILES

/var/spool/cron/crontabs	Directory containing the **crontab** files
/var/adm/cron/cron.allow	List of allowed users
/var/adm/cron/cron.deny	List of denied users
$HOME/.profile	User profile

NOTES

a. When entries are made to a **crontab** file, all previous entries are erased.

b. If **cron.allow** exists, a login name with appropriate privileges be listed for that user to be able to use the command.

EXIT VALUES

The **crontab** utility returns the following exit values:

0 (zero)	The **crontab** utility successfully finished its processing.
>0	An error occurred. Your **crontab** entry is not submitted, edited, or listed.

RELATED INFORMATION

Commands: **cron(8)**, **sh(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions. The following features are HP extensions to the XPG4 Version 2 specification:

- The **-v** flag is supported.

NAME

csplit - Splits files by context

SYNOPSIS

csplit [**-f** *prefix*] [**-ks**] [**-n** *number*] [*file* | **-**] *argument* ...

The **csplit** command reads the specified file and separates it into segments defined by the specified arguments.

FLAGS

-f *prefix* Specifies the prefix name (**xx** by default) for the created file segments. If the *prefix* argument would create a filename larger than **NAME_MAX** bytes, an error results, the **csplit** command exits with a diagnostic message, and no files are created.

-k Keeps created file segments intact in the event of an error.

-n *number*
Uses *number* decimal digits to form filenames for the file segments. The default value is 2.

-s Suppresses the display of character counts.

DESCRIPTION

If you specify **-** (dash) in place of the input filename, the **csplit** command reads from the standard input file.

By default, **csplit** writes the file segments to files named **xx00** ...**xx***n*, where *n* is the number of arguments listed on the command line. By default, these new files get the following segments of *file*:

00 From the start of *file* up to, but not including, the line referenced by the first *argument*.

01 From the line referenced by the first *argument* up to the line referenced by the second *argument*.

n+**1** From the line referenced by the last *argument* to the end of *file*.

The **csplit** command does not alter the original file.

The specified *arguments* can be a combination of the following:

/*pattern*/[*offset*]
Creates a file using the contents of the lines from the current line up to, but not including, the line that results from the evaluation of the regular expression with an offset, if included. The *offset* argument can be any integer (positive or negative) that represents a number of lines.

%*pattern***%**[*offset*]
Has the same effect as /*pattern*/ except no segment file is created.

+*number*

-*number* Move forward or backward the specified number of lines from the line matched by an immediately preceding *pattern* argument.

line_number
Creates a file containing the segment from the current line up to, but not including, *line_number*, which becomes the current line.

{*number***}**

Repeats the preceding argument the specified number of times. This number can follow any of the *pattern* or *line_number* arguments. If it follows a *pattern* argument, the **csplit** command reuses that pattern the specified number of times. If it follows a *line_number* argument, **csplit** splits the file from that point every *line_number* of lines for *number* times.

Place within quotation marks all *pattern* arguments that contain spaces or other characters special to the shell. Patterns may not contain embedded newline characters.

See the reference page for the **grep** command for information about creating patterns. In an expression such as **[a-z]**, the dash means "through" according to the current collating sequence. The collating sequence is determined by the value of the **LC_COLLATE** environment variable.

EXAMPLES

a. To split the text of a book into a separate file for each chapter, enter:

csplit book "/^Chapter *[0-9]/" {9}

This command creates files named **xx00**, **xx01**, **xx02**,...,**xx09**, and **xx10**, which contain individual chapters of the file **book**. The file **xx00** contains the front matter that comes before the first chapter. The **{9}** after the pattern causes the **csplit** command to create individual chapters up until file **xx09**; the remainder of **book** is then placed in file **xx10**.

b. To specify the prefix for the created filenames, enter:

csplit -f chap book "/^Chapter *[0-9]/" {9}

This command splits **book** into files named **chap00**, **chap01**,...**chap9**, and **chap10**.

RELATED INFORMATION

Commands: **ed(1), sh(1), split(1).**

Files: **locale(4)**.

NAME

cut - Displays selected parts from each line of a file

SYNOPSIS

cut -b *list* [**-n**] [*file* ...]

cut -c *list* [*file* ...]

cut -f *list* [**-d** *character*] [**-s**] [*file* ...]

The **cut** command locates the specified parts in each line of the specified file and writes the characters in them to the standard output file.

FLAGS

-b *list* Cuts selections based on a list of bytes. Each selected byte is output, unless you also specify the **-n** flag. For example, if you specify **-b 1-72**, the **cut** command writes out the first 72 bytes in each line of the file.

-c *list* Cuts selections based on a list of characters.

-d *character*
Uses the specified character as the field delimiter (separator) when you specify the **-f** flag. You must quote characters with special meaning to the shell, such as the space character. Any character can be used as *character*. The default field delimiter is a tab character.

-f *list* Specifies a list of fields assumed to be separated in the file by a field delimiter character, specified by the **-d** flag or the tab character by default. For example, if you specify **-f 1,7**, the **cut** command writes out only the first and seventh fields of each line. If a line contains no field delimiters, **cut** passes them through intact (useful for table subheadings) unless you specify the **-s** flag.

-n Does not split characters. When specified with the **-b** flag, each element in *list* of the form *low*-*high* (hyphen-separated numbers) is modified as follows:

- If the byte selected by *low* is not the first byte of a character, *low* is decremented to select the first byte of the character originally selected by *low*.
- If the byte selected by *high* is not the last byte of a character, *high* is decremented to select the last byte of the character prior to the character originally selected by *high*, or 0 (zero) if there is no prior character.

  If the resulting range element has *high* equal to 0 (zero) or *low* greater than *high*, the list element is dropped from *list* for that input line without causing an error.

Each element in *list* of the form *low*- is treated as previously described with *high* set to the number of bytes in the current line, not including the terminating newline character. Each element in *list* of the form -*high* is treated as previously described with *low* set to 1. Each element in *list* of the form *number* (a single number) is treated as previously described with *low* set to *number* and *high* set to *number*.

-s Suppresses lines that do not contain delimiter characters (use only with the **-f** flag). Unless you include this flag, lines with no delimiters are passed through.

DESCRIPTION

If you do not specify a file or if you specify a **-** (dash), the **cut** command reads the standard input file.

You must specify the **-b** flag (to select bytes), the **-c** flag (to select characters), or the **-f** flag (to select fields). The *list* argument (see the **-b**, **-c**, and **-f** flags) must be a space-separated or comma-separated list of positive numbers and ranges. Ranges can be in three forms:

- Two positive numbers separated by a **-** (dash), as in the form *low*-*high*, which represents all fields from the first number to the second number.
- A positive number preceded by a **-** (dash), as in the form **-***high*, which represents all fields from field number 1 to that number.
- A positive number followed by a **-** (dash), as in the form *low*-, which represents that number to the last field, inclusive.

The elements in *list* can be repeated, can overlap, and can be specified in any order.

Some sample *list* specifications are as follows:

1,4,7 or **1 4 7**
: First, fourth, and seventh bytes or fields.

1-3,8 First through third and eighth bytes or fields.

-5,10 First through fifth and tenth bytes or fields.

3- Third through last bytes or fields.

The fields specified by *list* can be a fixed number of byte positions, or the length can vary from line to line and be marked with a field delimiter character, such as a tab character.

You can also use the **grep** command to make horizontal cuts through a file and the **paste** command to put the files back together. To change the order of columns in a file, use the **cut** and **paste** commands.

EXAMPLES

To display several fields of each line of a file, enter:

cut -f 1,5 -d : /etc/passwd

This command displays the login name and full username fields of the system password file. These are the first and fifth fields (**-f 1,5**) separated by colons (**-d :**).

So, if the **/etc/passwd** file looks like this:

```
su:UHuj9Pgdvz0J":0:0:User with special privileges:/:
daemon:*:1:1::/etc:
bin:*:2:2::/usr/bin:
sys:*:3:3::/usr/src:
adm:*:4:4:System Administrator:/usr/adm:
pierre:*:200:200:Pierre Harper:/u/pierre:
joan:*:202:200:Joan Brown:/u/joan:
```

Then **cut -f 1,5 -d : /etc/passwd** produces this output:

```
su:User with special privileges
daemon:
bin:
sys:
adm:System Administrator
pierre:Pierre Harper
joan:Joan Brown
```

RELATED INFORMATION

Commands: **grep(1), paste(1).**

Section 3. User Commands (d - f)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letters **d** through **f**.

NAME

date - Display the date and time

SYNOPSIS

date [**-u**] [+*format*]

The **date** command displays the date and time.

FLAGS

-u Performs operations as if the **TZ** environment variable was set to the string **UTC0** or its equivalent historical value **GMT0**. Otherwise, the **date** command uses the time zone indicated by the **TZ** environment variable or the system default time zone if that variable is not set.

DESCRIPTION

The **date** command writes the current date and time to the standard output file if called with no flags or with a flag list that begins with a **+** (plus sign). Do not use the **date** command to change or set the date and time; use Guardian utilities instead.

The **LC_TIME** variable, if it is defined, controls the ordering of the day and month numbers in the date specifications. The default order is *mmddHHMM***.***SSyy*, where:

- *mm* is the month number (01=January).
- *dd* is the number of the day in the month.
- *HH* is the hour in the day (using a 24-hour clock).
- *MM* is the minute number.
- *SS* is the number of seconds.
- *yy* is the last two numbers of the year.

The *yymmddHHMM*[**.***SS*] format cannot be used if the year is in the range 01-12 (this is how the **date** command differentiates between the two formats).

If the locale defines ordering such that the day is specified before the month, the format is *ddmmHHMM*[**.***SS*[*yy*]], *yyddmmHHMM*[**.***SS*], or *ddmmHHMM*[*yy*].

The current month, day, hour, and year are default values. The system operates in Coordinated Universal Time (UTC).

If you follow keyword **date** with a **+** (plus sign) and a field descriptor, you can control the output of the command. You must precede each field descriptor with a **%** (percent sign). The system replaces the field descriptor with the specified value. Enter a literal **%** as **%%**. The **date** command copies any other characters to the standard output file without change. **date** always ends the string with a newline character. Output fields are fixed size (zero-padded if necessary).

The date command prints out a usage message on any unrecognized flags or input.

Field Descriptors

a Displays the locale's abbreviated weekday name (`Sun` through `Sat` or the nonEnglish equivalent).

A Displays the locale's full weekday name.

b Displays the locale's abbreviated month name.

B Displays the locale's full month name.

c Displays the locale's appropriate time and date representation.

C Displays the locale's century (the year divided by 100 and truncated to an integer) as a decimal number (`00` through `99`).

d Displays the day of the month as a decimal number (`01` through `31`).

D Displays the date in the format *mm/dd/yy* (the default format) or as specified by the **LC_TIME** environment variable, if defined.

e Displays the day of the month as a decimal number (`1` through `31` in a 2-digit field with leading space fill).

Ec Specifies the locale's alternative appropriate date and time representation.

EC Specifies the name of the base year (period) in the locale's alternative representation.

Ex Specifies the locale's alternative date representation.

Ey Specifies the offset from the display of the **%EC** field discriptor (year only) in the locale's alternative representation.

EY Specifies the full alternative year representation.

h Is a synonym for the **%b** field descriptor.

H Displays the hour as a decimal number (`00` through `23`).

I Displays the hour as a decimal number (`01` through `12`).

j Displays the day of the year as a decimal number (`001` through `366`).

m Displays the month of the year as a decimal number (`01` through `12`).

M Displays the minute as a decimal number (`00` through `59`).

n Inserts a newline character.

N Represents the alternative era name.

o Represents the alternative era year.

Od Specifies the day of the month using the locale's alternative numeric symbols.

Oe Specifies the day of the month using the locale's alternative numeric symbols.

OH Specifies the hour (24-hour clock) using the locale's alternative numeric symbols.

OI Specifies the hour (12-hour clock) using the locale's alternative numeric symbols.

Om Specifies the month using the locale's alternative numeric symbols.

OM Specifies the minutes using the locale's alternative numeric symbols.

OS Specifies the seconds using the locale's alternative numeric symbols.

OU Specifies the week number of the year (with Sunday as the first day of the week) using the locale's alternative numeric symbols.

Ow Specifies the weekday as a number in the locale's alternative representation (Sunday = 0).

OW Specifies the week number of the year (with Monday as the first day of the week) using the locale's alternative numeric symbols.

Oy Specifies the year (offset from the display of the **%C** field descriptor) in alternative representation.

p Displays the locale's equivalent of either AM or PM.

r Displays the time (12-hour clock) using AM/PM notation (or the nonhown in this nonEnglish equivalent) in the format *hh:mm:ss* AM or *hh:mm:ss* PM. In the OSS locale, this is equivalent to the field descriptors **%I: %m: %S %p**.

S Displays the seconds as a decimal number (00 through 61).

t Inserts a tab character.

T Displays the time in 24-hour clock format (00 through 23) as *hh:mm:ss* (the default format) or as specified by the **LC_TIME** environment variable, if defined.

U Displays the week number of the year (Sunday is the first day of the week) as a decimal number (00 through 53).

w Displays the day of the week as a decimal number (Sunday = 0).

W Displays the week number of the year (Monday is the first day of the week) as a decimal number (00 through 53).

x Displays the locale's appropriate date representation.

X Displays the locale's appropriate time representation.

y Displays the last two numbers of the year as a decimal number (00 through 99).

Y Displays the full year as a decimal number.

Z Displays the time zone name or no characters if the time zone cannot be determined.

%% Inserts a **%** character.

EXAMPLES

1. To display the current date and time, enter:

   date

   The output might look like the following:

   ```
   Thu Apr 9   13:21:30 EDT  1992
   ```

2. To display the date and time in a specified format, enter:

   date +"%r %d %h %y (%a)"

   This displays the date (assume the year is 1992) as:

   ```
   01:21:30  PM  09  Apr  92  (Thu)
   ```

RELATED INFORMATION

Files: **locale(4).**

STANDARDS CONFORMANCE

The date and time are set through the Guardian environment.

NAME

dc - Performs integer arithmetic with arbitrary precision

SYNOPSIS

dc [*file*]

DESCRIPTION

The **dc** command is an arbitrary-precision arithmetic calculator. **dc** takes its input from *file* or the standard input file until it reads an End-of-File character. It writes to the standard output file. **dc** operates on integers by default, but you can use subcommands to specify an input base, an output base, and a number of fractional digits to be maintained.

The **bc** command is a preprocessor for the **dc** command. The **bc** command provides infix notation and a syntax similar to the C language, which implements functions and reasonable control structures for programs.

SUBCOMMANDS

number Pushes the specified value onto the stack. *number* is an unbroken string consisting of the digits 0 through 9. To specify a negative number, precede *number* with _ (underscore). A number can contain a decimal point.

+ - * / % ^
Adds (+), subtracts (-), multiplies (*), divides (/), remainders (%), or exponentiates (^) the top two values on the stack. **dc** pops the top two values off the stack and pushes the result on the stack in their place.

! Interprets the rest of the line as a system command.

? Executes a line of input from the standard input file.

c Cleans the stack: pops all values on the stack.

d Duplicates the top value on the stack.

f Displays all values on the stack.

i Pops the top value on the stack and uses that value as the number radix for further input.

I Pushes the input base onto the stack.

k Pops the top value on the stack and uses that value as a nonnegative scale factor. The appropriate number of places is displayed on output and is maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base is reasonable if all are changed together.

l*x* Pushes the value in register *x* onto the stack. Register *x* is not changed. All registers start with 0 (zero) value.

L*x* Treats *x* as a stack and pops its top value onto the main stack.

o Pops the top value on the stack and uses that value as the number radix for further output.

O Pushes the output base onto the stack.

p Prints the top value on the stack. The top value remains unchanged.

P Interprets the top value on the stack as an ASCII string, prints it, and removes it.

q Exits the program. If **dc** is executing a string, it pops the recursion level by two.

Q Pops the top value on the stack and pop the string execution level by that value.

s*x* Pops the top value on the stack and stores it in a register named *x*, where *x* can be any single-byte character.

S*x* Treats *x* as a stack: pops the top value on the main stack and pushes that value onto stack *x*.

v Replaces the top value on the stack by its square root. Any existing fractional part of the argument is used, but otherwise the scale factor is ignored.

x Treats the top value on the stack as a character string and executes it as a string of **dc** commands.

X Replaces the top value on the stack by its scale factor.

z Pushes the number of elements in the stack onto the stack.

Z Replaces the top value on the stack with the number of digits in that value.

[*string*] Puts *string* onto the stack.

<*x* Pops the top two values on the stack and compares them. Register *x* is executed if the stated relationship is TRUE.

;: Are used for array operations.

EXAMPLES

1. To use **dc** as a calculator, proceed as follows:

   Enter:

   1 4 / p [Divide 1 by 4]

   The system responds as follows:

   ```
   0
   ```

   Enter:

   1 k [Keep 1 decimal place]s.
   1 4 / p

   The system responds as follows:

   ```
   0.2
   ```

   Enter:

   3 k [Keep 3 decimal places]s.
   1 4 / p

   The system responds as follows:

   ```
   0.250
   ```

Enter:

16 63 5 / + p [Divide 63 by 5, add the result to 16]

The system responds as follows:

```
28.600
```

Enter:

16 63 5 + / p [Add 63 and 5, divide the result by 16]

The system responds as follows:

```
0.235
```

You can type the comments (enclosed in brackets) into the command, but they are provided only for your information.

When you enter **dc** expressions directly from the keyboard, press the End-of-File key sequence to end the **dc** session and return to the shell command line.

2. To load and run a **dc** program file, proceed as follows:

Enter:

dc prog.dc
5 lf x p [5 factorial]s.

The system responds as follows:

```
120
```

Enter:

10 lf x p [10 factorial]s.

The system responds as follows:

```
3628800
```

This command interprets the **dc** program saved in **prog.dc** then reads from thestandard input file. The **lf x** evaluates the function stored in register **f**, which could be defined in the program file **prog.dc** as:

```
[ f: compute the factorial of n ]s.

[       (n = the top of the stack) ]s.

[ If 1>n do b;  If 1<n do r ]s.
        [d 1 >b d 1 <r] sf

[ Return f(n) = 1       ]s.
        [d - 1 +] sb

[ Return f(n) = n * f(n-1)  ]s.
        [d 1 - lf x *] sr
```

You can create **dc** program files with a text editor or with the **-c** (compile) flag of the **bc** command. When you enter **dc** expressions directly from the keyboard, press the End-of-File key sequence to end the **dc** session and return to the shell command line.

RELATED INFORMATION

Commands: **bc(1)**.

Files: **locale(4)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

del_define - Deletes one or more DEFINEs from the current OSS shell

SYNOPSIS

del_define {{*define-name*}... | **all**}

FLAGS

all Specifies that all existing DEFINEs except **=_DEFAULTS** are to be deleted.

DESCRIPTION

The **del_define** command is specific to OSS and an OSS shell built-in command. It deletes DEFINEs from the OSS shell.

The **del_define** command affects only DEFINEs for the current shell process. It is similiar to the TACL DELETE DEFINE command. Refer to the DELETE DEFINE command in the *TACL Reference Manual*.

define-name

Specifies the name of the DEFINE to be deleted. The name can be 2 through 24 characters long. The first character must be an equal sign (=) and the second must be a letter. The **=_DEFAULTS** DEFINE cannot be deleted.

EXAMPLES

1. To delete the DEFINE named **=DFILE**, enter:

   del_define =DFILE

2. To delete all defines in the current OSS shell (except **=_DEFAULTS**), enter:

   del_define all

EXIT VALUES

The following exit values are returned:

0 DEFINEs were deleted successfully.

>0 An error occurred.

NOTES

The **del_define** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **add_define(1)**, **info_define(1)**, **set_define(1)**, **show_define(1)**.

STANDARDS CONFORMANCE

The **del_define** command is an HP extension to the XPG4 Version 2 specification.

NAME

df - Displays statistics of filesets

SYNOPSIS

df [**-k**] [[*fileset*] ...]

FLAGS

-k Causes disk space numbers to be reported in 1024-byte (1-kilobyte) blocks. By default, all numbers are reported in 512-byte blocks.

DESCRIPTION

The **df** command displays the amount of used and available disk space on the fileset specified as *fileset*. It also displays the status (started, stopped, or unknown), how much of the total capacity of the fileset has been used, and the mount point of the fileset.

The value specified as *fileset* must be a fileset name as defined in the OSS Monitor subsystem of the Subsystem Control Facility (SCF). Valid values for *fileset* are not case-sensitive. If more than 15 values are specified for *fileset*, only the first 15 specified are displayed.

If no fileset name is specified, only statistics for the first 15 filesets are displayed. If more than 15 filesets exist, the **df** command displays a warning message.

EXAMPLES

Entering the command **df** with no flags on a node with 16 filesets displays:

```
WARNING: MORE THAN 15 FILESETS ARE PRESENT
Filesystem State   512-blocks Used   Avail Capacity Mounted on
DSAP       STOPPED    -         -     -         -    /dsap
R1         STOPPED    -         -     -         -    /home/ali
R2         STOPPED    -         -     -         -    /home/ali
R27        STOPPED    -         -     -         -    /home/ali
R28        STOPPED    -         -     -         -    /home/ali
R29        STOPPED    -         -     -         -    /home/ali
R3         STOPPED    -         -     -         -    /home/ali
R30        STOPPED    -         -     -         -    /home/ali
R31        STOPPED    -         -     -         -    /home/ali
R32        STOPPED    -         -     -         -    /home/ali
R33        STOPPED    -         -     -         -    /home/ali
R34        STOPPED    -         -     -         -    /home/ali
R35        STOPPED    -         -     -         -    /home/ali
R9         STOPPED    -         -     -         -    /home/ali
ROOT       STARTED 69927952 69620808 307144   100%  /
```

RELATED INFORMATION

Commands: **du(1)**.

Functions: **fstatvfs(2)**.

STANDARDS CONFORMANCE

The HP implementation does not support the **-P** or **-t** flag.

The HP implementation does not allow the specification of filenames.

NAME

diff - Compares text files

SYNOPSIS

diff [**-c** | **-C** *number* | **-e** | **-f** | **-n**] [**-br**] *directory1 directory2*

diff [**-c** | **-C** *number* | **-e** | **-f** | **-n**] [**-b**] *file1 file2*

FLAGS

The **-c**, **-C**, **-e**, **-f**, and **-n** flags are mutually exclusive.

The **-r** flag can be specified only with directory comparisons.

The **-b** flag can be used in combination with any other flags and in both file and directory comparisons.

-b Causes trailing spaces and tabs (blanks) to be ignored, and other strings of spaces and tabs to be considered to be identical.

-c Produces a listing with the default number of lines of context (3 lines). The output lists the files being compared and their last modification dates, then lists the differing lines. Lines that are changed from one file to the other are marked in both files with an **!** (exclamation point). Changes that lie within the specified number of lines of each other are grouped together on output.

-C *number*
Produces output that provides the number of lines of context specified by the *number* argument (where *number* is a positive decimal integer).

-e Produces a script of **a**, **c**, **d**, and **s** commands for the editor **ed**, which can re-create *file2* from *file1*. In connection with **-e**, the following shell program can help maintain multiple versions of a file. Only an ancestral file **($1)** and a chain of version-to-version **ed** scripts (**$2**, **$3**, ...) made by the **diff** command need be on hand. A "latest version" appears on the standard output, as shown below:

```
(shift; cat $*; echo '1,$p') | ed - $1
```

Extra commands are added to the output when comparing directories with **-e**; the result is an **sh** script for converting text files common to the directories from their state in *directory1* to their state in *directory2*.

-f Produces a script similar to that of **-e**, but not useful with the **ed** editor, and in the opposite order.

-r Checks files in common subdirectories recursively.

DESCRIPTION

Input Options

If neither the *file1* nor the *file2* argument is a directory, then either can be given as **-** (dash), in which case the standard input is used. If *file1* is a directory and *file2* is a file, or vice versa, a file in the specified directory with the same name as the specified file is used.

If both arguments are directories, the **diff** command sorts the contents of the directories by name, and then runs the regular **diff** file algorithm on text files that are different. Binary files that differ, common subdirectories, and files that appear in only one directory are also listed.

Output Options

There are several choices for output format. The default output format contains lines of these forms:

number1 **a** *number2,number3*
number1,number2 **d** *number3*
number1,number2 **c** *number3,number4*

These lines resemble **ed** commands to convert *file1* into *file2*. **a** indicates that a line or lines were *added* to one of the files; **d** indicates that a line or lines were *deleted*; and **c** indicates that a line or lines were *changed*. The numbers after the letters pertain to *file2*. In fact, by exchanging **a** for **d** and reading backward, one can ascertain how to convert *file2* into *file1*. As in the editor **ed**, identical pairs where *number1 = number2* or *number3 = number4* are abbreviated as a single number.

Following each of these lines are all the lines affected in the first file, flagged by < (left angle bracket), then all the lines that are affected in the second file, flagged by > (right angle bracket).

Except in rare circumstances, the **diff** command finds the smallest sufficient set of file differences.

EXAMPLES

1. To compare two files, enter:

   diff chap1.bak chap1

   This command displays the differences between the files **chap1.bak** and **chap1**.

2. To compare two files, ignoring differences in the amount of white space, enter:

   diff -b prog.c.bak prog.c

   If two lines differ only in the number of spaces and tabs between words, then the **diff** command considers them to be the same.

3. To create a file containing commands that the **ed** command can use to reconstruct one file from another, enter:

   diff -e ch2 ch2.old > new.old.ed

   This command creates a file named **new.to.old.ed** that contains the **ed** subcommands to change file **chap2** back into the version of the text found in file **chap2.old**. In most cases, **new.to.old.ed** is a much smaller file than **chap2.old**.

4. You can save disk space by deleting the file **chap2.old**, and you can reconstruct the file at any time by entering:

   (cat new.old.ed ; echo '1,$p') | ed - ch2 > ch2.old

   The commands in parentheses add **1,$p** to the end of the editing commands sent to the **ed** editor. The **1,$p** causes the **ed** command to write the file to standard output after editing it. This modified command sequence is then piped to the **ed** command (**|** **ed**), and the editor reads it as standard input. The **-** flag causes the **ed** command to not display the file size and other extra information, because it would be mixed with the text of file **chap2.old**.

NOTES

Editing scripts that are produced by the **-e** or **-f** flags cannot create lines that consist of a single . (dot) character.

Block, character, or FIFO special files cannot be used with the **diff** command because they cause the command to exit.

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no destinction between Guardian and OSS files or between local and remote files.

EXIT VALUES

An exit value of 0 (zero) indicates no differences; a value of 1 indicates differences were found, and value greater than 1 indicates an error.

RELATED INFORMATION

Commands: **comm**(1), **ed**(1), **pr**(1).

NAME

dircmp - Compares two directories

SYNOPSIS

dircmp [**-d**] [**-s**] *directory1 directory2*

FLAGS

-d Displays, for each common filename, the differing contents of the two files, if any. The display format is the same as that of the **diff** command.

-s Suppresses listing of the names of identical files.

DESCRIPTION

The **dircmp** command reads *directory1* and *directory2*, compares their contents, and writes the results to the standard output file.

First, **dircmp** compares the filenames in each directory. When the same filename appears in both, **dircmp** compares the contents of the two files.

In the output, **dircmp** lists the files unique to each directory. It then lists the files with identical names but different contents. If entered without a flag, **dircmp** also lists files that have both identical names and identical contents.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

1. To summarize the differences between the files in two directories, enter:

   dircmp proj.ver1 proj.ver2

   This command displays a summary of the differences between the directories **proj.ver1** and **proj.ver2**. The summary lists separately the files found only in one directory or the other, and those found in both. If a file is found in both directories, that file is listed. If the files are identical, **dircmp** displays `identical`; otherwise, **dircmp** displays `different`.

2. To show the details of the differences between files, enter:

   dircmp -d -s proj.ver1 proj.ver2

   The **-s** flag suppresses information about identical files. The **-d** flag displays a **diff** listing for each of the differing files found in both directories.

NOTES

In most cases, **diff -r**, rather than **dircmp**, is preferred.

EXIT VALUES

The following exit values are returned:

0 (zero) The command completed successfully.

>0 An error occurred.

RELATED INFORMATION

Commands: **cmp(1)**, **diff(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

dirname - Returns specified parts of pathnames

SYNOPSIS

dirname *string*

DESCRIPTION

The **dirname** command reads the string specified on the command line, deletes from the last / (slash) to the end of the line, and writes the remaining pathname to standard output.

The **dirname** command is generally used inside command substitutions within a shell procedure to specify an output filename that is some variation of a specified input filename.

EXAMPLES

1. To construct the name of a file located in the same directory as another, enter:

   AOUTFILE=‘dirname $TEXTFILE‘/a.out

   This command sets the **AOUTFILE** shell variable to the name of an **a.out** file in the same directory as **TEXTFILE**. If **TEXTFILE** is **/u/fran/prog.c**, then the value of **dirname $TEXTFILE** is **/u/fran**, and **AOUTFILE** becomes **/u/fran/a.out**.

RELATED INFORMATION

Commands: **basename**(1), **sh**(1).

NAME

dspcat - Displays all or part of a message catalog

SYNOPSIS

dspcat [**-g**] *catalog_name*
[*set_number* [*message_number*]]

FLAGS

-g formats the output so that it can be used as input to the **gencat** utility. You cannot use the *message_number* operand with the **-g** flag.

Operands

catalog_name identifies a file containing a message catalog. If you omit this operand, **dspcat** searches for the message catalog in the set of directories specified by the **NLSPATH** environment variable.

set_number specifies a set in the catalog.

message_number
specifies a message in the set. You cannot use the *message_number* operand with the **-g** flag. You must specify *set_number* if you specify *message_number*.

DESCRIPTION

The **dspcat** utility displays all or part of a message catalog. If you include all three operands, **dspcat** displays the specified message. If you do not include *message_number*, **dspcat** displays all the messages in the set. If you specify only *catalog_name*, **dspcat** displays all messages in the catalog. If you specify an invalid *message_number* or an *invalid set_number*, **dspcat**displays an error message.

Environment Variables

These environment variables affect the execution of the **dspcat** utility: **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXAMPLES

To display message number 2 in set number 1 of *test.cat*, enter:

```
dspcat test.cat 1 2
```

DIAGNOSTICS

The **dspcat** utility generates these errors:

```
Usage: dspcat [-g] catname [set#] [msg#]
dspcat: Invalid set number.\n
No message id allowed with -g option.
dspcat: Invalid set number.\n
dspcat: Invalid msg number.\n
Usage: dspcat [-g] catname [set#] [msg#]
Unable to open specified catalog \%d set(s).\n
dspcat: Invalid set - catalog only has \%d set(s).\n
dspcat: Invalid set - set \%d not found.\n
dspcat: Invalid message - message \%d not found.\n
```

RELATED INFORMATION

Commands: **gencat(1)**, **mkcatdefs(1)**.

STANDARDS CONFORMANCE

The **dspcat** utility is an HP extension to the XPG4 Version 2 specification.

NAME

dspmsg - Writes a message from a message catalog to standard output

SYNOPSIS

dspmsg [**-s** *set_number*]
catalog_name
message_number
[*'default_message*']
[*argument* . . .]

FLAGS

-s *set_number* specifies a set in the message catalog; the default set number is 1, if not specified by -s set_number.

Operands

catalog_name specifies the message catalog.

message_number
specifies the message to be written to standard output.

default_message
specifies the message to be written if **dspmsg** cannot find the specified message; *default_message* is optional.

argument supplies optional arguments to substitute into the message catalog or into the default message; there is no limit to the number of arguments.

DESCRIPTION

The **dspmsg** utility writes a message from a message catalog to standard output. You must specify the message catalog (*catalog_name*) and the message (*message_number*) to be written. You can specify the set number with the **-s** flag, or let **dspmsg** default to set number 1.

The **dspmsg** utility allows an unlimited number of string arguments for substituting into the specified message if it contains either the **%s** or the **%n$s printf()** function conversion specifications.

If **dspmsg** cannot find the specified message it writes the optionally specified **default_message**. If **dspmsg** cannot find the specified message and you do not supply a default, a system-generated error message is written. You must enclose the default message in single quotes if you are using the **%n$s** notation for message inserts.

You can follow the default message with optional arguments to substitute into the catalog message or the default message. Missing arguments for conversion specifications are replaced by null strings.

Environment Variables

These environment variables affect the execution of the **dspmsg** utility: **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **NLSPATH**.

EXAMPLES

To display the set number 1, message number 2, of the catalog `test.cat`, enter:

```
dspmsg -s 1 test.cat 2 'Message not found'
```

If you have not assigned message text to message number 2, **dspmsg** displays the default message **Message not found**.

DIAGNOSTICS

The **dspmsg** utility generates these errors:

```
\nNone or all arguments must use \%n$ format
\n$ missing from \%n$ format
\nNone or all arguments must use \%n$ format
\nInvalid argument index
\nInvalid format specifier
```

RELATED INFORMATION

Commands: **gencat(1)**, **mkcatdefs(1)**.

STANDARDS CONFORMANCE

The **dspmsg** utility is an HP extension to the XPG4 Version 2 specification.

NAME

du - Displays a summary of disk usage

SYNOPSIS

du [**-a** | **-s**] [**-klrx**] [**-Wuser=***username*] [*file* ...]

FLAGS

-a Displays disk use for each file. Without **-a**, **du** does not report on files, unless they are listed on the command line.

-k Displays the file sizes in units of 1024 bytes, instead of the default 512-byte units.

-l Allocates blocks, in files with multiple links, evenly among the links. By default, a file with two or more links is counted only once.

-r Displays an error message when **du** encounters an inaccessible directory, or an inaccessible file when used with **-a**. This is the default action.

-s Displays only the grand total for each of the specified files, but not for any subdirectories.

-x When evaluating file sizes, evaluates only those files that have the same device as the file specified by *file*. In other words, **du** does not cross mount points.

When two flags are mutually exclusive (such as **-a** and **-s**), the last flag you enter on the command line takes precedence.

HP Extension

-Wuser=*username*

Specifies that only the files or directories owned by that user are listed.

DESCRIPTION

The **du** command displays the number of blocks in all directories (listed recursively) within each specified *directory*. When the **-a** flag is specified, **du** reports the number of blocks in individual files. The block count includes the indirect blocks of each file and is in 512-byte units, independent of the cluster size used by the system. If no file or directory name is provided, the **du** command uses the current directory.

The size of the file space allocated to a directory is defined as the sum total of the space allocated to all files in the file hierarchy rooted in the directory, plus the space allocated to the directory itself.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

1. To display the disk usage of a directory tree and each of its subtrees, enter:

   du /u/fran

   This displays the number of disk blocks in the **/u/fran** directory and each of its subdirectories.

2. To display the disk usage of each file, enter:

   du -a /u/fran

   This displays the number of disk blocks contained in each file and subdirectory of **/u/fran**. The number shown beside a directory name is the disk usage of that directory tree. The number shown beside the name of a regular file is the disk usage of that file

alone.

3. To display only the total disk usage of a directory tree, enter:

   du -s /u/fran

   This displays only the sum total disk usage of **/u/fran** and the files it contains (**-s**).

4. To display only total disk usage used by user **grp.ram** of a directory tree, enter:

   du -s -Wuser=grp.ram /u/fran

NOTES

If too many files are distinctly linked, **du** counts the excess files more than once.

When **du** cannot execute the **stat( )** function on files or cannot read or execute **stat( )** on directories, it reports an error condition and the final exit status is affected. Files with multiple links are counted and written for only one entry.

EXIT VALUES

The **du** command returns the following values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **df(1)**, **ls(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

The following features are HP extensions to the XPG4 Version 2 specification:

- The **-l** and **-Wuser** flags are supported.

NAME

echo - Writes arguments to standard output

SYNOPSIS

echo [*string* ...]

DESCRIPTION

The **echo** command writes the string specified by the *string* argument to the standard output file.

The arguments are separated by spaces, and a newline character follows the last *string*. Use **echo** to produce diagnostic messages in command files and to send data into a pipe. If there are no arguments, **echo** outputs a newline character.

The **echo** command recognizes the following special characters:

\a Displays an alert character.

\b Displays a backspace character.

\c Suppresses the newline character. All characters following **\c** in the arguments are ignored.

\f Displays a formfeed character.

\n Displays a newline character.

\r Displays a carriage-return character.

\t Displays a tab character.

\v Displays a vertical tab character.

\\ Displays a backslash character.

number Displays an 8-bit character whose value is the 0-, 1-, 2-, or 3-digit octal number given by the *number* argument. The first digit of *number* must be a 0 (zero).

EXAMPLES

1. To write a message to standard output, enter:

   echo Please insert diskette . . .

2. To display a message containing special characters as listed in **DESCRIPTION**, enclose the message in quotes, as follows:

   echo "\n\n\nI'm at lunch.\nI'll be back at 1 p.m."

   This command skips three lines and displays the messages:

   ```
   I'm at lunch.
   I'll be back at 1 p.m.
   ```

   Note that you must enclose the message in quotation marks if it contains escape sequences such as **\n**. Otherwise, the shell treats the \ (backslash) as an escape character. The previous command line, entered without the quotation marks, results in the following output:

   ```
   nnnI'm at lunch.nI'll be back at 1 p.m.
   ```

3. To use **echo** with pattern-matching characters, enter:

   echo The back-up files are: *.bak

This command displays the message `The back-up files are:` and then displays the filenames in the current directory ending with **.bak**.

4. To add a single line of text to a file, enter:

   echo Remember to set the shell search path to $PATH. >>notes

   This command adds the message to the end of the file **notes** after the shell substitutes the value of the **PATH** shell variable.

5. To write a message to the standard error output (shell built-in command only), enter:

   echo Error: file already exists. >&2

   Use this command in shell procedures to write error messages. If the **>&2** is omitted, then the message is written to the standard output file.

NOTES

The OSS **echo** command has both a shell built-in version and a regular version. The two versions have the same features and functionality. The only difference between the two versions is that the shell built-in version does not start a new shell process when it is invoked. Both versions are described in the reference page for **echo**. The shell built-in version is the default. To specify the regular version, use the full pathname: **/bin/echo** For more information about shell built-in commands, refer to the reference page for **sh(1).**

RELATED INFORMATION

Commands: **sh(1).**

NAME

ecobol - Compiles TNS/E native COBOL85 programs

SYNOPSIS

ecobol

[**-c** | **-Wnolink**]
[**-g**]
[**-L** *directory*] ...
[**-l** *library*] ...
[**-O** [*optlevel*]]
[**-o** *outfile*]
[**-s**]
[**-Wansistreams**]
[**-Wcall_shared** | **-Wshared**]
[**-WBdllsonly** | **-WBdynamic** | **-WBstatic**]
[**-Wcobol**="*arg*[**,**...]"]
[**-Wcodecov**]
[**-Wcolumns**=*n*]
[**-Wcopylib**=*pathname1*]
[**-Wdryrun**]
[**-Weld**="*arg*[**,**...]"]
[**-Weld_obey**="*pathname2*"]
[**-Werrors**=*n*]
[**-Wglobalized**]
[**-Wheap**=*n*[**b** | **w** | **p**]
[**-Whelp** | **-Wusage**]
[**-Whighpin**={**on** | **off**}]
[**-Whighrequesters**={**on** | **off**}]
[**-W**[**no**]**include_whole**]
[**-W**[**no**]**innerlist**]
[**-W**[**no**]**inspect**]
[**-Wlines**=*n*]
[**-W**[**no**]**list**]
[**-W**[**no**]**map**]
[**-Wmigration_check**]
[**-WmoduleCatalog**="*catalog_spec*"]
[**-WmoduleGroup**["[*group_spec*]"]]
[**-WmoduleSchema**="*schema_spec*"]
[**-WmoduleTableSet**["[*tableset_spec*]"]]
[**-WmoduleVersion**["[*version_spec*]"]]
[**-Wmxcmp**["*arg*[**,**...]"]]
[**-Wmxcmp_add**="*arg*[**,**...]"]
[**-Wmxcmp_files**="*file*[**,**...]"]
[**-Wmxcmp_querydefault**="*attr_name*=*attr_value*[**,**...]"]
[**-Wnostdlib**]
[**-Woptimize**=*n*]
[**-W**[**no**]**optional_lib**]
[**-Wr**]
[**-W**[**no**]**reexport**]
[**-Wrunnamed**]
[**-W**[**no**]**saveabend**]
[**-Wsavetemps**]
[**-Wsettog**=*n*[**,** *n*] ...]
[**-Wsql**["*arg*[**,**...]"]]

[**-Wsqlcomp**[="*arg*[**,**...]"]]
[**-Wsqlmx**[="*arg*[**,**...]"]]
[**-Wsqlmxadd=**"*arg*[**,**...]"]
[**-W**[**no**]**suppress**]
[**-Wstandard=**{**1985** | **2002** }]
[**-Wsyntax**]
[**-Wsystype=**{**guardian** | **oss**}]
[**-Wtimestamp=***value*]
[**-Wu=**"*symbol_name*"]
[**-Wv**]
[**-Wverbose**]
[**-Ww**]
[**-Wx**]
operand ...

FLAGS

-c | **-Wnolink** Performs compilation of the specified source files but suppresses the linking phase. This flag does not delete any object files that are produced.

For source files of the form *file***.cbl**, creates object files with names of the form *file***.o** in the current directory.

Use this flag when an SQL compiler is to be invoked without having to rebuild the executable file.

-g Produces symbols information for symbolic debugging in the object or executable files. This is equivalent to specifying the SYMBOLS and INSPECT directives to the ECOBOL compiler.

-L *directory* Changes the algorithm for searching the libraries named in the **-l** flags to look in the directory named by the *directory* pathname before looking in the default directories **/lib**, **/usr/lib**, and **/usr/local/lib**. Directories named in **-L** options are searched in the order specified.

The order of specifying the **-l** and **-L** flags is significant. If the **-L** flag is specified, it should be specified before specifying any of the following flags, to affect the processing of **-l** flags related to those flags:

-WBdllsonly, **-WBdynamic**, or **-WBstatic**
-Wshared

-l *library* Specifies the filename of a library file to be used for linking. This flag can be specified more than once in a command line and is normally used following specification of **-WBdllsonly**, **-WBdynamic**, **-WBstatic**, or **-Wshared**.

In static linking mode, specifying this flag instructs the linker to search for the library named **lib***library***.a**. In dynamic linking mode, specifying this flag instructs the linker to search for the library named **lib***library***.dll** or **lib***library***.so**; if **lib***library***.dll** or **lib***library***.so** is not found, use **lib***library***.a**.

The position of **-l** *library* operands within a list of flags affects the order in which libraries are searched.

The order of specifying the **-l** and **-L** flags is also significant. If the **-L** flag is specified, it should be specified before specifying any of the following flags, to affect the processing of **-l** flags related to those flags:

-WBdllsonly, **-WBdynamic**, or **-WBstatic**
-Wshared

-O [*optlevel*] Specifies the optimization level to be used for the program file using one of the following values:

0 Specifies an OPTIMIZE 0 ECOBOL compiler directive

1 Specifies an OPTIMIZE 1 ECOBOL compiler directive

2 or no *optlevel* value
Specifies an OPTIMIZE 2 ECOBOL compiler directive

If a **-O** flag is not specified, an OPTIMIZE 1 ECOBOL compiler directive is specified.

-o *outfile* Uses the pathname *outfile* instead of the default pathname **a.out** for the executable file produced.

-s Strips symbolic and other information not required for proper execution from object and executable files. If both the **-g** and **-s** flags are used, symbolic information is kept in the object files but is stripped from the executable file. Do not specify the **-s** and **-Wsql** flags in the same **ecobol** invocation.

-Wansistreams
Generates a program that opens text files as file code 180 files instead of file code 101 (EDIT) files when a program is compiled for the Guardian environment and includes C or C++ modules compiled with the **c89** or **c99 flag** **-Wsystype=guardian**. (By default Guardian C or C++ modules open text files as file code 101 files.) This flag is ignored if **-Wsystype=oss** is specified. OSS C or C++ modules can open text files only as file code 180 files.

-WBdllsonly | **-WBdynamic** | **-WBstatic**
Specifies the type of linking to be performed:

-WBdllsonly Specifies that the **eld** linker should limit searches to position-independent code (PIC) files that are dynamic-link libraries (DLLs) when resolving the file names specified for the **-l** and **-L** flags.

If a file name is qualified, **eld** searches for a DLL with that name.

If a filename is unqualified, in each search path, **eld** first searches for a DLL with the file name as specified in the **-l** or **-L** flag. If **eld** cannot find a DLL, the file name is unqualified, and the search path is not in the Guardian file system (/G), then **eld** prefixes **lib** and suffixes **.so** to the file name and searches again. If **eld** still cannot find the DLL, it searches the path again with the same prefix but with **.dll** as the suffix. For more information on search paths, see the **Finding Libraries** subsection of the **eld(1)** reference page under **DESCRIPTION**.

When a DLL cannot be found, **eld** issues an error message unless its **-allow_missing_libs** flag is specified.

The **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags are search control toggles. Multiple flags can be specified in a single **eld** invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** and **-L** flags and search for just archive files for others. The default library search control is **-WBdynamic**.

-WBdynamic Specifies that the linker utility should use dynamic linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Dynamic linking is in effect until a **-WBstatic** flag is specified. **-WBdynamic** is the default setting. Refer to the **Differences Between Dynamic and Static Linking** subsection for details.

-WBstatic Specifies that the linker utility should use static linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Static linking is in effect until a **-WBdynamic** flag is specified. **-WBdynamic**, not **-WBstatic**, is the default setting. Refer to the **Differences Between Dynamic and Static Linking** subsection for details.

You cannot use these flags if you use the **-c** or **-Wnolink** flag.

-Wcall_shared | **-Wshared**
Specifies the kind of linked file that should be created:

-Wcall_shared Directs the compiler to create a position-independent code (PIC) program loadfile using the **eld** linker. If you also specify the **-c** or **-Wnolink** flag, the file created is a PIC linkfile instead.

This is the default behavior.

-Wshared Directs the compiler to create a PIC dynamic-link library (DLL) using the **eld** linker.

-Wcobol="*arg*[**,**...]**"**
Passes to the ECOBOL compiler the directives in the argument string enclosed in quotation marks. If more than one value is specified, they must be separated by commas without any white space. This string follows any directives generated by other flags. If you repeat this flag, arguments are passed to the compiler in the order specified.

-Wcodecov Directs the compiler to create an instrumented object file and to create or add to an existing SPI file. This flag has an effect only if you also specify either the **-Wtarget=ipf** flag or the **-Wtarget=tns/e** flag.

The first time the **-Wcodecov** flag is used to compile a program, the compiler creates a Static Profiling Information (SPI) file. This file is one of the input files for the Code Coverage tool. If the program is compiled in an OSS directory:

- The default name for the SPI file is **pgopti.spi**.
- If the default file is not write-accessible, the name of the SPI file created is **tpopti.spi**.
- A lock file called **pgopti.spl**. When compilation is complete, the compiler deletes this file.

If the program is compiled in a Guardian directory:

- The default name for the SPI file is **pgospi**.
- If the default file is not write-accessible, the name of the SPI file created is **tpgospi**.

- A lock file called **pgospl**. When compilation is complete, the compiler deletes this file.

If the SPI file already exists when the program is compiled with the **-Wcodecov** flag, the compiler updates or adds information to the existing SPI file. If more than one SPI file exists for the same program, you must concatenate the files manually before you can use the resulting file as input to the Code Coverage Tool.

For more information about the Code Coverage Tool, see the *Code Coverage Tool Reference Manual*.

-Wcolumns=*n* Sets the maximum number of columns for an input file to *n*, where *n* is a number in the range 12 through 32767. If *n* is greater than 132, 132 is used. The compiler ignores text in columns beyond *n*.

-Wcopylib="*pathname1***"**
Specifies *pathname1* as the source file to use as the default COPY library for any COPY statement in the source program that does not specify a library. If you repeat this flag, the last file specified is the default COPY library. The default is to look for a file called COPYLIB in the current working directory.

-Wdryrun Verifies the syntax and semantics of the flags and operands that were specified and enables the **-Wv** flag. No compilation system components are run.

-Weld="*arg***[,...]"**
Passes to the **eld** utility the directives in the argument string enclosed in quotation marks after any other arguments are passed. If more than one value is specified, they must be separated by commas without any white space. If you repeat this flag, arguments are passed to the **eld** utility in the order specified.

This flag is ignored when linking is suppressed.

-Weld_obey="*pathname2***"**
Passes *pathname2* (a file of **eld** utility commands) to the **eld** utility.

This flag is ignored when linking is suppressed.

-Werrors=*n* Stops compiling when *n* errors have been encountered.

-Wglobalized Specifies that the code generated by the compiler is preemptable. By default, compilers generate code that is not preemptable. Preemptable code allows named references in a DLL to resolve to externally-defined code and data items instead of to resolve to its own internally-defined code and data items. Preemptable code is less efficient than code that is not preemptable, and is only needed in a few instances when creating a DLL.

This flag has an effect only if you also specify either the **-Wtarget=ipf** flag or the **-Wtarget=tns/e** flag.

-Wheap=*n***[b | w | p]**
Specifies the value that the linker should use for the HEAP_MAX attribute of the output file. *n* can be any positive hexadecimal value that gives a size valid for the NonStop server node on which the file is used. The size can be specified in units of:

b Bytes; this is the default unit

w Words

p Pages

-Whelp | **-Wusage**
Displays information on how to run the **ecobol** utility. No compilation system components are run.

-Whighpin={**on** | **off** }
Directs the linker to set the HIGHPIN attribute to **on** or **off** in the output object files. This attribute specifies whether the object file will run at a high PIN or a low PIN.

If the program is compiled for execution in the Guardian environment, the default setting is **-Whighpin=off**. If the program is compiled for execution in the OSS environment, the default setting is **-Whighpin=on**. This flag is set only if an executable object file is produced.

-Whighrequesters={**on** | **off** }
Directs the linker to set the HIGHREQUESTERS attribute to **on** or **off** in the output object file. This attribute specifies whether the object file supports requests from requesters running at a high PIN.

The object file must contain a COBOL main program. If the COBOL main program was compiled with the **ecobol -Wsystype=guardian** flag set, the default setting is **-Whighrequesters=off**. If the COBOL main program was compiled with the **ecobol -Wsystype=oss** flag set, the default setting is **-Whighrequesters=on**. This flag is set only if an executable object file is produced.

-W[**no**]**include_whole**
Tells the **eld** linker whether to include in the loadfile all linkable archive members of all archive libraries encountered after this flag is specified.

Specifying **-Winclude_whole** begins this linking action. When **-Wnoinclude_whole** behavior is in effect, archive searches are controlled by the existence of undefined symbols. Archives are searched in the order specified on the command line. Symbols are marked as undefined by compilers or by the user through the **-Wu** flag or the **eld** linker **-u** flag. When an archive member is found that resolves an undefined symbol, the member´s symbols are merged into the external symbol table for the loadfile being created. After the merge, the undefined symbol that triggered the merge is resolved (marked as defined). The same merge might resolve other undefined symbols or result in more undefined symbols.

You can stop the linking action of **-Winclude_whole** by specifying the **-Wnoinclude_whole** flag later in the command line or an obey file.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

The default setting is **-Wnoinclude_whole**.

-W[**no**]**innerlist**
Enables [disables] the generation of instruction code mnemonics in the compiler listing immediately following each corresponding statement. This flag works only if the **-Wnosuppress** flag is specified. The default is **-Wnoinnerlist**.

-W[**no**]**inspect** Designates [does not designate] the Native Inspect debugger as the default debugger for the output object file. Use this flag with the **-g** flag. The default setting is **-Wnoinspect**. This flag is set only if an executable object file is produced.

-Wlines=*n* Sets the maximum number of lines on a listing page to *n*, if a listing is generated. *n* is a number in the range 10 through 32767.

-W[**no**]**list** Temporarily enables [disables] the generation of the compiler listing. This flag works only if the **-Wnosuppress** flag is specified. The default is **-Wlist**.

-W[**no**]**map** Temporarily enables [disables] the generation of identifier maps in the compiler listing. This flag works only if the **-Wnosuppress** flag is specified. The default is **-Wnomap**.

-Wmigration_check
When used with the -Wstandard=1985 flag, causes the compiler to issue a warning message when it encounters a user-defined COBOL word that is a reserved word in the COBOL-2002 standard. The default is not to issue migration warning messages.

-WmoduleCatalog="*catalog_spec***"**
Specifies a NonStop SQL/MX module catalog name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a catalog name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor release 2.0 and newer.

-WmoduleGroup[**="**[*group_spec*]**"**]
Specifies a string for a module group specification to use as a prefix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a group name. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-WmoduleSchema="*schema_spec***"**
Specifies a NonStop SQL/MX module schema name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a schema name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor version 2.0 and newer.

-WmoduleTableSet[**="**[*tableset_spec*]**"**]
Specifies a string for a tableset specification to use as the first suffix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a tableset name The string cannot contain more than 31 characters.

This flag is valid only for preprocessor version 1.8 and newer.

-WmoduleVersion[**="**[*version_spec*]**"**]
Specifies a string for a tableset specification to use as the second suffix to the externally qualified module name that is written to the module file. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-Wmxcmp[="*arg*[**,**...]"]
Invokes the NonStop SQL/MX compiler after the NonStop SQL/MX preprocessor is invoked.

If a value is supplied for *arg*, it must be one of the following:

replace Directs the NonStop SQL/MX compiler to replace the existing module or create a new one. The default action does not replace an existing module.

warn Directs the NonStop SQL/MX compiler to generate a warning rather than an error if a table does not exist at compilation time.

verbose Directs the NonStop SQL/MX compiler to display summary information as well as error and warning messages.

If the **-Wmxcmp** flag is specified more than once, only the last occurrence has an effect. If the **-Wmxcmp** flag is specified without the **-Wsqlmx** flag, and if a file specified for *operand* has a name of the form *file***.m**, that file is passed to the NonStop SQL/MX compiler.

If the **-Wmxcmp** flag is specified, you cannot use the **-Wsql** or **-Wsqlcomp** flag. The **-Wmxcmp** flag is ignored when a flag, such as **-Wsyntax**, that prevents compilation is specified.

-Wmxcmp_add="*arg*[**,**...]**"**
Specifies a string to pass to the NonStop SQL/MX compiler without validation or change. If more than one value is specified, they must be separated by commas without any white space.

-Wmxcmp_files="*file*[**,**...]**"**
Passes MDF files specified to **mxcmp** in release 1 compilation mode. Passes all specified files without the **.m** extension to **mxCompileUserModule** in release 2 compilation mode.

-Wmxcmp_querydefault="*attr_name***=***attr_value*[**,**...]**"**
Specifies attribute settings (CONTROL QUERY DEFAULT settings) to pass to the NonStop SQL/MX compiler. These attribute settings override any corresponding entries in the **SYSTEM_DEFAULTS** table.

-Wnostdlib Suppresses the searching of the standard library directories to locate libraries for any C or C++ modules in the program. Refer to the **Standard Library Directories** subsection of the **c89** or **c99 reference page for details.**

-Woptimize=*n* Sets the optimization level to *n*. *n* is one of the following:

0 Disables all optimizations and therefore yields code with relatively poor performance. Optimization level 0 is useful when you are developing and debugging your program and is recommended for serious debugging. Statements are well-defined when debugging; breakpoints and stepping occurs in a manner that the user would expect when viewing the related source.

1 Generates partially optimized code sequences. Object code compiled at optimization level 1 can be symbolically debugged with the Visual Inspect debugger; however, statement boundaries might be blurred. The Visual Inspect debugger chooses a sensible location when a user requests a breakpoint on a source statement,

but its definition of statement boundaries does not always coincide directly with source statements. The debugger emits a warning when a process is held at a statement for which the code associated with a previous source statement has not yet executed.

2 Generates fully optimized code sequences. Machine-level debugging might be required, because symbolic debugging capability will be limited.

The default is 1.

-W[**no**]**optional_lib**

Indicates whether a library specified in the command stream should be considered optional when the **eld** linker creates a loadfile.

When **-Wnooptional_lib** behavior is in effect, any library specified in a **-l** or **-lib** flag is included in the **.liblist** section of the loadfile being created. When **-Woptional_lib** behavior is in effect, a specified library can be omitted from the **.liblist** section of the loadfile being created if omitting it would not affect how symbolic references are resolved.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

If a library is specified more than once, and at least one specification occurs when **-Wnooptional_lib** is in effect, the library is included in the **.liblist** section of the loadfile being created.

The default behavior is **-Wnooptional_lib**.

-W[**no**]**reexport**

Tells the **eld** linker whether to mark any library specified in an **-l** or **-L** flag after this flag for reexport in its **libList** entry in the loadfile being created. Specifying **-Wnoreexport** leaves the library unmarked; specifying **-Wreexport** marks the library. Reexport is a run-time attribute that is used by the **rld** loader to decide what DLLs it needs to load.

-Wnoreexport is the default action.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

-Wrunnamed Directs the linker to set the RUNNAMED ON attribute in the current object file. This attribute specifies that the object file runs as a named process. The default is RUNNAMED OFF.

-W[**no**]**saveabend**

Specifies that a process snapshot (saveabend) file is [not] created if the program terminates abnormally. The default is **-Wnosaveabend**.

-Wsavetemps Saves all temporary and intermediate files created by compilation system components. Use the **-Wv** flag to display the filenames.

-Wsettog=*n*[**,** *n*] ...

Specifies a numeric toggle in the range 1 through 15 that is defined only during the NonStop SQL/MX preprocessing step. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about the NonStop *SQL/MX* **-d** *toggle* option.

All **-Wsettog** values that are supplied to **ecobol** are automatically passed as **-d** options to the NonStop SQL/MX preprocessor. The **-d** options control the processing of ?IF directives by the preprocessor; the options do not pass ?SETTOG directives to the ECOBOL compiler.

This flag is ignored unless the **-Wsqlmx** flag is also specified. This flag can be specified more than once.

-Wsql[="*arg*[**,**...]"]
Enables NonStop SQL/MP support when processing COBOL85 source files, using the arguments in the argument string enclosed in quotation marks. If more than one value is specified, they must be separated by commas without any white space. Refer to the NonStop *SQL/MP Programming Manual for COBOL85* for a description of the arguments that can be passed to the NonStop SQL/MP compiler. If no errors occur, **-Wsql** also runs the SQLCOMP compiler after the link step.

If you specify the **-Wsql** flag, you cannot use the **-s**, **-Wmxcmp**, or **-Wsqlmx** flag.

-Wsqlcomp[="*arg*[**,**...]"]
Runs the NonStop SQL/MP SQLCOMP compiler after the link step, using the arguments specified. If more than one value is specified, they must be separated by commas without any white space.

If you specify the **-Wsqlcomp** flag, you cannot use the **-Wmxcmp** or **-Wsqlmx** flag.

-Wsqlmx[="*arg*[**,**...]"]
Invokes the NonStop SQL/MX **mxsqlco** preprocessor, using the arguments specified. If an *arg* value is specified, it must be one of the following; if more than one value is specified, they must be separated by commas without any white space:

ansi_format Directs the preprocessor to assume ANSI fixed format for the source file that it reads.

double_quotes Directs the preprocessor to accept SQL string literals delimited by double quotes in addition to literals delimited by single quotes.

listing Directs the preprocessor to write its diagnostic messages to a file named *file***.eL**, where *file* is the name of the primary source file.

preprocess_only
Directs the preprocessor to suppress all steps after preprocessing.

This option is valid only for preprocessor release 2.0 and newer.

refrain_r2 Directs the SQL/MX preprocessor to use only the rules and features that apply to preprocessors prior to release 2.0. The default action is to use only the rules and features that apply to preprocessors beginning with release 2.0.

This option is valid only for preprocessor release 2.0 and newer.

If you specify the **-Wsqlmx** flag, you cannot use the **-Wsql** or **-Wsqlcomp** flag.

-Wsqlmxadd="*arg*[**,**...]"
Specifies a string to pass to the SQL/MX preprocessor without validation or change. If more than one value is specified, they must be separated by commas without any white space.

-W[**no**]**suppress**
Disables [enables] the generation of identifier maps in the compiler listing. The compiler listing is written to standard output. The default is **-Wsuppress**.

-Wstandard={**1985** | **2002** }
Specifies the COBOL standard the compiler should use. The default is **1985**, which specifies that the compiler follow the COBOL-1985 standard. When **2002** is specified, the compiler reserves all the COBOL words that are reserved words in the COBOL-2002 standard. The compiler does not support all COBOL-2002 features. For a list of the features supported, see the *COBOL Manual for TNS/E Programs*.

-Wsyntax Checks the syntax of the source program, but does not generate any code.

-Wsystype={**guardian** | **oss** }
Specifies the target execution environment. This flag selects definitions used during compilation, program startup code, default libraries, and system routines used during linking. The default setting is **-Wsystype=oss**. (To run files compiled for a Guardian target execution environment, you must set the file code to 800 with a `FUP ALTER` *filename*`, CODE 800` command from a TACL prompt.)

-Wtimestamp=*value*
Provides a creation timestamp for the NonStop SQL/MX preprocessor that is written to the two output files created by the preprocessor. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about the formats allowed for *value*. If this flag is specified more than once, only the last occurrence has an effect. Note that **ecobol** does not check that *value* is valid; it relies on the NonStop SQL/MX preprocessor to validate this argument.

This flag is ignored unless the **-Wsqlmx** flag is also specified.

-Wu="*symbol_name***"**
Tells the **eld** linker to add *symbol_name* as an undefined symbol. This causes **eld** to search for this symbol in any archive libraries that are specified after this flag on the command line or in an obey file.

The search constraint specified by the **-Wu** flag is overridden by use of the **-Winclude_whole** flag.

-Wv Echoes to the standard error file the command line as each component of the compilation system is run.

-Wverbose Displays detailed information from the ECOBOL compiler and linker utility.

-Ww Suppresses the printing of compiler warning messages.

-Wx Strips part of the symbol table from the output object file, but keeps information necessary for the object file to be used as input to the linker utility.

Do not include a space before or after the = (equal sign).

Quotation marks around string values in flags are optional but recommended to avoid errors caused by shell substitutions or deletions.

DESCRIPTION

The **ecobol** utility is the interface to the ECOBOL compilation system; it accepts source code conforming to the ISO COBOL85 standard. The system consists of an ECOBOL compiler and a linker utility (**eld**), with additional program components supporting SQL.

ecobol performs simple validation of the flags and operands on its command line and, depending on those items, invokes components of the language compilation system. **ecobol** does not verify the existence of files it passes to compilation system components. It does verify that *operand* identifies valid files to pass to compilation system components. **ecobol** and the components it runs issue messages to the standard error file.

ecobol performs the following steps:

1. If the **-Wsqlmx** flag is specified, invokes the NonStop SQL/MX preprocessor to preprocess any COBOL source files that contain embedded NonStop SQL/MX statements to create either of the following:
    - COBOL source files with module definitions (using the release 2 module management method)
    - COBOL-only source files and module definition files (MDFs) (using the release 1 module management method)
2. Compiles any specified COBOL source files or source files produced by Step 1 into object files.
3. If the **-Wmxcmp** flag is specified, invokes the NonStop SQL/MX compiler to compile any module definitions or MDFs.
4. Links the object files together with any libraries specified on the command line. This step occurs if no flags that prevent linking (such as **-c** or **-Wnolink**) are specified and if the source files are compiled without errors.
5. If the **-Wsqlcomp** flag is specified, invokes the NonStop SQL/MP compiler to process any embedded NonStop SQL/MP statements in files created by Step 1 or specified in the command.
6. Writes an executable object file or dynamic link library (DLL) specified by the **-o** flag (if present) or to the file **a.out**.

The files specified in the *operand* list are operated on by the appropriate program components of the compilation system, depending on the command line flags and the type of file operands.

If the **-c** flag is specified, then for all pathname operands of the form *file***.cbl**, the files **$(***basename pathname***.c).o** are created as the result of successful compilation.

If **-c** is not specified, the object files created after successful compilation are combined by the link operation into a program file, dynamic-link library (DLL), or user library. When linking is performed and either the **-Wsqlmx** or **-Wmxcmp** flag is specified, the list of libraries searched automatically includes **zclidll**. Object files created are not deleted after successful generation of the executable program file.

The executable file is created according to OSS file creation rules, except that the file permissions are set to **S_IRWXO | S_IRWXG | S_IRWXU** and the bits specified by the **umask** value of the process are cleared.

HP Extensions

The **-W** flags are specific to HP for supporting the HP compilation environment. The argument strings within these flags are passed to the program components unchanged, along with default argument strings and argument strings corresponding to **ecobol** command line flags meaningful to the program components. Do not specify conflicting instructions in **-W** flag argument strings or **ecobol** command line flags. The results of conflicting instructions are undefined.

Operands

An *operand* is a pathname. At least one pathname must be specified. The following operands are supported:

file.**a**	A library of object files typically produced by the **ar** command, and passed directly to the linker utility.
file.**cbl**	A COBOL85 language source file to be compiled and optionally linked. Embedded NonStop SQL/MP information might be present.
file.**cob**	A COBOL85 language source file to be compiled and optionally linked. Embedded NonStop SQL/MP information might be present.
file.**ECBL**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**ecbl**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**ECOB**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**ecob**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**m**	A module definition file (MDF) containing NonStop SQL/MX information for a corresponding COBOL source file.
file.**o**	An object file produced by a previous ECOBOL compilation, to be passed directly to the linker utility.

Input Files

Input files are one of the following:

- A text file containing a COBOL85 language source program
- An object file in the format produced by the command **ecobol -c**
- A library of object files in the format produced by archiving zero or more object files using the **ar** command
- A library of object files produced by the **eld** utility

When **-Wsqlmx** is specified, **ecobol** uses the source file extension to determine whether a file requires preprocessing and the names of the source files created by the NonStop SQL/MX preprocessor. The name of the source file created is the name of the primary source file with the following transformation to the file extension:

- Each source file with the extension **.ecbl** or **.ecob** is given to the **mxsqlco** program for preprocessing. The resulting source files have the extensions **.cbl** and **.m**, where the file named *file*.**cbl** contains the COBOL source to be compiled and the file named *file*.**m** contains the corresponding module definition file (MDF).

- Source files with the extensions **.cbl** or **.cob** are not given to the **mxsqlco** program; these files are assumed to contain no embedded SQL statements.

Files created by the NonStop SQL/MX preprocessor overwrite any existing files with the same name in the current working directory.

Output Files

Output files are object files, executable files, log files, NonStop SQL/MX module definition files created by the NonStop SQL/MX preprocessor, or all four. Log files have names of the form *file***.eL**. Module definition files have names of the form *file***.m**.

Standard Output

The standard output file is a text file that contains the compiler listing, if generated.

Standard Error

The standard error file is used for diagnostic and informational messages. If more than one file operand is specified, then for each such file, `"%s: \n",`*file* might be written. These messages precede the processing of each input file.

Environment Variables

The following environment variables affect the execution of **ecobol**. The **ecobol** utility and its program components do not support locale variables.

COMP_ROOT

Changes the default pathnames for the **ecobol** compilation system components. In the OSS environment, the string specified in **COMP_ROOT** is added to the beginning of the default pathnames. If a component's environment variable is set explicitly, the **COMP_ROOT** environment variable does not modify the component's environment variable.

ECOBFE Determines the pathname of the **ecobol** compiler. **/G/system/system/ecobfe** is the default.

ELD Determines the pathname of the **eld** utility invoked by **ecobol**. **/usr/bin/eld** is the default location for the OSS environment.

MXCMP Determines the pathname of the NonStop SQL/MX release 1 compiler. **/G/system/system/mxcmp** is the default.

MXCMPUM Determines the pathname of the NonStop SQL/MX release 2 compiler. **/usr/tandem/sqlmx/bin/mxCompileUserModule** is the default.

MXSQLCO Determines the pathname of the NonStop SQL/MX preprocessor, **mxsqlco**. **/usr/tandem/sqlmx/bin/mxsqlco** is the default.

SQLCOMP Determines the pathname of the NonStop SQL/MP compiler invoked by **ecobol**. **/G/system/system/sqlcomp** is the default.

SQLCIO Determines the pathname of the object file for the NonStop SQL/MX application program interface to the ECOBOL compiler. **/usr/tandem/lib/sqlci.o** is the default.

SQLMX_PREPROCESSOR_VERSION

Indicates the preprocessor rules and features to be used. Specifying the value 800 causes rules and features associated with release 1.8 to be used; the **mxcmp** compiler is used and only MDF files and annotated source files are produced, while rules and features associated with release 2.0 and later are ignored. Specifying a value of 1200 or larger or not specifying a value causes rules and features associated with release 2.0 and later to be used; the **mxCompileUserModule** compiler

is used and annotated source files that contain embedded module definitions are produced instead of MDF files, while restrictions associated with release 1.8 or earlier are ignored.

TMPDIR Determines the pathname that overrides the default directory for temporary files created by **ecobol** and components it invokes. By default, temporary files are stored in the **/tmp** directory. If **TMPDIR** is set to a directory that does not exist or is not writable, **ecobol** uses the default directory as described on the **tempnam(3)** reference page.

Processes

With the exception of the ECOBFE and SQLCOMP processes, which is invoked as a Guardian process, all components are invoked as OSS processes.

Standard Libraries

The following libraries are available for COBOL85 programs in the OSS environment.

-l cob Contains COBOL library and utility routines described in the *COBOL Manual for TNS/E Programs*.

-l cre Contains C run-time library routines.

-l cli Contains SQL/MX support routines.

In the absence of flags that inhibit invocation of a linker utility, such as **-c** or **-Wnolink**, **ecobol** passes **-l cob** and **-l cre** operands to the linker utility, causing the COBOL and CRE libraries to be searched. If the **-Wsqlmx** or **-Wmxcmp** flags are present, **ecobol** also passes a **-l cli** operand to the linker utility, causing the SQL/MX support library to be searched.

If you want the libraries to be searched in a specific order or you want linking options to be processed in a specific order, you should invoke the linker using the **eld** command from the OSS shell and not use **ecobol** to do the linking.

Differences Between Static and Dynamic Linking

The **-WBdllsonly** and **-WBdynamic** operands specify dynamic linking. The **-WBstatic** operand specifies static linking.

In dynamic linking, the **eld** utility first searches for a dynamic-link library (DLL). If a DLL cannot be found, the linker utility searches for an archive file. If no archive file can be found, an error is issued.

In static linking, the linker utility searches for an archive file but does not search for a DLL. If the archive file cannot be found, an error is issued.

Dynamic and static linking are not exact opposites. Dynamic linking accepts either a DDL or an archive file, but static linking accepts only an archive file.

Unlike **ecobol** flags, multiple **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** operands can be specified in a single **ecobol** invocation; thus, it is possible to perform dynamic linking for some **-l** operands and static linking for others.

-WBdllsonly, **-WBdynamic** and **-WBstatic** operands specified to **ecobol** are temporarily overridden by linking arguments specified in the **-Weld** or**-Weld_obey** flags.

Refer to the **eld(1)** reference page for more information.

Using the c89 or c99 and ecobol Utilities

OSS COBOL85 programs can contain COBOL85 modules and C modules. Compile COBOL85 modules using the **ecobol** utility and C modules using the **c89** or the **c99** utility. To produce a program containing COBOL85 and C modules, first compile all the modules written in either COBOL85 or C. You can also link these modules together or with other libraries at this time, but do not SQL-compile the modules. After you have compiled all the modules of one language, compile the modules written in the other language, specifying any necessary linking or SQL-compiling options.

For example, to produce an executable object file made up of COBOL85 modules **cobol1.cbl** and **cobol2.cbl** and C modules **c1.c** and **c2.c**, you can first run the C compiler using the **c89** utility with:

```
c89 -c -o cprog.o c1.c c2.c
```

This command directs **c89** to compile the two modules but not link them. The output object file is **cprog.o**.

You can then invoke the **ecobol** utility to compile the two COBOL85 modules and link the ECOBOL compiler output with the previously produced C object file and the standard C library to produce the executable object **myprog** with:

```
ecobol -o myprog cprog.o cobol1.cbl cobol2.cbl
```

Refer to the *C/C++ Programmer's Guide* and the *Open System Services Programmer's Guide* for details on writing and compiling C programs in the OSS environment.

EXAMPLES

1. The command

   ```
   ecobol test1.cbl
   ```

   compiles the source file **test1.cbl** and links the object file into a program file **a.out**.

2. The command

   ```
   ecobol -c test1.cbl
   ```

   compiles the source file **test1.cbl** into an object file **test1.o**.

3. The command

   ```
   ecobol -g -o test2 x.cbl y.cbl z.cbl
   ```

   compiles source files **x.cbl**, **y.cbl**, and **z.cbl** and links the object files into a program file **test2**. Symbolic information is generated by the compiler and retained by the linker utility for debugging.

4. The command

   ```
   ecobol -o xyz -Wsqlmx x.o y.o z.o
   ```

   links the object files **x.o**, **y.o**, and **z.o** into a program file **xyz**. The NonStop SQL/MX compiler is then invoked to compile **xyz**.

5. The command

   ```
   ecobol -Wnolink -Wsql="catalog \$abc.def" xyz
   ```

   invokes the NonStop SQL/MP compiler, **sqlcomp**, on program file **xyz** without going through the linking process. In addition to the input filename **xyz**, the catalog option is passed to the NonStop SQL/MP compiler.

6. The command

   ```
   ecobol -o testprog -L .  -L /usr/test/lib testprog.cbl -l tdm
   ```

   compiles the COBOL85 language source program **testprog.cbl** and links the object file with the library specified in the **-l** operand. It also links the object file with a DLL, if found. If a DLL is not found, it uses the standard C run-time library. The **eld** utility produces a program file named **testprog**.

   By default, dynamic linking is selected. **ecobol** searches directories for the library **tdm** specified by the **-l** flag in the following order and selects the first copy found:

   libtdm.so in the current directory (**-L .**)

   libtdm.a in the current directory (**-L .**)

   libtdm.so in **/usr/test/lib** (**-L /usr/test/lib**)

   libtdm.a in **/usr/test/lib** (**-L /usr/test/lib**)

   libtdm.so in **/lib** (by default)

   libtdm.a in **/lib** (by default)

   libtdm.so in **/nonnative/usr/lib** (by default)

   libtdm.a in **/nonnative/usr/lib** (by default)

   libtdm.so in **/usr/lib** (by default)

   libtdm.a in **/usr/lib** (by default)

   libtdm.so in **/usr/local/lib** (by default)

   libtdm.a in **/usr/local/lib** (by default)

7. The command

   ```
   ecobol -Wsqlmx -Wmxcmp -o sqlprog.exe sqlprog.ecbl
   ```

   processes the single COBOL module named **sqlprog.ecbl** containing embedded NonStop SQL/MX statements using the release 2 compilation method as follows:

   a. The NonStop SQL/MX preprocessor is invoked to process the source file. The preprocessor creates the file **sqlprog.cbl**. The file **sqlprog.cbl** is the COBOL-only equivalent of **sqlprog.ecbl**; that is, the preprocessor translates all embedded NonStop SQL/MX statements to the appropriate COBOL code.

   b. If no errors occurred in Step a, the ECOBOL compiler processes the file **sqlprog.cbl** to create the file **sqlprog.o**.

   c. If no errors occurred in Step b, the NonStop SQL/MX compiler is invoked to process the module definitions.

   d. If no errors occurred in Step c, **eld** is invoked to link the file **sqlprog.o** with the standard COBOL library and produces the executable file **sqlprog.exe**.

8. The command

   ```
   ecobol -c -Wsqlmx file1.ecob file2.ecob file3.ecob
   ```

   preprocesses the three specified files and also compiles them, but does not link the resulting object files. Using the release 2 module management method, if no errors are detected during either preprocessing or compilation, the following files are created: **file1.cob**,

file2.cob, **file3.cob**, **file1.o**, **file2.o**, and **file3.o**.

9. The command

   ```
   ecobol -c -Wsqlmx file1.cbl file2.ecbl file3.ecob file4.cob
   ```

   mixes COBOL source files with and without embedded NonStop SQL/MX statements. All files are compiled but not linked. Using the release 2 module management method, if no errors are detected during either preprocessing or compilation, the following files are created: **file2.cob**, **file3.cob**, **file1.o**, **file2.o**, **file3.o**, **file4.o**.

10. The command

    ```
    ecobol -Wmxcmp -Wmxcmp_files="test1.m,test1.o"
    ```

    SQL-compiles the MDF file **test1.m** using the NonStop SQL/MX **mxcmp** compiler and processes the file **test1.o** using the NonStop SQL/MX **mxCompileUserModule** without also linking it.

DIAGNOSTICS

If **ecobol** encounters a compilation error that prevents an object file from being created, it writes a diagnostic message to the standard error file and continues to compile other source code operands; however, it does not perform program linking and returns a nonzero exit status. If the linking is unsuccessful, cobol writes a diagnostic message to the standard error file and returns a nonzero exit status.

EXIT VALUES

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

RELATED INFORMATION

Commands: **ar(1)**, **c89(1)**, **c99(1)**, **eld(1)**, **nmcobol(1)**, **strip(1)**.

Functions: **tempnam(3)**.

STANDARDS CONFORMANCE

The **ecobol** utility is an HP extension to the XPG4 Version 2 specification.

NAME

ed - Edits a file line by line

SYNOPSIS

ed [**-p** *string*] [**-s**] [*file*]

The **ed** command invokes a line-editing program that works on one file at a time by copying it into a temporary edit buffer and making changes to that copy.

FLAGS

-p *string* Sets the **ed** prompt to *string*. The default value for *string* is null (no prompt).

-s Suppresses byte counts that the editor displays with the **e**, **E**, **r**, and **w** subcommands, suppresses diagnostic messages for the **e**, **E**, and **q** subcommands, and suppresses the **!** (exclamation point) prompt after a **!***system_command*. The same things are suppressed when you call **ed** with a filename.

Note that if you quit without writing the file, you do not get an error message. (Generally, before ending the program, the **q** subcommand checks to determine whether the buffer was written to a file since the last time it was changed. If not, **q** displays the ? message.)

DESCRIPTION

The **ed** command does not alter the file itself until you use the **w**rite subcommand. You can specify on the command line the *file* you want to edit, or you can use **ed** subcommands to read a file into the buffer. When **ed** reads a new file into the buffer, the contents of that file replace the buffer's previous contents, if any.

The ed Modes

The **ed** program operates in one of two modes, *command mode* and *text mode*. In command mode, **ed** recognizes and executes subcommands. In text mode, **ed** adds text to the file buffer, but does not recognize subcommands. You enter text mode by using the **a**, **c**, or **i** subcommand. To leave text mode, enter **.** (dot) alone at the beginning of a line.

Subcommand Syntax

An **ed** subcommand consists of zero, one, or two addresses, followed by a single-character subcommand, possibly followed by arguments to that subcommand. These addresses specify one or more lines in the buffer. Because every subcommand has default addresses, you frequently do not need to specify addresses.

Pattern Matching

The **ed** command supports a limited form of special *pattern-matching characters* that you can use as *regular expressions* (REs) to construct *pattern strings*. You can use these patterns in addresses to specify lines and in some subcommands to specify portions of a line.

For information about regular expressions (REs), see the reference page for the **grep** command.

Addressing

There are three types of **ed** addresses: line number addresses, addresses relative to the current line, and pattern addresses.

The current line is the point of reference in the buffer and is represented by a **.** (dot). When you start the **ed** program, the current line is the last line in the buffer. As you edit a file, the current line changes to the last line affected by a subcommand. The current line is the default address for several **ed** commands. (See **SUBCOMMANDS** to find out how each subcommand affects the current line.)

Subcommands for determining and changing the current line are described in the section **SUBCOMMANDS**. The following are guidelines for constructing addresses:

- **.** (dot) addresses the current line.
- **$** addresses the last line of the buffer.
- *n* addresses the *n*th line of the buffer.
- ´*x* addresses the line marked with a lowercase letter, *x*, by the **k** subcommand.
- /*pattern*/ addresses the next line that contains a matching string. The search begins with the line after the current line and stops when it finds a match for the pattern. If necessary, the search moves to the end of the buffer, wraps around to the beginning of the buffer, and continues until it either finds a match or returns to the current line.
- **?***pattern***?** addresses the previous line that contains a match for the pattern. The **?***pattern***?** construct, like /*pattern*/, can search the entire buffer, but it does so in the opposite direction.
- An address followed by +*number* or -*number* specifies an address plus or minus the indicated number of lines. The + (plus sign) is optional.
- An address that begins with + or - specifies a line relative to the current line. For example, **-5** is the equivalent of **.-5** (five lines above the current line).
- An address that ends with - or + specifies the line immediately before (-) or immediately after (+) the addressed line. Used alone, the + character addresses the line immediately before the current line. The + character addresses the line immediately after the current line; however, the + character is optional. The + and - characters have a cumulative effect; for example, the address **--*O addresses the line two lines above the current line.**
- For convenience, a **,** (comma) stands for the address pair **1,$** (first line through last line) and a **;** (semicolon) stands for the pair **.,$** (current line through last line).

Commands that do not accept addresses regard the presence of an address as an error. Commands that do accept addresses can use either given or default addresses. When given more addresses than it accepts, a command uses the last (rightmost) ones.

In most cases, a **,** (comma) separates addresses (for example **2,8**). A **;** (semicolon) can also separate addresses. A **;** between addresses causes the **ed** command to set the current line to the first address and then calculate the second address (for example, to set the starting line for a search based on guidelines 5 and 6). In a pair of addresses, the first must be numerically smaller than the second.

The ed Limits

The following is a list of **ed** command size limitations:

- 64 bytes per filename.
- 512 bytes per line.
- 256 bytes per global subcommand list.
- 128-kilobyte buffer size. (Note that the buffer not only contains the original file but also editing information. Each line occupies one word in the buffer.)

The maximum number of lines depends on the amount of memory available to you. The maximum file size depends on the amount of physical data storage (disk or tape drive) available or on the maximum number of lines permitted in user memory.

SUBCOMMANDS

In most cases, only one **ed** subcommand can be entered on a line. The exceptions to this rule are the **n**, **p**, and **l** subcommands, which can be added to any subcommand except **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

The **e**, **E**, **f**, **r**, and **w** subcommands accept filenames as arguments. The **ed** command stores the last filename used with a subcommand as a default filename. The next **e**, **E**, **f**, **r**, or **w** subcommand given without a filename uses the default filename.

The **ed** command responds to an error condition with one of two messages: ? (question mark) or ?*file*. When the **ed** command receives an **INT** signal, it displays a ? and returns to command mode. When **ed** reads a file, it discards NULL characters and all characters after the last newline character.

Unless otherwise noted, all subcommands work by default on the current line; an address is optional. If you specify an address, you do not type the brackets.

When used as an address, a **.** (dot) refers to the current line. When a **.** (dot) is shown in the first position on an otherwise blank line, it terminates text mode and returns to command mode. *text* denotes user input in text mode. Note that *address* need not be a number; it can be a regular expression of the form /*RE*/**,**/*RE*/ or /*RE*/**;**/*RE*/.

[*address*]**a**

text

. The **a** (append) subcommand adds text to the buffer after the addressed line. Enter a **.** (dot) to return to command mode. The **a** subcommand sets the current line to the last inserted line or, if no lines were inserted, to the addressed line. Address **0** causes the **a** subcommand to add text to the beginning of the buffer.

[*address1***,***address2*]**c**

text

. The **c** (change) subcommand deletes the addressed lines, then replaces them with new input. Enter a **.** (dot) to return to command mode. The **c** subcommand sets the current line to the last new line of input or, if there were none, to the first line that was not deleted.

[*address1*,*address2*]**d**
The **d** (delete) subcommand removes the addressed lines from the buffer. The line after the last line deleted becomes the current line. If the deleted lines were originally at the end of the buffer, the new last line becomes the current line.

e *file* The **e** (edit) subcommand first deletes any contents from the buffer, then loads another file into the buffer, sets the current line to the last line of the buffer, and displays the number of bytes read in to the buffer. If the buffer was changed since its contents were last saved (with the **w** subcommand), **e** displays ? before it clears the buffer.

The **e** subcommand stores *file* as the default filename to be used, if necessary, by subsequent **e**, **E**, **r**, or **w** subcommands. (See the **f** subcommand.)

When the **!** (exclamation point) character replaces *file*, **e** takes the rest of the line as a shell (**sh**) command and reads the command output. The **e** subcommand does not store the name of the shell command as a default filename.

E *file* The **E** (Edit) subcommand works like the **e** subcommand, with one exception: **E** does not check for changes made to the buffer since the last **w** subcommand.

f [*file***]** The **f** (filename) subcommand changes the default filename (the stored name of the last file used) to *file*, if *file* is given. If *file* is not given, the **f** subcommand prints the default filename.

[*address1***,***address2*]**g/***pattern***/***subcommand_list*
The **g** (global) subcommand first marks every line that matches the pattern. Then, for each marked line, this subcommand sets the current line to that line and executes *subcommand_list*. Place a single subcommand, or the first subcommand of a list, on the same line with the **g** subcommand; place subsequent subcommands on separate lines. Except for the last line, end each of these lines with a \ (backslash).

The *subcommand_list* can include the **a**, **i**, and **c** subcommands and their input. If the last command in *subcommand_list* is normally the **.** (dot) that ends input mode, the **.** (dot) is optional. If there is no *subcommand_list*, the **ed** subcommand displays the current line. The *subcommand_list* cannot include the **g**, **G**, **v**, **V**, or **!** subcommands.

The **g** subcommand is similar to the **v** subcommand, which executes *subcommand_list* for every line that does not contain a match for the pattern. Note that the **g** subcommand defaults to the entire file, not to the current line.

[*address1***,***address2*]**G/***pattern***/**
The interactive **G** (Global) subcommand first marks every line that matches the pattern, then displays the first marked line, sets the current line to that line, and waits for a subcommand. **G** accepts any but the following **ed** subcommands: **a**, **c**, **g**, **G**, **i**, **v**, **V**, and **!**. After the subcommand finishes, **G** displays the next marked line, and so on. **G** takes a newline character as a null subcommand. **&** (ampersand) causes **G** to execute the previous subcommand again, if there is one. Note that subcommands executed within the **G** subcommand can address and change any lines in the buffer. The **G** subcommand can be terminated by pressing the Interrupt key sequence. Note that this subcommand defaults to the entire file, not to the current line.

h The **h** (help) subcommand displays a short message that explains the reason for the most recent **?** notification. The current line number is unchanged.

H The **H** (Help) subcommand causes the **ed** command to enter a mode in which help messages (see the **h** subcommand) are displayed for all subsequent **?** notifications. The **H** subcommand toggles this mode, and it is initially set to "off." The current line number is unchanged.

[*address*]**i**

text

. The **i** (insert) subcommand inserts text before the addressed line and sets the current line to the last inserted line. Enter **.** (dot) to return to command mode. If no lines are inserted, **i** sets the current line to the addressed line. This subcommand differs from the **a** subcommand only in the placement of the input text. Address **0** is not legal for the **i** subcommand.

[*address1***,***address1+1*]**j**
The **j** (join) subcommand joins contiguous lines by removing the intervening newline characters. If given only one address, **j** does nothing. (For splitting lines, see the **s** subcommand.) Note that lines that exceed the line length limit cannot be joined. If lines are joined, the current line number is set to the address of the joined line; otherwise, the current line number is unchanged.

[*address*]**k***x*
The **k** (mark) subcommand marks the addressed line with name *x*, which must be a

lowercase letter. The address *'x* (single quotation mark before the marking character) then addresses this line. The **k** subcommand does not change the current line. Note that marks attached to lines are deleted with the line.

[*address1*,*address2*]**l**
The **l** (list) subcommand displays the addressed lines. The **l** subcommand wraps long lines and, unlike the **p** subcommand, represents nonprinting characters as 3-digit octal numbers with a \ (backslash) preceding each byte in the character. The following characters, however, are written as escape sequences:

\\	Backslash
\a	Alert
\b	Backspace
\f	Formfeed
\n	Newline
\r	Carriage return
\t	Tab
\v	Vertical tab

A **$** (dollar sign) character is placed at the end of each line so that a real (literal) **$** at the end of a line cannot be misinterpreted.

An **l** subcommand can be appended to any **ed** subcommand except **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

[*address1*,*address2*]**m***address3*
The **m** (move) subcommand repositions the addressed lines. The first moved line follows the line addressed by *address3*. Address **0** for *address3* causes **m** to move the addressed lines to the beginning of the file. The line specified by *address3* cannot be one of the lines to be moved. The **m** subcommand sets the current line to the last moved line.

[*address1*,*address2*]**n**
The **n** (number) subcommand displays the addressed lines, each preceded by its line number and a tab character (displayed as spaces); the **n** subcommand leaves the current line at the last line displayed. An **n** subcommand can be appended to any **ed** subcommand except **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

[*address1*,*address2*]**p**
The **p** (print) subcommand displays the addressed lines and sets the current line to the last line displayed. A **p** subcommand can be appended to any **ed** subcommand except **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**. For example, the subcommand **dp** deletes the current line and displays the new current line.

P The **P** (Prompt) subcommand turns the **ed** prompt string ***** or the string specified by the **-p** flag on or off. Initially, **P** is off.

q The **q** (quit) subcommand exits the **ed** program. Before ending the program, **q** checks to determine whether the buffer was written to a file since the last time it was changed. If not, **q** displays the ? message. Note that you do not get more than one prompt in a row; a second consecutive **q** quits the **ed** program without displaying a prompt.

Q The **Q** (Quit) subcommand exits the **ed** program without checking for changes to the buffer since the last **w** subcommand (compare with the **q** subcommand).

[*address*]**r** *file*

The **r** (read) subcommand reads a file into the buffer after the addressed line; **r** does not delete the previous contents of the buffer. When entered without *file*, **r** reads the default file, if any, into the buffer (see the **e** and **f** subcommands). **r** does not change the default filename. Address **0** causes **r** to read a file in at the beginning of the buffer. After it reads a file successfully, **r** displays the number of bytes read into the buffer and sets the current line to the last line read.

If **!** (exclamation point) replaces *file* in a **r** subcommand, **r** takes the rest of the line as an OSS shell (**sh**) command whose output is to be read. The **r** subcommand does not store the names of shell commands as default filenames.

[*address1***,***address2*]**s**/*pattern*/*replacement*/*flags*

The **s** (substitute) subcommand searches each addressed line for a string that matches the *pattern* and then replaces the string with the specified *replacement* string. Without a number *n* or the global indicator **g**, **s** replaces only the first matching string on each addressed line. With *n*, **s** replaces the *n*th occurrence of *pattern* on the addressed line. With the **g** indicator, **s** replaces every occurrence of the matching string on each addressed line.

If **s** does not find a match for the pattern, it returns the error message ?. Any character except a space or a newline character can separate (delimit) the *pattern* and *replacement* arguments. The **s** subcommand sets the current line to the last line changed.

An **&** (ampersand) in the *replacement* string is a special symbol that has the same value as the *pattern* string. So, for example, the subcommand **s/out/&ing/** has the same effect as the subcommand **s/out/outing/** and replaces **out** with **outing** on the current line. A backslash before the ampersand (**&**) removes this special meaning of **&** in *replacement*.

The **%** (percent sign), when used by itself as *replacement*, causes **s** to use the previous *replacement* again. The **%** character does not have this special meaning if it is part of a longer *replacement* or if it is preceded by a \ (backslash).

Lines can be split by substituting newline characters into them. In *replacement*, the sequence \<**Return**> quotes the newline character (not displayed) and moves the cursor to the next line for the remainder of the string.

The value of *flags* can be the following:

count Substitutes for the *count*th occurrence only of the regular expression that is found on each addressed line.

g Substitutes globally for all nonoverlapping instances of the regular expression, instead of just substituting for the first instance.

l Displays the final line in which a substitution was made in the format specified for the **l** subcommand.

n Displays the final line in which a substitution was made in the format specified for the **n** subcommand.

p Displays the final line in which a substitution was made in the format specified for the **p** subcommand.

[*address1***,***address2*]**t***address3*

The **t** (transfer) subcommand inserts a copy of the addressed lines after *address3*. The **t**

subcommand accepts address **0** (for inserting lines at the beginning of the buffer). The **t** subcommand sets the current line to the last line copied.

u The **u** (undo) subcommand restores the buffer to the state it was in before it was last modified by an **ed** subcommand. The subcommands that **u** can undo are **a**, **c**, **d**, **g**, **G**, **i**, **j**, **m**, **r**, **s**, **t**, **u**, **v**, and **V**. All changes made to the buffer by a **g**, **G**, **v**, or **V** global subcommand are undone as a single change. The current line number is set to the value it had before the command being undone started.

[*address1*,*address2*]**v**/*pattern*/*subcommand_list*
The **v** subcommand executes the subcommands in *subcommand_list* for each line that does not contain a match for the pattern. The **v** subcommand is a complement for the global subcommand **g**, which executes *subcommand_list* for every line that does contain a match for the pattern.

[*address1*,*address2*]**V**/*pattern*/
The **V** subcommand first marks every line that does not match the pattern, then displays the first marked line, sets the current line to that line, and waits for a subcommand. The **V** subcommand complements the **G** subcommand, which marks the lines that do match the pattern.

[*address1*,*address2*]**w** *file*
The **w** (write) subcommand copies the addressed lines from the buffer to the file named in *file*. If the file does not exist, the **w** subcommand creates it with permission mode 666 (read and write permission for everyone), unless the **umask** command setting specifies another file creation mode. (For information about file permissions, see the description of the builtin command **umask** in the reference page for **sh** and the reference page for the **chmod** command.)

The **w** subcommand does not change the default filename (unless *file* is the first filename used since you invoked **ed**). If you do not provide a filename, **ed** uses the default filename, if any (see the **e**, **E**, and **f** subcommands). The **w** subcommand does not change the current line.

If the **ed** command successfully writes the file, it displays the number of characters written. When **!** (exclamation point) replaces *file*, **ed** takes the rest of the line as a shell (**sh**) command whose output is to be read; **w** does not save shell command names as default filenames (the same effect as **!**). The use of the write subcommand with **!** (exclamation point) is not considered the last **w** subcommand that wrote the entire buffer. Thus, this alone does not prevent the warning to the user if an attempt is made to destroy the editor buffer through the **e** or **q** subcommands.

The address **0** is not a legal address for the **w** subcommand. Therefore, it is not possible to create an empty file with the **ed** command.

[*address*]**=**
Without an address, the = (equal sign) subcommand displays the current line number. With the address **$**, **=** (equal sign) displays the number of the last line in the buffer. The **=** subcommand does not change the current line and cannot be included in a **g** or **v** subcommand list.

!*system_command*
The **!** (exclamation point) subcommand allows system commands to be run from within the **ed** program. Anything following **!** on an **ed** subcommand line is interpreted as a system command. Within the text of that command string, **ed** replaces the unescaped character **%** with the current filename, if there is one.

When the **!** is used as the first character of a shell command (after the **!** that runs a

subshell), the **ed** command replaces the **!** character with the previous system command; for example, the command **!!** repeats the previous system command. If the command interpreter (the **sh** command) expands the command string, **ed** echoes the expanded line. The **!** subcommand does not change the current line. If any replacements of **%** or **!** are performed, the modified line is written to the standard output file before the command is executed.

number

+number

-number The **ed** command interprets a number alone on a line as an address and displays the addressed line. Addresses can be absolute (line numbers or **$**) or relative to the current line (*+number* or *-number*). Entering a newline character (a blank line) is equivalent to **+1p** and is useful for stepping forward through the buffer one line at a time.

EXIT VALUES

The **ed** editor returns a value of 0 (zero) if execution is successfully completed; if an error occurs, a value greater than 0 (zero) is returned.

RELATED INFORMATION

Commands: **chmod(1), edit(1), grep(1), sed(1), sh(1), stty(1), vi(1).**

Functions: **regexp**(3).

NAME

egrep - Searches a file for a pattern that is a full regular expression

SYNOPSIS

egrep
[**-c** | **-l**]
[**-bhinqsvx**]
{ *pattern* ... | **-e** *pattern* ... | **-f** *pattern_file* ... }
[*file* ...]

FLAGS

While most flags can be combined, some combinations result in one flag overriding another. For example, if you specify both the **-n** and **-l** flags, the output includes only filenames (as specified by the **-l** flag) and thus does not include line numbers (as specified by the **-n** flag).

-b Precedes each line by the block number of the block in which it was found. Use this flag to help find disk block numbers by context.

-c Displays only a count of matching lines.

-e *pattern* ...
Specifies a *pattern*. This flag works the same as a simple *pattern* but is useful when the pattern begins with a - (dash).

-f *pattern_file* ...
Specifies a file that contains patterns. Each pattern terminates with a newline character.

-h Suppresses reporting of filenames when multiple files are processed.

-i Ignores the case of letters in locating *pattern*; that is, uppercase and lowercase letters in the input are considered to be identical.

-l Lists the name of each file that has lines matching *pattern*. Each filename is listed only once; filenames are separated by newline characters.

-n Precedes each line with its relative line number in the file.

-q Suppresses all output except error messages. This flag is useful for easily determining whether a pattern or string exists in a group of files. When searching several files, it provides a performance improvement, because it can quit as soon as it finds the first match, and it requires less care by the user in choosing the set of files to supply as arguments, because it exits with a 0 (zero) exit status if it detects a match, even if the **egrep** command detected an access or read error on earlier file arguments.

-s Suppresses error messages about inaccessible files.

-v Displays all lines except those that match the specified pattern. This flag is useful for filtering unwanted lines out of a file.

-x Displays lines that match the pattern exactly with no additional characters.

DESCRIPTION

The **egrep** command searches the specified files (the standard input file by default) for lines containing characters that match the specified pattern and then write matching lines to the standard output file.

The **egrep** command is an obsolescent version of the command **grep -E**, which searches for patterns that are full regular expressions, except for \(and \), and with the addition of the following rules:

- A regular expression followed by a **+** (plus sign) matches one or more occurrences of the regular expression.
- A regular expression followed by a **?** (question mark) matches zero or one occurrence of the regular expression.
- Two regular expressions separated by a **|** (vertical bar) or by a newline character match either expression.
- A regular expression can be enclosed in **( )** (parentheses) for grouping.

The order of precedence of operators is as follows:

- **[= =] [: :] [. .]** (collation-related bracket symbols)
- \<*special_character*> (escaped characters)
- **[ ]** (bracket expressions)
- **( )** (grouping)
- *** + ? {***m***,** *n***}** (single-character-ERE duplication)
- concatenation
- **ˆ $** (anchoring)
- **|** (alternation)

Regular Expressions (REs)

Regular expressions (REs) cannot contain newline characters, because these signal a new pattern. The following REs match a single character:

character
: An ordinary character (one other than one of the special pattern-matching characters) matches itself.

.
: A **.** (dot) matches any single character except the newline character.

[*string***]**
: A string enclosed in **[ ]** (brackets) matches any one character in that string. In addition, certain pattern-matching characters have special meanings within brackets:

  ^
  : If the first character of *string* is a **ˆ** (circumflex), the RE **[ˆ***string***]** matches any character except the characters in *string* and the newline character. A **ˆ** has this special meaning only if it occurs first in the string.

  -
  : You can use a **-** (dash) to indicate a range of consecutive characters. The characters that fall within a range are determined by the current collating sequence, which is defined by the **LC_COLLATE** environment variable. For example, **[a-d]** is equivalent to **[abcd]** in the traditional ASCII collating sequence.

    A range can include a multicharacter collating element enclosed within bracket-period delimiters ([. .]). The bracket-period delimiters in the RE syntax distinguish multicharacter collating elements from a list of the individual characters that make up the element.

    A collating sequence can define equivalence classes for characters. An equivalence class is a set of collating elements that all sort to the same primary location. They are enclosed within bracket-equal delimiters (**[= =]**). An

equivalence class generally is designed to deal with primary-secondary sorting. For example, if **e**, **è**, and **ê** belong to the same equivalence class, then **[[=e=]fg]**, **[[=è=]fg]**, and **[[=ê=]fg]** are each equivalent to **[eèêfg]**.

The **-** (dash) character loses its special meaning if it occurs first (**[-***string***]**), if it immediately follows an initial circumflex (**[^-***string***]**), or if it appears last (**[***string***-]**) in the string.

] When the **]** (right bracket) is the first character in the string (**[]***string***]**) or when it immediately follows an initial circumflex (**[^]***string***]**), it is treated as a part of the string rather than as the string terminator.

special_character

A \ (backslash) followed by a special pattern-matching character matches the special character itself (as a literal character). These special pattern-matching characters are as follows:

. * [\ Always special, except when they appear within **[ ]** (brackets).

^ Special at the beginning of an entire pattern or when it immediately follows the left bracket of a pair of brackets (**[^...]**).

$ Special at the end of an entire pattern.

[: :] A character class name enclosed in bracket-colon delimiters matches any of the set of characters in the named class. Members of each of the sets are determined by the current setting of the **LC_CTYPE** environment variable. The supported classes are **alpha**, **upper**, **lower**, **digit**, **xdigit**, **space**, **print**, **punct**, **graph**, and **cntrl**. Here is an example of how to specify one of these classes:

[[:lower:]]

This matches any lowercase character for the current locale.

Forming Patterns

The following rules describe how to form patterns from REs:

- An RE that consists of a single, ordinary character matches that same character in a string.
- An RE followed by an ***** (asterisk) matches zero or more occurrences of the character that the RE matches. For example, the following pattern:

  ab*cd

  matches each of the following strings:

  acd
  abcd
  abbcd
  abbbcd

  but not the following string:

  abd

  If there is any choice, the leftmost longest matching string is chosen. For example, given the following string:

  122333444

  the pattern **.*** matches **122333444**, the pattern **.*3** matches **122333**, and the pattern **.*2**

matches **122**.

- An RE followed by:

  \{*number*\}
  Matches exactly *number* occurrences of the character matched by the RE.

  \{*number*,\}
  Matches at least *number* occurrences of the character matched by the RE.

  \{*number1*,*number2*\}
  Matches any number of occurrences of the character matched by the RE from *number1* to *number2*, inclusive.

  The values of *number1* and *number2* must be integers in the range 0 through 255. Whenever a choice exists, this pattern matches as many occurrences as possible.

  Note that if *number* is 0 (zero), *pattern* matches zero occurrences of *pattern*. For example:

  ```
  $ echo abc | grep 'aX\{0\}bX\{0\}cX\{0\}'
  abc
  $
  ```

- You can combine REs into patterns that match strings containing the same sequence of characters. For example, **AB*CD** matches the string **ABCD**, and **[A-Za-z]*[0-9]*** matches any string that contains any combination of ASCII alphabetic characters (including none), followed by any combination of numerals (including none).

- The character sequence **\(***pattern***\)** matches *pattern* and saves it into a numbered holding space. Using this sequence, up to nine patterns can be saved on a line. Counting from left to right on the line, the first pattern saved is placed in the first holding space, the second pattern saved is placed in the second holding space, and so on.

  The character sequence *n* matches the *n*th saved pattern, which is placed in the *n*th holding space. (The value of *n* is an integer in the range 1 through 9.) Thus, the following pattern:

  ```
  \(A\)\(B\)C\2\1
  ```

  matches the string **ABCBA**. You can nest patterns to be saved in holding spaces. Whether the enclosed patterns are nested or in a series, *n* refers to the *n*th occurrence, counting from the left, of the delimiting characters, **\)**.

Restricting What Patterns Match

A pattern can be restricted to match either from the beginning of a line, up to the end of the line, or the entire line:

- A ^ (circumflex) at the beginning of a pattern causes the pattern to match only a string that begins in the first character position on a line.

- A **$** (dollar sign) at the end of a pattern causes that pattern to match only if the last matched character is the last character (not including the newline character) on a line.

- The construction ^*pattern*$ restricts the pattern to matching only an entire line.

EXAMPLES

1. To display all lines in a file that begin with an ASCII letter, enter:

   ```
   egrep  '^[a-zA-Z]'  pgm.s
   ```

2. To display all lines that contain ASCII letters in parentheses or digits in parentheses (with spaces optionally preceding and following the letters or digits), but not letter-digit combinations in parentheses, enter:

   ```
   egrep \
   '\(  *([a-zA-Z]* | [0-9]*)  *\)'  my.txt
   ```

   This command displays lines in **my.txt** such as **(y)** or **(783902)**, but not **(alpha19c)**.

3. To display all lines that do not match a pattern, enter:

   ```
   egrep  -v  '^#'
   ```

   This displays all lines that do not begin with a # (number sign).

4. To display all lines that contain uppercase characters, enter:

   ```
   w
   egrep '[[:upper:]]' pgm.s
   ```

EXIT VALUES

The **egrep** command returns the following exit values:

0 (zero) A match was found.

1 No match was found.

2 A syntax error occurred or a file was inaccessible, even if matches were found.

RELATED INFORMATION

Commands: **ed(1)**, **ex(1)**, **grep(1)**, **sed(1)**, **sh(1)**.

Files: **locale(4)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification.

The following features are HP extensions to the XPG4 Version 2 specification:

- The **-b**, **-h**, **-q**, and **-x** flags are supported.

NAME

eld - Runs the TNS/E native linker utility for position-independent code

SYNOPSIS

eld

[**-alf** *filename1*]
[{ **-all** | **-include_whole** }
| { **-no_include_whole** | **-none** }]
[**-allow_duplicate_procs**]
[**-allow_missing_libs**]
[**-allow_multiple_mains**]
[**-ansistreams**]
[**-bdllsonly** | **-bdynamic** | **-bstatic**]
[**-bglobalized**]
[**-blocalized**]
[**-bsymbolic** | **-bsemi_globalized**]
[**-call_shared** | { **-dll** | **-shared** } | **-r**]
[**-change** *attribute_name attribute_value filename3*]
[**-check_registry** *filename4*]
[**-cross_dll_cleanup**]
[**-d** *address1*]
[**-data_resident**]
[{ **-dllname** | **-soname** } *DLL_name*]
[**-e** *symbol_name1*]
[{ **-export** | **-exported_symbol** } *symbol_name2*]
[**-export_all**]
[{ **-export_not** | **-hidden_symbol** } *symbol_name3*]
[**-first_l** *pathname1*]
[{ **-fl** | **-obey** } *location1*]
[**-grow_data_amount** *number1*]
[**-grow_text_amount** *number2*]
[**-grow_limit** *number3*]
[**-grow_percent** *number4*]
[**-import_lib** *filename5*]
[**-import_lib_stripped** *filename6*]
[**-instance_data** *keyword*]
[{ **-l** | **-lib** } *filename7*]
[{ **-L** | **-libvol** } *pathname2*]
[**-libname** *Guardian_filename*]
[**-limit_runtime_paths**]
[**-local_libname** *filename8*]
[**-m** | **-map**]
[**-make_implicit_lib**]
[**-make_import_lib** *filename9*]
[**-must_preset**]
[**-must_use_iname**]
[**-must_use_oname**]
[**-must_use_rname**]
[**-no_optional_lib** | **-optional_lib**]
[**-no_reexport** | **-reexport**]
[**-nostdfiles** | **-no_stdfiles**]
[**-nostdlib** | **-no_stdlib**]
[**-noverbose** | **-no_verbose**]
[**-o** *filename10*]

[**-public_registry** *filename11*]
[**-rename** *old_name new_name*]
[{ **-rld_l** | **-rpath** } *path_list1*]
[**-rld_first_l** *path_list2*]
[{ **-s** | **-x** }]
[**-set** *attribute_name attribute_value*]
[**-show_multiple_defs**]
[**-stdin**]
[**-strip** *filename12*]
[**-t** *address2*]
[**-temp_i** *filename13*]
[**-temp_o** *filename14*]
[**-temp_r** *filename15*]
[**-u** *symbol_name4*]
[**-ul**]
[**-unres_symbols** { **error** | **ignore** | **warn** }]
[**-update_code**]
[**-update_registry** *filename16*]
[**-verbose**]
[**-warn**]
[**-warn_common**]
[**-y** *symbol_name5*]
[*filename17*] ...

FLAGS

-alf *filename1* Tells ld to adjust a load file by rebasing or rebinding the loadfile *filename1*.

Defined globalized symbols are updated if necessary in the adjusted load file. The update timestamp of the new loadfile is set to the current date and time if any changes are made to the file.

If you do not use the **-o** flag or if the **-o** flag specifies the same filename as *filename1*, the new loadfile has the same name as the existing loadfile.

If you do not use the **-t** flag, the new loadfile is rebound if necessary to the libraries specified by the **-libvol** flag, but not rebased.

The **-alf** flag resolves symbolic references similar to the way **eld** resolves them when it is creating a new loadfile from a set of linkfiles. However, with the **-alf** flag, if you do not specify your preference for the treatment of unresolved references with a flag such as **-unres_symbols**, the default comes from what was specified for the **-set rld_unresolved** flag when the loadfile was originally built by **eld**.

If you specify the **-t** flag then the existing loadfile is rebased. Otherwise, the **-alf** processing does not change the preferred addresses of the loadfile.

Flags such as **-must_preset** have the same meanings as when **eld** originally creates a loadfile.

When you use this flag, you can use only the following flags in the same command:

-allow_missing_libs
-check_registry
-first_l
-fl or **-obey**
-L or **-libvol**

-local_libname
-must_use_oname and **-must_use_rname**
-must_preset
-nostdlib or **-no_stdlib**
-noverbose, **-no_verbose**, **-verbose**, and **-warn**
-o
-public_registry
-rld_first_l and **-rld_l**
-stdin
-t
-temp_o and **-temp_r**
-unres_symbols
-update_code and **-update_registry**

{**-all** | **-include_whole** } | { **-no_include_whole** | **-none**}
Tells **eld** whether to include in the loadfile all linkable archive members of all archive libraries encountered after this flag is specified.

Specifying **-all** or **-include_whole** begins this linking action. When **-none** or **-no_include_whole** behavior is in effect, archive searches are controlled by the existence of undefined symbols. Archives are searched in the order specified on the command line. Symbols are marked as undefined by compilers or by the user through the **-u** flag.

When an archive member is found that resolves an undefined symbol, that member is used by **eld** as if it had been specified as a linkfile in the command line or obey file. That linkfile might lead to more undefined symbols, more members might be found in the same archive to resolve them, and so forth.

You can stop the linking action of **-all** or **-include_whole** by specifying the **-no_include_whole** or **-none** flag later in the command line or obey file.

These flags can be specified as many times as needed in the command line or obey file. Providing a flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

The default setting is **-none**.

-allow_duplicate_procs
Tells **eld** to unconditionally accept multiple copies of a procedure. The only check made is that all copies of the procedure have the same procedure attributes (other than the RESIDENT, PRIVILEDGED, or OPT_LEVEL attributes); for example, it is acceptable if they have different sizes. The first copy of the duplicated procedure is the one that is kept, unless there are both RESIDENT and not RESIDENT copies; **eld** always keeps the first RESIDENT copy. When building an executable file, no space is allocated for the unused copies.

The default action is to accept multiple copies of only procedures specifically marked as duplicatable by C++.

-allow_missing_libs
Tells **eld** not to consider it an error when it cannot find an archive or a dynamic-link library (DLL) after searching for the name specified by a **-l** or **-lib** flag. Instead, a warning message is issued and processing continues.

The default action when a needed DLL or archive cannot be found is to stop considering the condition an error.

-allow_multiple_mains

Directs **eld** not to issue an error message if more than one procedure has the MAIN attribute. All main procedures are included in the output file. Only the first procedure having the MAIN attribute is listed as the main entry point in the file header.

The default action is to report an error when more than one procedure has the MAIN attribute.

-ansistreams Specifies that C run-time library functions create files of file code 180 (C text as binary) instead of file code 101 (EDIT). The type of files created can also be set with the `ANSISTREAMS` C and C++ compiler pragma.

See the *C/C++ Programmer's Guide* for more information.

-bdllsonly | **-bdynamic** | **-bstatic**

Tells **eld** what type of file to look for in each directory when **eld** searches for the file names specified for the **-l** or **-lib** flags. The three possible values are:

-bdllsonly Tells **eld** to search only for DLLs when it needs to search for the file name specified in the **-l** or **-lib** flag.

If the file name is unqualified, in each directory searched, **eld** first tries to open a file with the name specified for the **-l** or **-lib** flag. If eld cannot find a file with the specified name and the searched directory is not in the Guardian file system (/**G**), **eld** prefixes **lib** and suffixes **.so** to the file name and attempts to open a file with that modified name. As its final attempt to find the file, **eld** attempts to open a file that has the specified name prefixed with **z** and suffixed with **dll**.

An error occurs if the opened file does not contain a DLL.

-bdynamic Directs **eld** to search for DLLs and archive files when it needs to search for the file name specified in the **-l** or **-lib** flag.

If the file name is unqualified, in each directory searched, **eld** first tries to open a file with the name specified for the **-l** or **-lib** flag. If **eld** cannot find a file with the specified name and the searched directory is not in the Guardian file system (/**G**), **eld** prefixes **lib** and suffixes **.so** to the file name and attempts to open a file with that modified name. If **eld** still cannot find the file and the searched directory is not in the Guardian file system (/**G**), it prefixes the specified name with the **lib** prefix but with **.a** as the suffix and attempts to open a file with that modified name. As its final attempt to find the file, **eld** attempts to open a file that has the specified name prefixed with **z** and suffixed with **dll**.

An error occurs if:

- **eld** finds a DLL in a file whose name has the form **lib***filename7***.a**
- **eld** finds an archive in a file whose name has the form **lib***filename7***.so** or **z***filename7***dll**
- **eld** finds a linkfile rather than a DLL or an archive

-bstatic Directs **eld** to search only for archive files when it needs to search for the file name specified in the **-l** or **-lib** flag.

If the file name is unqualified, in each directory searched, **eld** first tries to open a file with the name specified for the **-l** or **-lib** flag). If **eld** cannot find a file with the specified name and the search path is not in the Guardian file system (/G), then **eld** prefixes **lib** and suffixes **.a** to the file name and attempts to open a file with that modified name.

An error occurs if the opened file does not contain an archive.

For more information on search paths, see the **Finding Libraries** subsection of this reference page under **DESCRIPTION**.

When a specified file cannot be found, **eld** issues an error message unless the **-allow_missing_libs** flag is specified.

The **-bdllsonly**, **-bdynamic**, and **-bstatic** flags are search control toggles. Multiple flags can be specified in a single **eld** invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** and **-lib** flags and search for just archive files for others.

The default library search control is **-bdynamic**.

-bglobalized Controls creation of the searchList for resolving the symbols in the loadfile. This flag causes the system to use the following sequence:

- For link-time resolutions:
  1. The loadfile itself
  2. The user library, if the loadfile is a program and has a user library
  3. Breadth-first transitive closure of DLLs on the libList
  4. Implicit libraries
- For load-time resolutions:
  1. The main program
  2. The user library
  3. Breadth-first transitive closure of DLLs on the libList
  4. Implicit libraries

You cannot use this flag when you use the **-blocalized**, **-bsemi_localized**, or **-bsymbolic** flag. The default action is the action for the **-blocalized** flag.

-blocalized Controls creation of the searchList for resolving symbols in the loadfile. This flag causes the system to use the following sequence for both link-time and load-time resolutions:

1. The loadfile itself
2. The user library, if the loadfile is a program and has a user library
3. Breadth-first transitive closure of re-exported libList-specified DLLs
4. Implicit libraries

This is the default **eld** action.

You cannot use this flag when you use the **-bglobalized**, **-bsemi_localized**, or **-bsymbolic** flag.

-bsymbolic | **-bsemi_globalized**
Controls creation of the searchList for resolving symbols in the loadfile. This flag causes the system to use the following sequence:

- For link-time resolutions:
  1. The loadfile itself
  2. The user library, if the loadfile is a program and has a user library
  3. Breadth-first transitive closure of DLLs on the libList
  4. Implicit libraries
- For load-time resolutions:
  1. The loadfile itself
  2. The main program
  3. The user library
  4. Breadth-first transitive closure of DLLs on the libList
  5. Implicit libraries

This is the same search order as for **-bglobalized**. However, when the searchList is created at load time, the loadfile is placed before the program at the beginning of the searchList.

You cannot use this flag when you use the **-bglobalized** or **-blocalized** flag. The default action is the action for the **-blocalized** flag.

-call_shared | { **-dll** | **-shared** } | **-r**
Tells **eld** what type of file to create as *filename11*. The possible values are:

-call_shared The file is to be a program loadfile. This is the default **eld** action.

You cannot use this flag when you use the **-dll** or **-shared** flag or the **-r** flag.

-dll | **-shared** The file is to be a DLL.

You cannot use this flag when you use the **-call_shared** or **-r** flag.

-r The file is to be a linkfile. If there is only one input file, the new linkfile has the same fingerprint as the input file.

You cannot use this flag when you use the **-call_shared**, **-dll**, or **-shared** flag.

-change *attribute_name attribute_value filename3*

Changes the value of the run-time attribute specified in *attribute_name* to the value specified in *attribute_value* in the existing file specified by *filename3*. See the **-set** flag for a description of *attribute_name* and *attribute_value*.

For a linkfile, you can only change the value for **FLOATTYPE** and **DATA_MODEL**. The value for **LIBNAME** can only be changed for program files. If **LIBNAME** is specified with a null value ("") for *attribute_value*, the existing user library name is removed from the program file *filename3*.

The **INCOMPLETE** attribute only applies to an import library; the **-change** flag can be used to turn it **ON** but cannot be used to turn it off.

You cannot specify other loadfile filenames or flags other than the following with the **-change** flag:

-noverbose, **-verbose**, or **-warn**
-fl or **-obey**
-stdin

The resulting loadfile has the same **eld** timestamp as before.

-check_registry *filename4*

Tells **eld** to check the **-range** entry in the private DLL registry identified as *filename4* for an address range for the file specified as *filename11* and verify that the DLL being created can fit into the range found.

If neither the **-check_registry** flag nor the **-update_registry** flag is specified, **eld** does not use a private DLL registry.

You must also use the **-dll** flag when you use this flag. You cannot use the following flags when you use the **-check_registry** flag:

-d or **-t**
-grow_data_amount, **-grow_limit**, **-grow_percent**, or **-grow_text_amount**
-update_registry

-cross_dll_cleanup

Discards a procedure in an input linkfile if a procedure of the same name exists in one of the other DLLs you specified. The reference to the symbol is resolved to the copy of the symbol that is in the DLL.

Without the use of the **-cross_dll_cleanup** flag, input linkfiles can contain multiple copies of the same global procedure, all marked STO_MULTIPLE_DEF_OK, and **eld** discards the code for all except one copy of the procedure.

When the **-cross_dll_cleanup** flag is used and a procedure of the same name is found in a DLL, **eld** discards the code for even the last copy of the procedure, and instead resolves references to the procedure to the copy found in the DLL. Using this flag allows you to reduce the size of the loadfile (program or DLL) you create.

When this flag is used, all the input linkfiles must have been compiled using the **-Wglobalized** flag.

For best results, list the DLLs that are indirectly used by this loadfile in addition to the DLLs that are directly used by this loadfile.

This flag is available on systems running H06.21 or a later H-series RVU, J06.10 or a later J-series RVU, or systems that have installed SPR T0608H01_AAL.

-d *address1* Specifies the hexadecimal virtual address at which the data constant segment starts. When creating a program file, the default value for *address1* is 08000000. When creating an explicit DLL, the default value is set to the next multiple of 64 kilobytes after the end of the text segment. When creating an implicit DLL, the default value is set to the next multiple of 128 kilobytes after the end of the text segment.

The value specified for *address1* is always hexadecimal and can optionally be prefixed by 0H. The specified value is automatically rounded up to a multiple of 128K bytes if **eld** is building an implicit DLL, or to a multiple of 64K bytes if **eld** is building any other type of file. If no more than 8 hexadecimal digits are specified, the number specified is sign-extended to 16 digits.

If you use this flag, you must also use the **-t** flag.

If you are building a DLL and do not use the **-d** flag, **eld** begins the data segment at the next 64K boundary after the end of the text segment. The **-d** flag can be used to place the data segment at some other address, rather than directly after its text segment. You should do this only if there is a special need to do so; HP recommends using the default placement whenever possible.

You cannot use this flag if you use the **-check_registry** or **-update_registry** flags.

-data_resident Specifies that the loadfile being created can contain both resident code and variable data.

When you create a loadfile with both resident code and variable data, you must either specify the **-data_resident** flag or the **-instance_data** flag with one of the values **data2**, **data2hidden**, or **data2protected**. Otherwise, an error occurs and no loadfile is created.

When you use the **-data_resident** flag, **eld** also sets the **EF_TANDEM_DATA_RESIDENT** bit in the **e_flags** field of the file header. This is a special flag that can be used when building a proto-process.

{**-dllname** | **-soname** } *DLL_name*
Tells **eld** the DLL name to store in the DLL being created. When this DLL is specified in the link step of another loadfile, the DLL name stored in this DLL is placed in the libList of the loadfile for later use by **rld** or the **eld -alf** flag when searching for DLLs.

If the DLL being created will reside in the Guardian file system, *DLL_name* must conform to Guardian filename rules. If the DLL being created will reside in the OSS file system, *DLL_name* must conform to OSS pathname rules. If you want the DLL to be able to reside in either file system, use an unqualified Guardian file identifier as both its *DLL_name* and its OSS filename.

If you specify both a *DLL_name* and the **-o** flag, the output loadfile file name is determined by the **-o** specification and *DLL_name* is saved in the DLL being created. HP recommends that you do not do this unless there is a specific need to do so; making *DLL_name* differ from the DLL's file name can cause problems, especially when the DLL needs to reside in both the OSS and Guardian file systems.

If you specify *DLL_name* but do not use the **-o** flag, the output loadfile file name

uses the *DLL_name* value.

If you do not specify *DLL_name* but use the **-o** flag, the output loadfile unqualified filename is used as the DLL name stored in the DLL being created; that is, only the unqualified part (rightmost part) of the output file pathname is used.

If you omit both a *DLL_name* and the **-o** flag, the output loadfile filename and *DLL_name* in the libList both default to **a.out**. HP recommends against using this default value because it is too generic to allow for proper administration of files.

-e *symbol_name1*
Specifies a function identifier. The specified function is the entry point, that is, the point at which to begin executing the program when the program is loaded.

You should use this flag only when linking a program that will execute without standard run-time support facilities and without linking a module such as CCPLMAIN (in the Guardian file system) or **ccplmain.o** (in the OSS file system) that contains a function with the MAIN attribute. Do not use this flag for libraries.

{ **-export** | **-exported_symbol** } *symbol_name2*
Tells **eld** to mark *symbol_name2* for export in the output loadfile in addition to those normally marked. This flag can be used with the **-export_not** or **-hidden_symbol** flag to create sets of symbols to be exported.

The same symbol cannot be specified as *symbol_name2* in an **-export** flag and as *symbol_name3* in an **-export_not** flag.

The default action when the **-ul** flag is not used is to export only those symbols marked by a compiler as requiring export.

-export_all Tells **eld** to mark as exported all symbols in the dynamic symbol table of the output loadfile except for the following:

- Linker-defined symbols other than **_MCB** (for the master control block)
- Procedures whose names begin with **__sti__** (global constructors), **__std__** (global destructors), **__INIT__** (initialization functions), or **__TERM__** (termination functions)

This flag can be used with the **-export_not** or **-hidden_symbol** flag to create a subset of symbols to be exported.

The default action when the **-ul** flag is not used is to export only those symbols marked by a compiler as requiring export.

{ **-export_not** | **-hidden_symbol** } *symbol_name3*
Tells **eld** not to mark *symbol_name3* for export in the output loadfile. This flag can be used with the **-export_all** flag to create sets of symbols to be exported.

The same symbol cannot be specified as *symbol_name3* in an **-export_not** flag and as *symbol_name2* in an **-export** flag.

The default action when the **-ul** flag is not used is to export only those symbols marked by a compiler as requiring export.

-first_l *pathname1*
Tells **eld** to use the specified pathname when searching for libraries. *pathname1* is used in library searches before the public libraries are searched.

pathname1 is either the relative or absolute pathname of an OSS directory.

eld does not verify the names of locations specified in the **-first_l** flag.

This flag can be specified more than once in a command line or an obey file. If you specify it more than once, the specified pathnames are searched in the order specified.

See the **Finding Libraries** subsection under **DESCRIPTION** for details about the effect of this flag on search order.

{**-fl** | **-obey** } *location1*

Specifies the name of an **eld** command file containing **eld** command tokens (such as filenames and command flag specifications).

The command file must be a C text file or an EDIT file. Tokens in the file can be separated by spaces, tabs, or ends of lines. Tokens can contain double quotation marks (") to group items into a single string, consistent with OSS shell usage. Within the command file, two hyphens indicate a comment that extends to the end of the current line.

Command files can be nested. There is no limit to the depth of nesting. Direct or indirect recursive nesting is allowed; **eld** keeps a stack of command file names encountered while processing the **-fl** or **-obey** flag and ignores any nested **-fl** or **-obey** flag that specifies a file name currently on the stack.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-grow_data_amount *number1*

Specifies the absolute amount in bytes of slack space to be reserved in virtual memory to allow for the growth of the data segment. The number specified is assumed to be decimal unless prefixed by **0x**; the **0x** prefix allows you to specify the number of bytes in hexadecimal.

This flag is used with the **-update_registry flag**. When **eld** makes the entry in the private DLL registry, the **-grow_data_amount** flag tells it to record a larger size for the data segment than is actually present in the DLL being built. The amount specified for **-grow_data_amount** tells how much larger to make the data segment. When this registry is used to choose addresses for other DLLs, the linker will not put them directly after this one in virtual memory, but rather leave space for future versions of this DLL to be larger.

The value specified for this flag might be overridden by the value used for the **-grow_percent** flag. The default value is 0 bytes.

When you use this flag, you must also use the **-update_registry** flag. You cannot use this flag with the **-grow_limit** flag.

-grow_text_amount *number2*

Specifies the absolute amount in bytes of slack space to be reserved in virtual memory to allow for the growth of the text segment. The number specified is assumed to be decimal unless prefixed by **0x**; the **0x** prefix allows you to specify the number of bytes in hexadecimal.

This flag is used with the **-update_registry** flag. When **eld** makes the entry in the private DLL registry, the **-grow_text_amount** flag tells it to record a larger size for the text segment than is actually present in the DLL being built. The amount specified for **-grow_text_amount** tells how much larger to make the text segment. When this registry is used to choose addresses for other DLLs, the linker will not put them directly after this one in virtual memory, but rather leave space for future versions of this DLL to be larger.

The value specified for this flag might be overridden by the value used for the

-grow_percent flag. The default value is 0 bytes.

When you use this flag, you must also use the **-update_registry** flag. You cannot use this flag with the **-grow_limit** flag.

-grow_limit *number3*

Specifies the absolute amount in bytes of slack space to be reserved in virtual memory to allow for the growth of the data and text segments. The number specified is assumed to be decimal unless prefixed by **0x**; the **0x** prefix allows you to specify the number of bytes in hexadecimal.

This flag is used with the **-update_registry flag**. When **eld** makes the entry in the private DLL registry, the **-grow_limit** flag tells the total size to record for the DLL being built. When this registry is used to choose addresses for other DLLs, the linker will not put them directly after this one in virtual memory, but rather leave space for future versions of this DLL to be larger.

The default value is 0 bytes.

When you use this flag, you must also use the **-update_registry** flag. When you use this flag, you cannot use the **-grow_data_amount**, **-grow_text_amount**, or **-grow_percent** flag.

-grow_percent *number4*

Specifies the relative amount of slack space to be reserved in virtual memory to allow for the growth of the data and text segments.

This flag is used with the **-update_registry** flag. When **eld** makes the entry in the private DLL registry, the **-grow_percent** flag tells a percentage to add to the sizes of each of the text and data segments, when calculating the total size to record for the DLL being built.

eld calculates an amount to use for a data segment, based on the **-grow_data_amount** flag, then calculates an amount to use based on the **-grow_percent** flag. The larger of the two amounts is used as the size for a data segment. The amount used is rounded up to a multiple of 64KB (or 128KB for an implicit DLL).

eld calculates an amount to use for a text segment, based on the **-grow_text_amount** flag, then calculates an amount to use based on the **-grow_percent** flag. The larger of the two amounts is used as the size for a text segment. The amount used is rounded up to a multiple of 64KB (or 128KB for an implicit DLL).

When this registry is used to choose addresses for other DLLs, the linker will not put them directly after this one in virtual memory, but rather leave space for future versions of this DLL to be larger.

The default value is 10 percent.

When you use this flag, you must also use the **-update_registry** flag. You cannot use this flag with the **-grow_limit** flag.

-import_lib *filename5*

Tells **eld** to create an import library named *filename5* in addition to creating a new DLL.

The new import library can be incomplete. The import library contains symbols for use by debugging utilities only if the new DLL also contains them.

You cannot use this flag when you use the **-import_lib_stripped** or **-make_implicit_lib** flag.

-import_lib_stripped *filename6*

Tells **eld** to create an import library named *filename6* in addition to creating a new DLL.

The new import library can be incomplete. The import library will not contain symbols for use by debugging utilities, regardless of whether the new DLL contains them.

You cannot use this flag when you use the **-import_lib** or **-make_implicit_lib** flag.

-instance_data *keyword*

Tells the linker whether to create one or two data segments, and whether to require that the loadfile contain no data that must go into the data variable segment if two segments are created.

The value of *keyword* can be:

data1 Create only one data segment for constant and variable data. The segment can be both read and written without restriction.

data1constant Flag as an error any situation where the loadfile contains the kind of data that would go into the data variable segment if **eld** was told to create two data segments.

This is the only permitted value when the **-make_implicit_lib** flag is used.

data2 Create both a data constant segment and a data variable segment. The data constant segment is read-only after being updated by the runtime loader; the data variable segment can be both read and written without restriction.

data2hidden Create both a data constant segment and a data variable segment. The data constant segment is read-only after being updated by the runtime loader; only privileged code can read or write the data variable segment.

data2protected

Create both a data constant segment and a data variable segment. The data constant segment is read-only after being updated by the runtime loader; only privileged code can write the data variable segment but read access is unrestricted.

The **data2protected** value is not supported on systems running H06.21 or later H-series RVUs or J06.10 or later J-series RVUs.

The default value for *keyword* is **data1**.

If you are creating a loadfile with both resident code and variable data, then you must either specify the **-data_resident** flag or the **-instance_data** flag with one of the values **data2**, **data2hidden**, or **data2protected**.

You cannot specify this flag if you use the **-r** flag.

{ **-l** | **-lib** } *filename7*

Specifies the name of a DLL or archive file to use to resolve external references from the file being linked. If you specify a three or four-character abbreviation (containing only letters or numbers) for an unqualified filename, **eld** will search for it using the naming rules described for the **-bdllsonly**, **-bdynamic**, and **-static** flags.

The **-l** flag must be specified in lowercase type. **-l** is a synonym for **-lib**.

If you specify the **-verbose** flag, **eld** writes to its output listing the locations where it found a DLL or archive file.

Other flags affect how *filename7* is used. See the **Finding Libraries** subsection under **DESCRIPTION** for details.

{ **-L** | **-libvol** } *pathname2*

Tells **eld** to use the specified pathname when searching for libraries. *pathname2* is used in library searches after the public libraries are searched.

pathname2 is either the relative or absolute pathname of an OSS directory.

eld does not verify the names of locations specified in the **-libvol** or **-L** flag.

This flag can be specified more than once in a command line or an obey file. If you specify it more than once, the specified pathnames are searched in the order specified.

See the **Finding Libraries** subsection under **DESCRIPTION** for details about the effect of this flag on search order.

-libname *Guardian_filename*

Stores the specified name within the program being built, to tell the operating system that this is the name of the user library to load into memory along with this program. *Guardian_filename* must be a Guardian file identifier qualified with a disk name and subvolume name; that is, of the form **$***disk***.***subvol***.***fileID*. Use an escape character before the special character **$** in the name so that the OSS shell does not process it, or enclose the entire flag specification in quotation marks.

The **-set** and **-change** flags can also associate a native user library with an executable native program.

The value specified for *Guardian_filename* cannot be the Guardian name of an OSS file.

-limit_runtime_paths

Tells **eld** to mark the loadfile so that **rld** will omit certain locations when searching for DLLs to resolve symbols. See **Finding Libraries** in the **DESCRIPTION** section of this reference page for more information.

The default action is to search all locations described in **Finding Libraries**.

-local_libname *filename8*

Specifies the name of a user library that can be used to resolve references at link time in the program being created.

If this flag is omitted, **eld** uses the value specified for **-set libname** as the user library. If this flag is specified and the **-set libname** flag is omitted, then the value of *filename8* is also stored within the program as the user library name and therefore must name a file in the Guardian filesystem (**/G**); **eld** converts this name to meet the **-set libname** syntax requirements for a Guardian filename.

If the user library cannot be found or cannot be opened, a warning message is issued and the program file is not preset at link time.

-m | **-map** Tells **eld** to produce a memory map of the program or DLL being created.

The default behavior does not produce a memory map.

-make_implicit_lib

Specifies that **eld** should mark the DLL being created as an implicit library. The default action does not mark the DLL as an impicit library.

You cannot use this flag when you use the **-import_lib** flag.

-make_import_lib *filename9*

Tells **eld** to make *filename9* a DLL import library using the DLL files specified elsewhere in the same command line or obey file.

You use this flag to make an import library that represents a:

- Single explicit DLL
- Set of implicit DLLs

If one or more DLLs are specified in the command line or obey file and they are all implicit DLLs, then **eld** creates an import library to represent them. If there is only one DLL in the command line or obey file and it is not an implicit DLL, then **eld** makes the import library to represent it. If there are multiple DLLs in the command line or obey file and they are not all implicit DLLs, an error occurs.

If you are creating an import library to represent a set of implicit DLLs, (all of the specified DLLs have the implicit bit set in their headers), then you can also use a private DLL registry. The value specified for *filename9* is used as the entry name for the implicit libraries in the private DLL registry.

To use this flag, at least one DLL file must be specified somewhere in the command line or obey file.

-must_preset Tells **eld** to treat the situation as an error when presetting fails. If this flag is omitted, the linker continues.

-must_use_iname

Tells **eld** to treat the situation as an error when it cannot delete an existing file to create an import library file of the same name.

The default behavior is to issue a warning message.

-must_use_oname

Tells **eld** to treat the situation as an error when it cannot delete an existing file to create the new object file created from a set of linkfiles, or the output file of the **-alf** or **-strip** flags, with the same name.

The default behavior is to issue a warning message.

-must_use_rname

Tells **eld** to treat the situation as an error when it cannot delete an existing file to create a private DLL registry file of the same name.

The default behavior is to issue a warning message.

-no_optional_lib | **-optional_lib**

Specifies whether a DLL specified in the command line or obey file should be considered optional when creating a loadfile.

When **-no_optional_lib** behavior is in effect, any specified library is included in the **.liblist** section of the loadfile being created. When **-optional_lib** behavior is in effect, a specified library can be omitted from the **.liblist** section of the loadfile being created if omitting it would not affect how symbolic references are resolved. For example, a DLL can be omitted when all of the following are true:

- The DDL does not export symbols that resolve any references in the loadfile being created
- The DLL does not cause other DLLs to be added to the search list, where the other DLLs resolve references in the loadfile being created
- The DLL does not cause other DLLs to be placed in a different order within the search list when those DLLs resolve references in the loadfile being created

These flags can be specified as many times as needed in the command line or obey file. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

If a library is specified more than once, **eld** uses the specification in effect for the first instance of the DLL that it processes and ignores the other specifications.

The default behavior is **-no_optional_lib**.

-no_reexport | **-reexport**
Tells **eld** whether to mark any DLL specified in a **-l** or **-lib** flag after this flag for reexport in its **libList** entry in the loadfile being created. Specifying **-no_reexport** leaves the library unmarked; specifying **-reexport** marks the library. Reexport affects how **eld** does its transitive closure on searches and is used by **rld** to decide how to fix-up symbolic references.

-no_reexport is the default action.

These flags can be specified as many times as needed in the command line or obey file. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

-nostdfiles | **-no_stdfiles**
Specifies that C run-time library functions do not automatically open the standard input and standard output files.

-nostdlib | **-no_stdlib**
Prevents **eld** from searching the standard library locations for DLLs and archive files.

-noverbose | **-no_verbose**
Prevents **eld** from writing warning and informational messages to its output listing. Only error messages and output specifically requested by other options appears in the listing.

The default value is **-no_verbose**.

If you specify more than one of the flags **-warn**, **-verbose**, and **-noverbose** or **-no_verbose** in the command line or an obey file, **eld** displays an inconsistent usage error message.

-o *filename10* Specifies the filename of the output loadfile. *filename10* can be the same as an input file name.

When *filename10* is the same as an existing file and linking is successful, **eld** deletes the existing file and then writes the output file. An error occurs if you do not have permission to delete the existing file and the **-must_use_oname** flag is also specified. (See the description of the **-temp_o** flag in this reference page for a description of the behavior when the **-must_use_oname** flag is not specified.)

If you do not specify a **-o** flag, the default output loadfile filename depends on

whether a **-dllname** or **-soname** flag is specified. *filename10* can also become the DLL name used for the file in the libList. See the description of the **-dllname** flag in this reference page for more information.

-public_registry *filename11*
Tells **eld** to use the file identified as *filename11* as the public DLL registry file.

If this flag is omitted, ld searches as follows to find the name of the public DLL registry file to use:

1. Its own directories, where it looks for a file named **zreg** (normally **/usr/bin/zreg** or **/usr/lib/zreg**)
2. Current data obtained from the operating system using definitions in **/usr/include/dpublib.h**

-rename *old_name new_name*
Changes the symbol name of an externally visible procedure or data item. *old_name* is the name of the procedure or data item to rename. *new_name* is the new name to give the procedure or data item. See the *eld Manual* for details.

{ **-rld_l** | **-rpath** } *path_list1*
Tells **eld** to set search paths in the loadfile for later use with the **-alf** flag or by the **rld** loader. *path_list1* identifies paths to be searched after using the loadfile location and before using the **rld** default locations.

path_list1 contains one or more pathname entries, separated by a colon (**:**). A pathname can be either an absolute OSS directory pathname or a fully qualified Guardian subvolume name.

This flag can be specified more than once in a command line or an obey file. Multiple *path_list1* specifications are concatenated into a single loadfile entry.

See the **Finding Libraries** subsection under **DESCRIPTION** for details about the effect of this flag on search order.

-rld_first_l *path_list2*
Tells **eld** to set search paths in the loadfile for later use with the **-alf** flag or by the **rld** loader. *path_list2* identifies paths to be searched after using the location specified by **-first_l** and before using the public libraries.

path_list2 contains one or more pathname entries, separated by a colon (**:**). A pathname can be either an absolute OSS directory pathname or a fully qualified Guardian subvolume name.

This flag can be specified more than once in a command line or an obey file. Multiple *path_list2* specifications are concatenated into a single loadfile entry.

See the **Finding Libraries** subsection under **DESCRIPTION** for details about the effect of this flag on search order.

-s | **-x** Omits symbol information used for symbolic debugging from the output linkfile or loadfile being created. A file stripped of all symbol information cannot be symbolically debugged with the Visual Inspect or Native Inspect debugger.

You can use this flag only when creating a file. To strip all symbol information from an existing loadfile, use the **-strip** flag.

-set *attribute_name attribute_value*

Sets the value of the run-time attribute specified in *attribute_name* to the value specified in *attribute_value* when creating a loadfile. Use the **-change** flag to change a run-time attribute in an existing loadfile.

Each *attribute_name* has a corresponding range of accepted *attribute_value*s as follows:

- **CPPDIALECT** or **CPLUSPLUSDIALECT** is:

  CPPNEUTRAL or **NEUTRAL**

  If **CPPDIALECT** or **CPLUSPLUSDIALECT** is not specified, the value used comes from the input linkfiles.

- **DATA_MODEL** is one of the following:

  ilp32
  lp64
  neutral

  If **DATA_MODEL** is not specified, the value used comes from the input linkfiles. This attribute is supported for systems running H06.24 or later H-series RVUs or J06.13 or later J-series RVUs only.

- **FLOAT_LIB_OVERRULE** is either **ON** or **OFF**. The default value is **OFF**. (A **FLOAT_TYPE_OVERRULE** value of **ON** is ignored for DLLs; this attribute only has meaning for programs.)

- **FLOATTYPE** is one of the following:

  IEEE_FLOAT
  NEUTRAL_FLOAT
  TANDEM_FLOAT

  If **FLOATTYPE** is not specified, the value used comes from the input linkfiles.

- **HEAP_MAX**, **MAINSTACK_MAX**, [**PROCESS_**]**SUBTYPE**, and **SPACE_GUARANTEE** are unsigned numbers. The default value is 0 (zero). For **HEAP_MAX**, **MAINSTACK_MAX**, and **SPACE_GUARANTEE**, the number of bytes specified is assumed to be decimal unless prefixed by **0x**; the **0x** prefix allows you to specify the size in hexadecimal.

- **HIGHPIN**, **HIGHREQUESTER**[**S**] or **HIGHREQUESTOR**[**S**], and **INSPECT** are either **ON** or **OFF**. The default value is **ON**.

- **INCOMPLETE** is either **ON** or omitted. This attribute can only be set when creating an import library. If it is not set, then the import library is complete. (The default value is omitted, so that a created import library is complete.)

- **INTERPOSE_USER_LIBRARY** is either **ON** or **OFF**. This attribute can be set only for a DLL; at run time, this attribute is disregarded unless the DLL is loaded as a user library. The default value is **OFF**.

- **LIBNAME** is the Guardian filename of a user library file, specified as described for the **-libname** flag. The default value is none.
- **RUNNAMED** and **SAVEABEND** are either **ON** or **OFF**. The default value is **OFF**.
- **RLD_UNRESOLVED** is **ERROR**, **IGNORE**, or **WARN**. The default value is **ERROR**.
- **USER_BUFFERS** is either **ON** or **OFF**. The default value is **OFF**.
- **SYSTYPE** is either **oss** or **guardian**. The default value is determined by the file system that contains the output file. For users of this reference page, the default value is probably **oss**. (If the output loadfile is created in the Guardian file system through the **/G** directory, the default is **guardian**.)

See the *eld Manual* for a description of each run-time attribute.

-show_multiple_defs
Tells **eld** to produce a listing of any symbols with multiple definitions within the input linkfiles.

The default action does not display instances of multiple definitions.

-stdin Reads the contents of the standard input file at the place in the command line where the flag is specified.

-strip *filename12*
Removes information used for symbolic debugging from an existing loadfile with the name *filename12*. A file stripped of all symbol information cannot be symbolically debugged with the Visual Inspect or Native Inspect debugger.

You can use this flag only on an existing loadfile. To strip all symbol information when creating a loadfile, use the **-s** or **-x** flag.

You cannot specify other loadfile filenames or flags other than the following with the **-strip** flag:

-noverbose, **-verbose**, or **-warn**
-fl or **-obey**
-must_use_oname
-stdin
-temp_o

The resulting file has the same **eld** timestamp as before.

-t *address2* Specifies the hexadecimal virtual address at which the text area starts. The default values for *address2* are:

- 70000000 for user programs
- 78000000 for a DLL when no DLL registry is used

The value specified for *address2* is always hexadecimal and can optionally be prefixed by 0H. The specified value is automatically rounded up to a multiple of 128 kilobytes for a implicit DLL library or 64 kilobytes for other types of loadfiles. If no more than 8 hexadecimal digits are specified, the number specified is sign-extended to 16 digits.

You cannot use this flag if you use the **-check_registry** or **-update_registry** flags.

You should not use this flag if you want the DLL address to be determined from a DLL registry file under your control.

-temp_i *filename13*
Tells **eld** to save its import library work with the specified file name until it has successfully rewritten the final file.

If an unqualified filename is used, **eld** saves the file in the directory where *filename10* is located.

If the **-temp_i** flag is not used and **eld** finishes creation of a new temporary file but cannot remove an existing file with the same name as the specified import library file, **eld** leaves the temporary file in the same directory as the output file and gives it a unique filename beginning with the letters **ZLD**.

This flag can be used only when the **-import_lib**, **-import_lib_stripped**, or **-make_import_lib** flag is used.

-temp_o *filename14*
Tells **eld** to save its output for the new object file created from a set of linkfiles, or the output file of the **-alf** or **-strip** flags, with the specified file name until it has successfully rewritten the final file.

If an unqualified filename is used and a new object file is being created or the **-strip** flag is used, **eld** saves the file in the directory where *filename10* is located. If an unqualified filename and an output file is being created when the **-alf** flag is used, **eld** saves the file in the directory where *filename1* is located.

If the **-temp_o** flag is not used and **eld** finishes creation of a new temporary file but cannot remove an existing file with the same name as the specified output file, **eld** leaves the temporary file in the same directory as the output file and gives it a unique filename beginning with the letters **ZLD**.

-temp_r *filename15*
Tells **eld** to save its private registry update work with the specified file name until it has successfully rewritten the final file.

If an unqualified filename is used, **eld** saves the file in the directory where *filename10* is located.

If the **-temp_r** flag is not used and **eld** finishes creation of a new temporary file but cannot remove an existing file with the same name as the specified registry file, **eld** leaves the temporary file in the same directory as the output file and gives it a unique filename beginning with the letters **ZLD**.

-u *symbol_name4*
Tells **eld** to add *symbol_name4* as an undefined symbol. This causes **eld** to search for this symbol in any archive libraries that are specified on the command line or in an obey file.

-ul Creates a user library. Specify this flag when linking modules to create a DLL as a native user library.

When you specify the **-ul** flag, the exported symbols are those described as exported by the **-export_all** flag, unless you also use the **-export_not** or **-hidden_symbol** flag.

-unres_symbols { **error** | **ignore** | **warn** }
Tells **eld** what action to take when a needed symbol cannot be resolved:

error	Issue an error message. This is the default action. The **-error_unresolved** flag is recognized as a synonym for this specification.
ignore	Ignore the missing symbol reference.
warn	Issue a warning message. The **-warn_unresolved** flag is recognized as a synonym for this specification.

Other flag specifications override these flags. If you specify more than one of these flags in the command line or an obey file, **eld** displays an inconsistent usage error message.

-update_code Tells **eld** to update relocation sites in the code segment when the **-alf** flag is also used.

-update_registry *filename16*
Tells **eld** to update the **-range** entry in the private DLL registry identified as *filename16* with the address range for the file specified as *filename10*.

If no **-range** entry is found, a range entry is appended to the registry file content.

If neither the **-check_registry** nor **-update_registry** flag is specified, **eld** does not use a private DLL registry.

You must also use the **-dll** flag when you use this flag.

-verbose Directs **eld** to write error, warning, and informational messages to its output listing, along with output specified by other options.

The default value is **-no_verbose**.

If you specify more than one of the flags **-warn**, **-verbose**, and **-noverbose** or **-no_verbose** in the command line or an obey file, **eld** displays an inconsistent usage error message.

-warn Directs **eld** to write only error and warning messages to its output listing, along with output specified by other options.

The default value is **-no_verbose**.

If you specify more than one of the flags **-warn**, **-verbose**, and **-noverbose** or **-no_verbose** in the command line or an obey file, **eld** displays an inconsistent usage error message.

-warn_common
Directs **eld** to issue a warning message when a common symbol is combined with another common symbol that has the same name but a different size.

This flag is supported for systems running H06.21 or later H-series RVUs or J06.10 or later J-series RVUs only.

-y *symbol_name5*

Tells **eld** to report which linkfiles define and use the symbol *symbol_name5*. The linkfiles are listed in the order encountered.

This information can be useful if a previous **eld** command produced error or warning messages about a symbol being either undefined or defined more than once.

Operands

filename17 Specifies one or more files for the **eld** utility to process.

The files used can be:

- Linkfiles that **eld** will combine into a new linkfile or loadfile
- DLLs that **eld** will use to resolve references in a new loadfile that it is creating, or in a loadfile being processed by the **-alf** flag
- DLLs that are input to the **-make_import_lib** flag

This operand is required for all flags except the **alf**, **-change**, and **-strip** flags.

DESCRIPTION

The **eld** utility links one or more TNS/E native position-independent code (PIC) linkfiles to produce an executable or nonexecutable native PIC loadfile. You can also modify existing loadfiles using **eld**. You can invoke **eld** directly or, if you are creating a C or C++ program, you can use the **c89** or the **c99** utility to invoke **eld** automatically for you.

Flag Names

Names of flags obey the following rules:

- If a flag name is one letter and the flag takes a value, the space between the flag name and the value is optional
- Flag names and keyword values are not case sensitive, except for the separate **-l** and **-L** flags

If no flags or operands are used, the **eld** command displays usage help in its output listing.

Input Files

eld cannot process a linkfile that contains a code section larger than 16 megabytes. This size restriction is imposed by the Itanium standard.

Output Files

eld creates loadfiles on the host platform with an OSS file mode of 777 (rwxrwxrwx), which is then ANDed with the **umask** value of the user creating the file. **eld** creates all other object files with an OSS file mode of 666 (r-xr-xr-x), which is then ANDed with the **umask** value of the user creating the file.

Any text file created by ld in the Guardian file system has a file code of 180. You can use the CTOEDIT utility to convert such files to code 101 files for viewing with TEDIT.

Loadfiles or linkfiles created in the Guardian file system have a Guardian file code of 800.

Maintaining DLL Registry Files

A DLL registry file is a C text file that contains attribute information associated with a set of DLLs. The file contains:

- An optional **-dllarea** specification that defines the first and last addresses of the virtual memory area that is to be assigned to all DLLs. This specification has the form:

  -dllarea *dllarea_start_address dllarea_end_address*

where both values are expressed in hexadecimal. If either value contains no more than 8 digits, it is automatically sign-extended to 16 digits. If *dllarea_start_address* is less than *dllarea_end_address*, new addresses are assigned in ascending order; otherwise, new addresses are assigned in descending order. The larger of the two address values is noninclusive while the smaller address is inclusive.

The default values are:

dllarea_start_address
0x080000000

dllarea_end_address
0x070000000

- A **-range** specification for each DLL in the set of DLLs. This specification has the form:

  -range *dll_file_name dll_start_address dll_max_size*

  where:

  dll_file_name is a loadfile filename (the **-o** flag *filename10* value or its equivalent)

  dll_start_address
  is the first location of the text space (which can be set using the **-t** flag)

  dll_max_size includes the text space, the data space, and space for growth

  The address range for a DLL begins at *dll_start_address* and extends through *dll_start_address+dll_max_size*. Range specifications that do not overlap allow for faster loading of DLLs. Overlapping range specifications must be created using a text editor.

Fields can be separated by spaces or tabs. Two successive hyphens indicate a comment that extends to the end of the current line.

The **eld** linker appends new **-range** specifcations and can maintain existing **-range** specifications. The **-dllarea** specification must be created using a text editor.

Saving Temporary Files

eld creates temporary working files while it processes command line or obey file information. These temporary working files are given names of the form **ZLDAF***nnn* (for all files except import libraries and private DLL registry files), **ZLDAI***nnn* (for import libraries), or **ZLDAR***nnn* (for private DLL registry files), where:

nnn is a unique sequentially assigned decimal number, beginning with 000

To create a final permanent file with the same name as an existing file, **eld** must first remove the existing file. If **eld** processing is interrupted when removing and recreating the final file, the working file is preserved as a file named **ZLD***nnn*.

The **temp_i**, **-temp_o**, or **-temp_r** flag allows you to save the completed working file as a temporary regular file with a known filename before the original file is removed. The temporary file is itself removed after the final permanent file is completely written.

Finding Libraries

If you specify an absolute or relative pathname for a flag's *filenamen* value, no search occurs. **eld** opens the specified file to see if it is a linkfile, an archive, or a DLL. If the file cannot be opened, an error occurs.

If you specify an unqualified filename (a filename value that does not contain a / character) for the **-l** or **-lib** flag, the OSS version of **eld** attempts to find the file. **eld** searches for libraries in the following locations when resolving the values specified for the **-l** or **-lib** flags:

1. Locations specified by the current **-first_l** flag
2. Public libraries (installed by the system operator) *** ****
3. Locations specified by the current **-libvol** and **-L** flags
4. Default locations in the OSS environment when building a 32-bit or neutral object: **/lib:/usr/lib:/usr/local/lib:/G/SYSTEM/ZDLL * ****

   Default locations in the OSS environment when building a 64-bit object: **/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib:/G/SYSTEM/ZDLL * ****

   The value of the **COMP_ROOT** environment variable is added to the beginning of **/lib**, **/usr/lib**, **/usr/local/lib**, **/lib64**, **/usr/lib64**, and **/usr/local/lib64**. By default, the value of **COMP_ROOT** is null in the OSS environment.

The steps marked by an asterisk (*****) are skipped when the **-nostdlib** or **-no_stdlib** flag is in effect, and the steps marked by two asterisks (******) are skipped only when the **-bstatic** flag is used. Archive libraries encountered in steps marked by ****** are reported as errors.

eld searches in each location for libraries by name. **eld** tries to open the specified file. If it cannot, it modifies the supplied value and tries to open a file with the modified name. (Unqualified names specified in the Guardian file system, **/G**, are not modified; **eld** uses only the supplied value.) The prefix **lib** and the following suffixes are added to the specified name to create the modified name:

.so To find a DLL, unless the **-bstatic** flag is in effect

.a To find an archive file, unless the **-dllsonly** flag is in effect

To find a DLL, the prefix **z** and the suffix **dll** are also tried unless the **-bstatic** flag is in effect.

When the **-alf** flag is used, the OSS version of **eld** searches for DLLs in the following locations:

1. Locations specified by the current **-first_l** flag
2. Locations specified by the current **-rld_first_l** flag
3. Public libraries (installed by the system operator) *****
4. The directory containing the loadfile (which might be a program or a DLL) ******
5. The files specified by **L**
6. Locations specified by the current **-rld_l** or **-rpath** flag
7. Default locations when the current loadfile is a 32-bit or neutral loadfile: **/lib:/usr/lib:/usr/local/lib:/G/SYSTEM/ZDLL * ****

   Default locations when the current loadfile is a 64-bit loadfile: **/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib:/G/SYSTEM/ZDLL * ****

   The value of the **COMP_ROOT** environment variable is added to the beginning of **/lib**, **/usr/lib**, **/usr/local/lib**, **/lib64**, **/usr/lib64**, and **/usr/local/lib64**. By default, the value of **COMP_ROOT** is null in the OSS environment.

The steps marked by an asterisk (*****) are skipped when the **-nostdlib** or **-no_stdlib** flag is in effect. When the **-limit_runtime_paths** flag has been used for the loadfile, the following are omitted from **rld**'s search:

- The steps marked by two asterisks (******) in the previously described search order

- Paths indicated by the TACL DEFINEs _RLD_FIRST_LIB_PATH and _RLD_LIB_PATH.

For More Information

eld is not an interactive tool like Binder. For more information on using **eld**, see the *eld Manual*. For more information on run-time library use, see the *rld Manual*.

EXAMPLES

1. The following example:

   eld objecta objectb -o objectc

   links together the input linkfiles named **objecta** and **objectb** to create a program named **objectc**.

2. The following example:

   eld -dll -o objecta objectb

   creates a DLL whose DLL name is **objecta** and whose filename is **objecta** from the linkfile named **objectb**.

3. The following example:

   eld obj1.o obj2.o -ul -o lib

   links the linkfiles named **obj1.o** and **obj2.o** together into a user library named **lib**.

4. The following example:

   eld obj3.o obj4.o -o prog -libname \$A.B.C

   links linkfiles named **obj3.o** and **obj4.o** together into a loadfile named **prog**. When **prog** runs, it has a user library with the Guardian name **$A.B.C**. The backslash (\) prevents the shell from misinterpreting the dollar sign ($).

5. The following example:

   eld obj6.o obj7.o -o prog -set systype guardian

   links the linkfiles named **obj6.o** and **obj7.o** into a loadfile named **prog** that you intend to run as a Guardian process.

6. The following example:

   eld -change highpin off exeobj

   changes the value of the **HIGHPIN** attribute in the loadfile **exeobj** to **OFF**.

7. The following is an example of a valid private DLL registry file:

   -- Specify the overall domain area that DLLs
   -- can reside in (high to low)
   -dllarea 0x68000000 0x60000000
   -- Set up two DLLs that overlap each other
   -- (one entry manually inserted)
   -range $SYSTEM.ZDLL.ABC 0x67D00000 0x100000
   -range /bin/usr/don/libmy.lib 0x67D00000 0x300000

FILES

/usr/bin/zimpimp
First default for the import library that represents the implicit DLLs

/usr/lib/zimpimp
Alternate default for the import library that represents the implicit DLLs

/usr/bin/zreg First default for the public DLL registry file

/usr/lib/zreg Alternate default for the public DLL registry file

NOTES

In the Guardian version of **eld**, MAP DEFINE names can be used to identify Guardian files wherever the ELD command allows entry of a filename. The OSS version of **eld** does not support the use of MAP DEFINEs. However, OSS filenames used with **eld** cannot begin with = because **eld** uses the = character to identify MAP DEFINEs for expansion into Guardian filenames.

EXIT VALUES

The **eld** command returns one of the following values:

0 (zero) No errors or warning conditions were detected.

1 One or more error or warning conditions were detected.

RELATED INFORMATION

Commands: **c89(1)**, **c99(1)**, **enoft(1)**, **ld(1)**, **nld(1)**, **noft(1)**.

Files: **float(4)**.

STANDARDS CONFORMANCE

The **eld** command is an HP extension to the Single UNIX specification and performs functions comparable to the UNIX **ld** command.

NAME

enoft - Reads and displays information from TNS/E native object files

SYNOPSIS

enoft

[-break key on keyboard]
[**-CD** [*dir_pathname1*]]
[{ **-CLOSE** | **-CL** } { ***** | *filenum* | *filename* }]
[**-COMMENT** [*text*]]
[**-COMP** [*ref_objfile*] *target_objfile* [**DETAIL** | **D**]]
[**-DBGINFO** { *proc_addr* | *proc_spec* }]
[{ **-DEMANGLE** | **-DE** } *proc_spec*]
[{ **-DIR** | **-FILES** } [*dir_pathname2*]]
[{ **-DUMPADDRESS** | **-DA** } *scope* [**IN** *format_spec*]]
[{ **-DUMPALL** | **-ALL** } [***** | **LIST**]]
[{ **-DUMPCODE** | **-DC** }
[{ **BRIEF** | **B** } | **IN** *format_spec*]]
[{ **-DUMPDATA** | **-DD** }
[{ **BRIEF** | **B** } | **IN** *format_spec*]]
[{ **-DUMPOFFSET** | **-DO** } *scope*
[**IN** *format_spec*]]
[{ **-DUMPPROC** | **-DP** } *proc_spec* [*scope*]
[**IN** *format_spec*]]
[{ **-DUMPSECTION** | **-DS** }
[***** [**DETAIL** | **D**] | *sect_name* | *sect_num*]
[**IN** *format_spec*]]
[**-DWARF** [***** | **ABBREV** |
INFO | **LINE** [**ORDINAL**] | **LOC**]
[**-DYNAMIC**]
[**-ENV**]
[{ **-EXIT** | **-E** | **-QUIT** | **-Q** }]
[{ **-FC** | **-!** } [*hist_num* | *-hist_offset* | *text*]]
[{ **-FILE** | **-F** } *ref_objfile*]
[**-FILEHDR**]
[**-FUNCDESC** | **-FD**]
[**-GOT**]
[**-HASH**]
[**-HASHVAL**]
[{ **-HELP** | **?** } [*flag* | *help_topic*]]
[{ **-HISTORY** | **-H** } [*hist_num*]]
[**-LAYOUT** [**CODE** | **DATA** | *****]]
[**-LIBLIST**]
[**-LIC**]
[{ **-LISTATTRIBUTE** | **-LA** } [**DETAIL** | **D**]]
[{ **-LISTCOMPILERS** | **-LC** } [**DETAIL** | **D**]]
[{ **-LISTDATA** | **-LD** }]
[{ **-LISTDEBUG** | **-LDE** }
[***** | { **PROC** | **P** } | {**DATA** | **D** }]
[**DETAIL** | **D**]]
[{ **-LISTEXPORTS** | **-LE** }]
[{ **-LISTFOPEN** | **-LFO** }]
[{ **-LISTOPTIMIZE** | **-LO** }
[**0** | **1** | **2** | ***** |
{ **EXCLUDE** | **E** } | { **BRIEF** | **B** } }]

[{ **-LISTPROC** | **-LP** } { *proc_spec* | ***** }
[{ **EXCLUDE** | **E** } | { **SUBPROC** | **SP** } |
{ **NOSUBPROC** | **NSP** }] [{ **DETAIL** | **D** }]]
[{ **-LISTSOURCE** | **-LS** }
[***** | *sourcendx* | *pathname3* | *file_num1*]
[{ **DETAIL** | **D** }]]
[{ **-LISTUNREFERENCED** | **-LUR** }
{ { **PROC** | **P** } | { **DATA** | **D** } | ***** }
[{ **DETAIL** | **D** }]]
[{ **-LISTUNRESOLVED** | **-LU** }
[{ **PROC** | **P** } | { **DATA** | **D** } | *****]
[{ **DETAIL** | **D** }]]
[**-LOG** [**OFF** | *pathname4* [**ASCII**]]]
[**-NOEXIT**]
[**-OBEY** *pathname5*]
[**-OUT** [**OFF** | *pathname6* [**ASCII**]]]
[{ **-PROCINFO** | **PI** }]
[**-PROGHDRS**]
[**-RELOC**]
[**-RESET** [*set_cmd* | *****]]
[**-RTDU** [{ **SOURCE** | **OBJECT** | ***** } [**DETAIL** | **D**]]]
[**-SECTHDRS**]
[**-SET** [*set_cmd* [*value*]]]
[**-SHOW** [*set_cmd* | *****]]
[**-STRTAB** [***** | **DYNSTR** | **DYNSTR2** | **PROCNAMES** |
RTDU | **SHSTRTAB** | **STRTAB** | **UNWIND**]
[{ **-SYMTAB** | **-SYMBOLS** } [***** | { **EXPORT** | **E** } |
{ **PROC** | **P** } | { **DATA** | **D** }]
[**-TANDEMINFO** | **-TDM**]
[**-UNWIND**]
[{ **-UNWINDINFO** | **UWI** }
[{ ***** | *proc_spec* } [**DETAIL** | **D**]]
[{ **-XREFPROC** | **-XP** } { *proc_spec* | ***** }
[**CALLEDBY** | **CALLS** | **BOTH**]
[{ **DETAIL** | **D** }]]
...
[*ref_objfile*]

FLAGS

-CD [*dir_pathname1*]

Changes the working directory **enoft** uses to search for relative pathnames. The specified pathname can be absolute or relative; relative pathnames are resolved using the current working directory.

Files in the Guardian file system can be used if a fully or partially qualified subvolume name is used in OSS pathname format (/**G**/*vol*/*subvol*). A subvolume on another node can be specified using the OSS pathname format for Expand-connected nodes (for example, a Guardian subvolume is specified as /**E**/**ode**/**G**/*vol*/*subvol*). When *dir_pathname1* is omitted, the default is the current working directory.

This flag is the equivalent of the **enoft** VOLUME command used in the Guardian environment. A valid value for *pathname_prefix* in the Guardian environment is a fully or partially qualified subvolume name and the default is the current volume

and subvolume.

{ **-CLOSE** | **-CL** } { ***** | *filenum* | *filename* }
Closes the specified files:

***** Closes all open files.

filenum Closes the file specified by the number **filenum**.

filename Closes the file specified by *filename*.

Closinig a log file or out file has the same effect as using the **-LOG OFF** or -LOUT OFF flag.

-COMMENT [*text*]
Allows comments in **enoft** command files. Comments are not displayed in output.

-COMP [*ref_objfile*] *target_objfile* [**DETAIL** | **D**]
Compares the two specified object files for major differences, including file headers and program headers. DWARF symbol tables are not compared.

If *ref_objfile* is not specified, the current object file is used to compare with *target_objfile*.

DETAIL or **D** provides additional, detailed information.

-DBGINFO { *proc_addr* | *proc_spec* }
Lists compilation source and debug file information for a given procedure name, index, or address. When *proc_addr* is used, the line number and instruction bundle address are also listed. If neither *proc_addr* nor *proc_spec* are specified, all procedures are listed.

proc_addr
Specifies the hexadecimal value of a code section address to be listed. This value must have the form 0*xxxxxxxxx*, where *x* is a hexadecimal digit. If the value specified is not on a 16-byte alignment boundary, **enoft** rounds the address down to the beginning of the address bundle. If line number information for the bundle is not available, **enoft** displays information from the nearest preceding bundle with line number information. Demangled names are listed when possible.

proc_spec
Specifies the procedure name or procedure number. Procedure names are case-sensitive. *proc_spec* is one of the following:

proc_name
Limits the scope to the specified procedure and subprocedures. The demangled form of the procedure name cannot be used because **enoft** does not support blank spaces in the name.

proc_num
Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

This command is used for loadfiles and import libraries.

{ **-DEMANGLE** | **-DE** } *proc_spec*

Displays the C++ symbol name specified by *proc_spec* in demangled format. An object file does not need to be open before using this flag.

proc_spec

Specifies the procedure name or procedure number. Procedure names are case-sensitive. *proc_spec* is one of the following:

proc_name

Limits the scope to the specified procedure and subprocedures. The demangled form of the procedure name cannot be used because **enoft** does not support blank spaces in the name.

proc_num

Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LIST-PROC** flag to list each procedure number.

{ **-DIR** | **-FILES** } [*dir_pathname2*]

Lists the files in the specified directory. The specified pathname can be absolute or relative; relative pathnames are resolved using the current working directory.

Files in the Guardian file system can be listed if a fully or partially qualified subvolume name is used in OSS pathname format (**/G/***vol*/*subvol*). A subvolume on another node can be specified using the OSS pathname format for Expand-connected nodes (for example, a Guardian subvolume is specified as **/E/ode/G/***vol*/*subvol*). When *dir_pathname2* is omitted, the default is the current working directory.

{**-DUMPALL** | **-ALL** } [***** | **LIST**]

Displays all nonzero size sections in the object file, including the file, program, and section headers. The sections are displayed in the order of their relative file offsets. When you do not specify an option, the display is similar to the output of the **-LAYOUT** flag but shows the section contents.

Unlike the **-DUMPSECTION * DETAIL** flag, output stops when an underlying dump command fails (for example, when a data error occurs).

***** Also lists the layout of the sections, a list of common file attributes, compiler information, various symbols, optimization levels, procedures, and source files.

If the **-DUMPALL** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

LIST Only lists the file, program, and section headers, the layout of the sections, a list of common file attributes, compiler information, various symbols, optimization levels, procedures, and source files.

{**-DUMPADDRESS** | **-DA** } *scope* [**IN** *format_spec*]

Displays code and data from a virtual address inside an object file's memory space. *scope* is the following:

start_address [*range_spec*]

start_address

Specifies the starting virtual address. The value specified is assumed to be in decimal format unless prefixed by **0x** to make it hexadecimal format. For **ICODE** or **INNERLIST** displays, *start_address* must be on a 16-byte boundary; for all other formats, **enoft** rounds the specified address down to the beginning of the bundle to be displayed.

range_spec

Specifies the amount of information to display. *range_spec* is one of the following:

TO *end_address*

Is the ending virual address. The value specified is assumed to be in decimal format unless prefixed by **0x** to make it hexadecimal format. For **ICODE** or **INNERLIST** displays, *end_address* must be on a 16-byte boundary; for all other formats, **enoft** rounds the specified address down to the beginning of the next bundle.

Valid values must be within the same section as the *start_address*. If the scope of the output is limited by another flag to a single procedure, only values within that procedure are valid.

If you omit **TO** *end_address*, only one 4-byte or 16-byte unit is displayed.

FOR *number* [{ **BYTES** | **B** } | {**UNITS** | **U** }]

Is the number of bytes or units (4-byte or 16-byte groups) to display. Valid values for *number* can be in decimal or hexadecimal format. The value specified is assumed to be in decimal format unless prefixed by **0x** to make it hexadecimal format.

If **BYTES**, **B**, **UNITS**, or **U** is omitted, the number displayed is the specified multiple of an 8-bit value, depending on the part of the object file being displayed; code instructions in a code section are displayed in multiples of 16-byte bundles, while all other data is displayed in multiples of 4 bytes.

FOR * [{ **BYTES** | **B** } | {**UNITS** | **U** }]

Displays information to the end of the procedure, subprocedure, or section. If **BYTES**, **B**, **UNITS**, or **U** is omitted, code instructions in a code section are displayed in multiples of 16-byte bundles, while all other data is displayed in multiples of 4 bytes.

If you use the **-DUMPADDRESS** flag on an OSS shell command line, the flag specification must be enclosed in quotation marks when you use the **FOR *** specifier.

IN *format_spec*

Specifies how the information is to be formatted. *format_spec* is one of the following:

ASCII | **A** Displays portions of the object file in ASCII format.

DECIMAL | **D**
Displays portions of the object file in decimal format.

HEX | **H** Displays portions of the object file in hexadecimal format.

ICODE | **IC** Displays portions of the object file in disassembled program code. This is the default format.

INNERLIST | **IN**
Displays portions of the object file in disassembled code and displays the source code interspersed with the assembler. This option can be used only for dumping text.

READABLE | **R**
Displays portions of the object file in an applicable format based on the item type and part of the object file being displayed. This is the default action.

{**-DUMPCODE** | **-DC** } [{ **BRIEF** | **B** } | **IN** *format_spec*]

Displays all code from the **.gateway**, **.plt**, **.text**, and **.restext** sections of an object file.

Specifying the **BRIEF** or **B** option is the same as specifying the **-LAYOUT CODE** flag.

Valid values for *format_spec* are identical to those of the **-DUMPADDRESS** flag. If you omit **IN** *format_spec*, the display is formatted in **ICODE** format.

{**-DUMPDATA** | **-DD** } [{ **BRIEF** | **B** } | **IN** *format_spec*]

Displays all initialized user data from the **.data**, **.sdata**, **.rdata**, and **.rconst** sections of an object file.

Specifying the **BRIEF** or **B** option is the same as specifying the **-LAYOUT DATA** flag.

Valid values for *format_spec* are identical to those of the **-DUMPADDRESS** flag. If you omit **IN** *format_spec*, the display is formatted in **HEX** format.

{**-DUMPOFFSET** | **-DO** } *scope* [**IN** *format_spec*]

Displays code and data from a physical offset within an object file. Valid values for *scope* are identical to those of the **-DUMPADDRESS** flag, except that the addresses are physical offsets within the file instead of virtual addresses. Valid values for *format_spec* are identical to those of the **-DUMPADDRESS** flag.

{**-DUMPPROC** | **-DP** } *proc_spec* [*scope*] [**IN** *format_spec*]

Displays the contents of a procedure or part of a procedure.

proc_spec

Specifies the procedure name or procedure number. Procedure names are case-sensitive in C and C++ but not in other languages. *proc_spec* is one of the following:

proc_name
Limits the scope to the specified procedure and subprocedures.

If *proc_name* is not completely specified, **enoft** resolves the name and lists conforming procedure names with numbers; wildcard matching (for example, *******partial_name* or *partial_name*******) can be used to search for items containing a match to the given pattern but only the first match found is displayed.

The demangled form of the procedure name cannot be used because **enoft** does not support blank spaces in the name.

proc_name.*subproc_name*[[**.subprocname**]...]]
Limits the scope to the specified subprocedure. If *proc_name* or *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified.

subproc_name[[**.subprocname**]...]]
Limits the scope to the specified subprocedure. If *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified. Only the first subprocedure with the specified name is displayed.

proc_num
Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

scope Specifies addresses in a form that is identical to that of the **-DUMPADDRESS** flag, except that the addresses are virtual offsets within the file instead of virtual addresses.

IN *format_spec*
Specifies how the information is to be formatted. Valid values for *format_spec* are identical to those of the **-DUMPADDRESS** flag.

{**-DUMPSECTION** | **-DS** } [***** [**DETAIL** | **D**] | *sect_name* | *sect_num*] [**IN** *format_spec*]
Displays the specified section of the object file. If you omit all options, the default display lists the sections in the file.

Valid display options are:

***** Displays all sections of the object file except the file, program, and section headers. If **DETAIL** or **D** is also specified, only nonzero size sections are displayed.

If you use the **-DUMPSECTION** flag on an OSS shell command

line, the flag specification must be enclosed in quotation marks when you use the ***** specifier.

sect_name Specifies any valid section name, as displayed by the **-SECTHDRS** or **-LAYOUT** flag.

sect_num Specifies any valid section number, as displayed by the **-SECTHDRS** flag.

IN *format_spec*

Specifies how the information is to be formatted. *format_spec* is one of the following:

ASCII | **A** Displays portions of the object file in ASCII format.

DECIMAL | **D**
Displays portions of the object file in decimal format.

HEX | **H** Displays portions of the object file in hexadecimal format.

ICODE | **IC** Displays portions of the object file in disassembled program code. This is the default format.

INNERLIST | **IN**
Displays portions of the object file in disassembled code and displays the sourc e code interspersed with the assembler. This option can be used only for dumping text.

READABLE | **R**
Displays portions of the object file in the format most appropriate to the type of data. This is the default action.

-DWARF [***** | **ABBREV** | **INFO** | **LINE** [**ORDINAL**] | **LOC**]

Displays the contents of the Debugging With Attribute Record Format (DWARF) debugging symbol table sections of an object file. These sections provide the symbolic information used by the Visual Inspect and Native Inspect debuggers.

***** Displays the contents of the **.debug_info**, **.debug_abbrev**, and **.debug_line** sections.

ABBREV Displays only the contents of the **.debug_abbrev** section.

INFO Displays only the contents of the **.debug_info** section, as constrained by the **-SET SCOPEPROC** flag.

LINE [**ORDINAL**]
Displays only the contents of the line number table (**.debug_line**) section. If a **.debug_line_nsk** section exists to support the use of Guardian EDIT utility line numbers, that section is displayed instead. To force display of **.debug_line** instead of **.debug_line_nsk**, specify the **ORDINAL** option.

LOC Displays only the contents of the **.debug_loc** section.

-DYNAMIC Displays the **.dynamic** section of a loadfile or import library.

-ENV Displays the current settings of the **enoft** environment.

{**-FILE** | **-F** } *ref_objfile*
Specifies the name of the target object file you want to use with **enoft**. *ref_objfile* can be a simple filename or an absolute or relative pathname.

A subsequent **-FILE** flag closes the current object file and opens the newly specified object file.

-FILEHDR Displays the contents of the Executable and Linking Format (ELF) file header from the beginning of the object file.

-FUNCDESC | **-FD**
Displays the contents of the **.IA_64.pltoff** and **.fptr** function descriptor sections of a loadfile.

-GOT Displays the global offset table section of a loadfile.

-HASH Displays the contents of the **.hash** and **.hash.gblzd** sections of a loadfile.

-HASHVAL Displays the contents of the **.hasval** and **.hashval.gblzd** sections of a loadfile.

{**-HELP** | **?** } [*flag* | *help_topic*]
Displays descriptions and syntax for **enoft** flags and operands. If no value is specified for *flag* or **help_topic**, **enoft** displays a single line description of each **enoft** flag. This information can be directed to an output file or log file.

flag Displays information about the specified flag, including syntax. *flag* can be any valid command name or command abbreviation.

help_topic Specifies one of the following **enoft** topics for which you want detailed information:

CMD_TYPE
CMD_ENTRY
OBJECT_FILES
FORMAT_SPEC
PROC_SPEC
SCOPE_RANGE
SOURCE_SPEC

-LAYOUT [**CODE** | **DATA** | *****]
Lists the parts of an object file in order by file offset, along with their virtual addresses.

CODE Lists only code sections.

DATA Lists only data sections.

***** Lists both code and data sections. If you omit an option for this flag, this is the default action.

-LIBLIST Displays the **.liblist** section of a loadfile or import library. This flag is an alias for specification of the **-DUMPSECTION** flag with the *section_name* value of **.liblist**.

-LIC Displays the library import characterization (LIC) section **.lic** of a preset loadfile. This flag is an alias for specification of the **-DUMPSECTION** flag with the *section_name* value of **.lic**.

{**-LISTATTRIBUTE** | **-LA** } [{ **DETAIL** | **D** }]
Lists common file and process attributes associated with an object file. **DETAIL** or **D** provides additional, detailed information about the debugging sections of the file.

{**-LISTCOMPILERS** | **-LC** } [{ **DETAIL** | **D** }]
Lists version information about the native compiler components and **eld** utility used to create an object file. **DETAIL** or **D** provides additional, detailed information.

{ **-LISTDATA** | **-LD** }
This flag is an alias for the **-SYMTAB DATA** flag. It lists all data symbols from the **.dynsym** and **.dynsym.gblzd** sections in a loadfile or import library, and the **.symtab** section in a linkfile.
[**DETAIL** | **D**]]

{ **-LISTDEBUG** | **-LDE** } [***** | { **PROC** | **P** } | {**DATA** | **D** }] [**DETAIL** | **D**]]
Lists all names in the **.debug_info** symbols table that meets variable (data) or subprogram (subprogram, subroutine, or alternate entry point) criteria. Specifying **PROC**, **P**, **DATA**,or **D** restricts the content of the listing to procedures or data; the default is *****, meaning all. **DETAIL** or **D** provides additional, detailed information.

{ **-LISTEXPORTS** | **-LE** }
This flag is an alias for the **-SYMTAB EXPORT** flag. This flag lists all exported symbols that are global and defined from the **.dynsym** and **.dynsym.gblzd** sections in a loadfile or import library. Symbols available only to other linkfiles and local to loadfiles (not exported) are not listed.

{n-**LISTFOPEN** | **-LFO** }
Lists the file numbers and filenames of all open files including object files, log files, and out files.

{**-LISTOPTIMIZE** | **-LO** } [**0** | **1** | **2** | ***** | { **EXCLUDE** | **E** } | { **BRIEF** | **B** }]
Lists procedures based on their optimization level. If no option is specified, all procedures in the object file are displayed, sorted by optimization level.

0, **1**, or **2** List only procedures with an optimization level corresponding to the specified number.

***** List all procedures sorted by optimization level. If the **-LISTOPTIMIZE** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

EXCLUDE | **E**
List all procedures sorted by optimization level but does not display information for symbols generated by a compiler or not found in a **debug_info** section.

BRIEF | **B**
Limit the display to counts of symbols matching the scope.

{**-LISTPROC** | **-LP** } { *proc_spec* | ***** }
[{ **EXCLUDE** | **E** } | { **SUBPROC** | **SP** } | { **NOSUBPROC** | **NSP** }]
[{ **DETAIL** | **D** }]
Lists procedures and their subprocedures.

proc_spec Specifies the procedure name or procedure number. Procedure names are case-sensitive in C and C++ but not in other languages. *proc_spec* is one of the following:

proc_name
Limits the scope to the specified procedure and subprocedures. If *proc_name* is not completely specified, **enoft** resolves the name and lists conforming procedure names with numbers. Wildcard matching (for example, *******partial_name* or *partial_name*******) can be used to search for items containing a match to the given pattern.

The demangled form of the procedure name cannot be used because **enoft** does not support blank spaces in the name.

*proc_name.subproc_name[[***.***subproc_name*]...]
Limits the scope to the specified subprocedure. If *proc_name* or *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified.

subproc_name[[**.***subproc_name*]...]
Limits the scope to the specified subprocedure. If *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified. Only the first subprocedure with the specified name is displayed.

proc_num
Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

***** Specifies all procedures in the current scope. If the **-LISTPROC** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

The content of the display can be controlled with the following options:

EXCLUDE | E Suppresses display of information for symbols generated by a compiler or not found in a **.debug_info** section.

SUBPROC | SP
Does not list parent procedures along with subprocedures. If procedure P contains subprocedure S, a **-LISTPROC S SUBPROC** flag lists only S and not P, even though S is contained within P.

NOSUBPROC | NSP
Does not list subprocedures along with parent procedures. If procedure P contains subprocedure S, a **-LISTPROC P NOSUBPROC** flag lists only P and not S, even though S is contained within P.

DETAIL | D Provides additional, detailed information about procedures and subprocedures.

{**-LISTSOURCE | -LS** } [*sourcendx* | *pathname3* | *file_num1* | *****]
[{ **DETAIL** | **D** }]
Lists compilation unit entries in the object file for a unit index value, for a source file, or for all source files used. If only one procedure is dumped, then the **-LISTSOURCE** flag dumps the entry for the source file containing the procedure.

sourcendx Narrows the scope to a single compilation unit within the object file. *sourcendx* must be a valid index value. Use the **-LISTSOURCE** flag to determine valid values.

pathname3 Narrows the scope to a single named source file. *pathname3* must be a fully qualified OSS pathname or a fully qualified Guardian filename.

file_num1 Specifies the file number. This number is determined by the order of procedures in the object files's procedure table.

***** Specifies that you want information for all procedures. This is the default action.

If the **-LISTSOURCE** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

DETAIL | D Displays additional, detailed information about the procedures.

{**-LISTUNREFERENCED | -LUR** } { { **PROC** | **P** } | { **DATA** | **D** } | ***** } [{ **DETAIL** | **D** }]
Lists the undefined and unreferenced symbols in an object file; this flag is only valid for pTAL object files. These symbols must be linked before the object file can be executed.

PROC | P Displays unresolved procedures.

DATA | **D** Displays unresolved data items.

***** Displays all unresolved items. When you do not specify an option, this is the default behavior.

If the **-LISTUNREFERENCED** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

DETAIL | **D** Displays additional, detailed information such as the type of each symbol. For C++ functions, **DETAIL** provides the "demangled" (original) names as well as the "mangled" internal equivalents.

{**-LISTUNRESOLVED** | **-LU** } [{ **PROC** | **P** } | { **DATA** | **D** } | *****] [{ **DETAIL** | **D** }]
Lists the undefined names in an object file. These references must be resolved before the file can be executed.

PROC | **P** Displays unresolved procedures.

DATA | **D** Displays unresolved data items.

***** Displays unresolved procedures and data items. When you do not specify an option, this is the default behavior.

If the **-LISTUNRESOLVED** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

DETAIL | **D** Displays detailed name information.

-LOG [**OFF** | *pathname4* [**ASCII**]]
Writes a copy of the current session's input and output to a file.

OFF Closes the current log file and stops all logging. This is the default action.

pathname4 Identifies the file to receive the copy of the command lines and output. If the file does not exist, **enoft** creates it. If the file already exists, **enoft** appends output to the end of the file.

ASCII Creates the log file as a file code 180 file in C text (ASCII) format instead of creating it as a code 101 file in its default EDIT file format. This option is valid only for use in the Guardian environment; the default log file format in the OSS environment is ASCII text.

-NOEXIT Runs **enoft** in interactive mode. After executing any commands preceding it on the command line, this flag causes **enoft** to issue a prompt and wait for additional input. Commands following **-NOEXIT** on the command line are ignored. Any previous commands that cannot apply to interactive mode are ignored; for example, **-SET LINES** is ignored.

The **NOEXIT** command is ignored while in interactive mode.

-OBEY *pathname5*
Directs **enoft** to read command lines from the file specified in *pathname5*. In the Guardian file system, the specified file must be an EDIT file (file code 101). The specified file can call other OBEY files; OBEY files can be nested to any depth. The specified file cannot call itself and files it calls cannot call it or themselves; OBEY files cannot be recursive.

-OUT [**OFF** | *pathname6* [**ASCII**]
Directs the input and output listings to a specified file.

OFF Turns off redirection to a file and reverts to the original output file. This is the default action.

pathname6 Specifies the partially or fully qualified name of the file. If the file does not exist, **enoft** creates it. If the file already exists, **enoft** appends output to the end of the file.

ASCII Creates the log file as a file code 180 file in C text (ASCII) format instead of creating it as a code 101 file in its default EDIT file format. This option is valid only for use in the Guardian environment; the default log file format in the OSS environment is ASCII text.

-PROCINFO | **PI**
Displays the contents of the **.procinfo** section of a linkfile. This flag is an alias for specification of the **-DUMPSECTION** flag with the *section_name* value of **.procinfo**.

-PROGHDRS Displays the contents of the program headers for a loadfile or import library.

-RELOC Displays the relocation tables in an object file. These tables are in sections with names that begin with **.rela**.

-RESET [*set_cmd* | *****]
Resets the target object file attributes previously specified with the **-SET** flag to their default values.

set_cmd Is one of the following target object file attributes:

CASE
DEMANGLE
FORMAT
HISTORYBUFFER
HISTORYWINDOW
LINES
SCOPEPROC
SCOPESOURCE
SORT

***** Specifies that all target object file attributes are reset to their default values. If you use the **-RESET** flag without specifying any value, this is the default action.

If you use the **-RESET** flag on an OSS shell command line, the flag specification must be enclosed in quotation marks when you use the ***** specifier.

Refer to the description of the **-SET** flag for descriptions of these object file attributes.

-RTDU [{ **SOURCE** | **OBJECT** | ***** } [**DETAIL** | **D**]]
Displays the header information for the run-time data unit (RTDU) sections of the object file. Linkfiles have only a **.source.rtdu** section, while loadfiles have both **.source.rtdu** and **.object.rtdu** sections.

If **SOURCE** is specified, only the **.source.rtdu** section is displayed. If **OBJECT**

is specified, only the **.object.rtdu** section is displayed. If ***** is specified or all options are omitted, all available sections are displayed.

If **DETAIL** or **D** is specified, the memory content of the RTDU section for each record is also displayed.

-SECTHDRS Displays the contents of the section headers for a loadfile.

-SET [*set_cmd* [*value*]]
Sets an **enoft** target object file attribute to the specified value. If you do not specify *value*, **enoft** displays the current value used for the *set_cmd* attribute. If you do not specify *set_cmd*, **enoft** displays the current settings for all attributes.

The **-SET** flag can be abbreviated by combining it with letters that abbreviate *set_cmd* attribute names; these abbreviated flags are shown in the following list of valid *set_cmd* attribute names:

{**CASE** | **-SC** } { **ON** | **OFF** }
Specifies the case sensitivity of the **enoft** environment.

ON Turns on case sensitivity in the **enoft** environment.

OFF Turns off case sensitivity in the **enoft** environment. This is the default setting.

If case sensitivity is turned off, some file and procedure names might not be correctly matched when commands or flags are entered.

If the target object file contains C or C++ code, **enoft** automatically turns on case sensitivity when the file is opened. For object files containing code from a mix of C or C++ and other languages, if the procedure or source scope for a flag or command is restricted to a source file that does not contain C or C++ code, case sensitivity is turned off but reverts to on when the scope restriction is removed.

{**FORMAT** | **-SF** } { { **ASCII** | **A** } | { **DECIMAL** | **D** }
| { **HEX** | **H**} | { **ICODE** | **IC** } | { **INNERLIST** | **IN** } |
{ **READABLE** | **R** } }
Specifies the format used to display the object file.

ASCII | **A** Displays portions of the object file in ASCII format.

DECIMAL | **D**
Displays portions of the object file in decimal format.

HEX | **H** Displays portions of the object file in hexadecimal format.

ICODE | **IC** Displays portions of the object file in disassembled program code.

INNERLIST | **IN**
Displays portions of the object file in disassembled program code and displays the source code interspersed with the assembler. This option can be used only for text dumps.

READABLE | **R**
Displays portions of the object file in an applicable format based on the item type and part of the object file being displayed. This is the default action.

Specifying **IN** *format_spec* in another flag overrides the setting of this attribute.

{**HISTORYBUFFER** | **-SHB** } *number*
Specifies the number of command lines in memory that are to be available to the **!**, **FC**, or **HISTORY** subcommands. If the **HISTORYBUFFER** attribute is not specified, the default value is 50 command lines.

Valid values for *number* are in the range from 0 (zero) to 65535. If *number* is greater than the current buffer size, **enoft** is unable to retrieve command lines that have already left the history buffer. If *number* is smaller than the current buffer size, the command lines lost from the buffer are not retrievable.

The **HISTORYBUFFER** setting is only meaningful when you use **enoft** interactively.

{**HISTORYWINDOW** | **-SHW** } *number*
Specifies the number of command lines displayed with the **HISTORY** subcommand.

Valid values for *number* are in the range from 0 (zero) to 50. If the **HISTORYWINDOW** attribute is not specified, the default value is 10 command lines.

The **HISTORYWINDOW** setting is only meaningful when you use **enoft** interactively.

{ **LINES** | **-SL** } *number*
Specifies the number of lines of output to display before pausing so that an area of output does not scroll out of the terminal or emulator display memory. A single line of output from **enoft** can result in multiple lines of output on a screen, so more lines than are specified by *number* might be displayed.

Valid values for *number* are in the range 0 (zero) through 65535. If the **LINES** attribute is not specified, the default value for *number* is zero. A zero value causes output to continue until all results are displayed.

This flag is ignored when entered on a command line. It is primarily for interactive use.

{ **SCOPEPROC** | **-SSP** } { *proc_spec* | ***** }

Narrows the scope to a single procedure or subprocedure. This is helpful when trying to find unique items within a procedure or subprocedure and when trying to limit output to a range within a single scope.

The setting of this attribute takes precedence over any other value of *proc_spec* used in the same command.

proc_spec

Specifies the procedure name or procedure number. Procedure names are case-sensitive in C and C++ but not in other languages. *proc_spec* is one of the following:

proc_name

Limits the scope to the specified procedure and subprocedures. If *proc_name* is not completely specified, **enoft** resolves the name and lists conforming procedure names with numbers; wildcard matching (for example, *******partial_name* or *partial_name*******) can be used to search for items containing a match to the given pattern.

The demangled form of the procedure name cannot be used because **enoft** does not support blank spaces in the name.

proc_name.*subproc_name*[[**.***subproc_name*]...]

Limits the scope to the specified subprocedure. If *proc_name* or *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified.

subproc_name[[**.***subproc_name*]...]

Limits the scope to the specified subprocedure. If *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified. Only the first subprocedure with the specified name is displayed.

proc_num
Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

***** Eliminates any scope limitations and allows you to view the entire object file. If you use the **-SET SCOPEPROC** flag on an OSS shell command line, the flag specification must be enclosed in quotation marks when you use the ***** specifier.

{ **SCOPESOURCE** | **-SSS** } { *sourcendx* | *pathname7* | *file_num2* | ***** }
Narrows the scope to a single compilation unit or source file, which is helpful when trying to find unique items within a source file, as well as limiting the output to a range within the designated scope.

sourcendx Narrows the scope to a single compilation unit within the object file. *sourcendx* must be a valid index value. Use the **-LISTSOURCE** flag to determine valid values.

pathname7 Narrows the scope to a single named source file for the object file. *pathname7* must be a fully qualified OSS pathname or a fully qualified Guardian filename. Wildcard matching (for example, *******partial_name* or *partial_name*******) can be used to search for items containing a match to the given pattern.

file_num2 Specifies the file number. This number specifies the ordering of file use in the object file. Use the **-LISTSOURCE** flag to list each file number.

***** Eliminates any scope limitations present and opens selections to the entire object file. This is the default action.

If you use the **-SET SCOPESOURCE** flag on an OSS shell command line, the flag specification must be enclosed in quotation marks when you use the ***** specifier.

The setting of this attribute takes precedence over any other source file pathname or file number used in the same command.

{ **SORT** | **-ST** } { { **ALPHA** | **A** } | { **LOC** | **L** } | { **NONE** | **N** } }
Specifies the sorting order of the output.

ALPHA | **A**
Sorts **enoft** output in alphabetic order.

LOC | **L** Sorts **enoft** output in virtual address order.

NONE | **N**
Sorts **enoft** output in numeric order determined in the relevant table. The default value is **NONE**.

-SHOW [***** | *set_cmd*]
Displays the current value of the specified **enoft** program environment parameter and the target object file parameter.

The ***** specifier indicates that all of the attributess should be displayed. This is the default action.

If you use the **-SHOW** flag on an OSS shell command line, the flag specification must be enclosed in quotation marks when you use the ***** specifier.

-STRTAB [***** | **DYNSTR** | **DYNSTR2** | **PROCNAMES** | **RTDU** | **SHSTRTAB** | **STRTAB** | **UNWIND**]
Displays the contents of string tables in the object file.

***** Displays all available string tables in the object file. If you omit an option for this flag, this is the default action.

DYNSTR Displays the **.dynstr** section pointed to from the **.dynsym** section.

DYNSTR2 Displays the **.dynstr2** section pointed to from the **.dynamic**, **.liblist**, and **.dynsym.gblzd** sections.

PROCNAMES
Displays the **.procnames** section pointed to from the **.procinfo** section.

RTDU Displays the **.rtdu** section.

SHSTRTAB Displays the **.shstrtab** section.

STRTAB Displays the **.strtab** section pointed to from the **.symtab** section.

UNWIND Displays the **.unwind.strings** section pointed to from the **.unwind** section.

{**-SYMTAB** | **-SYMBOLS**} [***** | { **EXPORT** | **E** } | { **PROC** | **P** } | { **DATA** | **D** }]
Displays the specified contents of the **.symtab** table and of the **.dynsym** and **.dynsym.gblzd** tables for a loadfile or import library.

***** Specifies all exported, data, and code symbols in the tables. If you use the **-SYMTAB** or **-SYMBOLS** flag on an OSS shell command line, the flag specification must be enclosed in quotation marks when you use the ***** specifier.

EXPORT or **E** Specifies only exported symbols that are global and defined.

PROC or **P** Specifies only code symbols.

DATA or **D** Specifies only data symbols.

-TANDEMINFO | **-TDM**
Displays the contents of the **.tandem_info** section of a loadfile or import library. The information displayed includes the export digest for a DLL. This flag is an alias for specification of the **-DUMPSECTION** flag with the *section_name* value of **.tandem_info**.

-UNWIND Displays the contents of the **.IA_64.unwind** sections of stack unwinding information for a linkfile or loadfile.

{**-UNWINDINFO** | **UWI**} [{ ***** | *proc_spec* } [**DETAIL** | **D**]]
Displays the contents of the **.IA_64.unwind_info** sections of stack unwinding information in both linkfiles and loadfiles.

DETAIL or **D** provides additional, detailed information.

proc_spec
Specifies the procedure name or procedure number. Procedure names are case-sensitive in C and C++ but not in other languages. *proc_spec* is one of the following:

proc_name
Limits the scope to the specified procedure and subprocedures. If *proc_name* is not completely specified, **enoft** resolves the name and lists conforming procedure names with numbers. Wildcard matching (for example, ***partial_name** or *partial_name****) can be used to search for items containing a match to the given pattern.

The demangled form of the procedure name cannot be used because **enoft** does not support blank spaces in the name.

proc_name.*subproc_name*[[**.***subproc_name*]...]
Limits the scope to the specified subprocedure. If *proc_name* or *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified.

subproc_name[[**.***subproc_name*]...]
Limits the scope to the specified subprocedure. If *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified. Only the first subprocedure with the specified name is displayed.

proc_num
Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

{**-XREFPROC** | **-XP** } { *proc_spec* | ***** }
[**CALLEDBY** | **CALLS** | **BOTH**] [{ **DETAIL** | **D** }]
Displays an alphabetical cross-reference listing of procedures.

proc_spec Specifies the procedure name or procedure number. Procedure names are case-sensitive in C and C++ but not in other languages. *proc_spec* is one of the following:

proc_name
Limits the scope to the specified procedure and subprocedures. If *proc_name* is not completely specified, **enoft** resolves the name and lists conforming procedure names with numbers. Wildcard matching (for example, *****_partial_name_ or _partial_name_*****) can be used to search for items containing a match to the given pattern.

The demangled form of the procedure name cannot be used because **enoft** does not support blank spaces in the name.

proc_name.*subproc_name*[[**.***subproc_name*]...]
Limits the scope to the specified subprocedure. If *proc_name* or *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified.

subproc_name[[**.***subproc_name*]...]
Limits the scope to the specified subprocedure. If *subproc_name* is not completely specified, **enoft** resolves the name and lists conforming subprocedure names with numbers.

This option is not valid for C or C++ programs. For EpTAL programs, only one level of subprocedure can be specified. For COBOL programs, more than one level can be specified. Only the first subprocedure with the specified name is displayed.

proc_num
Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

***** Specifies all procedures in the current scope. If you use the **-XREFPROC** flag on an OSS shell command line, the flag specification must be enclosed in quotation marks when you use the ***** specifier.

CALLEDBY Lists each procedure and the procedures that call it. A scope setting restricts the procedures that are the children of the given procedure. If you omit an option for this flag, this is the default action.

CALLS Lists each procedure and the procedures that it calls. A scope setting restricts the procedures that are the parents of the given procedure.

BOTH Lists the information for both **CALLEDBY** and **CALLS**.

DETAIL | D Lists the called or calling procedures referenced by the indicated procedures and the addresses where the calls are made. For C++ procedures, this option also provides the original external ("demangled") names as well as the internal ("mangled") equivalents used.

The virtual addresses of the call sites are shown with the **DETAIL** option specified. A **SORT** setting affects both lists and sublists of procedures.

Operands

ref_objfile Specifies the target object file. This operand is required unless you use the **-FILE** flag.

DESCRIPTION

The **enoft** utility reads and displays information from TNS/E native object files. **enoft** enables you to:

- Determine the optimization level of procedures in a file.
- Display object code with corresponding source code.
- Display specific sections of a loadfile, linkfile, or library. Using this capability requires knowledge of the structure and content of TNS/E object files, as described in the *eld and rld Manual*.
- List dynamic-link library (DLL) references in an object file.
- List object file attributes.

You can display the following object file components with **enoft**:

- Various file headers
- Program text and data segments
- Symbol table and relocation tables

These **enoft** capabilities are useful when developing and debugging programs.

The **enoft** utility can be used from the command line or interactively to examine object files. To use **enoft** interactively, enter the **enoft** command without specifying any flags; you can then specify the flags interactively as subcommands in the manner described in the **SUBCOMMANDS** subsection of this reference page. Alternatively, **enoft** launches and runs as an interactive process if the **-NOEXIT** flag is specified on the command line.

To use **enoft** from a command file, capture the flags listed in the **FLAGS** subsection of this reference page or the subcommands listed in the **SUBCOMMANDS** subsection of this reference page. Capture one flag or subcommand per line in the command file and then specify the command file as the standard input file to the **enoft** command using a redirection operator. Output can also be redirected to a file. The effective syntax for the **enoft** command in this instance is:

enoft < *command_file* [> *output_file*]

For complete information on using **enoft**, refer to the *enoft Manual*.

SUBCOMMANDS

enoft supports all flags listed in the **SYNOPSIS** section as subcommands for interactive use. Such subcommands consist of the flag without the prefixed dash (-).

The following subcommands are primarily for interactive use. These subcommands can be entered as OSS shell command line flags when prefixed by a dash (-) but are only meaningful when used interactively:

break key on keyboard
: Interrupts the processing of the current subcommand as soon as possible without corrupting any **enoft** internal tables. The **enoft** utility resumes operation with the next subcommand line.

CD [*dir_pathname1*]
: Changes current working directory to the specified directory.

{ **DIR** | **FILES** } [*dir_pathname2*]
: Lists the files in the specified directory.

EXIT | **E** | **QUIT** | **Q**
: Stops the **enoft** process.

FC [*hist_num* | *-hist_offset* | *text*]
: Allows you to edit or repeat a previously executed subcommand line.

hist_num	Specifies the number of a previously entered subcommand line. The default value is the previously entered subcommand line.
-hist_offset	Specifies a negative offset from the current subcommand line. The flag entered before the **FC** subcommand is -1.
text	Is a string of characters.

{ **HISTORY** | **H** } [*hist_num*]
: Displays previously entered subcommand lines. *hist_num* specifies the number of subcommand lines to be displayed. The default value is 10 subcommand lines.

{ **LINES** | **-SL** } *number*
: Specifies the number of lines of output to display before pausing so that an area of output does not scroll out of the terminal or emulator display memory. A single line of output from **enoft** can result in multiple lines of output on a screen, so more lines than are specified by *number* might be displayed.

  Valid values for *number* are in the range 0 (zero) through 65535. If the **LINES** attribute is not specified, the default value for *number* is zero. A zero value causes output to continue until all results are displayed.

NOTES

As an alternative syntax, **enoft** allows subcommands to be entered without the dash (-) that normally begins a command flag. When the dash is omitted, each subcommand after the first must be separated from the next subcommand in the command line by a semi-colon (**;**). Because semi-colons have significance to the OSS shell, they must be preceded by a backslash character or that portion of the command line must be enclosed in quotation marks; for example:

enoft file /usr/subdir/hello.out\; set format innerlist\; dumpcode

or

enoft "file /usr/subdir/hello.out; set format innerlist; dumpcode"

EXAMPLES

1. To find the names of procedures in a source file named **sample.c**:

   enoft -FILE sample.o -SET SCOPESOURCE sample.c
   "-LISTPROC *"
   or
   `enoft -F sample.o -SSS sample.c "-LP *"`

2. To find all the procedures that are called by source file **sample.c**:

   enoft -FILE sample.o -SET SCOPESOURCE sample.c
   "-XREFPROC * CALLEDBY"
   or
   `enoft -F sample.o -SSS sample.c "-XP * CALLEDBY"`

3. To look at the optimization levels for source file **sample.c**:

   enoft -FILE sample.o -SET SCOPESOURCE sample.c
   "-LISTOPTIMIZE *"
   or
   `enoft -F sample.o -SSS sample.c "-LO *"`

4. To look at the optimization level for a single procedure:

   enoft -FILE sample.o -SET SCOPEPROC *procedure-name*
   `"-LISTOPTIMIZE *"`
   or
   `enoft -F sample.o -SSP` *procedure-name* `"-LO *"`
   or
   `enoft -FILE sample.o -LISTPROC` *procedure-name* `DETAIL`
   or
   `enoft -F sample.o -LP` *procedure-name* `D`

5. To look at source file numbers for **sample.o**:

   enoft -FILE sample.o "-LISTSOURCE *"

6. To look at procedure numbers:

   enoft -F sample.o "-LP *"

7. To see the instructions for a procedure:

   enoft -FILE sample.o
   -DUMPPROC *procedure-name* `IN ICODE`
   or
   `enoft -F sample.o -DP` *procedure-name* `IN IC`

8. To look at a particular 20 bytes referenced by one of those instructions in hexadecimal:

   enoft -FILE sample.o
   -DUMPADDRESS 0x00000390 FOR 20 BYTES IN HEX
   or
   `enoft -F sample.o -0x00000390 FOR 20 B IN H`

9. To look at the first 30 bytes in an object file in ASCII:

   enoft -FILE sample.o
   -DUMPOFFSET 0x0 FOR 30 BYTES IN ASCII
   or
   `enoft -F sample.o -DO 0x0 FOR 30 B IN A`

10. To see all the data items external to the object file that need to be linked in and where they are used in alphabetic order:

    enoft -FILE sample.o -SET SORT ALPHA
    -LISTUNRESOLVED DATA DETAIL
    or
    `enoft -F sample.o -ST A -LU DATA D`

11. To find the data model of the object named **sample.o**:

    enoft -FILE sample.o
    -FILEHDR

    The data model is supported for systems running H06.24 or later H-series RVUs or J06.13 or later J-series RVUs only.

DIAGNOSTICS

enoft sends all diagnostic messages to standard output. Each diagnostic message has a unique message number. The following ranges of errors are reported:

Fatal errors	Fatal errors occur when memory cannot be allocated. Such messages are in the range from 1 to 999 and are prefixed by `FATAL ERROR ***`. Fatal errors always cause **enoft** to terminate with an exit value of 1.
Data errors	Data errors occur when an object file is incomplete or damaged or the specified command cannot be applied to the object type. Such messages are in the range from 1000 to 1999 and are prefixed by `DATA ERROR ***`.
Syntax Errors	Syntax errors occur when **enoft** cannot recognize a specified command or process the syntax used correctly. Such messages are in the range from 3000 to 3999 and are prefixed by `SYNTAX ERROR ***`.
Warnings	Warnings occur when **enoft** continues processing based upon its own assumptions about user intent. Such messages are in the range from 2000 to 2999 and are prefixed by `WARNING ***`.

EXIT VALUES

The **enoft** utility returns:

0	To indicate normal completion, usually in response to an **EXIT**, **E**, **QUIT**, or **Q** command.
1	To indicate fatal termination.

RELATED INFORMATION

Commands: **eld(1)**.

STANDARDS CONFORMANCE

The **enoft** command is an HP extension to the XPG4 Version 2 specification.

NAME

env - Displays or sets environment variables

SYNOPSIS

env [**-i**] [**-**] [*name=value ...*] [*command*] [*argument ...*]

FLAGS

-i Invokes *command* with the environment specified by the arguments; the **env** command ignores the inherited environment.

- Invokes *command* with the environment specified by the arguments; the **env** command ignores the inherited environment. (Obsolescent)

DESCRIPTION

The **env** command lets you get and change your current environment and then run the specified command with the changed environment. Changes in the form *name=value* are added to the current environment before the command is run. If the **-i** flag is used, the current environment is ignored, and the command runs with only the changed environment. Changes are only in effect while the specified command is running.

If *command* is not specified, the **env** command displays your current environment one *name=value* pair per line.

EXAMPLES

1. To replace one environment with another one, enter:

   env - PATH=$PATH IDIR=/u/jim/include LIBDIR=/u/jim/lib make

   This command runs the **make** command in an environment that consists only of these definitions for the **PATH**, **IDIR**, and **LIBDIR** parameters. You must redefine **PATH** so that the shell can find the **make** command.

   When **make** is finished, the original environment takes effect again.

EXIT VALUES

The **env** command exits with the following values:

0 (zero) The **env** command completed successfully.

1-125 An error occurred in the **env** command.

126 The specified utility was found but could not be invoked.

127 The specified utility could not be found.

RELATED INFORMATION

Commands: **sh(1).**

Functions: **exec(2).**

NAME

eval - Executes arguments as commands

SYNOPSIS

eval [*argument* ...]

DESCRIPTION

The arguments to **eval** are read as input to the shell, and the resulting commands are executed. **eval** concatenates the arguments and separates each with a space character.

EXAMPLES

In the following example, values are assigned to variables, and these variables are used as arguments to the **eval** command. The results of the **eval** command are printed to the screen by using the **print** command.

```
x=5 y=x
z='$'$x
print $y
x
eval y='$'$x
print $y
5
```

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **eval** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **exec**(1), **sh**(1).

NAME

ex - Edits lines in a file interactively

SYNOPSIS

ex [**-c** *subcommand*] [**-Rsv**] [**-w***number*] [**+***subcommand*] [**-**] [*file ...*]

ex [**-c** *subcommand*] [**-Rsv**] [**-t** *tag*] [*file ...*]

ex [**-c** *subcommand*] **-r**[*file*] [**-Rsv**] [*file*]

The **ex** command is a line-oriented text editor that is a subset of the **vi** screen editor.

FLAGS

-c *subcommand*
Executes the specified **ex** subcommand (*command*) before displaying the file for which the editor was invoked.

-r[*file*] Recovers *file* after an editor or system crash. If you do not specify *file*, a list of all saved files is displayed.

-R Sets the **readonly** option, preventing you from altering the file.

-s Does not display the filename or the **:** prompt upon entering **ex**. (Silent mode.)

-t*tag* Loads the file that contains *tag* and positions the editor at *tag*. To use this flag, you must first create a database of function names and locations using the **ctags** command. (OSS does not support the **ctags** command. However, it does support **ctags** files.)

-v Invokes the **visual** editor. When the **-v** flag is specified, an enlarged set of subcommands are available, including screen editing and cursor movement features. See **vi**.

-w*number*
Sets the default window size to *number* lines. This flag is useful only if used with the **-v** flag.

- Suppresses all interactive user feedback. If you use this flag, file input/output errors do not generate an error message.

+*subcommand*
Begins the edit with the specified editor subcommand. When *subcommand* is not entered, a + (plus sign) sets the current line to the bottom of the file. Normally **ex** sets the current line to the last line of the file, or to some specified tag or pattern. (Obsolescent)

DESCRIPTION

The **ex** editor is similar to **ed**, but is more powerful, providing multiline displays and access to a screen editing mode. You may prefer to call **vi** directly to have environment variables set for screen editing. Also **edit**, a limited subset of **ex**, is available for novices or casual use.

The *file* argument specifies the file or files to be edited. If you supply more than one file, the **ex** editor edits each file in the specified order.

To determine how your tty can perform more efficiently, **ex** uses the tty capability database **terminfo** and the type of tty you are using from the **TERM** environment variable.

The **ex** editor has the following features:

- You can view text in files. The **z** subcommand lets you access windows of text, and you can scroll through text by pressing **<Ctrl-d>** and **<Ctrl-u>** (visual (**-v**) mode only).

- The **undo** subcommand allows you to reverse the last subcommand, even if it is an **undo** subcommand. Thus, you can switch back and forth between the latest change in the edit file and the last prior file status and view the effect of a subcommand without that effect being permanent. Commands that affect the external environment cannot be undone, however. The **ex** command displays changed lines and indicates when more than a few lines are affected by a subcommand. The **undo** subcommand causes all marks to be lost on lines changed and then restored if the marked lines were changed. It does not clear the **buffer modified** condition.
- You can retrieve your work (except changes that were in the buffer) if the system or the editor crashes by reentering the editor with the **-r** flag and the filename.
- You can edit a sequence or group of files. You can use the **next** subcommand to edit each file on the command line in turn, or to specify a list of filenames to edit (using the shell pattern matching syntax). The wildcard character **%** (percent sign) represents the name of the current edit file and can be used to form filenames.
- You can copy and move text within a file and between files (see the **co**, **d**, **ya**, and **pu** subcommands). You use a group of buffers (that have the names of the ASCII letter **a** to **z**) to move text. You can temporarily place text in these buffers and copy or reinsert it in a file, or you can carry it over to another file. The buffers are cleared when you quit the editor. The editor does not notify you if text is placed in a buffer and not used before exiting the editor.
- You can use patterns that match words. A pattern can be a fixed character string or a regular expression.

  A regular expression is a string constructed of special pattern-matching characters. Using a regular expression to locate text in a file gives you more flexibility than trying to locate a fixed character string. For more information about regular expressions, see **grep**.

Editing Modes

Command mode

When you start the **ex** editor, it is in command mode. Enter **ex** subcommands at the : (colon) prompt.

Text entry mode

Entered by **a**, **i**, and **c**. In this state, you can enter text. Entry state ends normally with a line that has only a **.** (period) on it or ends abruptly if you press the Interrupt key sequence.

Visual and open mode

To use visual mode, use the following syntax:

line **vi** [*type*] [*count*]

Enters visual mode at the specified line. The *type* argument is optional, and can be a **-** (dash) or **.** (dot), as in the **z** subcommand, to specify the position of the specified line on the screen window. (The default is to place the line at the top of the screen window.) The *count* argument specifies an initial window size; the default is the value of the **window** option. The **Q** subcommand exits visual mode. For more information, see **vi**.

The **o** command opens a one-line window. All three commands share the input state of the **vi** editor. Press **<Esc>** to exit text entry mode. To return to the **ex** command state at the current line, enter **Q** while in command mode.

The ex Limits

The **ex** editor has the following maximum limits:

- 2048 bytes per line
- 256 bytes per global command list
- 128 bytes in the previous inserted and deleted text
- 128 bytes in a shell escape command
- 128 bytes in a string-valued option
- 32 bytes in a tag name
- 128 map macros with 2048 bytes total

SUBCOMMANDS

The **ex** subcommands affect the current line unless you specify otherwise. For information about how to address lines in a file, see **edit** and **vi**. For a complete description of edit options, see **Setting Options** on the **vi** reference page.

The ex Subcommands

ab[**brev**] *word abbrev*
Adds the specified abbreviation to the current abbreviation list.

[*line*] **a**[**ppend**][**!**]
Enters input mode and places text after the specified line. To place the text at the beginning of the buffer, specify line 0. The **!** (exclamation point) toggles the **autoindent** editor option setting for the execution of this subcommand.

ar[**gs**] Writes the argument list (the list of arguments on start-up) with the current argument inside **[** and **]** (left and right brackets). The argument list can later be replaced by the arguments of the **next** subcommand.

[*range*] **c**[**hange**][**!**] [*count*]
Enters input mode and replaces the lines in *range* with the input text. The current line is the last line input. The **!** (exclamation point) toggles the **autoindent** editor option setting for the execution of this subcommand.

chd[**ir**][**!**] [*directory*]

cd[**!**] [*directory*]
Changes the current working directory to *directory*. If the current buffer has been modified since the last write, the subcommand issues a warning and fails. You can override this warning by appending a **!** (exclamation point) to the subcommand name.

[*range*] **co**[**py**] *line* [*flags*]

[*range*] **t** *line* [*flags*]
Places a copy of the lines in *range* after the specified line. Line 0 causes the lines to be placed at the beginning of the buffer.

[*range*] **d**[**elete**] [*buffer*] [*count*] [*flags*]
Deletes the specified lines from the buffer. If you specify a named buffer, the deleted text is placed there; otherwise, the deleted text is placed in the unnamed buffer. The current line is the line following the deleted lines, or the last line if the deleted lines were at the end.

e[**dit**][**!**] [+*line*] [*file*]

ex[**!**] [+*line*] [*file*]
Edits *file*. If the current buffer has been modified since the last write, the subcommand writes a warning and terminates. You can override this action by appending the **!** (exclamation point) character to the subcommand (for example, **e!** *file*).

If the +*line* argument is specified, the current line is the specified position, where *line* can be a number (or **$**) or can be specified as /*pattern* or **?***pattern*. Preceding the pattern with a **/** (slash) starts a search from the beginning of the file. Preceding the pattern with a **?** (question mark) starts a search from the end of the file. This subcommand is affected by the **autowrite** and **writeany** editor options.

f[**ile**] [*file*]
Writes the current pathname, the number of lines, and the current position (if no *file* argument was specified). If *file* is specified, **ex** changes the current filename to *file* without changing the contents of the buffer or the previous current file.

[*range*] **g**[**lobal**] /*pattern*/ [*subcommands*]

[*range*] **v** /*pattern*/ [*subcommands*]
Marks the lines within the given range that match (**g**) or do not match (**v**) the given pattern. Then executes the **ex** subcommands with the current line set to each marked line.

You can specify multiple subcommands, one per line, by escaping each newline character with a \ (backslash). If the *subcommands* argument is not specified, each line is written. For the **append**, **change**, and **insert** subcommands, the input text is included as part of the **global** subcommand; in this case, you can omit the terminating period if it ends *subcommands*. The **visual** subcommand can be specified as part of *subcommands*. In this mode, input is taken from the terminal. Entering a **Q** from visual mode selects the next line matching the pattern and re-enters visual mode, until the list is exhausted.

You cannot use the **global** subcommand itelf and the **undo** subcommand in the *subcommands* argument. The **autoprint**, **autoindent**, and **report** editor options are inhibited for the duration of the **g** or **v** subcommand.

[*line*] **i**[**nsert**][**!**]
Enters input mode and places the input text before the specified line. The **!** (exclamation point) toggles the **autoindent** editor option setting for the execution of this subcommand.

[*range*] **j**[**oin**][**!**] [*count*] [*flags*]
Joins the text from the specified lines together into one line. In the POSIX locale, when the last character on the first line of a pair of lines to be joined is a **.** (period), two spaces are added following the period; when the last character of the first line is a space or when the first character on the second line of the pair is a **)** (right parenthesis), no spaces are added; otherwise, one space is added following the last character of the first line. Extra spaces at the start of a line are discarded.

Appending a **!** (exclamation point) character to the **join** subcommand causes a simpler join with no whitespace processing, independent of the current locale.

[*range*] **l**[**ist**] [*count*] [*flags*]
Writes the addressed lines; nonprintable characters are written as multicharacter sequences. The end of the line is marked with a **$** (dollar sign).

Long lines are folded. The current line is the last line written.

map[**!**] [*x rhs*]
Defines macros for use in visual mode. The first argument must be a single character or the sequence *#digit* (one of the terminal's numbered function keys). When this character or function key is entered in visual mode, the action is as if the corresponding *rhs* had been entered. If the **!** (exclamation point) character is appended to the subcommand name **map**, the mapping is effective during input mode rather than command mode. This allows *x* to have two different macro definitions at the same time: one for command mode and one for input mode. Nonprintable characters, except for the Tab character, require escaping with **<Ctrl-V>** (or **<Ctrl-Q>**) to be entered in the arguments. On certain block mode terminals, the mapping need not occur immediately (for example, it might occur after the terminal transmits a group of characters to the system), but it modifies the file as if it occurred immediately.

The **map** subcommand with no arguments writes all of the macros currently defined. If **!** (exclamation point) is appended to the subcommand, only the macros effective during input mode are written; otherwise, only the macros effective during command mode are written.

[*line*] **ma**[**rk**] *x*

[*line*] **k** *x* Gives the specified line the specified mark *x*, which must be a single lowercase letter of the POSIX locale. The current line position is not affected. The expression **'***x* can then be used as an address in any subcommand requiring one. For example, the following subcommand deletes all of the lines from the current one to the marked line:

.,'*x***d**

In addition, see the **vi ‘‘** and **’’** subcommands for uses of the mark in visual mode. If the **'***x* subcommand is used in nonvisual mode, the character marked is the first nonspace character of the current line; otherwise, the character marked is the character at the current column of the current line.

[*range*] **m**[**ove**] *line*
Moves the specified lines (*range*) after the target line (*line*). The current line is the first of the moved lines.

n[**ext**][**!**] [*file ...*]
Edits the next file from the argument list. If the current buffer has been modified since the last write, the subcommand writes a warning and terminates. You can override this action by appending the **!** (exclamation point) character to the subcommand name (**n!**). You can replace the argument list by specifying a new one as arguments to this subcommand. Editing then starts with the first file on this new list. The current line is reset as described for the **edit** subcommand. This subcommand is affected by the **autowrite** and **writeany** editor options.

[*range*] **nu**[**mber**] [*count*] [*flags*]

[*range*] **#** [*count*] [*flags*]
Writes the selected lines, each preceded with its line number in decimal. Nonprintable characters, except for **<Tab>**, are expanded as specified by the **print** subcommand.

The only meaningful flag is **l**, which allows additional expanded writing of tabs and End-of-Line characters by the **list** subcommand. The current line is the last line written.

[*line*] **o**[**pen**] */pattern/* [*flags*]
Enters open mode, which is equivalent to visual mode with a one-line window. All visual mode subcommands are available. If a match is found for the optional regular expression in *line*, the cursor is placed at the start of the matching pattern. The visual mode subcommand **Q** (see **vi**) exits open mode.

pre[**serve**]
Saves the current buffer in a form that can later be recovered by using **ex -r** or by using the **recover** subcommand. After the file has been preserved, a mail message is sent to the user. This message can be read by invoking **mailx**. The message contains the name of the file, the time of preservation, and an **ex** subcommand for recovering the file. Additional information can be included in the mail message.

[*range*] **p**[**rint**] [*count*] [*flags*]
Writes the addressed lines. Nonprintable characters, except for the Tab character, are written as multicharacter sequences.

Long lines are folded. The only meaningful flags are **#** and **l**. The current line is the last line written.

[*line*] **pu**[**t**] [*buffer*]
Puts back deleted or yanked lines after the specified line. A buffer can be specified; otherwise, the text in the unnamed buffer (where deleted or yanked text is placed by default) is restored. The current line is the first line put back.

q[**uit**][**!**] Terminates the editing session. If the current buffer has been modified since the last write, the subcommand writes a warning and terminates. You can override this warning and force an exit, discarding changes, by appending the character **!** to the subcommand name.

[*line*] **r**[**ead**][**!**] [*file*]
Places a copy of the specified file in the current buffer after the target line (line 0 places text at the beginning). If no file is named, the current file is the default. If there is no current file, the specified file becomes the current file. If there is neither current file nor *file* argument, the subcommand fails.

The current line is the last line read. In visual mode, the current line is the first line read. If *file* is preceded by **!**, *file* is taken to be an operating system command and passed to the program named in the **SHELL** environment variable. The resulting output is read in to the buffer. You can override the special meaning of **!** by escaping it with a \ (backslash) character.

rec[**over**] *file*
Attempts to recover *file* if it was saved as the result of a **preserve** subcommand, the receipt of a signal, or a system or editor crash. The current line is reset as described for the **read** subcommand.

rew[**ind**][**!**]
Rewinds the argument list; that is, sets the current file to the first file in the argument list. This is equivalent to a **next** subcommand with the current argument list as its argument. If the current buffer has been modified since the last write, the subcommand writes a warning and terminates. You can override the action by appending the **!** (exclamation point) character to the subcommand name (**rew!**). The current line is reset as described for the **read** editor subcommand. This subcommand is affected by the **autowrite** and **writeany** editor options.

se[**t**] [*option*[=[*value*]] ... [**no***option* ...] [*option***?** ...] [**all**]
When no arguments are specified, writes those options whose values have been changed from the default settings; when the argument **all** is specified, writes all of the option values.

Specifying an option name followed by the **?** character causes the current value of that option to be written. The **?** can be separated from the option name by zero or more spaces. The **?** is necessary only for Boolean valued options. Boolean options can be given values by the form **se** *option* to turn them on or **se no***option* to turn them off; string and numeric options can be assigned by the form **se** *option*=*value*. Spaces in strings can be included as they are by preceding each such character with a \ (backslash). More than one option can be set or listed by a single **set** subcommand by specifying multiple arguments, each separated from the next by one or more spaces.

sh[**ell**] Invokes the program named in the **SHELL** environment variable with the argument **-i** (interactive mode). You can resume editing when the program exits.

so[**urce**] *file*
Reads and executes subcommands from the file specified by the mandatory *file* argument. Such **source** subcommands can be nested.

[*range*] **s**[**ubstitute**] [/*pattern*/*repl*/[*options*] [*count*] [*flags*]
Replaces the first instance of *pattern* by the string *repl* on each specified line. If the /*pattern*/*repl*/ argument is not present, the /*pattern*/*repl*/ from the previous **substitute** subcommand is used.

If *options* includes the letter **g** (global), all nonoverlapping instances of the pattern in the line are substituted. If the option letter **c** (confirm) is included, then before each substitution the line is written with ^ characters written on the following line, adjacent to and identifying the pattern to be replaced; an affirmative response causes the substitution to be done, while any other input causes it to abort. An affirmative response consists of a line with the affirmative response (as defined by the current locale) at the beginning of the line. Such a line is subject to editing in the same way as the command line (the **/** or **:** line at the bottom of the screen).

The current line is the last line substituted. When the **c** option is used, typing the Interrupt character or receiving the **SIGINT** signal stops the substitute operation, and **ex** returns to command mode. All substitutions completed before the interrupt occurred are retained and none are made after that point. The current line is the last line substituted.

This subcommand is affected by the **LC_MESSAGES** environment variable and the **wrapscan** option.

su[**spend**][**!**]

st[**op**][**!**] Allows control to return to the invoking process; **ex** suspends itself as if it had received the **SIGTSTP** signal. The suspension occurs only if job control is enabled in the invoking shell.

Following either **suspend** or **stop** with the character **!** affects the operation of the **autowrite** editor option for this subcommand only.

The current suspend character (see **stty**) also causes the suspension.

ta[**g**][**!**] *tagstring*
Searches for the tag string, which can be in a different file. If the tag is in a different file, the new file is opened for editing. If the current buffer has been modified since the last write, the subcommand writes a warning and terminates. You can override the action by appending the **!** character to the subcommand name. The current line is reset to the line

indicated by the tag. This subcommand is affected by the **autowrite**, **tags**, and **writeany** editor options.

The **tag** subcommand searches for *tagstring* in the tag file referred to by the **tags** editor option until a reference to *tagstring* is found. The file pointed to by this reference is loaded into the buffer, and the current line is set to the first occurrence of the pattern specified in the tags file associated with the supplied *tagstring*. If the tags file contained a line number reference, the current line is set to that line. If the pattern or line number is not found, the subcommand writes an error message. If a file referred to by the **tags** editor option does not exist or is not readable, the subcommand also writes an error message.

una[**bbrev**] *word*
Deletes *word* from the list of abbreviations, as described by the **abbrev** subcommand.

u[**ndo**] Reverses the changes made by the previous editing subcommand (one that changes the contents of the buffer). For this purpose, **global** and **visual** are considered single subcommands. An **undo** can itself be reversed. Commands that affect the external environment, such as **write**, **edit**, and **next** cannot be undone.

unm[**ap**][**!**] *x*
If no **!** (exclamation point) is specified, removes the command-mode macro definition for *x*; otherwise, removes the input-mode macro definition for *x*. See the **map** subcommand.

[*line*] **vi**[**sual**] [*type*] [*count*] [*flags*]
Enters visual mode with the current line set to *line*. The *type* argument is optional, and can be a **-** (minus sign), **.** (period), **+** (plus sign), or ^ (circumflex), as in the **z** subcommand, to specify the position of the specified line on the screen window. (The default is to place the line at the top of the screen window.) The *count* argument specifies the number of lines that is initially written; the default is the value of the **window** editor option. The **Q** subcommand exits visual mode. (For more information about the **Q** subcommand, see the **vi** reference page.)

[*range*] **w**[**rite**][**!**] [>>] [*file*]

[*range*] **w**[**rite**] [**!**] [*file*]

[*range*] **wq**[**!**] [>>] [*file*]
Writes the specified lines (the whole buffer, if *range* is not specified) out to the file represented by pathname *file*, writing to standard output the number of lines and bytes written.

If *file* is specified and is not the current file, and the file named by *file* exists, then the write fails. If the current file has been changed by the **file** subcommand and that file exists, the write fails. In either case, you can force the write by appending the **!** (exclamation point) character to the subcommand name. You can append to an existing file by appending >> to the subcommand name.

If the *file* argument is preceded by a **!** (exclamation point) character, the program named in the **SHELL** environment variable is invoked with *file* as its second argument, and the specified lines are passed as standard input to the subcommand. The **!** in this usage must be separated from the **write** subcommand by at least one space character. You can override the special meaning of **!** by escaping it with a \ (backslash) character. This subcommand is affected by the **writeany** and **readonly** editor options.

The subcommand **wq** is equivalent to a **w** followed by a **q**; **wq!** is equivalent to **w!** followed by **q**. If the current buffer has no pathname associated with it, the **write** subcommand fails.

[*range*] **x**[**it**][**!**] [*file*]

Performs a **write** subcommand if any changes have been made to the current buffer since the last write to any file.

Unless the subcommand fails because an attempt to write lines to a file did not succeed, the **ex** program exits after an **x** subcommand. This subcommand is affected by the **writeany** and **readonly** editor options.

[*range*] **ya**[**nk**] [*buffer*] [*count*]

Places the specified lines in the named buffer. If no buffer is specified, the unnamed buffer is used (where the most recently deleted or yanked text is placed by default).

[*line*] **z** [*type*] [*count*] [*flags*]

If *type* is omitted, *count* lines following the specified line are written. The default for *count* is the value of the **window** editor option. The *type* argument changes the position at which *line* is written on the screen by affecting the number of lines written before and after *line*.

If *type* is specified, it is one of the following:

- (dash) Places *line* at the bottom of the screen.

+ (plus sign)
Places *line* at the top of the screen.

. (period) Places *line* in the middle.

^(circumflex)
Writes out *count* lines starting *count**2 lines before the addressed line; the net effect of this is that a **z^** subcommand following another **z** subcommand writes the previous page.

= (equal sign)
Centers the addressed line on the screen with a line of **-** (dash) characters written immediately before and after it. The number of preceding and following lines of text written are reduced to account for these lines of hyphens.

In all cases, the current line is the last line written, with the exception of the **=** type, which causes the current line to be that addressed in the subcommand.

! *subcommand*

[*range*]**!** *subcommand*

Passes the remainder of the line after the **!** (exclamation point) character to the program named in the **SHELL** environment variable for execution. A warning is issued if the buffer has been changed since the last write. A single **!** character is written when the subcommand completes. The current line position is not affected.

Within the text of *subcommand*, **%** (percent sign) and **#** (number sign) are expanded as pathnames (the current and alternative pathnames, respectively), and **!** is replaced with the text of the previous **!** subcommand. (Thus, **!!** repeats the previous **!** subcommand.) If any such expansion is performed, the expanded line is echoed.

You can override the special meanings of **%**, **#**, and **!** by escaping them with a \ (backslash) character. This subcommand is affected by the **autowrite** and **writeany** editor options.

In the second form of the **!** subcommand, the remainder of the line after the **!** is passed to the program named in the **SHELL** environment variable, as described previously. The specified lines are provided to the program as standard input; the resulting output

replaces the specified lines.

[*range*] **<** [*count*] [*flags*]
Shifts the specified lines to the left; the number of character positions to be shifted is determined by the **shiftwidth** editor option. Only leading spaces are lost in shifting; other characters are not affected. The current line is the last line changed.

[*range*] **>** [*count*] [*flags*]
Shifts the specified lines to the right, by inserting spaces, using tabs where possible, as determined by the **shiftwidth** editor option. Empty lines are not changed. The current line is the last line changed.

[*range*] **&** [*options*] [*count*] [*flags*]

[*range*] **s**[**ubstitute**] [*options*] [*count*] [*flags*]

[*range*] [*options*] [*count*] [*flags*]
Repeats the previous substitute subcommand, as if **&** were replaced by the previous **s**/*pattern*/*repl*/ subcommand. (The same effect can be obtained by omitting the /*pattern*/*repl*/ string in the **substitute** subcommand.) The version of the subcommand using (tilde) is the same as **&** and **s**, but the pattern used is the last regular expression used in any subcommand, not necessarily the one used in the last substitute subcommand. For example, in the following sequence, the (tilde) is equivalent to **s/green/blue/**:

s/red/blue/
/green

[*line*] **=** [*flags*]
Writes the line number of the specified line (the default is the last line). The current line position is not affected.

<Ctrl-d>
Writes the next *n* lines, where *n* is the value of the editor option **scroll**. The subcommand is invoked with the End-of-File character. The current line is the last line written.

@@ *buffer*

******* *buffer* Executes each line of the named buffer as an **ex** subcommand. If no buffer is specified, or is specified as **@@** or *****, the last buffer executed is used. If there is no last buffer, an error occurs.

Displays addressed lines with line numbers

" Starts comment

<Return>
Displays next line

The ex Subcommand Addresses

$ The last line

+ The next line

- The previous line

+*n* The *n*th line forward

-*n*	The *n*th previous line
%	The first through last lines
number	Line *number*
.	The current line
*x***-***number*	The *number*th line before line *x*
x,*y*	Lines *x* through *y*
´*m*	The line marked with *m*
´´	The previous context
/*pattern***$**	The next line with *pattern* at end of line
/^*pattern*	The next line with *pattern* at start of line
/*pattern*	The next line with *pattern*
?*pattern*	The previous line with *pattern*

Scanning Pattern Formation

^	The beginning of the line
$	The end of the line
.	Any character
\<	The beginning of the word
\>	The end of the word
[*string*]	Any character in *string*
[^*string*]	Any character not in *string*
[*x*-*y*]	Any character between *x* and *y*, inclusive
*****	Any number of the preceding character
	The replacement part of the last substitute subcommand.
/(*pattern***\)**	A regular expression pattern can be enclosed in escaped parentheses to identify them for substitution actions.

FILES

/bin/exrecover	The **recover** subcommand.
/bin/expreserve	The **preserve** subcommand.
.exrc or **$HOME/.exrc**	Editor start-up file.
/tmp/Ex*nnnnn*	Editor temporary file.
/tmp/Rx*nnnnn*	Names buffer temporary file.
/var/preserve	Preservation directory.

RELATED INFORMATION

Commands: **ed(1)**, **grep(1)**, **vi(1)**.

Files: **terminfo(4)**.

The **TERM** environment variable.

NAME

exec - Executes arguments as commands

SYNOPSIS

exec [*argument* ...]

DESCRIPTION

If *argument* is specified and is a valid command name, it is executed in place of the shell without creating a new process. Input/output arguments can affect the current process. If no arguments are given, the effect of **exec** is to modify file descriptors as prescribed by the input/output redirection list. Any file descriptor numbers greater than 2 that are opened with this mechanism are closed when invoking another program.

EXAMPLES

1. The following command executes the program **my_program**:

   exec my_program

EXIT VALUES

If a command is specified, **exec** does not return to the shell; instead, the exit status of the process is the exit status of the program implementing the command that overlaid the shell.

If the command specified by the arguments is not found, the exit status is 127.

If the command is found, but it is not an executable utility, the exit status is 126.

If a redirection error occurs, the shell exits with a value in the range 1-125; otherwise, **exec** returns an exit status of 0 (zero).

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **exec** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **eval**(1), **sh**(1).

NAME

exit - Causes the shell to exit

SYNOPSIS

exit [*n*]

DESCRIPTION

The **exit** command causes the shell to exit with the exit status specified by *n*. If *n* is omitted, the exit status is that of the last command executed. An End-of-File also causes the shell to exit, unless the shell has the **ignoreeof** option (see **set**) turned on.

EXAMPLES

1. The following command exits the OSS shell with a value of true.

   exit 0

EXIT VALUES

If *n* is not specified, the exit status will be the exit value of the last command executed, or 0 (zero) if no command was executed.

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **exit** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **set**(1), **sh**(1).

NAME

expand - Replace tab or space characters

SYNOPSIS

Current syntax

expand [**-t** *tablist*] [*file* ...]

Obsolescent syntax

expand [**-***tabstop* | **-***tab1,tab2,...,tabn*] [*file* ...]

The **expand** command changes tab characters to spaces in the named files or in the standard input file, and writes the result to the standard output file.

FLAGS

-t *tablist* Specifies the tab stops. The *tablist* argument consists of a single positive decimal integer or multiple positive decimal integers, separated by spaces or commas, in ascending order. If a single number is specified, tabs are set *tablist* column positions apart instead of the default width (8). If multiple numbers are specified, tabs are set at those specific column positions. Tabbing to tab stop position *n* thus causes the next character output to be in the (*n*+1)th column position on that line.

If the **expand** command has to process a tab character at a position beyond the last of those specified in a multiple tab stop list, the tab character is replaced by a single space in the output.

-*tabstop* Sets tab stops *tabstop* spaces apart instead of the default distance (8). (Obsolescent.)

-*tab1, tab2, ..., tabn*
Sets tab stops at specified columns. Columns are measured in bytes. (Obsolescent.)

DESCRIPTION

Backspace characters are preserved in the output and decrement the column count for tab calculations. The column position count cannot be decremented below zero. The **expand** command is useful for preprocessing character files (before sorting, looking at specific columns, and so on) that contain tab characters.

EXAMPLES

1. To replace tab characters in **file** with spaces, enter:

   expand file

RELATED INFORMATION

Commands: **unexpand**(1).

NAME

export - Allows values of variables to be used by other commands

SYNOPSIS

export [*name*[=*value* ...]]

export -p

FLAGS

-p Writes the names and values of all exported variables.

DESCRIPTION

The **export** command marks the name and value, specified as the *name* and *value* arguments, for automatic export to the shell environment.

If **-p** is specified, **export** writes the names and values of all exported variables to standard output. The command **export -p** allows portable access to the value that can be saved and then later restored by using, for example, a **.** (dot) script. The shell formats the output, including proper use of quoting, so that it is suitable for reinput to the shell, as commands that achieve the same exporting results.

EXAMPLES

1. The following example defines a umask value of 066 and exports this value to the shell where it will be used for all new files and directories created during the shell process.

   export umask=066

2. The following command lists all exported variables and their values in effect for the current shell process.

   export -p

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.
- Words, following a command specified as *command*, that are in the format of a parameter assignment are expanded with the same rules as a parameter assignment. This means that ~ (tilde) substitution is performed after the = (equal sign). Word splitting and filename generation are not performed.

The **export** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

NAME

expr - Evaluates arguments as expressions

SYNOPSIS

expr *expression*

DESCRIPTION

The **expr** command reads an expression, evaluates it, and writes the result to the standard output file. Within the *expression* argument, you must separate each term with spaces, precede characters special to the shell with a \ (backslash), and quote strings containing spaces or other special characters. Note that **expr** returns 0 (zero) to indicate a zero value, rather than the null string. Integers can be preceded by a unary minus sign. Internally, integers are treated as 32-bit, two's complement numbers.

The operators and keywords are described in the following listing. Characters that need to be escaped are preceded by a \ (backslash). The list is in order of increasing precedence, with equal precedence operators grouped within {} (braces).

expression1 **\|** *expression2*
Returns *expression1* if it is neither null nor 0 (zero); otherwise, returns *expression2*.

expression1 **\ &** *expression2*
Returns *expression1* if neither *expression1* nor *expression2* is null nor 0 (zero); otherwise, returns 0 (zero).

expression1 **{ =, \>, \>=, \<, \<=, != }** *expression2*
Returns the result of an integer comparison if both expressions are integers; otherwise, returns the result of a string comparison.

expression1 **{+, -}** *expression2*
Adds or subtracts integer-valued arguments.

expression1 **{ *, /, % }** *expression2*
Multiplies, divides, or provides the remainder from the division of integer-valued arguments.

expression1 **:** *expression2*
Compares *expression1* with *expression2*, which must be a regular expression, with syntax as described for the **grep** command, except that all patterns are *anchored* to the beginning of the string, so ^ (circumflex) (which anchors a pattern to the beginning of a line) is not a special character in this context.

Normally, the matching operator returns the number of characters matched. Alternatively, you can use the **\(**...**\)** symbols in *expression2* to return a portion of *expression1*.

(*expression*)
Provides expression grouping.

To avoid unpredictable results when using a range expression to match a class of characters, use a character class expression rather than a standard range expression. For information about character class expressions, see the discussion of this topic in the **grep** command reference page.

Each part of an expression is composed of separate arguments, so use of spaces is required. For example, enter the following command line instead of **expr 1+2**:

expr 1 + 2

EXAMPLES

1. To increment a shell variable, enter:

   COUNT=‘expr $COUNT + 1‘

   This command adds **1** to the **COUNT** shell variable (see the **sh** command reference page for details).

2. To find the length of a shell variable, enter:

   RES=‘expr "$VAR" : ".*"‘

   Note that the **VAR** variable is placed within double quotation marks to avoid problems where **VAR** is NULL or contains embedded spaces. The regular expression is also quoted to avoid expansion by the shell.

3. To use part of a shell variable, enter:

   RES=‘expr "$VAR" : "-*\(.*\)"‘

   This removes leading **-** (dashes), if any, from the **VAR** variable. If the **\(\)** characters are omitted, the **RES** variable would contain the length of **VAR**.

4. Special considerations:

   RES=‘expr "x$VAR" : "x-*\(.*\)"‘

   This command succeeds even if the **VAR** variable has the value **-** (dash).

   RES=‘expr "x$VAR" = "x="

   This succeeds even if the **VAR** variable has the value **=** (equal sign).

EXIT VALUES

The **expr** command returns the following exit values:

0	The expression is neither null nor 0 (zero).
1	The expression is null or 0 (zero).
2	The expression is invalid.
>2	An error occurred.

RELATED INFORMATION

Commands: **grep(1)**, **sh(1)**, **test(1)**.

NAME

false - Returns a standard exit value

SYNOPSIS

false

DESCRIPTION

The **false** command returns a nonzero exit value. This command is usually used in input to the **sh** command.

EXAMPLES

This procedure displays the date and time once a minute. To stop it, press the Interrupt key sequence.

EXIT VALUES

The nonzero value returned by **false** may vary from system to system.

RELATED INFORMATION

Commands: **sh**(1), **true**(1).

NAME

fc - Lists, edits, or reexecutes commands

SYNOPSIS

fc [**-r**] [**-e** *editor*] [*first* [*last*]]

fc -l[-nr] [*first* [*last*]]

fc -s[*old=new*] [*first*]

fc -e -[*old=new*] [*first*] (Obsolescent)

FLAGS

-e *editor* Uses the specified editor to edit the commands. The value in the **FCEDIT** variable is used as a default when **-e** is not specified. If **FCEDIT** is null or unset, **ed** is used as the editor.

-l Lists the commands rather than invokes an editor on them. The commands are written in the sequence indicated by the *first* and *last* arguments, as affected by **-r**, with each command preceded by the command number.

-n Suppresses command numbers when listing with **-l**.

-r Reverses the order of the commands listed with **-l** or editor (with neither **-l** or **-s**). An obsolescent version of **-r** is **fc -e -**.

-s Reexecutes the command without invoking an editor.

DESCRIPTION

The **fc** command lists or edits and reexecutes commands previously entered to an interactive shell.

The parameters *first* and *last* specify a range of commands to be listed or edited, where *first* and *last* are the names of previously executed commands or the numbers of commands in the list produced by the **history** command. A negative number is used as an offset to the current command number.

If *last* is not specified, then it will be set to *first*. If *first* is not specified, the previous command is the default for editing. The *old=new* argument replaces the first occurrence of a command name *old* in the commands to be reexecuted by the string *new*.

The number of previous commands that can be accessed is determined by the value of the **HISTSIZE** variable.

EXAMPLES

1. The following command reexecutes the command whose number in the **history** log corresponds to the number given for the argument.

   fc -s argument

2. The following command opens the **vi** editor on the command in the **history** log that corresponds to the number given for the argument.

   fc -e vi argument

NOTES

The **fc** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **history**(1), **sh**(1).

NAME

fg - Brings processes to the foreground

SYNOPSIS

fg [*job*]

DESCRIPTION

The **fg** command brings each process specified as *job* to the foreground. See the reference page for the **jobs** command for information on the format of **job**.

EXAMPLES

1. The following command restarts, as a foreground process, the stopped background process whose job number is 149.

   fg %149

NOTES

The **fg** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **jobs**(1), **sh**(1).

NAME

fgrep - Searches a file for a fixed-string pattern

SYNOPSIS

fgrep
[**-c** | **-l**]
[**-bhinqsvx**]
{ *pattern* ... | **-e** *pattern* ... | **-f** *pattern_file* ... }
[*file* ...]

FLAGS

While most flags can be combined, some combinations result in one flag overriding another. For example, if you specify both the **-n** and **-l** flags, the output includes only filenames (as specified by the **-l** flag) and thus does not include line numbers (as specified by the **-n** flag).

-b Precedes each line by the block number of the block in which it was found. Use this flag to help find disk block numbers by context.

-c Displays only a count of matching lines.

-e *pattern* ...
Specifies a *pattern*. This flag works the same as a simple *pattern* but is useful when the pattern begins with a - (dash).

-f *pattern_file* ...
Specifies a file that contains patterns. Each pattern terminates with a newwline character.

-h Suppresses reporting of filenames when multiple files are processed.

-i Ignores the case of letters in locating *pattern*; that is, uppercase and lowercase letters in the input are considered to be identical.

-l Lists the name of each file that has lines matching *pattern*. Each filename is listed only once; filenames are separated by newline characters.

-n Precedes each line with its relative line number in the file.

-q Suppresses all output except error messages. This flag is useful for easily determining whether a pattern or string exists in a group of files. When searching several files, it provides a performance improvement, because it can quit as soon as it finds the first match, and it requires less care by the user in choosing the set of files to supply as arguments, because it exits with a 0 (zero) exit status if it detects a match, even if the **fgrep** command detected an access or read error on earlier file arguments.

-s Suppresses error messages about inaccessible files.

-v Displays all lines except those that match the specified pattern. This flag is useful for filtering unwanted lines out of a file.

-x Displays lines that match the pattern exactly with no additional characters.

DESCRIPTION

The **fgrep** command searches the specified files (the standard input file by default) for lines containing characters that match the specified pattern and then writes the matching lines to the standard output file.

The **fgrep** command is an obsolescent version of the command **grep -F**, which searches for patterns that are fixed strings.

Command Usage

The **fgrep** command precedes the matched line with the name of the file containing it if you specify more than one file (except when the **-h** flag is specified).

Lines are limited to 2048 bytes; longer lines are broken into multiple lines of 2048 or fewer bytes.

Running the **fgrep** command on a nontext file (for example, an **.o** file) produces unpredictable results and is discouraged.

Regular Expressions (REs)

Regular expressions (REs) cannot contain newline characters, because these signal a new pattern. The following REs match a single character:

character
: An ordinary character (one other than one of the special pattern-matching characters) matches itself.

.
: A **.** (dot) matches any single character except the newline character.

[*string*]
: A string enclosed in **[ ]** (brackets) matches any one character in that string. In addition, certain pattern-matching characters have special meanings within brackets:

  ^
  : If the first character of *string* is a ^ (circumflex), the RE **[^***string***]** matches any character except the characters in *string* and the newline character. A ^ has this special meaning only if it occurs first in the string.

  -
  : You can use a **-** (dash) to indicate a range of consecutive characters. The characters that fall within a range are determined by the current collating sequence, which is defined by the **LC_COLLATE** environment variable. For example, **[a-d]** is equivalent to **[abcd]** in the traditional ASCII collating sequence.

    A range can include a multicharacter collating element enclosed within bracket-period delimiters ([. .]). The bracket-period delimiters in the RE syntax distinguish multicharacter collating elements from a list of the individual characters that make up the element.

    A collating sequence can define equivalence classes for characters. An equivalence class is a set of collating elements that all sort to the same primary location. They are enclosed within bracket-equal delimiters (**[= =]**). An equivalence class generally is designed to deal with primary-secondary sorting. For example, if **e**, **è**, and **ê** belong to the same equivalence class, then **[[=e=]fg]**, **[[=è=]fg]**, and **[[=ê=]fg]** are each equivalent to **[eèêfg]**.

    The **-** (dash) character loses its special meaning if it occurs first (**[-***string***]**), if it immediately follows an initial circumflex (**[^-***string***]**), or if it appears last (**[***string***-]**) in the string.

  ]
  : When the **]** (right bracket) is the first character in the string (**[]***string***]**) or when it immediately follows an initial circumflex (**[^]***string***]**), it is treated as a part of the string rather than as the string terminator.

special_character
A \ (backslash) followed by a special pattern-matching character matches the special character itself (as a literal character). These special pattern-matching characters are as follows:

. * [\ Always special, except when they appear within **[ ]** (brackets).

^ Special at the beginning of an entire pattern or when it immediately follows the left bracket of a pair of brackets (**[^...]**).

$ Special at the end of an entire pattern.

[: :] A character class name enclosed in bracket-colon delimiters matches any of the set of characters in the named class. Members of each of the sets are determined by the current setting of the **LC_CTYPE** environment variable. The supported classes are **alpha**, **upper**, **lower**, **digit**, **xdigit**, **space**, **print**, **punct**, **graph**, and **cntrl**. Here is an example of how to specify one of these classes:

[[:lower:]]

This matches any lowercase character for the current locale.

Forming Patterns

The following rules describe how to form patterns from REs:

- An RE that consists of a single, ordinary character matches that same character in a string.
- An RE followed by an ***** (asterisk) matches zero or more occurrences of the character that the RE matches. For example, the following pattern:

  ab*cd

  matches each of the following strings:

  acd
  abcd
  abbcd
  abbbcd

  but not the following string:

  abd

  If there is any choice, the leftmost longest matching string is chosen. For example, given the following string:

  122333444

  the pattern **.*** matches **122333444**, the pattern **.*3** matches **122333**, and the pattern **.*2** matches **122**.

- An RE followed by:

  \{*number*\}
  Matches exactly *number* occurrences of the character matched by the RE.

  \{*number*,\}
  Matches at least *number* occurrences of the character matched by the RE.

  \{*number1*,*number2*\}
  Matches any number of occurrences of the character matched by the RE from *number1* to *number2*, inclusive.

  The values of *number1* and *number2* must be integers in the range 0 through 255. Whenever a choice exists, this pattern matches as many occurrences as possible.

  Note that if *number* is 0 (zero), *pattern* matches zero occurrences of *pattern*. For example:

  ```
  $ echo abc | grep 'aX\{0\}bX\{0\}cX\{0\}'
  abc
  $
  ```

- You can combine REs into patterns that match strings containing the same sequence of characters. For example, **AB*CD** matches the string **ABCD**, and **[A-Za-z]*[0-9]*** matches any string that contains any combination of ASCII alphabetic characters (including none), followed by any combination of numerals (including none).

- The character sequence **\(***pattern***\)** matches *pattern* and saves it into a numbered holding space. Using this sequence, up to nine patterns can be saved on a line. Counting from left to right on the line, the first pattern saved is placed in the first holding space, the second pattern saved is placed in the second holding space, and so on.

  The character sequence *n* matches the *n*th saved pattern, which is placed in the *n*th holding space. (The value of *n* is an integer in the range 1 through 9.) Thus, the following pattern:

  \(A\)\(B\)C\2\1

  matches the string **ABCBA**. You can nest patterns to be saved in holding spaces. Whether the enclosed patterns are nested or in a series, *n* refers to the *n*th occurrence, counting from the left, of the delimiting characters, **\)**.

Restricting What Patterns Match

A pattern can be restricted to match either from the beginning of a line, up to the end of the line, or the entire line:

- A **^** (circumflex) at the beginning of a pattern causes the pattern to match only a string that begins in the first character position on a line.

- A **$** (dollar sign) at the end of a pattern causes that pattern to match only if the last matched character is the last character (not including the newline character) on a line.

- The construction **^***pattern***$** restricts the pattern to matching only an entire line.

EXAMPLES

1. To search several files for a string of characters, enter:

   fgrep 'strcpy' *.c

   This searches for the string **strcpy** in all files in the current directory with names ending in **.c**.

2. To count the number of lines that match a pattern, enter:

   fgrep -c '{' pgm.c
   fgrep -c '}' pgm.c

   This displays the number of lines in **pgm.c** that contain left and right braces.

   If you do not put more than one **{** or **}** on a line in your C programs, and if the braces are properly balanced, then the two numbers displayed will be the same.

3. To display all lines in a file that begin with an ASCII letter, enter:

   fgrep -e '^[a-zA-Z]' pgm.s

   Note that because the **fgrep** command searches only for fixed strings and does not interpret pattern-matching characters, the following command searches only for the literal string **^[a-zA-Z]** in file **pgm.s**:

   fgrep '^[a-zA-Z]' pgm.s

4. To display all lines that do not match a pattern, enter:

   fgrep -v '^#'

   This displays all lines that do not begin with a **#** (number sign).

5. To display the names of files that contain a pattern, enter:

   fgrep -l 'rose' *.list

   This searches the files in the current directory whose names end with **.list** and displays the names of those files that contain at least one line containing the string **rose**.

6. To display all lines that contain uppercase characters, enter:

   fgrep '[[:upper:]]' pgm.s

EXIT VALUES

The **fgrep** command returns the following exit values:

0 (zero) A match was found.

1 No match was found.

2 A syntax error occurred or a file was inaccessible, even if matches were found.

RELATED INFORMATION

Commands: **ed(1)**, **egrep(1)**, **grep(1)**, **sed(1)**, **sh(1)**.

Files: **locale(4)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification.

The following features are HP extensions to the XPG4 Version 2 specification:

- The **-b**, **-h**, **-q**, and **-s** flags are supported.

NAME

file - Determines file type from file content

SYNOPSIS

file *file* ...

DESCRIPTION

The **file** command reads files, performs a series of tests on each one, and attempts to classify them by type. **file** then writes the file types to standard output.

The **file** utility uses the following steps to determine the type of a file:

- Uses the *stat( )* function to determine if the file operand is a regular file or a special file such as a directory, a named pipe (FIFO), or a file in the Guardian file system.
- Checks regular files to determine if they are HP-generated object files.
- Examines the first 512 bytes of object files other than HP object files to determine the file type; uses the **/etc/magic** file to check for magic values.
- Invokes a Guardian procedure call for Guardian text files to retrieve the Guardian specific file attributes, file type, and file code.

If a file appears to be plain text, **file** examines the first 512 bytes and tries to determine what kind of text it is. If the first 512 bytes only contain ASCII characters, **file** returns either `ASCII text` or `English text`. If the file contains other characters (that is, European or Asian xtended characters), **file** returns `data`. If a file does not appear to be plain text, **file** attempts to distinguish a binary data file from a text file that contains extended characters.

/etc/magic File

The **file** command uses the **/etc/magic** file to identify files that have a magic number (that is, contain a string constant that indicates type).

The following magic values are supported:

Table 3–1. Supported Magic Values

Byte Offset	Value Type	Magic Value	Printed String
0	short	070707	**cpio** archive
0	string	070707	ASCII **cpio** archive
0	string	%!	PostScript document
0	string	<MakerFile	FrameMaker document
0	string	<MIFFile	FrameMaker MIF file
0	string	<MML	FrameMaker MML file
0	string	!<arch>	archive
257	string	ustar	tar archive
40	string	SunBin	Sun binary

You can add other magic values to the **/etc/magic** file.

Use on Guardian Objects

Specify Guardian files with the **/G** pathname convention.

The **file** command uses the *stat( )* function to determine the file type. Guardian processes with subtype 30 are identified as `character special` files. Guardian pathnames consisting of only **/G**, volume names, or subvolume names are identified as `directory`. Guardian nondisk devices and processes that are not of subtype 30 are identified as `block special`.

For disk files, the **file** utility checks the file format and returns its type. The file utility can only detect object files created by HP compilers and tools. Object files generated by other sources are

flagged as `unknown format`.

For text files, the **file** utility examines the file data and tries to determine what kind of text it contains. The string `empty` is printed for any file type that has no data.

Standard Output

The *type* value for each *file* operand is printed to standard output in the following format:

"`%s: %s\n`", *file*, *type*[,*type*]...

where *type* contains one of the following strings:

File Type	Strings
OSS and Guardian files that are not disk files	`block special`, `character special`, `directory`, `fifo`, `strings`
Empty regular OSS files	`empty`
Regular HP object files	`Archive member`, `axcel region`, `binder region`, `ELF object format`, `executable`, `Guardian Archive`, `Guardian COFF object format`, `Guardian target library`, `NonStop OSS target`, `TNS/E PIC object file`, `TNS/R object file`, `TNS/R PIC object file`, `OSS archive`, `process snapshot`, `shared runtime library`, `symbol region`, `TNS object format`, `uses shared runtime library`
Regular text files	`ASCII text`, `assembler program text`, `commands text`, `C program text`, `data or International Language text`, `English text`, `FORTRAN program text`, `English text`, `EQN`, `input text`, `[nr/tr]off`, `SCCS`, `TBL`
Files in the Guardian file system	`entry-sequenced`, *file codes*, `key-sequenced`, `relative`, `SQL object`, `unstructured`
Files with magic values, **cpio** or **tar** format files	`archive`, `CPIO archive`, `FrameMaker document`, `FrameMaker MIF file`, `FrameMaker MML file`, `PostScript document`, `Sun Binary`, `TAR archive`
All other files	`unknown format`

EXAMPLES

1. To show the file type of a Guardian text file, enter:

   file /G/vol/subvol/file

   This command displays the pathname of the file followed by the file type.

FILES

/etc/magic	File type database.

RELATED INFORMATION

Commands: **ls(1)**.

Files: **locale(4)**.

STANDARDS CONFORMANCE

This command has extensions to the XPG4 Version 2 specification in order to support files in the Guardian environment.

NAME

find - Finds files matching an expression

SYNOPSIS

find *pathname* ... [**-W NOG**] [**-W NOE**] [*expression* ...]

FLAGS

HP Extensions

-W NOG Specifies that the **/G** directory should be omitted when the initial directory is root. This flag is ignored when the initial directory is not **/**, **/E**, or **/E/***system*.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root. This flag is ignored when the initial directory is not root.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

The **find** command recursively searches the directory tree for each specified pathname, seeking files that match a Boolean *expression* written using terms given later. The output from **find** depends on the terms used in *expression*.

Expressions

In the following descriptions, the argument *number* is a decimal integer that can be specified as *+number* (more than *number*), *-number* (less than *number*), or *number* (exactly *number*).

The first argument that starts with a **-** (dash) or is a **!** (exclamation point) or a **(** (left parenthesis) and all subsequent arguments are interpreted as an expression.

-acl [*aclpatt* | *=aclpatt* | **opt**]
TRUE if the access control list (ACL) of the file matches the ACL pattern *aclpatt* or contains optional ACL entries. See "Access Control Lists (ACLs)."

-name *pattern*
TRUE if *pattern* matches the basename of a filename. You can use pattern-matching characters, provided they are quoted.

-perm [-]*mode*
TRUE if the file permission code of the file exactly matches *mode* (see the reference page for the **chmod** command). If the optional **-** (dash) is present, this expression evaluates to TRUE if *at least* these permissions are set. If the **-** is omitted, the expression evaluates to TRUE when the file permission bits exactly match the value of the resulting template.

The *mode* argument can be up to three octal digits. This argument is used to represent file mode bits. It is identical in format to the symbolic mode argument described for the **chmod** command and is interpreted as follows:

- To start, a template is assumed with all file mode bits cleared.
- The symbol **+** (plus sign) sets the appropriate mode bits in the template, whereas a symbol of **-** (minus sign) clears the appropriate bits. A **-** (minus sign) cannot be the first character of *mode* because it could create ambiguity with the optional leading **-** (dash). Because the initial mode is all bits off, there are no symbolic modes that need to use **-** (minus sign) as the first character.

- The symbol = (equal sign) sets the appropriate mode bits without regard to the contents of the process's file mode creation mask.

-perm [-]*octal_number*
If the **-** (dash) is omitted, TRUE when the file permission bits exactly match the value of the octal number *octal_number* and only the bits corresponding to the octal mask 07777 are compared. (For more information, see the description of the octal mode on the **chmod** command's reference page.) If *octal_number* is preceded by a **-** (dash), the expression evaluates as TRUE if *at least* all of the bits specified in *octal_number* that are also set in the octal mask 07777 are set.

-priv { PRIVSETID | PRIVSOARFOPEN }
(J06.11 or later J-series RVUs or H06.22 or later H-series RVUs only)
TRUE when the file has the specified privilege. For example, the flag:

-priv PRIVSETID

is TRUE for files that have the PRIVSETID privilege.

-prune Always TRUE. This expression prunes the search tree at the file. That is, if the current pathname is a directory, the **find** command does not descend into that directory. If the **-depth** expression is specified, the **-prune** expression has no effect.

-type *type*
TRUE if the file *type* is of the specified type as follows:

c	Character special file
d	Directory
f	Plain file
l	Symbolic link
p	FIFO (a named pipe)
s	Socket file (H-series and J-series RVUs only)

-links *number*
TRUE if the file has *number* links. The argument *number* is a decimal integer that can be specified as *+number* (more than *number*), *-number* (less than *number*), or *number* (exactly *number*). See the reference page for the **ln** command.

-user *user*
TRUE if the file belongs to *user*.

-nouser TRUE if the file belongs to a user ID for which the **getpwuid()** function returns null.

-group *group*
TRUE if the file belongs to *group*.

-nogroup
TRUE if the file belongs to a group ID for which the **getgrgid()** function returns null.

-size *number*[**c** | **k**]
TRUE if the file is *number* blocks long (512 bytes per block). For this comparison, the file size is rounded up to the nearest block. If the **c** argument is present, the expression evaluates to TRUE if the file is *number* bytes long. If the **k** argument is present, the expression evaluates to TRUE if the file is *number* kilobytes long. For this comparison, the file size is rounded up to the nearest kilobyte.

The argument *number* is a decimal integer that can be specified as *+number* (more than *number*), *-number* (less than *number*), or *number* (exactly *number*).

-atime *number*
TRUE if the file was accessed in the past *number* days. The argument *number* is a decimal integer that can be specified as *+number* (more than *number*), *-number* (less than *number*), or *number* (exactly *number*). For example **-atime 3** is TRUE if the file was accessed any time in the period from 72 to 48 hours ago.

-mtime *number*
TRUE if the file was modified in the past *number* days. The argument *number* is a decimal integer that can be specified as *+number* (more than *number*), *-number* (less than *number*), or *number* (exactly *number*).

-ctime *number*
TRUE if the file inode was changed in the past *number* days. The argument *number* is a decimal integer that can be specified as *+number* (more than *number*), *-number* (less than *number*), or *number* (exactly *number*).

-exec *command*
TRUE if the *command* runs and returns a 0 (zero) value as exit status. The end of *command* must be punctuated by a quoted or escaped **;** (semicolon). The command parameter { } is replaced by the current pathname. If shell quoting is used in *command*, each word in the command must be quoted separately. Also, the characters **;** (semicolon) and **{ }** (braces) must appear as separate words on a command line.

The current directory for the invocation of *command* is the same as the current directory when the **find** command was started.

-ok *command*
Equivalent to the **-exec** expression, except that the **find** command first asks you whether it should start *command*. If your response begins with **y**, or the locale's equivalent of a **y**, *command* is started. If the response is negative, *command* is not invoked and the expression evaluates as FALSE. The end of *command* must be punctuated by a quoted or escaped semicolon. If shell quoting is used in *command*, each word in the command must be quoted separately. Also, the characters **;** (semicolon) and **{ }** (braces) must appear as separate words on a command line.

-print Always TRUE; causes the current pathname to be displayed. The **find** command assumes a **-print** expression, unless the **-exec**, **ls**, or **-ok** expressions are present.

-newer *file*
TRUE if the current file was modified more recently than the file indicated by *file*.

-depth Always TRUE. This expression causes the descent of the directory hierarchy to be done so that all entries in a directory are affected before the directory itself. This expression can be useful when **find** is used with the **pax** utility to transfer files contained in directories without write permission.

If the **-depth** expression is not specified, all entries in a directory are affected after the directory itself. If **-depth** is specified, it applies to the entire expression, even if the **-depth** primary would not normally be evaluated.

\(*expression* **\)**
TRUE if *expression* is TRUE.

-ls Always TRUE; causes the *pathname* argument to be printed together with its associated statistics. These include, respectively, inode number, size in kilobytes (1024 bytes), protection mode, number of hard links, user, group, size in bytes, and modification time. If the file is a special file, the size field contains the major and minor device numbers. If the file is a symbolic link, the pathname of the linked-to file is printed, preceded by ->. The format of the **-ls** flag is identical to that of **ls -gilds** (note that formatting is done internally, without executing **ls**.)

-xdev Always TRUE; causes the **find** command to not traverse down into a file system different from the one on which the current pathname resides. If any **-xdev** expression is specified, it applies to the entire expression even if the **-xdev** expression would not normally be evaluated.

The primaries can be combined using the following operators (in descending order of precedence):

(*expression* **)**
TRUE if *expression* is TRUE.

! *expression*
The negation of a primary (! is the unary **not** operator).

expression [**-a**] *expression*
Concatenation of primaries (the **and** operation is implied by the juxtaposition of two primaries or can be explicitly stated as **-a**). The second expression is not evaluated if the first expression is FALSE.

expression **-o** *expression*
Alternation of primaries (**-o** is the **or** operator). The second expression is not evaluated if the first expression is TRUE.

If no *expression* is present, **-print** is used as the expression; otherwise, if the given expression does not contain any of the expressions **-exec**, **-ok**, or **-print**, the given expression is effectively replaced by the following:

(*expression* **) -print**

The **-user**, **-group**, and **-newer** expressions each evaluate their respective arguments only once.

To avoid unpredictable results when using a range expression to match a class of characters, use a character class expression rather than a standard range expression. For information about character class expressions, see the reference page for the **grep** command.

Access Control Lists (ACLs)

The **-acl** expression enables the user to search for access control list (ACL) entries. The expression TRUE if the file's access control list matches an access control list pattern or contains optional ACL entries (see the **acl(5)** reference page). The **-acl** expression has three forms:

-acl *aclpatt* Match all files that have ACLs that include all (zero or more) pattern entries specified by the *aclpatt* pattern.

-acl =*aclpatt* Match a file only if its ACL includes all (zero or more) pattern entries specified by the *aclpatt* pattern, and every entry in its ACL is matched by at least one pattern entry specified in the *aclpatt* pattern.

-acl opt Match all files that have ACLs that include optional ACL entries.

By default, **-acl** is TRUE for files that have access control lists that include all the access control list patterns in *aclpatt*. The ACL for a file can also contain unmatched entries.

If *aclpatt* begins with **=**, the remainder of the string must match all entries in the access control list of the file.

An *aclpatt* consists either of a type field, an ID field, and a mode field, or a type field and a mode field. The fields are separated by colons. You can specify multiple comma-separated *aclpatts*.

The type field is one of **user**, **group**, **class**, **other** or *****, optionally preceded by **default:**. The literals **user**, **group**, **class**, **other**, and **default** can be abbreviated to **u**, **g**, **c**, **o**, and **d**, respectively. A type field of ***** matches any of the preceding types. If the type field is **class**, **other**, or *****, the ID field is not allowed (for example, **-acl *:rwx**).

The ID field is either a numeric user or group ID, a user or group ID string from the system user authentication database or group database, respectively, or *****, which matches any ID.

The mode field consists of a string of three characters. The first character is either **r**, indicating that read permission is granted; **-**, indicating that read permission is denied; or **?**, which matches either state of read permission. The second character is either **w**, **-**, or **?**, which similarly indicate the state of write permission; and the third character is either **x**, **-**, or **?**, which indicates the state of execute permission.

As a special case, if *aclpatt* is the value **opt**, the expression is true for files with optional access control list entries.

Environment Variables

The following environment variables affect the execution of the **find** command:

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root and a recursive operation occurs in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE	Omit the **/E** directory.
NOG	Omit the **/G** directory.
NOG:NOE	Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

EXAMPLES

1. To list all files in the file system with a given base filename, enter:

   find / -name .profile

   This command searches the entire file system and writes the complete pathnames of all files named **.profile**. The **/** (slash) tells the **find** command to search the root directory and all of its subdirectories. This search may take a while, so it is best to limit the search by specifying the directories where you think the files might be.

2. To list the files with a specific permission code in the current directory tree, enter:

   find . -perm 0600

   This command lists the names of the files that have only owner-read and owner-write permission. The **.** (dot) tells the **find** command to search the current directory and its subdirectories. See the reference page for the **chmod** command for details about permission codes. Alternatively, you could enter the following:

   find . -perm u+rw

3. To search several directories for files with certain permission codes, enter:

   find manual clients proposals -perm -0600

   This command lists the names of the files that have owner-read and owner-write permission and possibly other permissions. The directories **manual**, **clients**, and **proposals**, and their subdirectories, are searched. Note that the expression **-perm 0600** in the previous example selects only files with permission codes that match **0600** exactly. In this example, the expression **-perm -0600** selects files with permission codes that allow at least the accesses indicated by **0600**. This command also matches the permission codes **0622** and **2744**.

4. To search for files on the local node that have the PRIVSETID privilege, enter:

   find /bin -priv PRIVSETID -print

5. To search for regular files with multiple links, enter:

   find . -type f -links +1

   This lists the names of the ordinary files (**-type f**) that have more than one link (**-links +1**). Note that every directory has at least two links: the entry in its parent directory and its own **.** (dot) entry. See the reference page for the **ln** command for details about multiple file links.

6. To search for the file **f1** among the OSS files on the remote node **node1**, enter:

   export UTILSGE=NOG
   find /E/node1 -name f1 -print

7. To search for the file **f1** among the OSS files on the local node, enter:

   find / -W NOG -W NOE -name f1 -print

8. To find all files not owned by user karl that have access control lists with at least one entry associated with karl, and one entry for no specific user in group bin with the read bit on and the write bit off, enter:

   find / ! -user karl -acl u:karl:???,g:bin:r-? -print

9. To find all files that have a read bit set in any access control list entry, enter:

   find / -acl *:r?? -print

10. To find all files that have the write bit unset and execute bit set in every access control list entry, enter:

    find / -acl =*:?-x -print

11. To find all files that have optional access control list entries, enter:

    find / -acl opt -print

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

EXIT VALUES

The **find** command returns a 0 (zero) if all the paths are visited without error. The **find** command returns a nonzero value if it encounters an error.

RELATED INFORMATION

Commands: **chmod(1)**, **getfilepriv(1)**, **grep(1)**, **ln(1)**, **setacl(1)**, **setfilepriv(1)**, **sh(1)**, **test(1)**.

Functions: **stat(2)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The **-W NOG** and **-W NOE** flags and the **UTILSGE** environment variable are HP extensions to the XPG4 Version 2 specification.

NAME

flex - Generates a C language lexical analyzer

SYNOPSIS

flex [**-bcdfinpstvFILT8**] **-C**[**efmF**] [**-S***skeleton*] [*file* ...]

FLAGS

-b Generates backtracking information to file **lex.backtrack**. This is a list of scanner states that require backtracking and the input characters on which they backtrack. By adding rules, you can remove backtracking states. If all backtracking states are eliminated and the **-f** or **-F** flag is used, the generated scanner will run faster.

-d Makes the generated scanner run in **debug** mode. Whenever a pattern is recognized and the global **yy_lex_debug** is nonzero (which is the default value), the scanner writes to the standard error file a line of the form:

```
--accepting rule at line 53 ("the matched text")
```

The line number refers to the location of the rule in the file defining the scanner (the input to the **flex** command). Messages are also generated when the scanner backtracks, accepts the default rule, reaches the end of its input buffer (or encounters a NULL), or reaches an End-of-File.

-f Specifies *full table* (no table compression is done). The result is large but fast. This flag is equivalent to **-Cf**.

-i Instructs the **flex** command to generate a case-insensitive scanner. The case of letters given in the **flex** input patterns is ignored, and tokens in the input are matched regardless of case. The matched text given in the **yytext** variable will have the original case (as read by the scanner).

-p Generates a performance report to the standard error file. This identifies features of the **flex** input file that will cause a loss of performance in the resulting scanner.

-s Causes the default rule (that unmatched scanner input is echoed to the standard output file to be suppressed. If the scanner encounters input that does not match any of its rules, it aborts with an error.

-t Instructs the **flex** command to write the scanner it generates to the standard output file instead of to the file **lex.yy.c**.

-v Specifies that the **flex** command should write to the standard error file a summary of statistics regarding the scanner it generates.

-F Specifies that the fast scanner table representation should be used. This representation is about as fast as the full table representation (the **-f** flag), and for some sets of patterns it will be considerably smaller (and for others, larger). This flag is equivalent to the **-CF** flag.

-I Instructs the **flex** command to generate an interactive scanner; that is, a scanner that stops immediately rather than looking ahead if it knows that the currently scanned text cannot be part of a longer rule's match. Note that the **-I** flag cannot be used with full or fast tables; that is, with the **-f**, **-F**, **-Cf**, or **-CF** flags.

-L Instructs the **flex** command not to generate **#line** directives in the file **lex.yy.c**. The default action is to generate such directives so error messages in the actions will be correctly located with respect to the original **flex** input files.

-T Makes the **flex** command run in trace mode. It generates a lot of messages to the standard output file concerning the form of the input and the resultant nondeterministic and deterministic finite automata. This flag is mostly for use in maintaining the **flex** command.

-8 Instructs the **flex** command to generate an 8-bit scanner (the default scanner is a 7-bit scanner).

-C[efmF]

Controls the degree of table compression. The default setting is **-Cem**, which provides the highest degree of table compression. Faster-executing scanners can be traded off at the cost of larger tables with the following generally being true:

Slowest and smallest

-Cem
-Cm
-Ce
-C
-C{f,F}e
-C{f,F}

Fastest and largest

-C flags are not cumulative; whenever the flag is encountered, the previous **-C** settings are forgotten. The **-f** or **-F** and **-Cm** flags do not make sense together; there is no opportunity for meta-equivalence classes if the table is not being compressed. Otherwise, the flags may be freely mixed.

-C Specifies that the scanner tables should be compressed and neither equivalence classes nor meta-equivalence classes should be used.

-Ce Directs the **flex** command to construct equivalence classes; for example, sets of characters that have identical lexical properties. Equivalence classes usually give dramatic reductions in the final table/object file sizes (typically a factor of 2 to 5) and are inexpensive in terms of cost versus performance (one array look-up per character scanned).

-Cm Directs the **flex** command to construct meta-equivalence classes, which are sets of equivalence classes (or characters, if equivalence classes are not being used) that are commonly used together. Meta-equivalence classes are often a benefit when using compressed tables, but they have a moderate performance impact (one or two "if" tests and one array look-up per character scanned).

-Cf Specifies that the full scanner tables should be generated; the **flex** command should not compress the tables by taking advantage of similar transition functions for different states.

-CF Specifies that the alternative fast scanner representation should be used.

-S*skeleton*
Overrides the default skeleton file from which the command constructs its scanners. This is useful for **flex** maintenance or development.

-c Specifies table-compression options. (Obsolescent)

-n Suppresses the statistics summaries that the **-v** flag typically generates. (Obsolescent.)

DESCRIPTION

The **lex** and **flex** commands have the same functionality.

The **flex** command is a tool for generating *scanners*: programs that recognize lexical patterns in text. **flex** reads the given input files, or its standard input file if no filenames are given or if a file operand is **-** (dash), for a description of a scanner to generate. The description is in the form of pairs of regular expressions and C code, called *rules*. **flex** generates as output a C source file, named **lex.yy.c**, which defines a routine **yylex()**. This file is compiled and linked with the **-ll** library to produce an executable. When the executable is run, it scans its input and the regular expressions in its *rules* looking for the best match (longest input). When it has selected a rule it executes the associated C code, which has access to the matched input sequence (commonly referred to as a token). This process then repeats until input is exhausted.

flex treats multiple input files as one.

Syntax for flex Input

This subsection contains a description of the **flex** input files, which are normally named with a **.l** suffix. It provides a listing of the special values, macros, and functions recognized by the **flex** command.

The **flex** input file consists of three sections, separated by a line with just **%%** in it:

[*definitions*]
%%
[*rules*]
[**%%**
[*user_functions*]]

where

definitions Contains declarations to simplify the scanner specification and declarations of start states, which are explained below.

rules Describes what the scanner is to do.

user_functions Contains user-supplied functions, which are copied directly to file **lex.yy.c**.

With the exception of the first **%%** sequence, all sections are optional. The minimal scanner, **%%**, copies its input to the standard output file.

Each line in the *definitions* section can be:

name regexp Defines *name* to expand to *regexp*. *name* is a word beginning with a letter or an underscore (**_**) followed by zero or more letters, digits, underscores, or dashes (**-**). In the regular-expression parts of the rules section, the command substitutes *regexp* wherever you refer to {*name*} (*name* within braces).

%x *state* [*state* ...] or **%s** *state* [*state* ...]
Defines names for states used in the rules section. A rule can be made conditionally active based on the current scanner state. Multiple lines defining states can appear, and each can contain multiple *state* names, separated by white space. The name of a *state* follows the same syntax as that of *regexp* names except that

dashes (**-**) are not permitted. Unlike *regexp* names, *state* names share the C **#define** namespace. In the *rules* section, states are recognized as <*state*> (*state* within angle brackets).

The **%x** directive names exclusive states. When a scanner is in an exclusive state, only rules prefixed with that state are active. Inclusive states are named with the **%s** directive.

%{ or **%}** When placed on lines by themselves, enclose C code to be passed verbatim into the global definitions of the output file. Such lines commonly include preprocessor directives and declarations of external variables and functions.

space or *tab* Appear at the beginning of lines in the definitions section that are to be passed directly into the **lex.yy.c** output file, as part of the initial global definitions.

The *rules* section follows the *definitions*, separated by a line consisting of **%%**. The rules section contains rules for matching input and taking actions, in the following format:

pattern [*action*]

pattern starts in the first column of the line and extends until the first nonescaped white space character. The command attempts to find the *pattern* that matches the longest input sequence and execute the associated *action*. If two or more *pattern*s match the same input, the one that appears first in the *rules* section is chosen. If no *action* exists, the matched input is discarded. If no *pattern* matches the input, the default action is to copy it to the standard output file.

All *action* code is placed in the **yylex( )** function. Text (C code or declarations) placed at the beginning of the *rules* section is copied to the beginning of the **yylex( )** function and can be used in actions. This text must begin with a space or a tab (to distinguish it from rules). In addition, any input (beginning with a space or within **%{** and **%}** delimiter lines) appearing at the beginning of the rules section before any rules are specified is written to file **lex.yy.c** after the declarations of variables for the **yylex( )** function and before the first line of code in **yylex( )**.

Elements of each rule are:

state A *pattern* can begin with a comma-separated list of *state* names enclosed by angle brackets (< *state* [**,***state*...] >). These states are entered through the **BEGIN** statement. If a *pattern* begins with a *state*, the scanner can recognize it only when in that state. The initial state is 0 (zero).

regexp A *pattern* can be a regular expression to match against the input stream. The regular expressions in the **flex** command provide a rich character-matching syntax.

The following characters, shown in order of decreasing precedence, have special meanings:

x Matches the character *x*.

"" (double quotes)
Enclose characters and treat them as literal strings. For example, **"*+"** is treated as the asterisk character followed by the plus character.

str (backslash)
If *str* is one of the characters **a**, **b**, **f**, **n**, **r**, **t**, or **v**, then represents the ANSI C interpretation (for example, **\n** is a newline). If *str* is a string of octal digits, it is interpreted as a character with octal value *str*. If *str* is a string of hexadecimal digits with a leading **x**, it is interpreted as a character with that value. Otherwise, it is interpreted literally with no special meaning. For example, **x*yz** represents the four characters **x*yz**.

[ ] (brackets)

Represent a character class in the enclosed range ([.-.]) or the enclosed list ([...]). The dash character (**-**) is used to define a range of characters from the ASCII value or the 8-bit class of the character that comes before the dash to the ASCII value or the 8-bit class of the character that follows the dash. For example, **[abcx-z]** matches **a**, **b**, **c**, **x**, **y**, or **z**.

The circumflex (^), when it appears as the first character in a character class, indicates the complement of the set of characters within that class. For example, **[^abc]** matches any character except **a**, **b**, or **c**, including special characters like newline. Similarly, **[^a-zA-Z]** is any character that is not a letter.

( ) (parentheses)

Group regular expressions. For example, **(ab)** is considered as a single regular expression.

{ } (braces)

When enclosing numbers, indicate a number of consecutive occurrences of the expression that comes before it. For example, **(ab){1,5}** indicates a match for from 1 to 5 occurrences of the string **ab**.

When enclosing a name, the name represents a regular expression defined in the *definitions* section. For example, **{digit}** will be replaced with the defined regular expression for **digit**. Note that the expansion takes place as if the definition were enclosed in parentheses.

. (dot) Matches any single character except newline.

? (question mark)

Matches zero or one of the preceding expressions. For example, **ab?c** matches both **ac** and **abc**.

***** (asterisk)

Matches zero or more of the preceding expressions. For example, **a*** is zero or more consecutive **a** characters. The utility of matching zero occurrences is more obvious in complicated expressions. For example, the expression **[A-Za-z][A-Za-z0-9]*** indicates all alphanumeric strings with a leading alphabetic character, including strings that are only one alphabetic character.

+ (plus sign)

Matches one or more of the preceding expressions. For example, **[a-z]+** is all strings of lowercase letters.

xy (concatenation)

Matches the expression *x* followed by the expression *y*.

| (vertical bar)

Matches either the preceding expression or the following expression. For example, **ab|cd** matches either **ab** or **cd**.

x/*y* (slash)

Matches expression *x* only if expression *y* (trailing context) immediately follows it. For example, **ab/cd** matches the string **ab** but only if followed by **cd**. Only one trailing context is permitted per *pattern*.

^ (circumflex)
When it appears at the beginning of the pattern, matches the beginning of a line. For example, **^abc** matches the string **abc** if it is found at the beginning of a line.

$ (dollar sign)
When it appears at the end of a pattern, matches the end of a line. It is equivalent to **/\n**. For example, **abc$** matches the string **abc** if it is found at the end of a line.

<<EOF>>
Matches an End-of-File.

<*x*> (angle bracket)
Identifies a state name (see earlier description of *state*) and can appear only at the beginning of a pattern. For example, **<done><<EOF>>** matches an End-of-File, but only if it is in the state **done**.

In addition, the following rules apply for bracket expressions:

Equivalence class expressions
These represent the set of collating elements in an equivalence class and are enclosed within bracket-equal delimiters (**[= =]**). An equivalence class generally is designed to deal with primary-secondary sorting; that is, for languages like French that define groups of characters as sorting to the same primary location, and then have a tie-breaking, secondary sort. For example, if **a, à** (a accent grave), and **â** (a circumflex) belong to the same equivalence class, then **[[=a=]b]**, **[[=à=]b]**, and **[[=â=]b]** are each equivalent to **[aàâb]**.

NOTE: If you are viewing this reference page online using the **man** command, the special characters are not displayed. See this reference page in the *Open System Services Shell and Utilities Reference Manual*.

Character class expressions
These represent the set of characters in the current locale belonging to the named ctype class. These are expressed as a ctype class name enclosed in bracket-colon delimiters **([: :])**.

In the C or OSS locale, the following character class expressions are supported: [**:alpha:**], [**:upper:**], [**:lower:**], [**:digit:**], [**:alnum:**], [**:xdigit:**], [**:space:**], [**:print:**], [**:punct:**], [**:graph:**], and [**:cntrl:**].

Other locales may define additional character classes.

Letters and digits never have special meanings. A character such as ^ or **-**, which has a special meaning in particular contexts, refers simply to itself when found outside that context. Spaces and tabs must be escaped to appear in a regular expression; otherwise they indicate the end of the expression.

action Each *pattern* in a rule has a corresponding *action*, which can be any arbitrary C statement. The pattern ends at the first nonescaped white space character; the remainder of the line is its *action*. If the action is empty, then when the pattern is matched, the input that matched it is discarded.

If the action contains a **{**, then the action scans till the balancing **}** is found, and the action may cross multiple lines. Using a **return** statement in an action returns from **yylex()**.

An action consisting solely of a vertical bar (|) means *same as the action for the next rule*.

flex variables that can be used within actions are:

yytext Is a string (**char ***) containing the current matched input. It cannot be modified.

yyleng Is the length (**int**) of the current matched input. It cannot be modified.

yyin Is a stream (**FILE ***) that the **lex** and **flex** commands reads from the standard input file by default. It can be changed, but because of the buffering **flex** uses, changing the stream makes sense only before scanning begins. Once scanning terminates because an End-of-File was found, **void yyrestart (FILE ****new_file***)** can be called to point **yyin** at a new input file. Alternatively, **yyin** can be changed whenever a new or different buffer is selected (see **yy_switch_to_buffer()**).

yyout Is a stream (**FILE ***) to which **ECHO** output is written (the standard output file by default). It can be changed by the user.

YY_CURRENT_BUFFER
Returns the current buffer (**YY_BUFFER_STATE**) used for scanner input.

flex macros and functions that can be used within actions are:

ECHO Copies the **yytext** variable to the scanner's output.

BEGIN *state*
Changes the scanner state to be *state*. This affects which rules are active. The *state* must be defined in a **%s** or **%x** definition. The initial state of the scanner is **INITIAL** or 0 (zero).

REJECT Directs the scanner to proceed immediately to the *next best* pattern that matches the input (which may be a prefix of the current match). The **yytext** and **yyleng** variables are reset appropriately. Note that **REJECT** is a particularly expensive feature in terms of scanner performance; if it is used in any of the scanner's actions, it slows down all the scanner's pattern matching operations. **REJECT** cannot be used if the command is invoked with either the **-f** or **-F** flag.

yymore( ) Indicates that the next matched text should be appended to the currently matched text in the **yytext** variable (rather than replace it).

yyless(*n*) Returns all but the first *n* characters of the current token back to the input stream, where they are rescanned when the scanner looks for the next match. The **yytext** and **yyleng** variables are adjusted accordingly.

yywrap( ) Returns 0 (zero) if there is more input to scan or 1 if there is not. The default **yywrap( )** always returns 1. It is implemented as a macro.

yyterminate( )
Can be used instead of a return statement in an action. It terminates the scanner and returns a 0 (zero) to the scanner's caller.

yyterminate( ) is automatically called when an End-of-File is encountered. It is a macro and can be redefined.

yy_create_buffer(*file, size*)
Returns a **YY_BUFFER_STATE** handle to a new input buffer large enough to accommodate *size* characters and associated with the given *file*. When in doubt, use **YY_BUF_SIZE** for the size.

yy_switch_to_buffer(*new_buffer*)
Switches the scanner's processing to scan for tokens from the given buffer, which must be a **YY_BUFFER_STATE**.

yy_delete_buffer(*buffer*)
Deletes the given buffer.

YY_NEW_FILE
Enables scanning to continue after the **yyin** variable has been assigned a new file to process.

YY_DECL Controls how the scanning function, **yylex( )**, is declared. By default, it is **int yylex( )** or, if prototypes are being used, **int yylex(void)**. This definition can be changed by redefining the **YY_DECL** macro. This macro is expanded immediately before the **{...}** (braces) that delimit the scanner function body.

YY_INPUT(*buf,result,max_size***)**
Controls scanner input. By default, **YY_INPUT** reads from the file-pointer **yyin** variable. Its action is to place up to *max_size* characters in the character array *buf* and return in the integer variable *result* either the number of characters read or the constant **YY_NULL** to indicate EOF. Following is a sample redefinition of **YY_INPUT**, in the definitions section of the input file:

```
%{
#undef YY_INPUT
#define YY_INPUT(buf,result,max_size)\
   {\
       int c = getchar();\
       result = (c == EOF) ? YY_NULL : (buf[0] = c, 1);\
   }
%}
```

When the scanner receives an End-of-File indication from **YY_INPUT**, it checks the **yywrap( )** function. If **yywrap( )** returns zero, it is assumed that the **yyin** has been set up to point to another

input file, and scanning continues. If it returns a nonzero value, then the scanner terminates, returning zero to its caller.

YY_USER_ACTION
Can be redefined to provide an action that is always executed prior to the matched pattern's action.

YY _USER_INIT
Can be redefined to provide an action that is always executed before the first scan.

YY_BREAK
Is used in the scanner to separate different actions. By default, it is simply a *break*, but it can be redefined if necessary.

The *user_functions* section consists of complete C functions, which are passed directly into the **lex.y.cc** output file (the effect is similar to defining the functions in separate **.c** files and linking them with **lex.y.cc**). This section is separated from the *rules* section by the **%%** delimiter.

Comments, in C syntax, can appear anywhere in the *user_functions* or *definitions* sections. In the *rules* section, comments can be embedded within actions. Empty lines or lines consisting of white space are ignored.

The following macros are not normally called explicitly within an action, but they are used internally by the **flex** command to handle the input and output streams.

input( ) Reads the next character from the input stream. You cannot redefine **input( )**.

output( ) Writes the next character to the output stream.

unput(*c*) Puts the character *c* back into the input stream. It will be the next character scanned. You cannot redefine **unput( )**.

libl.a contains default functions to support testing or quick use of a **flex** program without the **yacc** command; these functions can be linked in through **-ll**. They can also be provided by the user.

main( ) A simple wrapper that simply calls **setlocale( )** and then calls the **yylex( )** function.

yywrap( ) The function called when the scanner reaches the end of an input stream. The default definition simply returns 1, which causes the scanner in turn to return 0 (zero).

EXAMPLES

1. The following command processes the file **lexcommands** to produce the scanner file **lex.yy.c**:

   flex lexcommands

   This is then compiled and linked by the command:

   cc -oscanner lex.yy.c -ll

   to produce a program **scanner**.

2. The **scanner** program converts uppercase to lowercase letters, removes spaces at the end of a line, and replaces multiple spaces with single spaces. The **lexcommands** file contains:

```
%%
[A-Z]   putchar(tolower(yytext[0]));
[ ]+$
```

```
[ ]+  putchar(' ');
```

FILES

flex.skel Is the skeleton scanner.

lex.yy.c Is the generated scanner C source.

lex.backtrack Contains backtracking information generated from the **-b** flag.

NOTES

- Some trailing context patterns cannot be properly matched and generate warning messages:

  ```
  Dangerous trailing context
  ```

  These are patterns where the ending of the first part of the rule matches the beginning of the second part, such as **zx*/xy***, where the **x*** matches the **x** at the beginning of the trailing context.

- For some trailing context rules, parts that are actually fixed length are not recognized as such, leading to the previously mentioned performance loss. In particular, patterns using **{***n***}** (such as **test{3}**) are always considered variable length.

  Combining trailing context with the special | (vertical bar) action can result in fixed trailing context being turned into the more expensive variable trailing context. This happens in the following example:

  ```
  %%
  abc|
  xyz/def
  ```

- Use of the **unput( )** macro invalidates the contents of the **yytext** and **yyleng** variables within the current **flex** action.

- Use of the **unput( )** macro to push back more text than was matched can result in the pushed-back text matching a beginning-of-line (^) rule even though it did not come at the beginning of the line.

- Pattern matching of NULLs is substantially slower than matching other characters.

- The **flex** command does not generate correct **#line** directives for code internal to the scanner; thus, bugs in **flex.skel** yield invalid line numbers.

- Due to both buffering of input and read-ahead, you cannot intermix calls to **<stdio.h>** routines, such as **getchar( )**, with **flex** rules and expect it to work. Call **input( )** instead.

- The total table entries listed by the **-v** flag excludes the number of table entries needed to determine what rule was matched. The number of entries is equal to the number of deterministic finite-state automaton (DFA) states if the scanner does not use **REJECT**, and is somewhat greater than the number of states if it does.

- **REJECT** cannot be used with the **-f** or **-F** flag.

RELATED INFORMATION

Commands: **awk(1)**, **lex(1)**, **sed(1)**, **yacc(1)**.

Files: **locale(4).**

NAME

fold - Breaks lines in a file

SYNOPSIS

fold [**-bs**] [**-w** *width*] [*file* ...]

The **fold** command breaks lines in the specified files, or in the standard input file if no files are specified, to have maximum *width*.

FLAGS

-b Counts *width* in bytes rather than in column positions. In this case, the lines are not limited to **LINE_MAX** bytes.

-s If a segment of a line contains a blank character within the first *width* column positions (or bytes), breaks the line after the last such blank character, meeting the width constraints. If there is no blank character meeting the requirements, the **-s** flag does not affect that output segment of the input line.

-w *width* Specifies the maximum width to which lines should be folded in column positions (or bytes if **-b** is specified). The default value is 80.

DESCRIPTION

The **fold** command is a filter that folds lines from its input files, breaking the lines to have a maximum of *width* column positions (or bytes, if the **-b** flag is specified). The **fold** command breaks lines by inserting a newline character so that each output line is the maximum width possible that does not exceed the specified number of column positions (or bytes). A line cannot be broken in the middle of a character.

The **fold** command is often used to send text files to line printers that truncate, rather than fold, lines wider than the printer is able to print (usually 80 or 132 column positions).

If the **<backspace>**, **<tab>**, or **<carriage-return>** characters are encountered in the input and the **-b** flag is not specified, these characters are treated specially:

<backspace>
The current count of line width is decremented by one, although the count never becomes negative. The **fold** command does not insert a newline character immediately before or after any backspace character.

<tab> Each tab character encountered advances the column position pointer to the position of the next tab stop. Tab stops are at each column position *number*, such that *number* modulo 8 equals 1.

<carriage-return>
The current count of the line width is set to 0 (zero). The **fold** command does not insert a newline immediately before or after any carriage-return character.

Note that the **fold** command may affect any underlining that is present.

EXAMPLES

The **fold** command can be used to prepare files to be joined side-by-side with the **paste** command. You might want to display these two files, **az** and **AZ**, side-by-side:

```
aaaa bbbb cccc dddd eeee ffff gggg hhhh iiii jjjj kkkk llll mmmm
nnnn oooo pppp qqqq rrrr ssss tttt uuuu vvvv wwww xxxx yyyy zzzz

AAAA BBBB CCCC DDDD EEEE FFFF GGGG HHHH IIII JJJJ KKKK LLLL MMMM
NNNN OOOO PPPP QQQQ RRRR SSSS TTTT UUUU VVVV WWWW XXXX YYYY ZZZZ
```

The following command line:

fold -w 32 az > az2; fold -w 32 AZ > AZ2; paste -d" " az2 AZ2

results in the output below:

```
aaaa bbbb cccc dddd eeee ffff gg AAAA BBBB CCCC DDDD EEEE FFFF GG
gg hhhh iiii jjjj kkkk llll mmmm GG HHHH IIII JJJJ KKKK LLLL MMMM
nnnn oooo pppp qqqq rrrr ssss tt NNNN OOOO PPPP QQQQ RRRR SSSS TT
tt uuuu vvvv wwww xxxx yyyy zzzz TT UUUU VVVV WWWW XXXX YYYY ZZZZ
```

EXIT VALUES

The **fold** command returns the following values:

0 All input files were successfully processed.

1 A usage error occurred.

2 An input file cannot be opened. The **fold** command continues processing the other input files specified on the command line.

RELATED INFORMATION

Commands: **expand(1)**, **paste(1)**, **unexpand(1)**.

NAME

ftp - Transfers files between a local OSS file system and a remote host

SYNOPSIS

ftp [**-dginv**] [*host*]

The **ftp** command is the interface to the File Transfer Protocol (FTP). This command uses OSS FTP to transfer files between the local host and a remote host or between two remote hosts. OSS FTP only runs in an OSS shell environment. The Guardian FTP client runs in a Guardian environment.

FLAGS

The following flags can be entered on the shell command line. (The **ftp** command equivalents can also be entered at the `ftp>` prompt.)

-d Enables debugging by turning on the logging feature. See the **debug** subcommand.

-g Disables the expansion of metacharacters in filenames. Interpreting metacharacters may be referred to as expanding (sometimes called *globbing*) a filename. See the **glob** subcommand.

-i Disables interactive prompting during multiple file transfers. See the **prompt**, **mget**, **mput**, and **mdelete** subcommands for descriptions of prompting during multiple file transfers.

-n Prevents an automatic login on the initial connection. Otherwise, **ftp** searches for a **$HOME/.netrc** entry that describes the login and initialization process for the remote host. See the **user** subcommand.

-v Displays all the responses from the remote server and provides data transfer statistics. This is the default display mode when the output of the **ftp** command is to a device, such as the console or a display. However, if output is redirected, such as through a pipe or to a file, or if the **ftp** command is started by a daemon, such as the **cron** daemon, verbose mode is not in effect unless the **-v** flag or the **verbose** subcommand is used. See the **verbose** subcommand.

DESCRIPTION

The OSS FTP client transfers data between a local host with an OSS file system and a remote host that can use a dissimilar file system. Therefore, although the protocol provides a lot of flexibility for transferring data, it does not attempt to preserve file attributes that are specific to a particular file system (for example, the protection mode or modification times of a file). Additionally, the FTP protocol makes few assumptions about the overall structure of a file system and does not provide or allow such things as recursively copying subdirectories.

The **ftp** command provides subcommands for tasks such as listing remote directories, changing the current local and remote directory, transferring multiple files in a single request, creating and removing directories, and escaping to the local shell to perform shell commands. The **ftp** command also provides for security by sending passwords to the remote host and permits automatic login, file transfer, and logoff.

If you execute **ftp** and do not specify a hostname, **ftp** immediately displays the `ftp>` prompt and waits for an **ftp** subcommand. To connect to a remote host, you then execute the **open** subcommand. When the **ftp** command connects to the remote host, **ftp** then prompts for the username and password before displaying the `ftp>` prompt again. **ftp** fails if no password is defined at the remote host for the specified username.

If you do specify the name of a remote host, **ftp** immediately tries to establish a connection to the specified host. If **ftp** connects successfully, **ftp** searches for a local **$HOME/.netrc** file in your current directory or home directory. If the file exists, **ftp** searches the file for an entry that initiates

the login process and command macro definitions for the remote host. If the **$HOME/.netrc** file or autologin entry does not exist, **ftp** prompts you for a username and password. This occurs whether or not the hostname is entered on the command line.

If **ftp** finds a **$HOME/.netrc** autologin entry for the specified host, **ftp** attempts to use the information in that entry to automatically log into the remote host. The **ftp** command also loads any command macros defined in the entry. In some cases (for example, when the required password is not listed in an autologin entry), **ftp** prompts for the password before displaying the `ftp>` prompt. Once **ftp** completes the autologin process, **ftp** executes the **init** macro if the macro is defined in the autologin entry. If the **init** macro does not exist or does not contain a **quit** or **bye** command, **ftp** then displays the prompt and waits for a subcommand.

The remote username that you specify either at the prompt or in a **$HOME/.netrc** file must exist and have a password defined at the remote host, or **ftp** fails.

The **ftp** command interpreter, which handles all commands entered at the `ftp>` prompt, provides facilities that are not available with most file transfer programs, such as: the handling of filename arguments to **ftp** commands, the ability to collect a group of commands into a single command macro, and the ability to load macros from a **$HOME/.netrc** file. These facilities are designed to allow simplifying repeated tasks and to allow using **ftp** in unattended mode. The **ftp** command interpreter does not support entry of commands (including the arguments) longer than 198 characters at the `ftp>` prompt.

The command interpreter handles filename arguments according to the following rules:

- If a **-** (dash) is specified for the argument, standard input is used for read operations and standard output is used for write operations.
- Failing the preceding check, if globbing is enabled, local filenames are expanded according to the rules used in **csh**; (see the **glob** subcommand). If the **ftp** command expects a single local file (for example, **put**), the pattern-matching characters following the **put** command are interpreted as a file name.
- For **get** and **mget** subcommands with unspecified local filenames, the local filename is the same as the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename can then be altered if **runique** is on.
- For **mput** commands and **put** commands with unspecified remote filenames, the remote filename is the same as the local filename, which can be altered by a **ntrans** or **nmap** setting. The resulting filename can then be altered by the remote server if **sunique** is on.
- If the first argument is /G, the Guardian file space is accessed. The "G" must be capitalized (upper case) to access Guardian file space.

To end an **ftp** session when you are running interactively, use the **bye** or **quit** subcommand or the End-of-File key sequence at the `ftp>` prompt. To end a file transfer before it has been completed, use the Interrupt key sequence. The default Interrupt key sequence is **<Ctrl-c>**.

The **stty** command can be used to redefine this key sequence. Sending transfers (those from the local host to the remote host) are normally halted immediately. Receiving transfers are halted by sending an **FTP ABOR** instruction to the remote FTP server and discarding all incoming file transfer packets until the remote server stops sending them.

If the remote server does not support the **ABOR** instruction, the `ftp>` prompt will not appear until the remote server has sent all of the requested files. Additionally, if the remote server does something unexpected, the local **ftp** process may need to be ended manually.

SUBCOMMANDS

The following **ftp** subcommands can be entered at the prompt. If an argument for a subcommand includes spaces, enclose the argument within "" (double quotes).

![*command* [*argument* ...]]
Invokes an interactive shell on the local host. An optional command, with one or more optional arguments, can be specified.

$ *macro* [*argument* ...]
Executes the specified macro, previously defined with the **macdef** subcommand. Arguments are not expanded. See the **macdef** subcommand for further information.

? [*subcommand*]
Displays a help message describing the subcommand. If you do not specify *subcommand*, **ftp** displays a list of known subcommands.

account [*password*]
Sends a supplemental password that a remote host may require before granting access to its resources. If the password is not supplied with the command, you are prompted for the password. The password does not appear on the screen.

append *local_file* [*remote_file*]
Appends a local file to a file on the remote host. If the remote filename is not specified, the local filename is used, altered by any setting made with the **ntrans** or **nmap** subcommand. The **append** subcommand uses the current values for **form**, **mode**, **struct**, and **type** while appending the file. For more information on these subcommands, see their individual descriptions.

ascii Sets the file transfer type to network ASCII. This is the default. File transfer may be more efficient with binary-image transfer. File transfers to the Guardian (/G) file system will always arrive as Guardian file type 180.

bell Sounds a bell after the completion of each file transfer.

binary Sets the file transfer type to binary image. This can be more efficient than an ASCII transfer. Attempts to transfer binary-image files without specifying this subcommand can lead to corrupt data, since linefeeds and carriage returns are interpreted differently by the **ascii** subcommand. File transfers to the Guardian (/G) file system will always arrive as Guardian file type 180.

bye Ends the File Transfer session and exits **ftp**. Same as **quit**. An End of File key sequence (default control-D) also terminates the session and exits.

case Sets a toggle for the case of filenames. When **case** is on, remote filenames that appear in all capital letters are changed from uppercase to lowercase when written in the local directory. The default is off (uppercase remote filenames are written in uppercase in the local directory).

cd *remote_directory*
Changes the remote working directory to the specified directory.

cdup Changes the working directory on the remote host to the parent of the current directory.

close Ends the File Transfer session, but does not exit **ftp**. Defined macros are erased. Same as **disconnect**.

cr Strips the carriage-return character from a carriage-return/linefeed sequence when receiving records during ASCII-type file transfers. (**ftp** terminates each ASCII-type record with a carriage-return/linefeed sequence during file transfers.) This conforms with the UNIX system convention for terminating records with a single linefeed. Records on remote hosts that have different record termination conventions may have single linefeeds embedded in records. To distinguish these embedded linefeeds from record delimiters, set **cr** to **off**. **cr** toggles between **on** and **off**.

delete *remote_file*
Deletes the specified remote file.

debug [**on** | **off**]
Prints each command sent to the remote host preceded by the string **-->** when **debug on** is specified.

dir [*remote_directory*][*local_file*]
Writes a listing of the contents of *remote_directory* to the file *local_file*. If *remote_directory* is not specified, **dir** lists the contents of the current remote directory. If *local_file* is not specified or is a **-** (dash), **dir** displays the listing on the local terminal.

disconnect
See **close**.

form *format*
Specifies the form of the file transfer. The only form available is **file**.

get *remote_file* [*local_file*]
Copies the remote file to the local host. If *local_file* is not specified, the remote filename is used locally and is altered by any settings made by the **case**, **ntrans**, and **nmap** subcommands. The **ftp** command uses the current settings for **type**, **form**, **mode**, and **struct** while transferring the file. For additional information, refer to the description of each of these subcommands.

glob Toggles filename expansion (globbing) for **mdelete**, **mget**, and **mput**. If globbing is off, filename arguments for these subcommands are not expanded. When globbing is enabled and a pattern-matching character is used in a subcommand that expects a single filename, results may be different than expected. For example, the **append** and **put** subcommands perform filename expansion and then use only the first filename generated. Other **ftp** subcommands, such as **cd**, **delete**, **get**, **rename**, and **rmdir**, do not perform filename expansion and take the pattern-matching characters literally.

Globbing for the **mput** subcommand is done locally in the same way as for the **csh** command. For **mdelete** and **mget**, each filename is expanded separately at the remote machine and the lists are not merged. The expansion of a directory name may be different than the expansion of a filename, depending on the remote host and the **ftp** server.

To preview the expansion of a directory name, use the **mls** subcommand:

mls *remote_file* **-**

To transfer an entire directory subtree of files, transfer a **tar** archive of the subtree in binary form, rather than using **mget** or **mput**.

hash Toggles # (hash sign) printing. When **hash** is on, **ftp** displays one hash sign for each data block (1024 bytes) transferred.

help [*subcommand*]
Displays help information. Refer to the **?** subcommand.

lcd [*directory*]
Changes the working directory on the local host. If you do not specify a directory, **ftp** uses your home directory.

ls [*remote_directory*] [*local_file*]
See the **dir** subcommand.

macdef *macro*
Defines a subcommand macro. Subsequent lines up to a null line (two consecutive linefeeds) are saved as the text of the macro. Up to 16 macros containing at most 4096 bytes for all macros can be defined. Macros remain defined until redefined or a **close** is executed.

The special characters $ (dollar sign) and \ (backslash) have special uses in **ftp** macros. A **$** followed by one or more numbers is replaced by the corresponding macro parameter on the invocation line (refer to the **$** subcommand). A **$** followed by an **i** indicates that the macro is to loop, with **$i** being replaced by consecutive parameters on each pass. The first macro parameter is used on the first pass, the second parameter is used on the second pass, and so on. A \ prevents special treatment of the next character. Use the \ to turn off the special meanings of **$** and \.

mdelete *remote_files*
Expands *remote_files* and deletes the indicated remote files.

mdir [*remote_directory ... local_file*]
Expands *remote_directory* at the remote host and writes a listing of the contents of the *remote_directory* to the *local_file*. If the *remote_directory* argument contains a pattern-matching character, **mdir** prompts for a *local_file* if none is specified. If the *remote_directory* argument is a list of remote directories, separated by spaces, the last argument in the list must be either a local filename or a **-** (dash). If *local_file* is -, **mdir** displays the listing on the local terminal. If interactive prompting is on (refer to the **prompt** subcommand), **ftp** prompts you to verify that the last argument is a local file and not a remote directory.

mget *remote_file ...*
Expands *remote_files* at the remote host and copies the indicated remote files to the current directory on the local host. Refer to the **glob** subcommand for more information on filename expansion. The remote filenames are used locally and are altered by any settings made by the **case**, **ntrans**, and **nmap** subcommands. The **ftp** command uses the current settings for **type**, **form**, **mode**, and **structure** while transferring the files. Refer to the description of each of these subcommands for additional information.

mls [*remote_directory ... local_file*]
Expands *remote_directory* at the remote host and writes an abbreviated file listing of the indicated remote directories to a local file. If the *remote_directory* argument contains a pattern-matching character, **mls** prompts for a *local_file* if none is specified. If the *remote_directory* argument is a list of remote directories, separated by spaces, the last argument in the list must be either a local filename or a **-** (dash). If *local_file* is -, **mls** displays the listing on the local terminal. If interactive prompting is on (refer to the **prompt** subcommand), **ftp** prompts you to verify that the last argument is a local file and not a remote directory.

mode [*mode*]
Sets file transfer mode. The only mode available is **stream**.

modtime [*remote_file*]
Shows the last modification time of file *remote_file* on the remote machine.

mput [*local_file ...*]
Expands *local_file* at the local host and copies the indicated local files to the remote host. Refer to the **glob** subcommand for more information on filename expansion. The local filenames are used at the remote host and are altered by any settings made by the **ntrans** and **nmap** subcommands. The **ftp** command uses the current settings for **type**, **form**, **mode**, and **structure** while transferring the files. Refer to the description of each subcommand for additional information.

nmap [*inpattern outpattern*]
Sets or unsets the filename mapping mechanism. If no arguments are specified, filename mapping is turned off. If arguments are specified, source filenames are mapped for **mget** and **mput** operations and for **get** and **put** operations when the destination filename is not specified. This subcommand is useful when the local and remote hosts use different file naming conventions or practices. Mapping follows the pattern set by *inpattern* and *outpattern*.

The *inpattern* variable specifies the template for incoming filenames, which may have already been processed according to the **case** and **ntrans** settings. The template variables **$1** through **$9** can be included in *inpattern*. All characters in *inpattern* other than **$** and protected **$**s (that is, **\$**) define the values of the template variables. For example, if the *inpattern* is **$1.$2** and the remote filename is **mydata.dat**, the value of **$1** is **mydata** and the value of **$2** is **dat**.

The *outpattern* variable determines the resulting filename. The variables **$1** through **$9** are replaced by their values as derived from *inpattern* and the variable **$0** is replaced by the original filename. Additionally, the sequence [*sequence1*,*sequence2*] is replaced by the value of *sequence1* if *sequence1* is not null; otherwise, it is replaced by the value of *sequence2*. For example, the following subcommand would yield **myfile.data** from **myfile.data** or **myfile.data.old**, **myfile.file** from **myfile**, and **myfile.myfile** from **.myfile**:

nmap $1.$2.$3 [$1,$2].[$2,file]

Spaces can be included in *outpattern*. Use the \ (backslash) character to prevent the special meanings of **$**, **[**, **]**, and **,** (comma) in *outpattern*.

ntrans [*in_characters* [*out_characters*]]
Sets or unsets the filename character translation mechanism. If no arguments are specified, character translation is turned off. If arguments are specified, characters in source filenames are translated for **mget** and **mput** operations and for **get** and **put** operations when the destination filename is not specified. This subcommand is useful when the local and remote hosts use different file naming conventions or practices. Character translation follows the pattern set by *in_characters* and *out_characters*. Characters in a source filename matching characters in *in_characters* are replaced by the corresponding characters in *out_characters*. If the string *in_characters* is longer than the string *out_characters*, characters in *in_characters* are deleted if they have no corresponding character in *out_characters*.

open *host* [*port*]
Establishes a connection to the FTP server at the specified *host*. If the optional port number is specified, **ftp** will attempt to connect to a server at that port. If the autologin feature is set (that is, **-n** was not specified on the command line), **ftp** will attempt to automatically log you into the FTP server. You must also have a **$HOME/.netrc** file with the correct information in it and the correct permissions set.

prompt Toggles interactive prompting. If interactive prompting is on (the default), **ftp** will prompt for verification before retrieving, sending, or deleting multiple files during **mget**, **mput**, and **mdelete** operations. Otherwise, **ftp** will perform the operation on all files specified.

proxy [*subcommand*]
Executes an **ftp** command on a secondary control connection. This subcommand allows **ftp** to simultaneously connect to two remote FTP servers for transferring files between the two servers. To establish the secondary control connection, specify **open** as the first **proxy** subcommand. Enter the subcommand **proxy ?** to see the other **ftp** subcommands that are executable on the secondary connection. The following subcommands behave differently when prefaced by **proxy**:

- The **open** subcommand does not define new macros during the autologin process.
- The **close** subcommand does not erase existing macro definitions.
- The **get** and **mget** subcommands transfer files from the host on the primary connection to the host on the secondary connection.
- The **put**, **mput**, and **append** subcommands transfer files from the host on the secondary connection to the host on the primary connection.

File transfers require that the FTP server on the secondary connection support the **PASV** (passive) instruction.

put *local_file* [*remote_file*]
Stores a local file on the remote host. If you do not specify *remote_file*, **ftp** uses the local filename to name the remote file, and the remote filename is altered by any settings made by the **ntrans** and **nmap** subcommands. The **ftp** command uses the current settings for **type**, **form**, **mode**, and **structure** while transferring the files. Refer to the description of each subcommand for additional information.

pwd Displays the name of the current directory on the remote host.

quit Ends the file transfer session and exits **ftp**. A synonym for **bye**.

quote *string*
Sends the specified *string* verbatim to the remote host. Unpredictable results can occur when you quote commands that involve data transfers.

recv *remote_file* [*local_file*]
Copies the remote file to the local host. A synonym for **get**.

remotehelp [*subcommand*]
Requests help from the remote FTP server.

rename *from to*
Renames a file on the remote host.

reset Clears the reply queue. This command resynchronizes the command parsing.

restart *marker*
Restarts the immediately following **get** or **put** command at the indicated marker. On systems that treat files as unstructured byte arrays (such as OSF/1 and UNIX systems), marker is simply a byte offset into the file.

rmdir *remote_directory*
Removes the directory *remote_directory* at the remote host.

runique Toggles whether unique filenames are created for local destination files during **get** and **mget** operations. If creating unique local filenames is not enabled (the default), **ftp** overwrites local files. Otherwise, if a local file has the same name as specified for a local destination file, **ftp** modifies the specified name of the local destination file with **.1**. If a local file is already using the new name, **ftp** appends the postfix **.2** to the specified name. If a local file is already using this second name, **ftp** continues incrementing the postfix until it either finds a unique filename or reaches **.99** without finding a unique name. If **ftp** cannot find a unique name, **ftp** reports an error and the transfer does not take place. Note that **runique** does not affect local filenames generated from a shell command. When transferring a file ending in **.1** (or any other digit) to a /G Guardian file system, the period (**.**) is dropped from the resulting Guardian file name.

send *local_file* [*remote_file*]
Stores a local file on the remote host. A synonym for **put**.

sendport
Toggles the use of FTP **PORT** instructions. By default, **ftp** uses a **PORT** instruction when establishing a connection for each data transfer. When the use of **PORT** instructions is disabled, **ftp** does not use **PORT** instructions for data transfers.

size *filename*
Displays the size of the file, *filename*, on the remote system in bytes.

status Displays current status of **ftp**.

struct [*structure*]
Sets data transfer structure type. The only structure supported is **stream**.

sunique Toggles whether unique filenames are created for remote destination files during **put** and **mput** operations. If creating unique remote filenames is not enabled (the default), **ftp** overwrites remote files. Otherwise, if a remote file has the same name as specified for a remote destination file, the remote FTP server modifies the name of the remote destination file. Note that the remote server must support the **STOU** instruction.

trace Toggles packet tracing.

type [*type*]
Sets the file transfer type to *type*. If *type* is not specified, the current type is printed. The default type is **ascii**. Note that binary transfer (type **binary**) can be more efficient than ASCII transfer. File transfers to the Guardian (/G) file system will always arrive as Guardian file type 180.

user *user* [*password*] [*account*]
Identifies the local user as *user* to the remote FTP server. If *password* or *account* is not specified and the remote server requires it, **ftp** prompts for it locally. If *account* is required, **ftp** sends it to the remote server after the remote login process completes.

Note that, unless autologin is disabled by specifying **-n** on the command line, this process is done automatically for the initial connection to the remote server. You also need a **$HOME/.netrc** file in your home directory to issue an autologin.

verbose Toggles verbose mode. When verbose mode is on (the default), **ftp** displays all responses from the remote FTP server. Additionally, **ftp** displays statistics on all file transfers when the transfers are completed.

Aborting a File Transfer

To abort a file transfer, use the Interrupt key sequence (often <**Ctrl-c**>). Sending transfers will be immediately halted. Receiving transfers will be halted by sending an FTP protocol **ABOR** command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for **ABOR** processing. If the remote server does not support the **ABOR** command, the prompt `ftp>` does not appear until the remote server has completed sending the requested file.

The Interrupt key sequence is ignored when **ftp** has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the **ABOR** processing previously described, or from unexpected behavior by the remote server, including violations of the FTP protocol. If the delay results from unexpected remote server behavior, the local **ftp** program must be killed by hand.

EXAMPLES

1. This example shows how user **smith**, who is logged in on **host1**, can log in on the remote host **host2**, check the current working directory on **host2** and list its contents, transfer a file, and then end the session.

   $ **ftp host2**

   If the connection to **host2** is successful, a verification message is displayed on the local system:

   ```
   Connected to host2.abc.org
   220 host2 FTP server (Version 5.47 13 Mar 90 02:27) ready.
   Name (host2:smith): smith
   Password:
   ```

   Enter your name and password when prompted by the system. A message similar to the following is then displayed on your local system:

   ```
   230 User smith logged in

   ftp> _
   ```

   To set the file transfer type to binary, enter the **binary** subcommand after the `ftp>` prompt:

   `ftp>` **binary**

   A message similar to the following is displayed on your local system:

   ```
   200 Type set to I
   ```

To check the current working directory, enter the **pwd** command after the `ftp>` prompt:

```
ftp> pwd
```

A message similar to the following is displayed on your local system:

```
257 "u/smith" is current directory
```

To list the contents of the current working directory, enter the **ls** command after the `ftp>` prompt:

```
ftp> ls
```

A message similar to the following is displayed on your local system:

```
200 PORT command successful.
150 Opening data connection for /usr/bin/ls
        (555.5.55.555) (0 bytes)
printfile
testfile
226 Transfer complete.
```

(The `Opening data connection` message appears on one line, not on two lines as shown.)

To transfer a file from the remote host to the local host, enter the **get** or **mget** subcommand following the `ftp>` prompt:

```
ftp> get testfile tmp.testfile
```

A message similar to the following is displayed on your local system:

```
200 PORT command successful.
150 Opening data connection for testfile
        (555.5.55.555) (1201 bytes)
226 Transfer complete.
local:tmp.testfile remote:testfile
```

(The `Opening data connection` message appears on one line, not on two lines as shown.)

To end the **ftp** session, enter the **quit** subcommand after the `ftp>` prompt:

```
ftp> quit
221 Goodbye.
$ _
```

2. This example shows how user **smith**, who is logged in on **host1**, can log in as the user **smith** on the remote host **host2**:

```
$ ftp host2

Connected to host2.abc.org
220 host2 FTP server (Version 5.47 13 Mar 90 02:27) ready.
Name (host2:smith):  smith
```

```
331 Passwd required for smith
Password:
230 User smith logged in
ftp>
```

3. In this example, user **smith** makes a typing error:

   $ **ftp test**

```
Connected to test.abc.org
220 test FTP server (Version 5.47 13 Mar 90 02:27) ready.
Name (test:fred): msith
331 Passwd required for msith
Password:
530 User msith unknown
ftp> user smith
331 Passwd required for smith
Password:
230 User smith logged in
ftp>
```

4. In this example, user **fred** issues the **ftp** command without specifying a hostname, then connects to **host1** using the **open** subcommand:

   $ **ftp**
   ftp> **open host1**

```
Connected to host1.abc.org
220 host1 FTP server (Version 5.47 13 Mar 90 02:27) ready.
Name (host1:fred):  fred
331 Passwd required for fred
Password:
230 User fred logged in
ftp>
```

FILES

$HOME/.netrc	Contains automatic login information.

RELATED INFORMATION

Commands: **sh(1)**, **ftpserver(7)**, **stty(1)**.

Files: **netrc(4)**.

Section 4. User Commands (g - j)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letters **g** through **j**.

NAME

gencat - Creates and modifies a message catalog

SYNOPSIS

gencat *catalog_file* [*source_file* ...]

FLAGS

Operands

FLAGS

Operands

catalog_file is the name of a message catalog file. The naming convention for message catalog files uses the **.cat** extension.

source_file is a text file you create to hold messages printed by your program. The naming convention for message source files uses the **.msg** extension. You can use any text editor to enter messages into the text source file. Messages can be grouped into sets that represent general functional subsets of your program. Each message has a numeric identifier, which must be unique within its set. The message source file can also contain commands recognized by **gencat** for manipulating sets and individual messages.

DESCRIPTION

The **gencat** command can be used to create a message catalog (usually ending in **.cat**) from a message text source file (usually ending in **.msg**).

A message text source file is a text file that you create to hold messages printed by your program. Message source files usually have the .msg suffix. You can use any text editor to enter messages into the text source file. Messages can be grouped into sets, generally to represent functional subsets of your program. Each message has a numeric identifier, which must be unique within its set. The message text source file can also contain commands recognized by **gencat** for manipulating sets and individual messages.

The **gencat** utility does not recognize symbolic names for messages. If you use symbolic names rather than numeric constants to refer to messages, use the **mkcatdefs** utility to preprocess the *source_file*. The **mkcatdefs** utility accepts symbolic names and their associated messages; the output generated by the **mkcatdefs** utility is used as input to **gencat**.

If a message catalog with the name *catalog_file* exists, **gencat** modifies it according to the statements in the message source files. If it does not exist, **gencat** creates a catalog file with the name *catalog_file*.

You can specify any number of message text source files. The **gencat** command processes multiple source files one after the other in the sequence that you specify them. Each successive source file modifies the catalog. If you do not specify a source file, the **gencat** command accepts message source data from standard input. Note that you can specify a dash (-) for the catalog file (standard output) or the source file (standard input).

The **gencat** utility conforms to the XPG4 specification. In an XPG4-conforming application, set numbers must be integers in the range of 1 to **NL_SETMAX**, inclusive, and message numbers must be integers in the range of 1 to **NL_MSGMAX**, inclusive.

The *catalog_file* can contain the following commands. Each initial keyword or number must be followed by a space or a tab character. The **gencat** utility ignores any line beginning with a space, a tab, or a **$** (dollar sign) character followed by a space, a tab, or a newline character. Thus, you can use these sequences to start comments in your *catalog_file*. Blank lines are also ignored. Finally, you can place comments on the same line after the **$delset**, **$quote**, **$len**, or **$set** commands, because the **gencat** utility ignores anything following the preceding syntax elements.

message_number text
Inserts *text* as a message with the identifier *message_number*. There must be exactly one blank, space, or tab character between message number and text. Numbers must be ascending within each set, but need not be contiguous. If the message text is empty, and a space field separator is present, an empty string is stored in the message catalog. If a message source line has a message number and nothing else, the existing message associated with message number (if any) is deleted from the catalog. The length of text must be in the range 0 through **NL_TEXTMAX**.

$delset *set_number*
Deletes the set of messages indicated by *set_number*. You cannot use symbolic identifiers with the **$delset** command.

$quote *character*
Sets the quote character to *character*. See the explanation later in this section for more information. By default, or if the **$quote** command was last used with no argument, no quote character is defined.

$len [*max_length*]
Sets the maximum length allowed for messages in your catalogue. If this command is not used, or if you use it without the *max_length* argument, or if the specified value of *max_length* is not between 1 and the value of **NL_TEXTMAX**, inclusive, the maximum length defaults to the value of **NL_TEXTMAX**.

$set *set_number*
Indicates that all messages entered after this command are placed in the set indicated by *set_number*. You can change the set by entering another **$set** command. However, set numbers must be entered in ascending order; you can not go back to a lower-numbered set during the **gencat** session. If the command is not used, the default set number is 1.

A line beginning with a digit marks a message to be included in the catalog. The first blank (space or tab) character following the digit is the field separator. The rest of the line is considered to be message text unless the first character after the blank is the quote character. In this case the message text extends from just after the initial quote character to just before the next unescaped quotation mark character. The rest of the line is ignored.

Escape sequences, like those recognized by the C language, can be used in *text*; they are listed after the commands. Use a backslash (\) character to continue message text on the following line.

The **gencat** command does not accept symbolic identifiers. If you use symbolic identifiers you must use the **mkcatdefs** utility to preprocess a message source file and change symbolic identifiers into numeric constants.

The Escape character \ (backslash) can be used to include the following special characters in the message text:

\n Inserts a newline character.

\t Inserts a horizontal tab character.

\v Inserts a vertical tab.

\b Performs a backspace function.

\r Inserts a carriage return.

\f Inserts a formfeed character.

\\ Inserts a \ (backslash) character.

ddd Inserts the single-byte character associated with the octal value represented by the octal digits *ddd*. You can specify 1, 2, or 3 octal digits; however, you must include leading zeros if the characters following the octal digits are also valid octal digits. For example, the octal value for **$** (dollar sign) is 44. To insert **$5.00** into a message, use **\0445.00**, not **\445.00**, or the **5** will be parsed as part of the octal value.

\x*dddd* Inserts the single-byte or double-byte character associated with the hexadecimal value represented by the four valid hexadecimal digits *dddd*. You can specify either two or four digits. See the explanation of *ddd* for a way to avoid parsing errors when the hexadecimal value precedes an actual digit. This escape sequence is an Open System Services extension to the XPG4 specification.

You can also include **printf( )** conversion specifications in messages that are printed by the **printf( )** family of calls in C code. If you display a message from a shell script with the **dspmsg** command, the only conversion specifications that can be used in the message are **%s** and **%***n***$s**.

When you enter a number followed by a message, **gencat** removes the first space or tab character immediately following the number. Any spaces or tabs that follow the first space or tab are considered part of the message.

Environment Variables

The following environment variables affect the execution of the **gencat** utility: **LANG**, **LC_ALL**, **LC_TYPE, LC_MESSAGES**, **NLSPATH**.

EXAMPLES

1. To use the **$set** command in a source file to give a group of messages a set number, enter:

   $set 10 Communication Error Messages

   The message set number is **10**. All messages following the **$set** command are assigned that set number, up until the next occurrence of a **$set** command. (Set numbers must be assigned in ascending order, but need not be contiguous. Large gaps in the number sequence are discouraged in order to increase efficiency and performance. There is no performance advantage to using more than one set number in a catalog.)

   You can include a comment in the **$set** command, but it is not required.

2. To use the **$delset** command to remove all messages belonging to the specified set from a catalog, enter:

   $delset 10 Communication Error Messages

   The command set affected by the **$delset** command is specified by the number argument. **$delset** must be placed in the proper set number order with respect to any **$set** commands in the same message source file. You can include a comment in the **$delset** command.

3. To enter message text and assign message ID numbers, enter:

   12 file removed

   This assigns the message ID number **12** to the text that follows it.

   You must specify a single space or tab character between the message ID number and the message text, but you can include more spaces or tabs if you prefer. Any extra spaces or tabs included are treated as part of the message itself. Message numbers must be in ascending order within a single message set, but need not be contiguous.

   All text following the message number is included as message text, up to the end of the

line. If you place the escape character \ (backslash) as the last character on the line, the message text continues on the following line. Consider the following example:

**This is the text associated with **
message number 5.

The preceding two lines define this single-line message:

```
This is the text associated with message number 5.
```

4. The following example shows the use of the **$quote** command:

```
$quote "   Use a double quote to delimit message text
$set 10           Message Facility - Quote command messages
1 "Use the $quote command to define a character \
\n for delimiting message text" \n
2 "You can include the \"quote\" character in a message \n \
by placing a \ in front of it" \n
3 You can include the "quote" character in a message \n \
by beginning it with any other character  $quote
4 You can disable the quote mechanism by \n \
using the $quote command without \n a character \
after it \n
```

In this example, the **$quote** command defines the " (double quote) as the quote character. The quote character must immediately follow the message number and the single blank character. Any text following the next occurrence of the quote character is ignored.

The example shows two ways the quote character can be included in the message text:

- Place a single backslash (\) in front of the quote character.
- Begin the message text with another character, as in message 3. The quote character is not treated specially unless it is the first character after the blank.

The example shows the following:

- A backslash (\) is still required to split a quoted message across lines, as in messages 2, 3, and 4.
- To display a \ in a message, you must place another \ in front of it, as in message 1.
- You can format your message with a newline character by using **\n**, as in message 3.
- If you use the **$quote** command with no character argument, you disable the quote mechanism.

NOTES

The **$len** command is not described by X/OPEN. Other implementations of **gencat** may not recognize **$len**.

DIAGNOSTICS

The **gencat** utility generates these error messages:

```
Usage: gencat CatalogFile [SourceFile...]
gencat: Cannot open target file \%s.\n
gencat: The realloc system call failed.
gencat: The following message text is longer than the en
```

```
value:\n\t\%s\n
gencat: The set number in the following line is not valid:\n\t\%s\n
gencat: The length of the hex number in the following line is not
valid.\n\
It must be either two or four digits.\n\t\%s\n
gencat: Reached end of line before the defined closing quote.\n\t\%s\n
gencat: The following message string is longer than
NL_TEXTMAX:\n\t\%s\n
gencat: Reached end of string before expected.\n\t\%s\n
gencat: Internal error: The file pointer offset is not correct.
gencat: Cannot load the catalog file \%x.\n
gencat: Cannot the existing catalog file \%s.\n
gencat: There is not enough memory available now.
gencat: The following set uses a symbolic identifier:\n\t\%s\n
gencat: The following message uses a symbolic identifier:\n\t\%s\n
gencat: There is no message defined in a source file.
gencat: Set or message numbers are not in ascending sequence after:
\n\tMessage:    \%d, Set: \%d\n\t\%s
```

EXIT VALUES

The **gencat** utility returns a 0 (zero) on successful completion; otherwise it returns a 1 and generates a diagnostic message. When **gencat** reports an error, it takes no action on any commands and leaves an existing catalog unchanged.

RELATED INFORMATION

Commands: **mkcatdefs(1)**.

STANDARDS CONFORMANCE

The following are HP extensions to the XPG4 Version 2 specification:

- The escape sequence **x***dddd* is supported for compatability with OSF/1.

NAME

genxlt - Generates code-set translation table

SYNOPSIS

genxlt [**-f** *outputfile*]
[*inputfile*]

FLAGS

-f *outputfile*

Specifies that the generated version of the code-set conversion table be placed in the file specified by *outputfile*.

Operands

inputfile Specifies the code-set conversion table source file.

DESCRIPTION

The **genxlt** command reads a source code-set conversion table file from *inputfile* and writes the compiled version to *outputfile*.

If *inputfile* is not specified, standard input is used. If *outputfile* is not specified, standard output is used. The source code-set conversion table provided to **genxlt** as input contains directives that are acted upon by the utility to produce the compiled version.

The format of a code-set conversion table source file is:

- An optional **target_sub** directive in the form: **target_sub** *target* [*comment*]
- An optional **source_sub** directive in the form: **source_sub** *source* [*comment*]
- Conversion entry lines must be in the form: *source target* [*comment*]

The *source* field must be hexadecimal numbers. The *target* field must be hexadecimal numbers or the keyword is invalid. The *comment* field is optional. Lines whose initial non-space, non-tab character is the # (number sign) are treated as comment lines. Null lines and lines consisting only of white-space characters are also treated as comment lines.

The *source*, *target*, and *comment* fields can be separated by any combination of spaces and tabs. Comments can contain these characters because anything after the second separator is considered a comment. The maximum value of *source* is 65535 decimal. The maximum value of *target* is 4294967295 decimal. The hexadecimal representations of the values can be specified by anything acceptable to the **scanf() %x** conversion specification, from 1 to 8 hexadecimal digits. If a*source* value is found in the conversion table multiple times, the last entry is used in the compilation of the conversion table. If a *source* value is greater than 0xff (255 decimal), **genxlt** creates a conversion table for a double-byte input code set.

Include only valid and invalid conversions in the table. All values of *source* that are not present in the first column of the table are considered to be valid for input but have no corresponding *target* value. Specific values of *source* that are invalid for input must be declared invalid by including them in the *source* column with the string invalid in the *target* column on the same line.

Valid input characters that have no corresponding *target* value are converted to the *target* value specified by the **target_sub** directive. 0x1A, the ASCII SUB character, is the default if no **target_sub** is specified. Conversions of valid input characters to the **target_sub** character by the **iconv()** function are called non-identical conversions.

If the **target_sub** character has a corresponding character in the input code set, the source value for that character can be specified by the **source_sub** directive. 0x1A, the ASCII SUB character, is the default if no **source_sub** is specified. Specifying the **source_sub** character prevents conversions of **source_sub** characters to the **target_sub** character from being counted by **iconv()**

as non-identical conversions.

There is no requirement that the **target_sub** character for a conversion from a source code set to a target code set be the **source_sub** character in a table that specifies the inverse conversion.

Comments can contain any characters, but it is recommended that only characters in the ASCII code set be used. Using non-ASCII characters might cause **genxlt** to report an error if it finds characters that are invalid in the current locale.

EXAMPLES

The following is an excerpt of a code set conversion table:

```
#source_sub     0x1a    # SUB (default source_sub)
target_sub      0xffff  Replacement Character
0x0             0x00    NULL
0x1             0x01    SOH
0x001a          0x1A    SUB
0x80            0xc7    C cedilla
0x81            0xfc    u diaeresis
0x82            0xe9    e acute
0x83            0xe2    a circumflex
0x84            0xe4    a diaeresis
0x85            invalid not in use
0x86            0x40    a grave
0xa1a1          0xa2a3  special
0xa1a2          0xa2a4  special
0xa1a3          0xa2a5  special
0xa1a4          0xa2a6  special
```

In the preceding example, a comment serves only as a reminder that the default **source_sub** value is 0x1a. The **target_sub** is explicitly specified. The input values 0x1a and 0x001a are the same input value; input values can be padded with leading NULL bytes. The table indicates that the input character corresponding to the **source_sub** character is translated to the character SUB (0x1A), instead of to the **target_sub** character. It is not required that the **source_sub** character be translated to the **target_sub** character.

Note that the source value 0x85 in the preceding example is specifically declared invalid in the target column. Source values between 0x86 and 0xa1a1 are implicitly declared valid by their omission from the table. The omitted characters are converted to the **target_sub** character.

The name of the file generated by the **genxlt** utility must adhere to the following naming convention for the **iconv** subsystem to recognize it as a conversion-table file:

```
fromcode: "ISO8859-1-GL"
tocode:  "ISO8859-1"
conversion table file: "ISO8859-1-GL_ISO8859-1"
```

The code-set conversion table file name is formed by concatenating the *to_code* file name onto the *from_code* file name, with an underscore character between the two.

Input code sets representable by the tables are restricted to single-byte and double-byte code sets. It is not possible to represent a multibyte input code set, such as SJIS. Conversion of multibyte input code sets to other code sets can be done by the **iconv** command only, using **iconv** converters.

DIAGNOSTICS

The following error messages have an exit value of 1:

`Usage: genxlt [-f outputfile] [inputfile]`
An unknown flag was detected at the command line.

`genxlt: Unable to write to output file.`
A failure to write to the output file occurred.

`genxlt: Unable to open output file.`
`(file name): no such file or directory`
A failure to open the output file occurred.

`Failure to open the input file.`
`(file name): is a directory`
A directory name is supplied as the input file.

The following error messages have an exit value of 2:

`genxlt: Invalid format at line (line number)`
The input file has an invalid format.

`genxlt: There was no assignment for index (index number)`
The input file is missing a source value.

EXIT VALUES

If successful, the **genxlt** utility exits with a value of 0 (zero). If unsuccessful, the **genxlt** utility exits with a value greater than zero and writes an error message to the standard errror file.

RELATED INFORMATION

Commands: **iconv(1)**.

Functions: **iconv(3)**.

NAME

getacl - Lists access control lists (ACLs) for files

SYNOPSIS

getacl [-ad] *file* **...**

FLAGS

-a Displays the filename, owner, group, and any nondefault ACL entries for the file.

-d Displays the filename, owner, group, and any default ACL entries for the file. Only directories have default ACL entries.

If you do not specify any flags, the filename, owner, group, and both default and nondefault ACL entries are displayed.

DESCRIPTION

The **getacl** command displays the owner, group, and ACL entries for each *file* that is a directory, a regular file, a first-in, first-out (FIFO) special file, or a bound AF_UNIX socket.

When you specify multiple files, a blank line separates the ACL listing for each file. The format of a single ACL is:

 # file: *filename*
 # owner: *uid*
 # group: *gid*
user::*perm*
user:*uid***:***perm*
group::*perm*
group:*gid***:***perm*
class:*perm*
other:*perm*
default:user::*perm*
default:user:*uid***:***perm*
default:group::*perm*
default:group:*gid***:***perm*
default:class:*perm*
default:other:*perm*

The first three lines show the filename, the file owner, and the file-owning group. When you specify only the **-d** flag, and the file has no default ACL, only these three lines are displayed. Only directories have default ACL entries.

The **user** entry without a user ID indicates the permissions that are granted to the owner of the file. One or more additional **user** entries indicate the permissions that are granted to the specified users. The **group** entry without a group identifier indicates the permissions that are granted to the owning group of the file. One or more additional **group** entries indicate the permissions that are granted to the specified groups. The **other** entry indicates the permissions that are granted to others. The **class** entry provides a mask that you can use to restrict the permissions granted by additional **user** entries and any **group** entries.

The default entries (**default:user**, **default:group**, and **default:other**) can exist for directories only and contain ACL entries that are added to files and directories created within the directory. Default entries are added to new files as actual entries. Default entries are added to new directories both as actual entries and default entries.

The *uid* is the login name, *gid* is a group name, and *perm* is a three-character string of letters representing the separate discretionary access rights: **r** (read), **w** (write), **x** (execute/search), or the placeholder character **-** (dash). The value of *perm* is displayed in the order **rwx**. If a permission is not granted by an ACL entry, the placeholder character appears.

The **getacl** command displays ACL entries in the order in which the entries are evaluated when an access check is performed. Any default ACL entries for a directory have no effect on access checks.

The file owner (**user::**) permission bits represent the access that the owner of the file has. The file **class** permission bits represent the most access that any additional **user** entry, additional **group** entry, or the owning group entry can grant. The file **other** permission bits represent the access that the other ACL entry has. If a user invokes the **chmod** command or the **setacl** command and changes the file **class** permission bits, the access granted by the additional ACL entries might be restricted. For detailed information about ACLs, see the **acl(5)** reference page.

To indicate that the file group class permission bits restrict an ACL entry, **getacl** displays, after each affected entry, text in the form **#effective:***perm*, where *perm* shows only the permissions actually granted.

EXAMPLES

Given file **filea**, with an ACL six entries long, the command **getacl filea** displays:

```
# file: filea
# owner: fletcher
# group: us
user::rwx
user:spy:---
user:archer:rw-
group::r--
class:rw-
other:---
```

Given file **filea**, with an ACL six entries long, after the command **chmod 700 filea** was issued, the command **getacl filea** displays:

```
# file: filea
# owner: fletcher
# group: us
user::rwx
user:spy:---
user:archer:rw- #effective:---
group::r-- #effective:---
class:---
other:---
```

Given directory **fileb**, with an ACL containing default entries, the command **getacl -d fileb** displays:

```
# file: fileb
# owner: fletcher
# group: us
default:user::rwx
default:user:spy:---
default:group::r--
default:other:---
```

Given directory **fileb**, the command **getacl fileb** displays:

```
# file: fileb
# owner: fletcher
# group: us
user::rwx
```

```
user:spy:---
user:archer:rw-
group::r--
other:---
default:user::rwx
default:user:spy:---
default:group::r
```

NOTES

The output of the **getacl** command is in the correct format for input to the **setacl** command. If you direct the output from **getacl** to a file, you can use this file as input to **setacl**, allowing you to easily assign the ACL of one file to another file.

RELATED INFORMATION

Commands: **chmod(1)**, **ls(1)**, **setacl(1)**.

Functions: **acl(2)**, **aclsort(3)**, **getgrid(3)**, **getpwuid(3)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

This command is an HP extension to the XPG4 Version 2 specification.

NAME

getconf - Displays system configuration variable values

SYNOPSIS

getconf *system_configuration*

getconf *path_configuration pathname*

DESCRIPTION

The *system_configuration* argument specifies a system-wide configuration variable. The *path_configuration* argument specifies a system path-configuration variable. The *pathname* argument specifies a pathname for the *path_configuration* variable.

The *system_configuration* argument specifies system-configuration variables whose values are valid throughout the system. There are two kinds of system-wide configuration values:

- System-wide configuration variables
- System standards configuration variables

The *path_configuration* argument specifies system path-configuration variables whose values contain information about paths and the path structure in the system.

System-Wide Configuration Variables

System-wide configuration variables contain the minimum values met throughout all portions of the system. The following list defines the system-wide configuration variables used with the **getconf** command:

ARG_MAX
The maximum length, in bytes, of the arguments for one of the **exec** functions, including environment data. **atexit()** per process.

BC_BASE_MAX
The maximum value allowed for the **obase** variable with the **bc** command.

BC_DIM_MAX
The maximum number of elements permitted in an array by the **bc** command.

BC_SCALE_MAX
The maximum value allowed for the **scale** variable with the **bc** command.

BC_STRING_MAX
The maximum length of string constants accepted by the **bc** command.

CHILD_MAX
The maximum number of simultaneous processes for each real user ID.

CLK_TCK
The number of clock ticks per second. The value of **CLK_TCK** may be variable, and it should not be assumed that **CLK_TCK** is a compile-time constant.

COLL_WEIGHTS_MAX
The maximum number of weights that can be assigned to an entry in the **LC_COLLATE** locale-dependent information in a locale-definition file.

CS_PATH
A value for the **PATH** environment variable that finds all standard utilities.

EXPR_NEST_MAX
The maximum number of expressions that can be nested within parentheses by the **expr** command.

LINE_MAX
The maximum length, in bytes, of a command's input line (either standard input or another file) when the utility is described as processing text files. The length includes room for the trailing newline character.

NGROUPS_MAX
The maximum number of simultaneous supplementary group IDs for each process.

OPEN_MAX
The maximum number of files that one process can have open at one time.

PAGE_SIZE
The page size granularity for memory regions.

PASS_MAX
The maximum number of characters returned by **getpass()** (not including terminating null).

PATH A value for the **PATH** environment variable that finds all standard utilities.

RE_DUP_MAX
The maximum number of repeated occurrences of a regular expression permitted when using the interval-notation parameters, such as the **m** and **n** parameters, with the **ed** command.

STREAM_MAX
The number of streams that one process can have open at one time.

TZNAME_MAX
The maximum number of bytes supported for the name of a time zone (not the length of the **TZ** environmental variable).

System Standards Configuration Variables

System standards configuration variables contain the *minimum* values required by a particular system standard. The prefixes **POSIX_**, **POSIX2_**, and **XOPEN** indicate that the variable contains the minimum value for a system characteristic required by the POSIX 1003.1, POSIX 1003.2, and X/Open system standards, respectively. System standards are system-wide minimums that the system meets to support the particular system standard. Actual configuration values may exceed these standards. The system standards configuration variables for the **getconf** command are defined as follows:

POSIX_ARG_MAX
The length of the arguments for one of the **exec** functions, in bytes, including environment data.

POSIX_CHILD_MAX
The maximum number of simultaneous processes for each real user ID.

POSIX_JOB_CONTROL
This variable has a value of 1 if the system supports job control; otherwise, the variable is undefined.

POSIX_LINK_MAX
The maximum value of a file's link count.

POSIX_LOCALEDEF
This variable has a value of 1 if the system restricts supported locales to only those it supplies; otherwise, the variable has a value of 0 (zero).

POSIX_MAX_CANON
The maximum number of bytes in a terminal canonical input queue.

POSIX_MAX_INPUT
The maximum number of bytes for which space will be available in a terminal input queue.

POSIX_NAME_MAX
The maximum number of bytes in a filename.

POSIX_NGROUPS_MAX
The maximum number of simultaneous supplementary group IDs for each process.

POSIX_OPEN_MAX
The maximum number of files that one process can have open at one time.

POSIX_PATH_MAX
The maximum number of bytes in a pathname.

POSIX_PIPE_BUF
The maximum number of bytes that can be written atomically when writing to a pipe.

_POSIX_REENTRANT_FUNCTIONS
This variable has a value of 1 if the system supports POSIX reentrant functions; otherwise, the variable is undefined.

POSIX_SAVED_IDS
This variable has a value of 1 if each process has a saved set-user-ID and a saved set-group-ID; otherwise, the variable is undefined.

POSIX_SSIZE_MAX
The maximum value that can be stored in an object of type **ssize_t**.

POSIX_STREAM_MAX
The number of streams that one process can have open at one time.

_POSIX_THREAD_ATTR_STACKSIZE
This variable has a value of 1 if the system supports the POSIX threads stack size attribute; otherwise, the variable is undefined.

_POSIX_THREADS
This variable has a value of 1 if the system supports POSIX threads; otherwise, the variable is undefined.

POSIX_TZNAME_MAX
The maximum number of bytes supported for the name of a time zone (not the length of the **TZ** environmental variable).

POSIX_VERSION
The date of approval of the most current version of the POSIX 1 standard that the system supports. The date is a 6-digit number, with the first 4 digits signifying the year and the last 2 digits the month. Different versions of the POSIX 1 standard are periodically approved by the IEEE Standards Board, and the date of approval is used to distinguish between different versions.

POSIX2_BC_BASE_MAX
The maximum value allowed for the **obase** variable with the **bc** command.

POSIX2_BC_DIM_MAX
The maximum number of elements permitted in an array by the **bc** command.

POSIX2_BC_SCALE_MAX
The maximum value allowed for the **scale** variable with the **bc** command.

POSIX2_BC_STRING_MAX
The maximum length string constants accepted by the **bc** command.

POSIX2_CHAR_TERM
One or more terminal types capable of all operations described in ISO/IEC 9945. This value need not be present on a system not supporting the User Portability Utilities Option.

POSIX2_COLL_WEIGHTS_MAX
The maximum number of weights that can be assigned to an entry of the **LC_COLLATE** locale variable in a locale-definition file.

POSIX2_EXPR_NEST_MAX
The maximum number of expressions that can be nested within parentheses by the **expr** command.

POSIX2_LINE_MAX
The maximum length, in bytes, of a command's input line (either standard input or another file) when the utility is described as processing text files. The length includes room for the trailing newline character.

POSIX2_RE_DUP_MAX
The maximum number of repeated occurrences of a regular expression permitted when using the interval-notation parameters, such as the **m** and **n** parameters with the **ed** command.

POSIX2_UPE
This variable has a value of 1 if the system supports the User Portability Utilities Option; otherwise, the variable has a value of 0 (zero).

POSIX2_VERSION
The date of approval of the most current version of the POSIX 2 standard that the system supports. The date is a 6-digit number, with the first 4 digits signifying the year and the last 2 digits the month. Different versions of the POSIX 2 standard are periodically approved by the IEEE Standards Board, and the date of approval is used to distinguish between different versions.

POSIX2_C_BIND
This variable has a value of 1 if the system supports the optional C Language Development Facilities specified by POSIX 2 and the optional C Language Bindings Option from POSIX 2; otherwise, the variable is undefined.

POSIX2_C_DEV
This variable has a value of 1 if the system supports the optional C Language Development Utilities from POSIX 2; otherwise, the variable is undefined.

POSIX2_C_VERSION
The integer value 199209L. This value indicates the version of the interfaces described in the C-Language Bindings Option section of the XPG4 standard. This value changes with each published version of ISO/IEC 9945 to indicate the 4-digit year and 2-digit month that the standard was approved by the IEEE Standards Board.

POSIX2_FORT_DEV
This variable has a value of 1 if the system supports the FORTRAN Development Utilities Option from POSIX 2; otherwise, the variable is undefined.

POSIX2_FORT_RUN
This variable has a value of 1 if the system supports the FORTRAN Runtime Utilities Option from POSIX 2; otherwise, the variable is undefined.

POSIX2_LOCALEDEF
This variable has a value of 1 if the system supports the creation of new locales with the **localedef** command; otherwise, the variable is undefined.

POSIX2_SW_DEV
This variable has a value of 1 if the system supports the Software Development Utilities Option from POSIX 2; otherwise, the variable is undefined.

SSIZE_MAX
The maximum value that can be stored in an object of type **ssize_t**.

XOPEN_VERSION
An integer indicating the most current version of the X/OPEN standard that the system supports.

System Path Configuration Variables

LINK_MAX
The maximum value of a file's link count. If the *pathname* argument refers to a directory, the value returned applies to the directory itself.

MAX_CANON
The maximum number of bytes in a terminal canonical input queue. If the *pathname* argument does not specify a terminal file, the **getconf** command exits with a nonzero value.

MAX_INPUT
The maximum number of bytes for which space will be available in a terminal input queue. If the *pathname* argument does not specify a terminal file, the **getconf** command exits with a nonzero value.

NAME_MAX
The maximum number of bytes in a filename. If the *pathname* argument specifies a directory, the value returned applies to the filenames within the directory.

PATH_MAX
The maximum number of bytes in a pathname. If the *pathname* argument specifies a directory, the value returned is the maximum length of a relative pathname when the specified directory is the working directory.

PIPE_BUF
The maximum number of bytes that can be written atomically when writing to a pipe. If the *pathname* argument specifies a FIFO or a pipe, the value returned applies to that object. If the *pathname* argument specifies a directory, the value returned applies to any FIFO created in that directory. If the *pathname* argument does not specify a directory or a FIFO file, the **getconf** command exits with a nonzero value.

POSIX_CHOWN_RESTRICTED
This variable has a value of 1 when the use of the **chown** function is restricted to a process with appropriate privileges and the group ID of a file can only be changed to the effective group ID of the process or to one of its supplementary group IDs. If the variable is undefined, it varies in the system, depending upon the path.

POSIX_NO_TRUNC
This variable has a value of 1 when pathnames longer than the limit specified by the **NAME_MAX** variable will generate an error. If the variable is undefined, it varies in the system, depending upon the path.

POSIX_VDISABLE
When this variable has a value of 1, terminal special characters, which are defined in the **termios.h** header file, can be disabled. If the *pathname* argument does not specify a terminal file, the **getconf** command will exit with a nonzero value.

EXAMPLES

1. To display the value of the **ARG_MAX** environment variable, enter:

   ```
   getconf ARG_MAX
   ```

2. To display the value of the **PATH_MAX** environment variable for the **/usr** directory and check the exit codes for the command, enter the following sequence of shell commands:

   ```
   value=$ (getconf PATH_MAX /usr)
   status=$?
   if [ "X$value" = "x" ]
   then
           case $status in
           0)      echo PATH_MAX is NULL; ;
           1)      echo PATH_MAX in /usr not defined; assume infinity; ;
           *)      echo Error in the getconf command; ;
           esac
   else
           echo The value of PATH_MAX in /usr is $value
   fi
   ```

   This sequence returns the following message:

   ```
   The value of PATH_MAX in /usr is 1024
   ```

FILES

/usr/include/limits.h	Defines system configuration variables.
/usr/include/unistd.h	Defines system configuration variables.

RELATED INFORMATION

Commands: **env(1)**.

Functions: **confstr(3)**, **pathconf(3)**, **sysconf(3)**.

Files: **limits(4)**.

NAME

getfilepriv - Displays file privileges for an executable file

SYNOPSIS

getfilepriv *file* **...**

FLAGS

Operands

file Specifies the name of a file for which you want to display privileges.

DESCRIPTION

The **getfilepriv** command displays the file privileges for the specified file. This file can be either a Guardian file or an OSS file.

For a description of the values for file privileges, see the **setfilepriv(1)** reference page.

EXAMPLES

1. To list the privileges for file **/user/privexe**, enter:

   getfilepriv /user/priexe

   Sample output:

   #file: /usr/privexe
   PRIVSETID

2. To list the privileges for the Guardian file **$system.system.privobj**, enter:

   getfilepriv /G/SYSTEM/SYSTEM/PRIVOBJ

   Sample output:

   #file: /G/SYSTEM/SYSTEM/PRIVOBJ
   PRIVSETID
   PRIVSOARFOPEN

3. To list the privileges for all the files in the current directory, enter:

   getfilepriv *

   Sample output:

   #file: exe1_with_privs
   PRIVSETID
   PRIVSOARFOPEN

   #file: exe2_no_priv
   NONE

   #file: exe2_with_priv
   PRIVSETID

   #file: not_executable
   NONE

   #file: exe3_with_priv
   PRIVSETID

NOTES

You can use the output of the **getfilepriv** command used on a single file, saved as a file, as the **priv_file** for the **setprivfile** command. Lines that begin with **#** (number sign) in the output of the **getfilepriv** command are treated as comments in the input of the **setfilepriv** command.

This command is supported on systems running J06.11 or later J-series RVUs or H06.22 or later H-series RVUs only.

RELATED INFORMATION

Commands: **setfilepriv(1)**.

Functions: **setfilepriv(2)**, **stat(2)**.

STANDARDS CONFORMANCE

This command is an HP extension.

NAME

getopts - Parses command options

SYNOPSIS

getopts *optstring name* [*argument* ...]

DESCRIPTION

The **getopts** command checks a specified command for legal options.

Operands

optstring Specifies the letters that the **getopts** command will recognize as valid option values when parsing the command options. If a letter is followed by a **:** (colon), the option is expected to have an argument specified in *argument*. The options can be separated from the argument by spaces.

name Specifies the name of a shell environment variable into which the **getopts** command should place the value of the next option.

argument Specifies an option argument for the **getopts** command to parse. If *argument* is omitted, positional parameters are parsed.

An option argument begins with a **+** (plus sign) or a **-** (dash). Either an option not beginning with **+** or **-** or the argument **--** ends the options.

Each time it is invoked, **getopts** places the next option letter it finds into the variable *name*. The value stored has a **+** added in front when *argument* begins with a **+**.

The index of the next *argument* is stored in **OPTIND**. **OPTIND** is initialized to **1** when the shell is invoked.

The option argument, if any, gets stored in **OPTARG**.

If a required option argument is missing, a leading **:** in *optstring* causes **getopts** to store the letter of an invalid option in **OPTARG** and to set *name* to **:** (colon). Otherwise, *name* is set to a **?** (question mark), the shell variable **OPTARG** is not set, and **getopts** prints an error message. The exit status remains 0 (zero).

The exit status is nonzero when there are no more options.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **NLSPATH**, **OPTARG**, and **OPTIND** environment variables.

EXAMPLES

The following section from a shell script takes the command line passed from the invoking process and extracts the flag characters. There are two expected flags: the **-a** flag and the **-b** flag. The **-b** flag requires an argument. At the end, the script echoes the remaining arguments not extracted by the **getopts** command.

```
aflag=
bflag=
while getopts ab: name
do
        case $name in
```

```
 a)     aflag=1
        b)      bflag=1
                bval-"$OPTARG";;
        ?)      echo Usage: $0 [-a] [-b value] parameters
                exit 2;;
        esac
done
if [ ! -z "$bflag" ]; then echo Option -a specified; fi
if [ ! -z "$bflag" ]; then echo Option -b "bval" specified; fi
shift $[OPTIND - 1]
echo Remaining parameters are: "$*"
```

NOTES

The **getopts** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the **sh(1)** reference page.

EXIT VALUES

The **getopts** command returns the following values:

0 (zero)	The command completed successfully.
>0	An error occurred or there are no more options.

RELATED INFORMATION

Commands: **sh(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

gname - Displays the Guardian environment filename for an OSS file

SYNOPSIS

gname [**-s**] *pathname* ...

FLAGS

-s Supresses formatting and displays only the Guardian filename.

DESCRIPTION

The **gname** command displays the Guardian filename for the file specified by *pathname*.

Operands

pathname
Specifies the OSS pathname for the file whose Guardian filename is to be displayed.

EXAMPLES

1. The command:

   gname /bin/gname

   results in output such as the following:

   ```
   gname: /bin/gname ---> \KT22.$XPG.ZYQ00000.Z00005LS
   ```

2. The command:

   gname -s /bin/pname

   results in output such as the following:

   ```
   \KT22.$XPG.ZYQ00000.Z00005LN
   ```

DIAGNOSTICS

The following error messages can be returned:

`A prefix within the pathname refers to a file other than the dir`
The value specified for *pathname* specifies a file where a directory name is required.

`Failed with Guardian error: 4002`
The value specified for *pathname* does not correspond to a known file.

`Invalid pathname specified:` *pathname*
The value specified for *pathname* contains a character that is not permitted in a pathname.

`Pathname or name component too long.`
Either a component of the value specified for *pathname* exceeds the maximum number of characters allowed in that component, or symbolic links in the specified pathname expand until the maximum length for a valid pathname is exceeded.

`Root fileset not mounted.`
The value specified for *pathname* uses a root fileset that is not currently mounted. If the specified value is valid, contact your site administrator to have that root fileset mounted.

`Some non-root fileset not mounted.`
The value specified for *pathname* uses a directory for a fileset that is not currently mounted. If the specified value is valid, contact your site administrator to have that fileset mounted.

EXIT VALUES

The **gname** command returns the following exit values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **pname(1)**.

Miscellaneous: **filename(5)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

grep - Search a file for a pattern

SYNOPSIS

grep [**-E** | **-F**] [**-c** | **-l** | **-q**] [**-bhinsvwxy**] [**-p***paragraph_separator* ...]
{*pattern* ... | **-e** *pattern* ... | **-f** *pattern_file* ...} [*file* ...]

FLAGS

While most flags can be combined, some combinations result in one flag overriding another. For example, if you specify the **-n** and **-l** flags, the output includes filenames only (as specified by the **-l** flag) and thus does not include line numbers (as specified by the **-n** flag).

-b Precedes each line by the block number in which it was found. Use this flag to help find disk block numbers by context.

-c Displays only a count of matching lines.

-e *pattern ...*
Specifies a *pattern*. This flag works the same as a simple *pattern* but is useful when the pattern begins with a **-** (dash).

-E Uses extended regular expressions (EREs) to match patterns (treats each pattern as an ERE). A null ERE matches every line.

-f *pattern_file ...*
Specifies a file that contains patterns. Each pattern terminates with a newline character.

-F Uses fixed strings to match patterns (treats each pattern as a literal string instead of as a regular expression). A null string matches every line.

-h Suppresses reporting of filenames when multiple files are processed.

-i Ignores the case of letters in locating *pattern*; that is, uppercase and lowercase letters in the input are considered identical (same as the **-y** flag).

-l Lists the name of each file with lines matching *pattern*. Each filename is listed only once; filenames are separated by newline characters.

-n Precedes each line with its relative line number in the file.

-p*paragraph_separator ...*
Displays the entire paragraph containing matched lines. Paragraphs are delimited by paragraph separators specified by *paragraph_separator*, which is a BRE pattern. Lines containing the paragraph separators are used only as separators; they are never included in the output. The default paragraph separator is a blank line. No space is permitted between the flag and the paragraph separator.

-q Suppresses all output except error messages. This is useful for easily determining whether or not a pattern or string exists in a group of files. When searching several files, it provides a performance improvement because it can quit as soon as it finds the first match, and it requires less care by the user in choosing the set of files to supply as arguments because it exits with a 0 (zero) exit status if it detects a match, even if the **grep** command detected an access or read error on earlier file

arguments.

-s Suppresses error messages about inaccessible files.

-v Displays all lines except those that match the specified pattern. This flag is useful for filtering unwanted lines out of a file.

-w Searches for the expression as a word (the pattern bracketed by nonalphanumeric characters or by the beginning or end of the line). See the reference page for the **ex** command.

-x Displays lines that match the pattern exactly with no additional characters.

-y Ignores the case of letters in locating *pattern*; that is, uppercase and lowercase letters in the input are considered to be identical (same as the **-i** flag).

DESCRIPTION

The **grep** command searches the specified files (the standard input file by default) for lines containing characters that match the specified pattern and then write matching lines to standard output.

The **grep** command searches for patterns that are limited regular expressions as described under **Regular Expressions (REs)**.

Command Usage

The **grep** command precedes the matched line with the name of the file containing it if you specify more than one file (except when the **-h** flag is specified).

Lines are limited to 2048 bytes; longer lines are broken into multiple lines of 2048 or fewer bytes. Paragraphs (under the **-p** flag) are currently limited to a length of 5000 bytes.

Running the **grep** command on a nontext file (for example, an **.o** file) produces unpredictable results and is discouraged.

Regular Expressions (REs)

Regular expressions (REs) cannot contain newline characters, because these signal a new pattern. The following REs match a single character:

character
An ordinary character (one other than one of the special pattern-matching characters) matches itself.

. A **.** (dot) matches any single character except the newline character.

[*string***]** A string enclosed in **[ ]** (brackets) matches any one character in that string. In addition, certain pattern-matching characters have special meanings within brackets:

^ If the first character of *string* is a ^ (circumflex), the RE **[^***string***]** matches any character *except* the characters in *string* and the newline character. A ^ has this special meaning only if it occurs first in the string.

- You can use a **-** (dash) to indicate a range of consecutive characters. The characters that fall within a range are determined by the current collating sequence, which is defined by the **LC_COLLATE** environment variable. For example, **[a-d]** is equivalent to **[abcd]** in the traditional ASCII collating

sequence.

A range can include a multicharacter collating element enclosed within bracket-period delimiters ([. .]). The bracket-period delimiters in the RE syntax distinguish multicharacter collating elements from a list of the individual characters that make up the element.

A collating sequence can define equivalence classes for characters. An equivalence class is a set of collating elements that all sort to the same primary location. They are enclosed within bracket-equal delimiters (**[= =]**). An equivalence class generally is designed to deal with primary-secondary sorting. For example, if **e**, **è**, and **ê** belong to the same equivalence class, then **[[=e=]fg**, **[[=è=]fg]**, and **[[=ê=]fg]** are each equivalent to **[eèêfg]**.

The **-** (dash) character loses its special meaning if it occurs first (**[-***string***]**), if it immediately follows an initial circumflex (**[^-***string***]**), or if it appears last (**[***string***-]**) in the string.

] When the **]** (right bracket) is the first character in the string (**[]***string***]**) or when it immediately follows an initial circumflex (**[^]***string***]**), it is treated as a part of the string rather than as the string terminator.

special_character

A \ (backslash) followed by a special pattern-matching character matches the special character itself (as a literal character). These special pattern-matching characters are as follows:

. * [\ Always special, except when they appear within **[ ]** (brackets).

^ Special at the beginning of an entire pattern or when it immediately follows the left bracket of a pair of brackets (**[^...]**).

$ Special at the end of an entire pattern.

[: :] A character class name enclosed in bracket-colon delimiters matches any of the set of characters in the named class. Members of each of the sets are determined by the current setting of the **LC_CTYPE** environment variable. The supported classes are **alpha, upper, lower, digit, xdigit, space, print, punct, graph, and cntrl.**

Here is an example of how to specify one of these classes:

[[:lower:]]

This matches any lowercase character for the current locale.

Forming Patterns

The following rules describe how to form patterns from REs:

- An RE that consists of a single, ordinary character matches that same character in a string.

- An RE followed by an ***** (asterisk) matches zero or more occurrences of the character that the RE matches. For example, the following pattern:

  ab*cd

  matches each of the following strings:

  acd
  abcd
  abbcd
  abbbcd

  but not the following string:

  abd

  If there is any choice, the longest matching leftmost string is chosen. For example, given the following string:

  122333444

  the pattern **.*** matches **122333444**, the pattern **.*3** matches **122333**, and the pattern **.*2** matches **122**.

- An RE followed by:

  \{*number***\}**
  Matches exactly *number* occurrences of the character matched by the RE.

  \{*number***,\}**
  Matches at least *number* occurrences of the character matched by the RE.

  \{*number1***,***number2***\}**
  Matches any number of occurrences of the character matched by the RE from *number1* to *number2*, inclusive.

  The values of *number1* and *number2* must be integers in the range 0 through 255, inclusive. Whenever a choice exists, this pattern matches as many occurrences as possible.

  Note that if *number* is 0 (zero), *pattern* matches zero occurrences of *pattern*; for example:

  ```
  $ echo abc | grep 'aX\{0\}bX\{0\}cX\{0\}'
  abc
  $
  ```

- You can combine REs into patterns that match strings containing the same sequence of characters. For example, **AB*CD** matches the string **ABCD** and **[A-Za-z]*[0-9]*** matches any string that contains any combination of ASCII alphabetic characters (including none), followed by any combination of numerals (including none).

- The character sequence **\(***pattern***\)** matches *pattern* and saves it into a numbered holding space. Using this sequence, up to nine patterns can be saved on a line. Counting from left to right on the line, the first pattern saved is placed in the first holding space, the second pattern is placed in the second holding space, and so on.

The character sequence *n* matches the *n*th saved pattern, which is placed in the *n*th holding space. (The value of *n* is a digit, 1-9.) Thus, the following pattern:

\(A\)\(B\)C\2\1

matches the string **ABCBA**. You can nest patterns to be saved in holding spaces. Whether the enclosed patterns are nested or in a series, *n* refers to the *n*th occurrence, counting from the left, of the delimiting characters, **\)**.

Restricting What Patterns Match

A pattern can be restricted to match from the beginning of a line, up to the end of the line, or the entire line:

- A ^ (circumflex) at the beginning of a pattern causes the pattern to match only a string that begins in the first character position on a line.
- A **$** (dollar sign) at the end of a pattern causes that pattern to match only if the last matched character is the last character (not including the newline character) on a line.
- The construction **^***pattern***$** restricts the pattern to matching only an entire line.

EXAMPLES

1. To search several files for a string of characters, enter:

   grep -F 'strcpy' *.c

   This command searches for the string **strcpy** in all files in the current directory with names ending in **.c**.

2. To count the number of lines that match a pattern, enter:

   grep -c -F '{' pgm.c
   grep -c -F '}' pgm.c

   This command displays the number of lines in **pgm.c** that contain left and right braces.

   If you do not put more than one **{** or **}** on a line in your C programs, and if the braces are properly balanced, then the two numbers displayed will be the same. If the numbers are not the same, then you can display the lines that contain braces in the order that they occur in the file with the command:

   grep -n -E '{|}' pgm.c

3. To display all lines in a file that begin with an ASCII letter, enter:

   grep '^[a-zA-Z]' pgm.s

   Note that because the command **grep -F** searches only for fixed strings and does not interpret pattern-matching characters, the following command searches only for the literal string **^[a-zA-Z]** in file **pgm.s**:

   grep -F '^[a-zA-Z]' pgm.s

   A space character is required between the **-F** flag and the literal string specification.

4. To display all lines that contain ASCII letters in parentheses or digits in parentheses (with spaces optionally preceding and following the letters or digits), but *not* letter-digit combinations in parentheses, enter:

   **grep -E **
   '\(*([a-zA-Z]* | [0-9]*) *\)' my.txt

   This command displays lines file in **my.txt** such as **(y)** or **(783902)**, but not **(alpha19c)**.

   Note that with the command **grep -E**, **\(** and **\)** match parentheses in the text and **(** and **)** are special characters that group parts of the pattern. With the **grep** command without the **-E** flag, the reverse is true; use **(** and **)** to match parentheses and **\(** and **\)** to group characters.

5. To display all lines that do *not* match a pattern, enter:

   grep -v '^#'

   This displays all lines that do not begin with a **#** (number sign).

6. To display the names of files that contain a pattern, enter:

   grep -l -F 'rose' *.list

   This command searches the files in the current directory whose names end with **.list** and displays the names of those files that contain at least one line containing the string **rose**.

   A space character is required between the **-F** flag and the literal string specification.

7. To display all lines that contain uppercase characters, enter:

   grep '[[:upper:]]' pgm.s

EXIT VALUES

The exit values of the **grep** command are as follows:

0	A match was found.
1	No match was found.
2	A syntax error was found or a file was inaccessible, even if matches were found.

RELATED INFORMATION

Commands: **ed(1), sed(1), sh(1).**

Files: locale(4).

NAME

gtacl - Runs a process in the Guardian environment from the OSS environment

SYNOPSIS

gtacl [*option* ...] [*operands*]

FLAGS

Operands used with the **gtacl** command must follow **gtacl** option specifications.

Options

All filename and pathname arguments used with **gtacl** options must be specified using OSS pathname syntax. In the current release, filenames and pathnames within the **/E** directory are not supported by the OSS file system. These specifications allow the user to identify Guardian environment objects on other HP nodes.

The specification of options ends when **gtacl** finds one of the following:

- The end of the input line
- Any operand; that is, any specification that does not start with a minus (-) or plus (+) and is not a redirection specification
- -- option
- **-p** option

Options cannot be grouped after a single minus (-) or plus (+). That is, **-debugnowait** is not a valid option specification.

The **gtacl** options can be specified in any order; **gtacl** processes options from left to right. When you specify a **gtacl** option more than once, the rightmost specification on the command line determines the value used.

-? | -help Displays usage information for the **gtacl** command. If this option is specified, all other **gtacl** options and operands on the command line are ignored.

-c *command* | **-cv** *command*

Submits *command* to the child process through the Guardian environment STDIN file of the child process.

The *command* string must conform to OSS **/bin/sh** quoting conventions. Refer to the **sh**(1) reference page for a description of those conventions.

When you specify the **-c** *command* option, **gtacl** suppresses all input and output operations except for those that occur after *command* is read and before the end-of-file (EOF) indication is returned. Use this option when you do not want Guardian program banner information or similar output that is not data generated by running *command*.

When you specify the **-cv** *command* option, **gtacl** displays all output, regardless of whether it occurs after *command* is read and before the end-of-file (EOF) indication is returned. Use this option when you do not want to discard Guardian program banner information or similar output that is not data generated by running *command*.

If you specify either of these options, the OSS environment

standard input file for **gtacl** is not connected to the Guardian environment standard input file of the child process. Refer to **Input/Output Filtering** under **NOTES** later in this reference page for more information.

IMPORTANT: Do not use the **-c** option with the **-i** or **+i** option. However, if the **-c** option is used with **-i** or **+i**, the last option specified on the command line takes effect.

-cpu *n*
Specifies the processor (0-15) in which the child process is to run.

The default processor is the processor in which **gtacl** is running, unless the child process is being created on another HP node. When the child process runs on another HP node, the operating system on that node assigns the default processor.

-debug
Enters a Guardian environment debugging tool at the first executable instruction of the child process. The default action is to start the child process without a debugging tool active.

Refer to the manual for the debugger in use for additional information.

-defmode on | -defmode off
Specifies the Guardian DEFINE mode for the child process.

If you use **-defmode on**, all Guardian DEFINE values of the process executing **gtacl** are inherited by the child process.

If you use **-defmode off**, only Guardian =_DEFAULTS DEFINE values are inherited by the child process.

The default mode is the DEFINE mode in effect for **gtacl**.

Refer to the *TACL Reference Manual* for additional information about Guardian DEFINEs.

-extswap *pathname*
Specifies a Guardian swap file or swap volume for the extended data segment of the child process. (Used for G-series TNS or accelerated child processes only.) The *pathname* must be specified in OSS pathname syntax.

The default action is to use the swap volume specified in the =_DEFAULTS DEFINE for the child process. If that volume is not available, the operating system chooses a swap volume.

Refer to the *Guardian Programmer's Guide* for additional information about swap files.

-f *pathname*
Specifies a text file containing the environment variables to be passed to the TACL process as PARAMs.

The environment variables must be listed in *name=value* format, with each pair on a new line; white space is ignored before the equal sign. Variable definitions that follow this format are passed, one PARAM for each definition, to the Guardian TACL process. Environment variable definitions that do not follow this format are ignored.

If the **-f** flag is omitted, **gtacl** attempts to pass all existing

environment variables unless the **-s** flag is used. If both the **-f** and **-s** flags are specified, the **-f** option is ignored. If the **-f** flag is specified more than once, only the rightmost specification in the command line is used.

-gpri *n*
Assigns the initial Guardian execution priority *n* (1-199) to the child process. The value 1 is the lowest priority; the value 199 is the highest priority.

The default priority is the priority used for **gtacl**.

-highpin on | -highpin off
Specifies whether the child process can run with a Guardian process identification number (PIN) greater than 255.

Specifying **-highpin on** means that the child process can run with a PIN greater than 255. Specifying **-highpin off** means the child process must run with a PIN between 0 and 254.

The default value is **-highpin on**, unless an OFF value for the Guardian process attribute is inherited from **gtacl**.

Refer to the *TACL Reference Manual* for additional information about high and low PINs.

-i
Specifies that **gtacl** should do the following:

- Send the child process the **gtacl** Guardian process name qualifier to use as both its Guardian environment IN and OUT filenames. (Many Guardian processes operate in an interactive mode when their Guardian environment IN and OUT filenames are the same.)
- Attempt to filter data passing through all OSS environment standard input, output, and error files. Refer to **Input/Output Filtering** under **NOTES** later in this reference page for more information.

The default action is to filter only the standard input and output files that are connected to an OSS regular file, device file, pipe, or FIFO that cannot be accessed through a Guardian environment filename.

IMPORTANT: Do not use the **-i** option with the **-c** option. However, if the **-i** option is used with **-c**, the last option specified on the command line takes effect.

+i
Specifies that **gtacl** should do the following:

- Send the child process different Guardian process name qualifiers for the Guardian environment IN and OUT filenames. (Many Guardian processes operate in a noninteractive mode when their Guardian environment IN and OUT filenames are different.)

- Attempt to filter data passing through all OSS environment standard input, output, and error files. Refer to **Input/Output Filtering** under **NOTES** later in this reference page for more information.

The default action is to filter only the standard input and output files that are connected to an OSS regular file, device file, pipe, or FIFO that cannot be accessed through a Guardian environment filename.

IMPORTANT: Do not use the **+i** option with the **-c** option. However, if the **+i** option is used with **-c**, the last option specified on the command line takes effect.

-inspect on | -inspect off | -inspect saveabend
Indicates the debugging mode to be used for the child process.

Specifying **inspect on** causes the child process to enter the currently selected symbolic debugger when the **-debug** option is specified or if a debug event occurs.

Specifying **-inspect off** causes the child process to enter the default debugger when the **-debug** option is specified or if a debug event occurs.

Specifying **-inspect saveabend** causes creation of a saveabend file (sometimes called a core file or a process snapshot file) when the process terminates abnormally.

The default debugging mode is the mode specified in the program file of the child process.

Refer to the manual for the debugging tool in use for additional information about debug events and debugger use.

-jobid 0 | -jobid -1
Controls the job ID to be assigned to the child process.

Specifying **-jobid 0** prevents the child process from running as part of a batch job (the **gtacl** process cannot function as a batch job ancestor, so no other value than **-jobid -1** is supported).

Specifying **-jobid -1** causes the child process to inherit its job ID (if any) from **gtacl**.

The default value is **-jobid -1**.

-lib *pathname* Specifies the OSS pathname of a user library file in the Guardian file system to be used by the child process; the program file for the child process is modified to point to the specified library.

Use of this option requires write access to the program file for the child process. The library file must be in the Guardian file system on the same HP node as the program file for the child process.

This option is needed only when a child process requires a user library and an alternate is needed. The default action is to run the child process with no modification to its user library usage.

+lib Specifies that the child process is to run without any user library file; the program file for the child process is modified so that it does not point to a library.

Use of this option requires write access to the program file for the child process.

This option is needed only when a child process has used a user library that is no longer needed. The default action is to run the child process with no modification to its user library usage.

-mem *n* Specifies the amount of memory to allocate for the data stack of the child process in 2048-byte virtual memory pages (1-64). (Used for G-series TNS or accelerated child processes only.) The default amount is determined by the program file executed as the child process.

-name {**/G/**}*processname*
Starts the child process as a named process using the specified name; *processname* must conform to Guardian process name rules for length.

If **/G/** is omitted, the full filename of the process is resolved using Guardian environment rules. Refer to the RUN command description in the *TACL Reference Manual* for the rules affecting process name length and resolution.

The rules for mapping between Guardian filenames and OSS pathnames mean that *processname* cannot begin with the dollar sign (**$**) used in the Guardian environment.

If only **-name /G** is specified, the operating system creates a unique four-character process name.

The default action is to use the process name attribute for the program file of the child process.

+name Starts the child process as an unnamed process. This specification is ignored if the Guardian RUNNAMED process attribute is set in the program file for the child process.

If you specify neither the **-name** nor the **+name** option, the default behavior is **+name**.

-nowait Exits without waiting for the child process to terminate. Refer to **Input/Output Filtering** under **NOTES** later in this reference page.

If you use this option, the effects of the following options are restricted:

"**-c** *command*"
"**-cv** *command*"
"**-i**"
"**+i**"

If you do not specify the **-nowait** option, the default action is to wait for the child process to terminate.

-p *pathname* Runs the specified program as the child process. This option is an alternate syntax for the **-prog** option followed by the **--** option (described later in this reference page). For example, **gtacl -p** *pathname operands* is equivalent to specifying **gtacl -prog** *pathname* **--** *operands*.

No other option can be specified after this option on the **gtacl** command line.

-pfs *n* Specifies the size of the operating system process file segment (PFS) for the child process in 2048-byte virtual memory pages (64-512). The default size is determined from the program file executed as the child process.

Refer to the *TACL Reference Manual* for additional information about PFS pages.

-pmsg on | -pmsg off

Specifies whether the **gtacl** command displays status information for the child process on the OSS standard output file of the **gtacl** process.

Specifying **-pmsg on** means the name (for a named child process) or the operating system cpu, pin (for an unnamed child process), the program file name, and exit status information is displayed. Specifying **-pmsg off** means no information is displayed.

The default setting for display is **-pmsg off**.

-prog *pathname*

Runs the specified program file as the child process.

If this option is specified, the **gtacl** command uses the OSS environment variable **PMSEARCHLIST** to resolve (expand) *pathname* when it consists of a single Guardian filename component. If the OSS environment variable **PMSEARCHLIST** is not defined, then **gtacl** uses the **/G/system/system** subvolume. If the named file cannot be found in **/G/system/system**, then the current version of **/G/system/sys***nn* is used. If the file is not found, an error diagnostic is printed and **gtacl** terminates.

If this option is not specified, the default action is to run the file **/G/system/system/tacl** or **/G/system/sys***nn/tacl*.

-reclen *length* Specifies the record length (1-4096) in bytes to be reported when a device inquiry request occurs for a file that **gtacl** is filtering.

The default record length is 80 bytes. For systems running J06.05 and later J-series RVUs, H06.16 and later H-series RVUs, or G06.32 and later G-series RVUs, if you start an SQLCI process from the **gtacl** utility, the record length can be up to 255 bytes, and the default record length is 132 bytes.

-s Suppresses propagation of all current OSS environment variables to the TACL process. This option takes precedence if the **-f** option is also used.

-swap *pathname*

Specifies the name of a Guardian swap file or swap volume for the data segment of the child process. This option is no longer used and is provided for compatibility with previous releases. If specified, the name must be

- in OSS pathname syntax
- valid for an existing file

but is otherwise ignored. The operating system chooses a swap volume.

-term *pathname*

Specifies the filename of a Guardian terminal device to be used as the home terminal of the child process. The name must be specified in OSS pathname syntax.

The default action is to use the home terminal of the **gtacl** process.

Refer to the *TACL Reference Manual* for additional information about Guardian terminal device names.

If the **gtacl** process is to be run in the background, HP recommends using **/G/zhome** as the home terminal for the child process. $ZHOME is a reliable home terminal on which the process can perform write operations. For more information about $ZHOME, see the *NonStop NS-Series Operations Guide*.

-- Specifies that there are no more options on the **gtacl** command line. Any information following this option is either processed as redirection specifications or passed to the child process as arguments.

Operands

All operands that the **gtacl** command does not interpret as *option* arguments are passed in the Guardian environment Startup message sent to the child process. **gtacl** does not expand operands or interpret special characters.

If the command line is entered through the OSS environment **/bin/sh** shell, the shell can process or expand operands on the command line before passing the line to **gtacl** to interpret. When an operand contains blanks or special characters normally processed by the shell, you must use quotation marks or escape characters correctly or unexpected actions might result. Refer to the **sh**(1) reference page for more information.

Up to 980 bytes of arguments can be passed to the child process.

DESCRIPTION

The **gtacl** command executes a Guardian program, TACL macro, TACL routine, TACL alias, or TACL command from the OSS environment within the same HP node. **gtacl** is an OSS process that spawns a Guardian process. **gtacl** allows you to specify the environment and initial process attributes of the child process.

EXAMPLES

1. Running an interactive Guardian TACL process:

   gtacl

2. Running a TACL command:

   gtacl -c 'status *, user'

3. Running a Guardian program directly, without using the TACL command interpreter, using shell quotes to preserve special characters for interpretation by the Guardian process:

   gtacl -p fup 'info \sys.$vol.svol.*'

4. Running a Guardian program directly, without using the TACL command interpreter (the shell escape character \ prevents the shell from expanding the * character):

   gtacl -p fup rename *, newsvol.*

5. Running a Guardian program directly and passing a string containing a blank within an operand:

   gtacl -p locate '"two words" $vol.svol.file'

6. Running a server program ($NULL) using the **-nowait** option. In this example the null program is passed a value of 1 for the backup CPU.

   gtacl -nowait -name /G/null -cpu 0 -p /G/system/system/null 1

7. Compiling a C program from a file in the Guardian file system:

   gtacl -p c < /G/vol/svol/file > listing.output

8. Compiling a C program from a file in the Guardian file system, with output to your Guardian default subvolume:

   gtacl -p c < /G/vol/svol/bit1C '; nolist,runnable'

9. Looking up the meaning of an **errno** value:

   gtacl -p 'error' $ERRNO

FILES

/G/system/system/tacl
Used as the default Guardian program to be executed. If the operating system cannot find the program with this pathname, it attempts to uses a copy from the current **SYS***nn* subvolume.

NOTES

The **gtacl** command is commonly used to:

- Start an interactive TACL process (**gtacl**)

- Execute a single Guardian environment command (**gtacl -c** *command* or **gtacl -cv** *command*)
- Run a Guardian environment program (**gtacl -p** *prog args*)

Using the **-c** or **-cv** option to run a Guardian process has the following advantages:

- You can use Guardian environment user defaults and macros set up in a TACLCSTM file.
- You need not distinguish among TACL built-in functions, TACL macros, or programs external to TACL.
- You can use TACL RUN option syntax to direct input or output to Guardian files (such as spooler locations) that are not available through shell redirection.

Using the **-c** or **-cv** option to run a Guardian process has the following disadvantages:

- You cannot redirect standard input using the shell.
- If you do not need TACL facilities, these options add the unnecessary overhead of TACL process creation.

Using the **-p** option to run a Guardian process has the following advantages:

- It runs the program without the overhead of TACL process creation.
- There are no restrictions on redirection of OSS files using the shell.

Using the **-p** option to run a Guardian process has the following disadvantages:

- TACL facilities (such as built-in functions or macros) cannot be used.

Redirecting Input or Output

The **gtacl** process does not have its own run options for redirecting output. Instead, standard **/bin/sh** redirection operators can be used to redirect the input or output of the **gtacl** process for any file that can be opened using the OSS-environment **open**( ) function.

You cannot use the OSS shell to redirect input or output for files that can be opened only using the Guardian file system (with the FILE_OPEN_ procedure call). For such files, the **gtacl -c** *command* or **-cv** *command* option must be used with TACL file redirection. For example, the following command fails because the OSS shell cannot directly open the Guardian spooler process:

```
gtacl -p tgal < /G/vol/subvol/file > /G/S/#TITAN
```

Instead, the following command must be used:

```
gtacl -c 'tgal /IN $vol.subvol.file, OUT $S.#TITAN/'
```

Many Guardian processes allow input or output from a process file. The **gtacl** process uses this feature by running as a named process and passing its own name in place of the OSS standard files that cannot be opened by the Guardian FILE_OPEN_ procedure call. For example, if **gtacl** is invoked with the following command and runs with a system-generated process name of $X123:

```
gtacl -c 'fileinfo $vol.subvol.*' > fileinfo.output
```

then the **gtacl** process runs TACL with an IN file named **$X123.#CMD** and an OUT file named **$X123.#OUT**. When TACL first reads from **$X123.#CMD**, **gtacl** returns the string **fileinfo $vol.subvol.***; on the next read, **gtacl** returns an end-of-file indication. Any output from TACL to **$X123.#OUT** is written by **gtacl** to its own OSS environment standard output file, which has been connected to the OSS environment file **fileinfo.output** by shell redirection.

Input/Output Filtering

The **gtacl** process automatically converts line endings for data passing between the two environments. When **gtacl** encounters a newline indicator in input from the OSS environment, it removes that character and forwards the line as a separate record to the Guardian environment. When **gtacl** encounters the end of a record in input from the Guardian environment, it forwards the line with a newline indicator to the OSS environment. This conversion is the primary filtering done to the data.

The following conditions limit the effectiveness of **gtacl** filtering:

- Filtering can be used only for data passing through OSS environment standard input, output, and error files and Guardian environment IN, OUT, and STDERR files. Any files directly opened by a Guardian environment process cannot be filtered.
- Some Guardian processes do not accept a process file as an input or output file. Filtering will not work with such Guardian processes.
- If filtering is used with redirection and either the **-c** *command* or **-cv** *command* option, it cannot be used for filtering standard input. When either of these options is specified, **gtacl** ignores its standard input file. For example, the following command will not work because the input from **myfile.tgal** is ignored:

  ```
  gtacl -c 'tgal /OUT $s.#hold/' < myfile.tgal
  ```

  Instead, either use an explicit Guardian input file:

  ```
  gtacl -c 'tgal /IN MYFILE, OUT $S.#HOLD/'
  ```

  or use the **-p** *pathname* option to run TGAL directly:

  ```
  gtacl -p tgal < myfile.tgal | lp -d hold
  ```

- When the **-nowait** option is used with input/output filtering, the child process can fail to do either of the following:

  — open one or more of the **gtacl** standard files because **gtacl** has stopped running

  — access one or more of the **gtacl** standard files because **gtacl** has stopped running

  To avoid these problems, do not use the **-c**, **-cv**, **-i**, or **+i** options. Use only files that can be opened from within the Guardian environment for standard input and output when you use the **-nowait** option.

Use in Shell Scripts

Guardian processes typically open a Guardian environment disk file for output by requesting protected or exclusive access. This practice can conflict with use of exclusion mode by an OSS process.

For example:

```
gtacl -p FUP 'INFO *'
```

fails if it is invoked from an OSS shell script and the output of the script is redirected to a Guardian disk file. Because the output file can be opened from the Guardian environment, **gtacl** does not filter the output. However, because the output file is still open by the shell process executing the script, FUP cannot open the file for exclusive access and terminates abnormally.

To avoid this problem when **gtacl** is used within a shell script, use the **-i** or **+i** option to force input/output filtering, as follows:

```
gtacl -i -p FUP 'INFO *'
```

OSS Environment Variables

The following OSS environment variables affect the execution of the **gtacl** command.

PMSEARCHLIST

If this variable is defined, **gtacl** uses the value if necessary to resolve a Guardian file identifier to find the program specified with the **-p** or **-prog** option. The value can be one or more of the following:

- a list of Guardian subvolume names in Guardian external file name format, separated by spaces. These subvolumes are searched when resolving a Guardian file identifier to find the program specified with the **-p** or **-prog** option.
- the TACL command interpreter #DEFAULTS built-in variable.
- the TACL command interpreter identifiers #DEFAULTS/CURRENT/ or #DEFAULTS/SAVED/.

If this variable is not defined, **gtacl** uses the **/G/system/system** subvolume to resolve a relative filename.

PWD

This variable must resolve to a valid Guardian filesystem subvolume name. If the value for this variable appears to be an OSS pathname outside of the **/G** directory, **gtacl** ignores the value.

If this variable is correctly specified, **gtacl** interprets the value as a Guardian volume and subvolume name. The **gtacl** process passes the value to the child process in the default volume and subvolume part of the Guardian environment Startup message.

If this variable is not defined or is incorrectly specified, then **gtacl** uses the inherited default volume and subvolume names from the =_DEFAULTS DEFINE for the Guardian environment Startup message.

Unless the **-s** option is used, OSS environment variables are converted into

Guardian PARAMs and passed to the child process in a Guardian-environment PARAM system message. Underscores in an OSS environment variable name are converted to circumflex (^) characters in the equivalent Guardian PARAM name.

A single PARAM name and value can contain up to 255 bytes of character information for one environment variable. If the length of an OSS environment variable name plus the value for that variable exceeds 254 bytes, the variable is not converted and an error message is sent to the standard output file for **gtacl**.

Up to 1024 bytes of PARAM names and values are supported. PARAM names and values are accumulated from the current shell environment variables by default, in the order defined. If more OSS environment variables need to be converted and passed than fit within the 1024-byte limit, only those that fit are passed. The remaining OSS environment variables are ignored and **gtacl** issues a warning message.

If the 1024-byte limit does not allow needed environment variables to be passed, the **-f** flag can be used to specify a specific set to be passed. When the **-f** flag is used, no accumulated environment variables are passed. Instead, each environment variable specified in the text file associated with the **-f** flag is converted to a separate Guardian PARAM. The converted PARAMs from the file must still fit within the 1024-byte limit.

Guardian Environment Variables

The following Guardian environment variables affect the execution of the **gtacl** command.

DEFINEs

=_DEFAULTS Provides the default values for the current Guardian volume and subvolume names.

If the **-defmode on** option is used, all Guardian DEFINEs inherited by **gtacl** are inherited by the child process. Up to 256K bytes of DEFINEs can be inherited. The actual maximum depends on the size of the PFS for the child process.

If the **-defmode off** option is used, only the Guardian =_DEFAULTS DEFINE values inherited by **gtacl** are inherited by the child process.

DIAGNOSTICS

Error diagnostics are written to the OSS environment standard error file of the **gtacl** process. All **gtacl** error messages are prefixed with `gtacl[`*n*`]:`, where *n* is a unique message number. The following messages can appear:

`gtacl[1]: unrecognized option` *option_name*
You specified an option that **gtacl** does not recognize.

Check for typographical errors and reenter a corrected command line.

`gtacl[2]: unable to open $RECEIVE, error` *n strerror(n)*
Guardian file-system error *n* was returned when **gtacl** attempted to open its own Guardian $RECEIVE file. The meaning of that error number as returned by the **strerror( )** function is displayed.

The recovery action depends on the Guardian file-system error

number. Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`gtacl[3]: unable to get default volume, DEFINEINFO error` *n*
Guardian file-system error *n* was returned from a call to the Guardian DEFINEINFO procedure that attempted to get the information for the =_DEFAULTS DEFINE. A system software problem might exist; the =_DEFAULTS DEFINE should exist for all user processes.

Refer to the DEFINEINFO procedure description in the *Guardian Procedure Calls Reference Manual* for the meaning of the error returned. Check the current value of your =_DEFAULTS DEFINE by using the OSS shell **info_define** command.

`gtacl[4]: the` *option_name* `option must be followed by` *permissible_values*
You specified the indicated option with an unrecognized value.

Check for typographical errors and reenter a corrected command line.

`gtacl[5]: Unable to run` *pathname*`, error (`*error*`,` *err_detail*`) :` *explanation*
The child process for the program file *pathname* could not be started by **gtacl** using the Guardian PROCESS_LAUNCH_ procedure call. The Guardian error *error* was returned by PROCESS_LAUNCH_. A brief explanation of that error number is displayed as *explanation*. The associated error detail return value is shown as *err_detail*. For many values of *error*, the value of *err_detail* is also a Guardian file-system error number.

Refer to the PROCESS_LAUNCH_ description in the *Guardian Procedure Calls Reference Manual* and to the error description in the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`gtacl[6]: Unable to open child process, error` *n*`:` *strerror(n)*
The **gtacl** command detected an error while trying to open the child process and send the child process a sequence of Guardian environment startup messages. The Guardian file-system error *n* was returned. The meaning of that error number as returned by the **strerror( )** function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions. A typical error is 201, because the child process terminated as soon as it started. If this is the case, use the **-inspect saveabend** option to create a saveabend (process snapshot) file or the **-debug** option to start the child process within a debugging tool.

`gtacl[7]: Unable to read from <$RECEIVE|`*stdfile*`>, error` *n*`:` *strerror(n)*
The **gtacl** process detected an error while reading its Guardian $RECEIVE file or the standard file indicated by *stdfile*. The Guardian file-system error *n* was returned. The meaning of that error number as returned by the **strerror( )** function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`gtacl[8]: Unable to write to` *stdfile*`, error` *n*`:` *strerror(n)*
The **gtacl** process detected an error while reading the standard file indicated by *stdfile*. The Guardian file-system error *n* was returned. The meaning of that error number as returned by the **strerror( )** function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`gtacl[9]:warning: unable to propagate all environment variables`
The **gtacl** process cannot convert all of the OSS environment variables into Guardian environment PARAMs. Some variables have not been passed to the child process.

No action is necessary if the full set of OSS environment variables is not needed by the child process. To eliminate this message, reduce the number of OSS environment variables to a set that will convert and fit into a single Guardian-environment PARAM message and then reissue the **gtacl** command. Refer to the section on interprocessor command interpreter messages in the *Guardian Procedure Errors and Messages Manual* for more information about the PARAM message format and usage.

`gtacl[10]: unable to get process information,` *procedure* `error` *n*
The **gtacl** process received the Guardian file-system error *n* when trying to determine its own Guardian process attributes. The **gtacl** process uses many of its own Guardian process attributes (such as processor number and home terminal name) as default values when creating the child process.

The Guardian procedure call indicated by *procedure* encountered the problem. This message occurs when one of the following conditions is true:

- The version of **gtacl** used is not compatible with the version of the operating system.
- The NonStop Kernel message system does not have enough resources to provide the information.
- A coding error exists within this version of **gtacl**.

Recovery action depends on the error returned. Refer to the description of *procedure* in the *Guardian Procedure Calls Reference Manual* for more detailed information about the error

when returned by that procedure.

`gtacl[11]:` `internal error -` *description*
The **gtacl** process has detected the situation described by *description*. This error should also create a saveabend file for the **gtacl** process.

Report this problem to your service provider. Give the service provider a copy of the saveabend (process snapshot) file and describe the conditions necessary to reproduce the problem.

`gtacl[12]:` `unable to allocate` *n* `bytes for` *purpose*
The **gtacl** process could not allocate the indicated amount of memory from its heap. This probably indicates that **gtacl** could not allocate a Guardian file-system extent on the extended segment file used for the heap.

If this problem occurs because of disk allocation failure, run **gtacl** using an extended swap volume or extended segment file that has more free space.

`gtacl[13]:` `unable to send` *msgtype* `message, error` *n*`:` *strerror(n)*
The **gtacl** process could not send a Guardian-environment Startup, ASSIGN, or PARAM message to its child process. The Guardian file-system error *n* was returned. The meaning of that error number as returned by the **strerror( )** function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`gtacl[14]:` *procedure* `error` *n* `on` *file*`:` *strerror(n)*
The indicated procedure returned Guardian file-system error *n* for the named file. The meaning of that error number as returned by the **strerror( )** function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`gtacl[15]:` *procedure* `returned error` *n*`:` *strerror(n)*
The indicated procedure returned Guardian file-system error *n* for the named file. The meaning of that error number as returned by the **strerror( )** function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`gtacl[16]:` `Unable to open` *pathname*`, error` *n*`:` *strerror(n)*
The file with the specified *pathname* could not be opened when the **-f** option was used. The Guardian file-system error *n* was returned for the file. The meaning of that error number as returned by the **strerror( )** function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

EXIT VALUES

The **gtacl** process returns the following exit values (the Guardian environment completion code always equals the **gtacl** exit value plus 256):

0	The child process terminated with an exit value of 0 or 5, or the **-nowait** option was specified and the child process started successfully. This value corresponds to the Guardian environment completion codes 256 and 231.
170	An error prevented **gtacl** from starting the child process. This value corresponds to the Guardian environment completion code 426.
171	Warnings occurred during the start of the child process. This value corresponds to the Guardian environment completion code 427.
172	An error prevented **gtacl** from receiving the termination status of the child process. This value corresponds to the Guardian environment completion code 428.
173	The child process terminated with a negative completion code. This value corresponds to the Guardian environment completion code 429.
174	The **gtacl** process terminated abnormally. This value corresponds to the Guardian environment completion code 430.

In all other cases, **gtacl** returns the exit status value and Guardian completion code that it receives from the child process.

RELATED INFORMATION

Commands: **info_define(1)**, **osh(1)**, **sh(1)**.

Functions: **open(2)**, **strerror(3)**.

Files: **core(4)**.

STANDARDS CONFORMANCE

The **gtacl** command is an HP extension to the XPG4 Version 2 specification.

NAME

hash - Affects memory of where utilities are located

SYNOPSIS

hash [*utility* ...] **-r**

FLAGS

-r Causes the **hash** command to forget all locations of utilities and commands.

DESCRIPTION

The **hash** command affects the way the current shell environment remembers the locations of utilities.

When the name of a command or utility is specified for *utility*, the **hash** command adds the location of that command or utility to its list of remembered locations.

When the **-r** flag is used, **hash** forgets all previously remembered locations of utilities and commands.

When *utility* is not specified and the **-r** flag is not used, **hash** reports the contents of its list of remembered locations.

EXAMPLES

1. The following command lists the commands whose locations the **hash** command currently remembers:

   hash

NOTES

The **hash** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

NAME

head - Displays the beginning of a file

SYNOPSIS

head [**-c** *bytes*] [**-n** *lines*] [*file* ...]

FLAGS

-c *bytes* Specifies the number of bytes (not characters) to display. If the last byte written is not a newline character, a newline character is appended to the output.

-n *lines* Specifies the number of lines to display. The default value for *lines* is 10.

DESCRIPTION

The **head** command prints the first *lines* lines or *bytes* bytes of each of the specified files to the standard output file.

If you do not specify *file*, **head** displays from the standard input file.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

To display the first 5 lines of a file named **test**, enter:

```
head -n 5 test
```

EXIT VALUES

The **head** command returns the following values:

0 (zero)	The command completed sucessfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **cat(1)**, **more(1)**, **tail(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification.

The following feature is an extension to the XPG4 Version 2 specification:

- The **-c** flag is supported.

NAME

history - Lists previously executed commands

SYNOPSIS

history

DESCRIPTION

The **history** command displays the contents of the **history** file, which contains a list of previously executed commands. The **history** command is an exported alias for the **fc -l** command and is compiled into the OSS shell. (See the reference page for the **fc** command.) **history** can be unset or redefined. See "Command Aliasing" in the **sh** reference page.

EXAMPLES

1. The following command lists the contents of the **history** file:

   history

NOTES

The **history** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **fc**(1), **sh**(1).

NAME

iconv - Converts encoded characters to another code set

SYNOPSIS

iconv

-f *from_code*
-t *to_code*
[*file* ...]

FLAGS

-f *from_code*
Specifies the input code set

-t *to_code*
Specifies the output code set

Operands

file specifies the file to be converted

DESCRIPTION

The **iconv** command converts the encoding of **characters** in *file* from one coded character set to another and writes the results to the standard output file.

The input and output coded character sets are identified by the arguments *from_code* and *to_code*. If the *file* operand is not specified on the the command line, the **iconv** command reads the standard input file.

If you specify invalid characters in the input stream, **iconv** returns an error message, writes the already converted data to standard output, and exits with a status greater than 0 (zero.)

If you specify characters in the input stream that are valid members of the input code set but are invalid members of the output code set, the input value is translated into a substitute character. Such conversions are called non-identical conversions. The value of the substitute character depends on the output code set specified.

The default value is 0x1A (the ASCII character SUB); the default value is 0xFFFD (the UCS-2 REPLACEMENT CHARACTER) for conversions to the UCS-2 code set. If the conversion is done by using a table, the substitute character may be specified by a **target_sub** directive in the source translation table. The default is 0x1A if there is no **target_sub** directive.

If a non-identical conversion occurred while converting an input file, the **iconv** utility will attempt to print a warning message to the standard error file, once per file. If an input file is large it is possible that the non-identical conversion warning will not be printed, even if such conversions occurred. The exit status is not affected by a non-identical conversion.

If the **LOCPATH** environment variable is not defined or is set to the empty string (""), **iconv** looks for converters and conversion tables in the default path: **/usr/lib/nls/loc:/etc/nls/loc**.

Otherwise, **iconv** looks in the directories specified by **LOCPATH**. If no valid converters or conversion tables are found in the directories specified by **LOCPATH**, the default path is searched too. If the superuser is using **iconv**, **LOCPATH** is ignored. **$LOCPATH** is an environment variable which, if set, specifies directories containing the **iconv** and **iconvTable** subdirectories to be used instead of the default directories. The value of **$LOCPATH** must be a colon-separated list of directories in which **iconv** can find the subdirectories containing converters and conversion tables. Specified directories must have the same organization as the **/usr/lib/nls/loc** directory.

conversion_directory/**iconv/***
is a directory containing algorithmic converters, where *conversion_directory* is specified in **$LOCPATH**.

conversion_directory/**iconvTable/***
is a directory containing **iconv** conversion tables generated by **genxlt**, where *conversion_directory* is specified in **$LOCPATH**.

Environment Variables

The following environment variables affect the execution of the **iconv** command: **LANG**, **LC_ALL**, **LC_MESSAGES**, **NLSPATH**.

DIAGNOSTICS

The following error messages have an exit value of 1:

`Usage: iconv -f <from> -t <to> [<infile>]`
Invalid command line

`iconv: Cannot open converter`
Failure to open a converter

`iconv: Unable to allocate enough memory`
Memory allocation failure

`iconv: Input file cannot be opened`
Failure to open the input file

`iconv:read` Failure to read an input file

`iconv: write`
Failure to write to the output file

The following error messages have an exit value of 2:

`iconv: truncated character found`
Uncompleted multibyte character in the input file

`iconv: invalid character found`
Invalid character in the input file

EXIT VALUES

If successful, the **iconv** command exits with a value of 0 (zero). If unsuccessful, the **iconv** command exits with a value greater than zero and writes an error message to the standard error file.

EXAMPLES

To convert the contents of the file `eucjp.file` from code set **eucJPC** to **SJIS**, and store the results in the file `sjis.file`, enter:

iconv -f eucJP -t SJIS eucjp.file > sjis.file

To convert the contents of the file `latin1.file` from code set ISO 8859-1 to code set ISO 8859-2 and store the results in the file *latin2.file*, enter:

iconv -f ISO8859-1 t-UCS-2 latin1.file | iconv -f UCS-2 -t ISO8859-2 > latin2.file

This conversion uses an intermediate code set (UCS-2) because there are no conversion tables or algorithmic converters available for converting directly from ISO 8859-1 to ISO 8859-2.

FILES

/usr/lib/nls/loc/iconv/*
Contains algorithmic converters.

/usr/lib/nls/loc/iconvTable/*
Contains table converters.

RELATED INFORMATION

Commands: **genxlt(1)**.

NAME

id - Displays the user's system identity

SYNOPSIS

id [*user*]

id -G [**-n**] [*user*]

id -g [**-nr**] [*user*]

id -u [**-nr**] [*user*]

FLAGS

-g Outputs only the effective group ID by using the **printf** format "**%u\n**".

-G Outputs all different group IDs (effective, real, and supplementary) only by using the **printf** format "**%u\n**". If there is more than one distinct group affiliation, this flag outputs each such affiliation by using the **printf** format **%u** before the new line is output.

-n Outputs the name in the **printf** format **%s** instead of the group ID by using the **printf** format **%u**.

-r Outputs the real ID instead of the effective ID.

-u Outputs the effective user ID only by using the **printf** format "**%u\n**".

DESCRIPTION

The **id** command writes a message containing the user and group IDs and the corresponding names of the invoking process or the specified user to the standard output file. When effective names and IDs do not match the real ones, the **id** command writes both.

If multiple groups are supported (according to the **NGROUPS_MAX** variable), the supplementary group affiliations of the invoking process are also written.

If the selected user has more than one allowable group membership listed in the group database, these are written in the same manner as the supplementary groups.

EXAMPLES

To display your user and group IDs, enter:

id

Information similar to the following is displayed:

```
uid=9259(tom) gid=1000(mcm) groups=0(system),1(daemon),
3(kmem),5(cf), 8(news),20(dumper),28(operator),
1003(engtools),1016(ddts),1052(mailuni), 1059(utils),
1062(lisbon),1065(perf)
```

RELATED INFORMATION

Commands: **logname(1).**

Functions: getuid(2).

NAME

info_define - Displays attributes and values of existing DEFINEs

SYNOPSIS

info_define [**-detail**]
{ { *define-name*}[*define-name*] ... | **all** }

FLAGS

-detail Displays all the values and attributes of each DEFINE specified by *define-name*.

define-name
Specifies the name of the DEFINE whose information is to be displayed. The name can be 2 through 24 characters long. The first character must be an equal sign (=) and the second character must be an alphabetic character. A list of DEFINE names, separated by blanks, can be specified.

all Displays information on all existing DEFINEs.

DESCRIPTION

The **info_define** command is specific to HP and is an OSS shell built-in command. It displays attributes and values associated with the specified DEFINEs. It is similiar to the TACL INFO DEFINE command. Refer to the *TACL Reference Manual* (INFO DEFINE) for more information on the output of **info_define**.

Environment Variables

LANG Determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale.

LC_ALL
Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variable beginning with **LC**.

LC_CTYPE
Determines the locale for interpretation of bytes of text data as characters (for example, single-byte characters as opposed to multibyte characters in arguments).

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to the standard error file and informational messages written to the standard output file.

EXAMPLES

1. To display the principal attributes of all DEFINEs, enter:

   info_define all

This command might result in the following display:

```
DEFINE NAME     ==ACK
CLASS           =MAP
FILE            =$BILL.WORK.FILE

DEFINE NAME     ==_DEFAULTS
CLASS           =DEFAULTS
VOLUME  =$BILL.WORK
```

2. To display all attributes of an existing DEFINE that have a value, enter:

   info_define -detail =TEST3

   This command might result in the following display:

```
DEFINE NAME     ==TEST3
CLASS           =TAPE
VOLUME  =SCRATCH
LABELS  =OMITTED
USE             =IN
DEVICE          =$TAPE2
MOUNTMSG        =Transferring files-Mr. Smith
```

NOTES

The **info_define** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the **sh(1)** reference page.

EXIT VALUES

The following exit values are returned:

0	DEFINEs displayed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **add_define(1)**, **del_define(1)**, **reset_define(1)**, **set_define(1)**, **show_define(1)**.

STANDARDS CONFORMANCE

The **info_define** command is an HP extension to the XPG4 Version 2 specification.

NAME

initfilepriv - Sets file privileges for selected system files

SYNOPSIS

initfilepriv *sysnn_path*

FLAGS

Operands

sysnn_path Specifies the path to the system subvolume that contains the system files that require file privileges (such as the Backup and Restore 2 product files). Typically, this is the current system subvolume, usually specified as **/G/system/sys***nn*, where *nn* indicates the system installation that is currently running.

DESCRIPTION

The **initfilepriv** command sets the file privileges on selected system files. For example, it sets the file privileges on the Backup and Restore 2 product such that, when started by a user with the appropriate privilege, it can back up and restore files that are in restricted-access filesets.

This command is supported on systems running J06.11 or later J-series RVUs or H06.22 or later H-series RVUs only. The Backup and Restore 2 product must also be installed on a system running J06.11 or later J-series RVUs or H06.22 or later H-series RVUs only.

Only Members of Safeguard SECURITY-PRV-ADMINISTRATOR (SEC-PRIV-ADMIN or SPA) group are permitted to use this command.

For more information about restricted-access filesets and file privileges, see the *Open System Services Management and Operations Guide*.

EXAMPLE

To set the file privileges for Backup and Restore 2 application files in the **sys21** subvolume, enter:

```
initfilepriv /G/system/sys21
```

RELATED INFORMATION

Commands: **getfilepriv(1)**, **setfilepriv(1)**.

Functions: **setfilepriv(2)**.

STANDARDS CONFORMANCE

This command is an HP extension.

NAME

ipcrm - Removes message queues, semaphore identifiers, or shared memory identifiers and deallocates their data structures

SYNOPSIS

ipcrm [**-m** *shared_memory_ID*] [**-M** *shared_memory_key*]
[**-q** *message_queue_ID*] [**-Q** *message_key*]
[**-s** *semaphore_ID*] [**-S** *semaphore_key*]

FLAGS

-m *shared_memory_ID*
Removes the shared memory identifier specified by the *shared_memory_ID* value, and removes the associated shared memory segment and data structure after the final detach operation.

-M *shared_memory_key*
Removes the shared memory identifier created with the specified *shared_memory_key* value, and removes the associated shared memory segment and data structure after the final detach operation.

-q *message_queue_ID*
Removes the message queue identifier specified by the *message_queue_ID* value, and removes the associated message queue and the data structure.

-Q *message_key*
Removes the message queue identifier created with the specified *message_key* value, and removes the associated message queue and the data structure.

-s *semaphore_ID*
Removes the semaphore set identifier specified by the *semaphore_ID* value, and removes the associated set of semaphores and data structure.

-S *semaphore_key*
Removes the semaphore set identifier created with the specified *semaphore_key* value, and removes the associated set of semaphores and data structure.

DESCRIPTION

The **ipcrm** command removes one or more message queues, semaphore identifiers, or shared memory identifiers from the processor on which it runs. The details of the remove operations are described on the reference pages for the **msgctl( )**, **semctl( )**, and **shmctl( )** functions. The identifiers and keys can be determined by using the **ipcs** command.

EXAMPLES

1. To remove the shared memory segment associated with shared memory identifier **128** from the processor on which your terminal session runs, enter:

   ipcrm -m 128

2. To remove the shared memory segment associated with shared memory identifier **128** on processor 4, enter:

   run -cpu=4 ipcrm -m 128

3. To remove the semaphore set associated with semaphore identifier **222** from the processor on which your terminal session runs, enter:

   ipcrm -s 222

4. To remove the semaphore set associated with semaphore identifier **222** on processor 4, enter:

   run -cpu=4 ipcrm -s 222

5. To remove the message queue associated with message queue identifier **4** from the processor on which your terminal session runs, enter:

   ipcrm -q 4

6. To remove the message queue associated with message queue identifier **4** on processor 4, enter:

   run -cpu=4 ipcrm -q 4

NOTES

The **ipcs** and **ipcrm** commands have no provision for managing interprocess communication (IPC) facilities on processors other than the one in which they run. To manage IPC facilities across all the processors of a node, you need to use these commands with other OSS shell commands and command flags, such as the **run** command **-cpu=** flag.

RELATED INFORMATION

Commands: **ipcs(1)**, **run(1)**.

Functions: **msgctl(2)**, **msgget(2)**, **semctl(2)**, **semget(2)**, **semop(2)**, **shmctl(2)**, **shmdt(2)**, **shmget(2)**.

STANDARDS CONFORMANCE

This utility is an HP extension to the XPG4 Version 2 specification.

NAME

ipcs - Reports interprocess communication (IPC) facilities status

SYNOPSIS

ipcs [**-a** | **-bcopt**] [**-mqs**]

FLAGS

-a Is the same as specifying the **-b**, **-c**, **-o**, **-p**, and **-t** flags.

-b Writes the maximum number of bytes in message queues, the maximum allowed size of segments for shared memory, and the number of semaphores in each semaphore set.

-c Writes the username and group name of the user who created the facility.

-m Writes information about active shared memory segments.

-o Writes the following usage information:

- Number of messages on queue.
- Total number of bytes in message queues.
- Number of processes attached to shared memory segments.

-p Writes the following usage information:

- OSS process ID of the last process to receive a message and OSS process ID of the last process to send a message currently on message queues
- Process identifier of the creating process for a facility
- Process identifier of the last process to call the **shmat( )**, **shmdt( )**, or **shmctl( )** function for the segment

-q Writes information about active message queues.

-s Writes information about active semaphore sets.

-t Writes the times of the following actions:

- Either:
  - — The creation of the facility
  - — The last control operation that changed the facility (either by modifying permissions information for the facility or by removing the facility)
- The last use of the **msgsnd( )** function and of the **msgrcv( )** function on message queues
- The last use of the **shmat( )** function and of the **shmdt( )** function on shared memory
- The last use of the **semop( )** function on semaphore sets

DESCRIPTION

The **ipcs** command writes information to the standard output file about active interprocess communication (IPC) facilities on the processor in which the command runs. If you do not specify any flags, **ipcs** writes information in short form about the currently active message queues, shared memory segments, semaphore sets, and local message queue headers.

Column Headings

The column headings and the meanings of the columns in an **ipcs** listing follow. The letters in parentheses indicate the flags that cause the corresponding heading to appear. The word "all" means that the heading always appears. The flags determine only what information is provided for each facility; they do not determine which facilities are listed.

T (all) Type of facility:

- **q** Message queue
- **m** Shared memory segment
- **s** Semaphore

ID (all) The identifier for the facility entry.

KEY (all)

The key used as a parameter in the **msgget( )**, **semget( )**, or **shmget( )** function call that created the facility entry.

Note that the key of a shared memory segment is changed to **IPC_PRIVATE** when the segment is removed. The entry disappears when the last process attached to the segment detaches it.

MODE (all)

The facility access modes and flags. The mode consists of 11 characters that are interpreted as follows. The first character is one of the following:

- **R** If a process is waiting on a **msgrcv()** function call.
- **S** If a process is waiting on a **msgsnd()** function call.
- **D** The associated shared memory segment was removed. The entry disappears when the last process attached to the segment detaches it.
- **-** The corresponding special flag is not set.

The second character is reserved for future use.

The next nine characters are interpreted as three sets of three characters each. The first set refers to the owner's permissions; the second set to permissions of others in the user group of the facility entry; and the third set to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.

The permissions are indicated as follows:

- **r** Read permission is granted.

w Write permission is granted (for shared memory segments).

a Alter permission is granted (for semaphore sets).

- The indicated permission is not granted.

OWNER (all)
The username of the owner of the facility entry.

GROUP (all)
The group name of the owner of the facility entry.

CREATOR (**a**,**c**)
The username of the creator of the facility entry.

CGROUP (**a**,**c**)
The group name of the creator of the facility entry.

Note that when the **OWNER**, **GROUP**, **CREATOR**, and **CGROUP** columns all appear, the user IDs and group IDs are displayed instead of the usernames and group names.

CBYTES (**a**,**o**)
The number of bytes in messages currently outstanding on the associated message queue.

QNUM (**a**,**o**)
The number of messages currently outstanding on the associated message queue.

QBYTES (**a**,**b**)
The maximum number of bytes allowed in messages outstanding on the associated message queue.

LSPID (**a**,**p**)
The ID of the latest process that sent a message to the associated queue.

LRPID (**a**,**p**)
The ID of the latest process that received a message from the associated queue.

STIME (**a**,**t**)
The time when the last message was sent to the associated queue.

RTIME (**a**,**t**)
The time when the last message was received from the associated queue.

CTIME (**a**,**t**)
The time when the associated entry was created or changed.

NATTCH (**a**,**o**)
The number of processes attached to the associated shared memory segment.

SEGSZ (**a**,**b**)
The size of the associated shared memory segment.

CPID (**a**,**p**)
The process ID of the creator of the shared memory entry.

LPID (**a**,**p**)
The process ID of the latest process to call the **shmat( )**, **shmdt( )**, or **shmctl( )** function for the associated shared memory segment.

ATIME (**a**,**t**)
The time when the latest call to the **shmat( )** function was made for the associated shared memory segment.

DTIME (**a**,**t**)
The time when the latest call to the **shmdt( )** function was made for the associated shared memory segment.

NSEMS (**a**,**b**)
The number of semaphores in the set associated with the semaphore entry.

OTIME (**a**,**t**)
The time when the latest semaphore operation was completed on the set associated with the semaphore entry.

EXAMPLES

1. Sample output from entering the **ipcs** command without flags follows:

```
Message Queues:

T       ID      KEY             MODE            OWNER   GROUP

q           0   0x00010381      R rw-rw-rw-     SUPER   SUPER
q       65537   0x00010307      R rw-rw-rw-     SUPER   SUPER
q       65538   0x00010311      R rw-rw-rw-     SUPER   SUPER
q       65539   0x0001032f      R rw-rw-rw-     SUPER   SUPER
q       65540   0x0001031b      R rw-rw-rw-     SUPER   SUPER
q       65541   0x00010339      - rw-rw-rw-     SUPER   SUPER
q           6   0x0002fe03      R rw-rw-rw-     SUPER   SUPER

Shared memory:

T       ID      KEY             MODE            OWNER   GROUP

m        65537  0x00000000      D rw-------     SUPER   SUPER
m       720898  0x00010300      - rw-rw-rw-     SUPER   SUPER
m        65539  0x00000000      D rw-------     SUPER   SUPER

Semaphores:

T       ID      KEY             MODE            OWNER   GROUP

s       131072  0x4d02086a      - ra-ra----     SUPER   SUPER
s        65537  0x00000000      - ra-------     SUPER   SUPER
s       131076  0x00010301      - ra-ra-ra-     SUPER   SUPER
```

2. To display information on a processor other than the one for your terminal session, start the **ipcs** utility using the **run** command **-cpu=** flag for the appropriate processor. For example, to see information for processor 4, enter:

   run -cpu=4 ipcs

NOTES

The **ipcs** and **ipcrm** commands have no provision for managing IPC facilities on processors other than the one in which they run. To manage IPC facilities across all the processors of a node, you need to use these commands with other OSS shell commands and command flags, such as the **run** command **-cpu=** flag.

RELATED INFORMATION

Commands: **ipcrm(1)**, **run(1)**.

Functions: **msgrcv(2)**, **msgsnd(2)**, **semctl(2)**, **semget(2)**, **semop(2)**, **shmat(2)**, **shmctl(2)**, **shmdt(2)**, **shmget(2)**.

STANDARDS CONFORMANCE

This utility is an HP extension to the XPG4 Version 2 specification.

NAME

jobs - Lists processes

SYNOPSIS

jobs [**-lnp**] [*job* ...]

FLAGS

-l Lists process IDs in addition to the normal information.

-n Lists jobs that have stopped or exited since last invocation.

-p Lists only the process group.

DESCRIPTION

The **jobs** command lists information about the process specified as *job*. If *job* is not specified, the **jobs** command provides information on all active processes.

EXAMPLES

1. The following command lists all jobs, including job numbers and PIDs.

   jobs -l

NOTES

The **jobs** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh(1)**.

NAME

join - Joins the lines of two files

SYNOPSIS

Current syntax

join [**-a** *filenum_a*]
[**-e** *string*]
[**-o** *number*.*field*, ...]
[**-t** *character*]
[**-v** *filenum_v*]
[**-1** *field1*] [**-2** *field2*]
file1 file2

Obsolescent syntax

join [**-a** *filenum_a*]
[**-e** *string*]
[**-j** *num* | *field* | *num fld*]
[**-o** *number*.*field*, ...]
[**-t** *character*]
file1 file2

FLAGS

-1 *field1* Specifies the number of the join field for *file1*. *field1* is a decimal integer starting with 1.

-2 *field2* Specifies the number of the join field for *file2*. *field2* is a decimal integer starting with 1.

-a *filenum_a*
Produces an output line for each unmatched line found in *file1* if *filenum_a* is **1** or in *file2* if *number* is **2**. Without **-a**, **join** produces output only for lines containing a matching field. If both **-a 1** and **-a 2** are used, the output contains all unmatched lines.

-e *string* Replaces empty output fields with *string*.

-j *num* | *fld* | *num fld*
(Obsolescent) Specifies the join field *fld* for file *num*, where *num* is **1** for *file1* or **2** for *file2*. If you do not specify *num*, **join** uses *fld* in both files. Without **-j**, **join** uses the first field in each file. The default value for both *num* and *fld* is **1**.

If you enter only a **1** or a **2** as an argument to **-j**, **join** interprets this argument as the file number (*num*); integers greater than 2 are interpreted as the field number (*fld*). Therefore, if you want to specify a field number of **1** or **2**, you must precede the *fld* specification with a *num* argument; otherwise, the **join** command interprets the **1** or **2** as the file number (*num*).

-o *number*.*field*
Produces output lines consisting of the fields given by one or more *number*.*field* arguments in the specified order, where *number* is **1** for *file1* or **2** for *file2* and *field* is a field number. Multiple **-o** arguments should be separated with commas.

-t *character*

Uses *character* (a single character) as the field separator character in the input and the output. Every appearance of *character* in a line is significant. The default separator is a space. If you do not specify **-t**, **join** also recognizes the tab and newline characters as separators.

With default field separation, the collating sequence is that of **sort -b**. If you specify **-t**, the sequence is that of a plain sort. To specify a tab character, enclose it in **' '** (single quotes).

-v *filenum_v*

Produces an output line for each unmatched line in *filenum_v* (where *filenum_v* is **1** for *file1* or **2** for *file2*), instead of the default output. If both **-v 1** and **-v 2** are specified, **join** produces output lines for all unmatched lines.

DESCRIPTION

The **join** command reads *file1* and *file2*, joins the lines in those files that contain common fields or otherwise according to the flags, and writes the results to the standard output file.

The join field is the field in the input files that **join** searches to determine what will be included in the output. One line appears in the output for each identical join field appearing in both *file1* and *file2*. The output line consists of the join field, the rest of the line from *file1*, then the rest of the line from *file2*.

You can specify the standard input file in place of *file1* or *file2* by substituting a **-** (dash) for the operand.

Both input files must be sorted according to the collating sequence specified by the **LC_COLLATE** environment variable, if set, for the fields on which they are to be compared (by default, the first field in each line).

Fields are normally separated by a space, a tab character, or a newline character. **join** treats consecutive occurrences of these separators as a single separator and discards leading separators. Use the **-t** flag to specify another field separator.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

Note that the vertical alignment shown in these examples might not be consistent with your output.

1. To perform a simple **join** operation on two files, **phonedir** and **names**, whose first fields contain employee last names, enter:

   join phonedir names

   If **phonedir** contains the following telephone directory:

   ```
   Binst          555-6235
   Dickerson      555-1842
   Eisner         555-1234
   Green          555-2240
   Hrarii         555-0256
   Janatha        555-7358
   Lewis          555-3237
   Takata         555-5341
   ```

```
Wozni           555-1234
```

and **names** contains this listing of names and department numbers:

```
Eisner          Dept. 389
Frost           Dept. 217
Green           Dept. 311
Takata          Dept. 454
Wozni           Dept. 520
```

then **join phonedir names** displays:

```
Eisner          555-1234        Dept. 389
Green           555-2240        Dept. 311
Takata          555-5341        Dept. 454
Wozni           555-1234        Dept. 520
```

Each line consists of the join field found in both files (the last name), followed by the rest of the line found in **phonedir**, followed by the rest of the line found in **names**.

2. To display unmatched lines in **names** as well as matched lines in both files, enter:

   join -a 2 phonedir names

   If **phonedir** contains:

```
Binst           555-6235
Dickerson       555-1842
Eisner          555-1234
Green           555-2240
Hrarii          555-0256
Janatha         555-7358
Lewis           555-3237
Takata          555-5341
Wozni           555-1234
```

   and **names** contains:

```
Eisner          Dept. 389
Frost           Dept. 217
Green           Dept. 311
Takata          Dept. 454
Wozni           Dept. 520
```

   then **join -a 2 phonedir names** displays:

```
Eisner          555-1234        Dept. 389
Frost                           Dept. 217
Green           555-2240        Dept. 311
Takata          555-5341        Dept. 454
Wozni           555-1234        Dept. 520
```

   This command performs the same **join** operation as in the first example and also lists the lines of **names** that have no match in **phonedir** (the entry for `Frost`).

3. To perform a **join** operation and display selected fields, enter:

   join -o 2.3,2.1,1.2 phonedir names

   This command displays the following fields in this order:

   Field 3 of **names** (department number without "`Dept.`")

   Field 1 of **names** (last name)

   Field 2 of **phonedir** (telephone number)

   If **phonedir** contains:

```
Binst           555-6235
Dickerson       555-1842
Eisner          555-1234
Green           555-2240
Hrarii          555-0256
Janatha         555-7358
Lewis           555-3237
Takata          555-5341
Wozni           555-1234
```

   and **names** contains:

```
Eisner          Dept. 389
Frost           Dept. 217
Green           Dept. 311
Takata          Dept. 454
Wozni           Dept. 520
```

   then **join -o 2.3,2.1,1.2 phonedir names** displays:

```
389     Eisner  555-1234
311     Green   555-2240
454     Takata  555-5341
520     Wozni   555-1234
```

4. To perform a **join** operation on a field other than the first field, enter:

   sort -b +1 -2 phonedir | join -1 2 - numbers

   This command combines the lines in **phonedir** and **names**, comparing the second field of **phonedir** to the first field of **numbers**.

   First, this command sorts **phonedir** by the second field, because both files must be sorted by the join field. The output of **sort** is then piped to **join**. The **-** (dash) by itself causes the **join** command to use this sorted output as its *file1*. **-1 2** defines the second field of the sorted **phonedir** as the join field. This field is compared to the first field of **numbers** (its join field is not specified with a **-2** flag).

   If **phonedir** contains:

```
Binst           555-6235
Dickerson       555-1842
Eisner          555-1234
Green           555-2240
Hrarii          555-0256
Janatha         555-7358
Lewis           555-3237
```

```
Takata          555-5341
Wozni           555-1234
```

and **numbers** contains:

```
555-0256
555-1234
555-5555
555-7358
```

then **sort ... | join ...** displays:

```
555-0256        Hrarii
555-1234        Eisner
555-1234        Wozni
555-7358        Janatha
```

Each telephone number in **numbers** is listed with the name listed in **phonedir** for that telephone number. Note that **join** lists all the matches for a given field. In this case, **join** lists both `Eisner` and `Wozni` as having the telephone number `555-1234`. The telephone number `555-5555` is not listed, because it does not appear in **phonedir**.

EXIT VALUES

The **join** command returns the following values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **awk(1)**, **cmp(1)**, **comm(1)**, **cut(1)**, **diff(1)**, **grep(1)**, **paste(1)**, **sort(1)**, **uniq(1)**.

Files: **locale(4)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification. The obsolescent version of the command does not conform to the XPG4 Version 2 specification, and its **-j** flag should be considered a temporary extension to the XPG4 Version 2 specification.

Section 5. User Commands (k - l)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letters **k** through **l**.

NAME

kill - Sends a signal to a running process

SYNOPSIS

kill -l [*exit_status*]

kill -s *signal_name process_ID ...*

Obsolescent Versions

kill -s *-signal_name | -signal_number process_ID ...*

The **kill** command sends a signal to one or more running processes.

FLAGS

-l [*exit_status*]

Lists all supported signal names. If the *exit_status* argument is specified and it is a value corresponding to an OSS process terminated by a signal, the name of the signal that terminated the OSS process is listed.

-s *signal_name*

Specifies the signal to send, using one of the symbolic names defined for required signals or job control signals. Values of *signal_name* are recognized in both uppercase and lowercase letters and with or without the prefix **SIG**. The symbolic name **0** (zero), which represents the value 0 (zero), is also recognized. The corresponding signal is sent instead of the **SIGTERM** signal.

-signal_name

Equivalent to **-s** *signal_name*. (Obsolescent.)

-signal_number

Specifies a nonnegative decimal integer representing the signal to be used instead of the **SIGTERM** signal as the *sig* argument in the call to the **kill** command for OSS process IDs. (Obsolescent.)

The effects of specifying any signal number other than those listed by the command **kill -l** is unknown. In the obsolescent versions, if the first argument is a negative integer, it is interpreted as an option to **-signal_number**, not as a negative **process_ID** operand specifying a process group.

DESCRIPTION

The **kill** command sends a signal to one or more running processes, specified by each *process_ID* argument.

The OSS shell contains a built-in command named **kill** that functions in the same way as the regular OSS command named **kill**, except that a new shell process is started for each execution of the regular form of **kill**. The shell built-in version is the default. Both the regular form and the shell built-in form of the **kill** command are described in this reference page.

Specify OSS processes to be signaled by giving their process identification number as the *process_ID* argument.

Specify the signal to be sent with the **-s** *signal_name* argument, the **-signal_name** option, or the **-signal_number** option.

If neither the **-s** *signal_name* argument, the **-signal_name** option, nor the **-signal_number** option is specified, the **SIGTERM** signal is sent by default.

All numeric signal specifications except 0 (zero) are obsolescent.

The shell reports the PID of each process running in the background (unless you start more than one process in a pipeline, in which case the shell reports the number of the last process). You can

also use the **ps** command to find the process ID of commands.

The sensitivity level of the target process must equal that of the process sending the signal, unless you have the **sysadmin** command authorization. If you have the **writeupclearance** or **writeupsyshi** base privileges, you can send signals to processes that dominate your process and are dominated by your clearance of the System High sensitivity level, respectively.

Unless you are are operating with superuser authority, the process you want to signal must belong to you. When operating with superuser authority, you can signal any process.

HP Extensions

For each Guardian *process_ID* argument ([**/E***/systemname*]**/G***/process* or [**/E***/systemname*]**/G***/cpu,pin*), the **kill** command performs actions equivalent to the Guardian **PROCESS_STOP** procedure called with the process handle of the given process. This TACL-type stop action can be accessed only by invoking the **kill** command with the *signal_name* set to **SIGGUARDIAN**.

If you identify a process by its CPU and PIN numbers you must use the following form:

/E*/system-name*/**G***/cpu,pin*

Specifying a Guardian process ID with any of the other signal names results in an error and an unsuccessful exit.

Special Process IDs

There are several special process IDs (PIDs) you can specify to cause the following special actions:

0	The signal is sent to all processes having a process group ID equal to the process group ID of the sender, except those with PIDs 0 and 1.
-*pid*	The signal is sent to all processes whose process group number is equal to the absolute value of *pid*.

Note that when you specify any negative PID, you must also specify the signal to be sent, even the default signal **SIGTERM**.

Arguments

The **kill** command supports the following arguments:

process_ID	A decimal integer specifying a process or process group to be signaled. Job control job identification notation is applicable only for invocation of the **kill** command in the current shell execution environment. The definition of *process_ID* is true for all signal numbers and values except the **SIGGUARDIAN** signal. If signal name **SIGGUARDIAN** is specified, the *process_ID* operand must be of the following form: [**/E***/systemname*]{**/G***/process* | **/G***/cpu,pin*}
exit_status	A decimal integer that specifies a signal number or the exit status of an OSS process terminated by a signal.

Environment Variables

The following environment variables affect the execution of the **kill** command:

LANG	Determines the locale to use for the locale categories when both the **LC_ALL** variable and the corresponding environment variable (whose name begins with **LC_**) do not specify a locale.

LC_ALL
Determines the locale to be used to override any values for locale categories specified by the settings of the LANG variable or any environment variable whose name begins with **LC_**.

LC_CTYPE
Determines the locale for the interpretation of bytes of text data as characters (for example, a single-byte character rather than a multibyte character in an argument).

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to the standard error file and informational messages written to the standard output file.

Standard Output

When the **-l** flag is not specified, the standard output file is not used.

When the **-l** flag is specified, the symbolic name of each signal is written in the following format:

%s%c,signal_name,separator

where *signal_name* is in uppercase without the prefix **SIG**, and *separator* is a blank space. For the last signal written, *separator* is a newline.

When both the **-l** flag and the *exit_status* argument are specified, the symbolic name of the corresponding signal is written in the following format:

%s,signal_name

EXAMPLES

1. To terminate a given process, enter:

   kill 1095

   This terminates process 1095 by sending it the default **SIGTERM** signal. Note that process **1095** might not actually terminate if you have programmed it to ignore or catch the **SIGTERM** signal.

2. To terminate several processes that ignore the default signal, enter:

   kill -s KILL 17285 15692

   This sends the **SIGKILL** signal to processes 17285 and 15692. **SIGKILL** is a special signal that normally cannot be ignored or caught.

3. To terminate all of your background processes, enter:

   kill 0

   This sends the **SIGTERM** signal to all members of the shell process group. This includes all background processes started with **&**. Although the signal is sent to the shell, it has no effect there because the shell ignores the default signal **15**.

4. To terminate all your processes and log out, enter:

   kill -s KILL 0

   This sends the **SIGKILL** signal to all members of the shell process group. Because the shell cannot ignore **SIGKILL**, this also terminates the login shell and logs you out. If you are using multiple windows, this closes the active window.

5. To send a different signal to a process, enter:

   kill -s USR1 1103

   This sends the **SIGUSR1** signal to process 1103. The action taken on the **SIGUSR1** signal is defined by the particular application you are running. (The name of the **kill** command is misleading because many signals, including **SIGUSR1**, do not terminate processes.)

6. To list the signal names in numerical order, stripped of the prefix **SIG**, enter:

   kill -l

   1) HUP **17) USR2**
   2) INT **18) CHLD**
   3) QUIT **19) bad trap**
   4) ILL **20) STOP**
   5) URG **21) TSTP**
   6) ABRT **22) MEMERR**
   7) IO **23) NOMEM**
   8) FPE **24) MEMMGR**
   9) KILL **25) STK**
   10) bad trap **26) TIMEOUT**
   11) SEGV **27) LIMIT**
   12) WINCH **28) CONT**
   13) PIPE **29) TTIN**
   14) ALRM **30) TTOU**
   15) TERM **31) ABEND**
   16) USR1 **32) GUARDIAN**

   This list may vary from system to system.

7. To stop a Guardian process, enter:

   kill -s GUARDIAN /G/cmon

   This issues a **PROCESS_STOP_()** against the named Guardian process **$cmon**.

8. To stop a Guardian process on another system or to stop a Guardian process that you identify by its cpu,pin, enter:

   kill -99 /E/tsii/G/3,57

   This issues a **PROCESS_STOP_()** against the Guardian process identified by the cpu,pin (3,57) running on the system named \tsii.

FILES

/usr/include/signal.h Specifies signal names.

NOTES

The OSS **kill** command has both a shell built-in version and a regular version. The two versions have the same features and functionality. The only difference between the two versions is that the shell built-in version does not start a new shell process when it is invoked. Both versions are described in the reference page for **kill**. The shell built-in version is the default. To specify the regular version, use the full pathname: **/bin/kill** For more information about shell built-in commands, refer to the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **ps(1)**, **sh(1)**.

Functions: **kill(2)**.

STANDARDS CONFORMANCE

Extensions have been added to the **kill** command in order to support the Guardian environment.

NAME

ksh - Describes the OSS shell

SYNOPSIS

ksh [**-i**] [**-c** *command_string* | **-s**] [**+** | **-abCefmnosuvx**] [**+** | **-o**][*option* ...] | [*argument* ...] | [*file*] [*argument* ...]

The OSS shell is an interactive command interpreter and a command programming language. The OSS shell is based on the UNIX Korn shell.

FLAGS

-c *command_string*
Causes **ksh** to read commands from *command_string*.

-i Causes **ksh** to run as an interactive shell. The **SIGTERM** signal is thus ignored, and the **SIGINT** signal is caught, causing the current command to be terminated and a new prompt to be output.

-r Causes **ksh** to run as a restricted shell.

-s Causes **ksh** to read commands from standard input. If you do not specify the **-c** flag or do not specify any arguments to **ksh** other than flags, **ksh** automatically invokes the **-s** flag. The **-c** flag overrides the **-s** flag, however.

The rest of the flags that can be used with **ksh** are described under the **set** subcommand, in the subsection **Special ksh Commands**.

DESCRIPTION

ksh is an alias for **sh**.

The OSS shell carries out commands either interactively from a terminal keyboard or from a file.

Some important features of the OSS shell are as follows:

- Command aliasing
- Filename substitution
- Tilde substitution
- Command substitution
- Parameter substitution
- Job control
- Inline editing

A file from which the shell carries out commands is usually called a *shell script*, a *shell procedure*, or a *command file*.

A *simple command* is a sequence of words separated by spaces or tabs. A *word* is a sequence of characters that contains no unquoted spaces or tabs. The first word in the sequence (numbered as 0) usually specifies the name of a command. Any remaining words, with a few exceptions, are passed to that command. A *space* refers to both spaces and tabs.

The *value* of a simple command is its exit value if it ends normally or (octal) 200 added to the signal number if it terminates due to a signal. For a list of *status* values, see the **signal()** system call.

A *pipeline* is a sequence of one or more commands separated by a | (vertical bar). In a pipeline, the standard output of each command becomes the standard input of the next command. Each

command runs as a separate process, and the shell waits for the last command to end. A *filter* is a command that reads its standard input, transforms it in some way, then writes it to its standard output. A pipeline normally consists of a series of filters. Although the processes in a pipeline (except the first process) can execute in parallel, they are synchronized to the extent that each program needs to read the output of its predecessor.

The exit value of a pipeline is the exit value of the last command.

A *list* is a sequence of one or more pipelines separated by ; (semicolon), **&** (ampersand), **&&** (two ampersands), or || (two vertical bars) and optionally ended by a ; (semicolon), an **&** (ampersand), a **|&** (coprocess), or a newline. These separators and terminators have the following effects:

; Causes *sequential execution* of the preceding pipeline; the shell waits for the pipeline to finish.

& Causes *asynchronous execution* of the preceding pipeline; the shell does *not* wait for the pipeline to finish.

&& Causes the list following it to be executed *only* if the preceding pipeline returns a 0 (zero) exit value.

|| Causes the list following it to be executed *only* if the preceding pipeline returns a nonzero exit value.

The **cd** command is an exception; if it returns a nonzero exit value, no subsequent commands in a list are executed, regardless of the separator characters.

The **;** and **&** separators have equal precedence, as do **&&** and ||. The single-character separators have lower precedence than the double-character separators. An unquoted newline character following a pipeline functions the same as a ; (semicolon).

Access Control Lists (ACLs)

If the shell creates a file, for example when you redirect **stdout** or **stderr** to a file, and the parent directory for that file has an ACL that contains default ACL entries, the file inherits ACL entries from the parent directory as described in the **acl(5)** reference page.

Comments

The shell treats as a comment any word that begins with a **#** character and ignores that word and all characters following up to the next newline character.

Shell Flow Control Statements

Unless otherwise stated, the value returned by a command is that of the last simple command executed in the command.

for *identifier* [**in** *word*...] **;do** *list* **;done**

Each time a **for** command is executed, *identifier* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, the **for** command executes the **do** *list* once for each positional parameter that is set. (See **Parameter Substitution**.) Execution ends when there are no more words in the list.

select *identifier* [**in** *word*...] **;do** *list* **;done**

Prints on standard error (file descriptor 2) the set of words, each word preceded by a number. If **in** *word*... is omitted, then the positional parameters are used instead. (See **Parameter Substitution**.) The **PS3** prompt is printed and a line is read from the standard input. If this line consists of the number of one of the listed words, then the value of the parameter *identifier* is set to the word corresponding to this number. If this line is empty, the selection list is printed again. If neither of these cases is true, the value of the parameter *identifier* is set to null. The contents of the line read from standard input

is saved in the **REPLY** parameter. The *list* is executed for each selection until a **break** or end-of-file character is encountered.

case *word* **in** [[**(**]
pattern [| *pattern*] ...**)** *list* **;;**] ... **esac**" Executes the list associated with the first pattern that matches *word*. The form of the patterns is the same as that used for filename generation. (See **Filename Generation**.)

if *list* **;then** *list* [**elif**
list **;then** *list*] ... [**;else** *list*] **;fi**" Executes the list following **if** and, if it returns a 0 (zero) exit status, executes the list following the first **then**. Otherwise, the list following **elif** is executed and, if its value is 0 (zero), the list following the next **then** is executed. Failing that, the **else** list is executed. If no **else** list or **then** list is executed, then the **if** command returns a 0 (zero) exit status.

while *list* **;do** *list* **;done**

until *list* **;do** *list* **;done**
Executes the **while** list repeatedly, and if the exit status of the last command in the list is 0 (zero), executes the **do** list; otherwise the loop terminates. If no commands in the **do** list are executed, then the **while** command returns a 0 (zero) exit status; **until** can be used in place of **while** to negate the loop termination test.

(*list***)** Executes *list* in a separate environment. Note that if two adjacent open parentheses are needed for nesting, a space must be inserted to avoid arithmetic evaluation.

{*list***;**} Executes *list*. Note that unlike the metacharacters **(** and **)**, **{** and **}** are reserved words and must be at the beginning of a line or after a **;** (semicolon) in order to be recognized.

[[*expression*]]
Evaluates *expression* and returns a 0 (zero) exit status when *expression* is TRUE. See **Conditional Expressions** for a description of *expression*.

function *identifier* **{***list***;}**

identifier **() {***list***;}**
Defines a function that is referenced by *identifier*. The body of the function is the list of commands between **{** and **}**. (See **Functions**.)

time *pipeline*
Executes *pipeline* and prints the elapsed time as well as the user and system time on standard error.

The following reserved words are recognized only when they appear, without single or double quotation marks, as the first word of a command:

case
do
done
elif
else
esac
fi
for
function
if
select
then

time
until
while
{ }
[[ ]]

Command Aliasing

The first word of each command is replaced by the text of an alias (if an alias for this word was defined). The first character of an alias name can be any nonspecial printable character, but the rest of the characters must be the same as for a valid *identifier*. The replacement string can contain any valid shell script, including the metacharacters previously listed. The first word of each command in the replaced text, other than any that are in the process of being replaced, will be tested for aliases. If the last character of the alias value is a space, the word following the alias will also be checked for alias substitution.

Aliases can be used to redefine special built-in commands but cannot be used to redefine the reserved words previously listed. Aliases can be created, listed, and exported with the **alias** command and can be removed with the **unalias** command. Exported aliases remain in effect for scripts invoked by name, but must be reinitialized for separate invocations of the shell. (See **Invocation**.)

Aliasing is performed when scripts are read, not while they are executed. Therefore, for an alias to take effect, the **alias** definition command has to be executed before the command that references the alias is read.

Aliases are frequently used as shorthand for full pathnames. An option to the aliasing facility allows the value of the alias to be automatically set to the full pathname of the corresponding command. These aliases are called *tracked* aliases.

The value of a tracked alias is defined the first time the corresponding command is looked up and becomes undefined each time the **PATH** environment variable is reset. These aliases remain tracked so that the next subsequent reference will redefine the value. Several tracked aliases are compiled into the shell. The **-h** flag of the **set** command makes each referenced command name into a tracked alias.

The following exported aliases are compiled into the shell, but can be unset or redefined:

autoload='typeset -fu'
command='command '
functions='typeset -f'
hash='alias -t -'
history='fc -l'
integer='typeset -i'
local=typeset
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
type='whence -v'

Tilde Substitution

After alias substitution is performed, each word is checked to see if it begins with an unquoted ~ (tilde). If it does, then the word up to a / (slash) is checked to see if it matches a username in the **/etc/passwd** file. If a match is found, the tilde and the matched name are replaced by the login directory of the matched user. This is called a *tilde substitution*. If no match is found, the

original text is left unchanged. A tilde by itself, or in front of a /, is replaced by the value of the **HOME** parameter. A tilde followed by a **+** (plus sign) or **-** (dash) is replaced by **$PWD** and **$OLDPWD**, respectively.

In addition, tilde substitution is attempted when the value of a variable assignment parameter begins with a tilde.

Command Substitution

The standard output from a command enclosed in parentheses preceded by a dollar sign **$( )** or a pair of **‘‘** (grave accents) can be used as part or all of a word; trailing newlines are removed. In the second (archaic) form, the string between the grave accents is processed for special quoting characters before the command is executed. (See **Quoting**.) The command substitution **$(cat** *file*) can be replaced by the equivalent but faster **$(<***file*). Command substitution of most special commands that do not perform input/output redirection are carried out without creating a separate process. An arithmetic expression enclosed in double parentheses preceded by a dollar sign (**$(( ))**) is replaced by the value of the arithmetic expression within the double parentheses.

Parameter Substitution

A *parameter* is an identifier, one or more digits, or any of the characters *****, **@@@@**, **#**, **?**, **-**, **$**, and **!**. A *named parameter* (a parameter denoted by an identifier) has a *value* and 0 (zero) or more *attributes*. Named parameters can be assigned values and attributes by using the **typeset** special command. The attributes supported by the shell are described later with the **typeset** special command. Exported parameters pass values and attributes to the environment.

The shell supports a 1-dimensional array facility. An element of an array parameter is referenced by a *subscript*. A subscript is denoted by an *arithmetic expression* enclosed with [ ] (brackets). To assign values to an array, use values of subscripts in the range of 0 to 1023. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array will be created if necessary. Referencing an array without a subscript is equivalent to referencing the element 0 (zero).

The value of a named parameter can be assigned by the following:

name=*value* [*name*=*value*]

If the integer attribute, **-i**, is set for *name*, the value is subject to arithmetic evaluation. *Positional parameters*, which are denoted by a number, can be assigned values with the **set** special command. Parameter **$0** is set from argument 0 (zero) when the shell is invoked. The **$** (dollar sign) character is used to introduce substitutable parameters.

${*parameter***}**

Reads all the characters from the **${** (dollar sign left brace) to the matching **}** (right brace) as part of the same word even if it contains braces or metacharacters. The value, if any, of the parameter is substituted. The braces are required when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name or when a named parameter is subscripted. If *parameter* is one or more digits, it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces. If *parameter* is ***** (asterisk) or **@@@@** (at sign), all the positional parameters, starting with **$1**, are substituted (separated by a field separator character). If an array identifier with subscript ***** or **@@@@** is used, the value for each of the elements is substituted (separated by a field separator character).

${#*parameter***}**

Substitutes the number of positional parameters if *parameter* is ***** or **@@@@**; otherwise, the length of the value of the *parameter* is substituted.

${#*identifier*[*****]**}**
Substitutes the number of elements in the array *identifier*.

${*parameter***:-***word***}**
Substitutes the value of *parameter* if it is set and non-null; otherwise, substitutes *word*.

${*parameter***:=***word***}**
Sets *parameter* to *word* if it is not set or is null; the value of the parameter is then substituted. Positional parameters cannot be assigned values in this way.

${*parameter***:?***word***}**
Substitutes the value of *parameter* if it is set and is non-null; otherwise, print *word* and exit from the shell. If *word* is omitted, a standard message is printed.

${*parameter***:+***word***}**
Substitutes *word* if *parameter* is set and is non-null; otherwise, substitutes nothing.

${***parameter***#***pattern***}** | **${*parameter***##***pattern***}**
Causes the value of this substitution to be the value of *parameter* with the matched portion deleted if the shell *pattern* matches the beginning of the value of *parameter*; otherwise the value of *parameter* is substituted. In the first form, the smallest matching pattern is deleted and in the second form, the largest matching pattern is deleted.

${***parameter***%***pattern***}** | **${*parameter***%%***pattern***}**
Causes the value of this substitution to be the value of *parameter* with the matched part deleted if the shell *pattern* matches the end of the value of *parameter*; otherwise, substitute the value of *parameter*. In the first form, the smallest matching pattern is deleted and in the second form, the largest matching pattern is deleted.

If the **:** (colon) is omitted from the previous expressions, then the shell checks only whether *parameter* is set or not.

In the previous expressions, *word* is not evaluated unless it is to be used as the substituted string, so that, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo ${d:-$(pwd)}
```

The following parameters are automatically set by the shell:

(hash mark)
The number of positional parameters in decimal.

- (dash)
Flags supplied to the shell on invocation or by the **set** command.

? (question mark)
The decimal value returned by the last executed command.

$ (dollar sign)
The process number of this shell.

_ (underscore)
Initially, an absolute pathname of the shell or script being executed as passed in the environment. Subsequently, the value is assigned the last argument of the previous command. This parameter is not set for commands that are asynchronous.

! (exclamation point)
The process number of the last background command invoked.

ERRNO The value of *errno* as set by the most recently failed system call. This value is system dependent and is intended for debugging purposes.

LINENO
The line number of the current line within the script or function being executed.

OLDPWD
The previous working directory set by the **cd** command.

OPTARG
The value of the last option argument processed by the **getopts** special command.

OPTIND
The index of the last option argument processed by the **getopts** special command.

PPID The process number of the parent of the shell.

PWD The present working directory set by the **cd** command.

RANDOM
Each time this parameter is referenced, a random integer, uniformly distributed between 0 and 32767, is generated. The sequence of random numbers can be initialized by assigning a numeric value to **RANDOM**.

REPLY This parameter is set by the **select** statement and by the **read** special command when no arguments are supplied.

SECONDS
Each time this parameter is referenced, the number of seconds since shell invocation is returned. If this parameter is assigned a value, then the value returned upon reference will be the value that was assigned plus the number of seconds since the assignment.

The following parameters are used by the shell:

CDPATH
The search path for the **cd** command.

COLUMNS
If this variable is set, the value is used to define the width of the edit window for the shell edit modes and for printing **select** lists.

EDITOR
If the value of this variable ends in **vi** and the **VISUAL** variable is not set, then the corresponding option (see **set** under **Special sh Commands**) will be turned on.

ENV If this parameter is set, then parameter substitution is performed on the value to generate the pathname of the script that will be executed when the shell is invoked. (See **Invocation**.) This file is typically used for **alias** and **function** definitions.

FCEDIT
The default editor name for the **fc** command.

FPATH The search path for function definitions. This path is searched when a function with the **-u** attribute is referenced and when a command is not found. If an executable file is found, then it is read and executed in the current environment.

IFS Internal field separators, normally spaces, tabs, and newlines that are used to separate command words which result from command or parameter substitution and for separating words with the **read** special command. The first character of the **IFS** parameter is used to separate arguments for the **$*** substitution. (See **Quoting**.)

HISTFILE
If this parameter is set when the shell is invoked, then the value is the pathname of the file that will be used to store the command history. (See **Command Reentry**.)

HISTSIZE
If this parameter is set when the shell is invoked, the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128.

HOME The default argument (home directory) for the **cd** command.

LANG Specifies the locale of your system, which is comprised of three parts: language, territory, and code set. The default locale is the **C** locale, which specifies the value **English** for language, **U.S.** for territory, and **ASCII** for code set.

LC_ALL
Specifies the behavior for all aspects of the locale.

LC_COLLATE
Specifies the collating sequence to use when sorting names and when character ranges occur in patterns. The default value is the collating sequence for American English. If absent, the collating sequence can be taken from the **LANG** parameter. If both **LC_COLLATE** and **LANG** are absent, the ANSI C collating sequence is used.

LC_CTYPE
Specifies the character classification information to use on your system. The default value is American English.

LC_MESSAGES
Specifies the language in which system messages appear and the language that the system accepts for user input of yes and no strings. The default is American English.

LC_MONETARY
Specifies the monetary format for your system. The default value is the monetary format for American English.

LC_NUMERIC
Specifies the numeric format for your system. The default value is the numeric format for American English.

LC_TIME
Specifies the date and time format for your system. The default value is the date and time format for American English.

LINES If this variable is set, the value is used to determine the column length for printing **select** lists. Select lists will print vertically until about two-thirds of **LINES** lines are filled.

LOCPATH
Specifies a series of colon-separated search rules that describe where to look for locales. These rules override the default search path of **/usr/lib/nls/loc**.

NLSPATH
Specifies a list of directories to search to find message catalogs.

PATH The search path for commands. (See **Execution**.) You cannot change **PATH** if executing under **rsh**, except in **.profile**.

PS1 The value of this parameter is expanded for parameter substitution to define the primary prompt string which by default is the **$** (dollar sign). The **!** (exclamation point) in the primary prompt string is replaced by the command number. (See **Command Reentry**.)

PS2 Secondary prompt string, by default > (right angle bracket).

PS3 Selection prompt string used within a **select** loop, by default **#?** (number sign, question mark).

PS4 The value of this parameter is expanded for parameter substitution and precedes each line of an execution trace. If omitted, the execution trace prompt is **+** (plus sign).

SHELL The pathname of the shell is kept in the environment.

TMOUT
If set to a value greater than 0 (zero), the shell terminates if a command is not entered within the prescribed number of seconds after issuing the **PS1** prompt. (Note that the shell can be compiled with a maximum bound for this value that cannot be exceeded.)

TZ Current value for the time zone, if any.

VISUAL
If the value of this variable ends in **vi**, the corresponding option (see the **set** command in **Special ksh Commands**) will be turned on.

The shell gives default values to **PATH**, **PS1**, **PS2**, **TMOUT**, and **IFS**, while **HOME**, **SHELL**, and **ENV** are not set by the shell.

Interpretation of Spaces
After parameter and command substitution, the results of substitutions are scanned for the field separator characters (those found in **IFS**), and split into distinct arguments where such characters are found. Explicit null arguments (**‘‘** or **’’**) are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

Filename Generation
Following substitution, each command *word* is scanned for the characters ***** (asterisk), **?** (question mark), and **[ ]** (brackets), unless the **-f** option was set. If one of these characters appears, the word is regarded as a pattern. The word is replaced with lexicographically sorted filenames that match the pattern. If no filename is found that matches the pattern, the word is left unchanged. When a pattern is used for filename generation, the **.** (dot) character at the start of a filename or immediately following a / (slash), as well as the / character itself, must be matched explicitly. In other instances of pattern matching, the / and **.** are not treated specially.

***** Matches any string, including the null string.

? Matches any single character.

[...] Matches any one of the enclosed characters. In an expression such as [**a-z**], the **-** (dash) means "through" according to the current collating sequence. The collating sequence is determined by the value of the **LC_COLLATE** environment variable. If the first character following the **[** (left bracket) is a **!** (exclamation point), then any character not enclosed is matched. A **-** can be included in the character set by putting it as the first or last character.

A *pattern_list* is a list of one or more patterns separated from each other with a **|** (vertical bar). Composite patterns can be formed with one or more of the following:

?(*pattern_list***)**
Optionally matches any one of the given patterns.

(pattern_list***)**
Matches zero or more occurrences of the given patterns.

+(*pattern_list***)**
Matches one or more occurrences of the given patterns.

@@@@(*pattern_list***)**
Matches exactly one of the given patterns.

!(*pattern_list***)**
Matches anything, except one of the given patterns.

Character Classes

You can use the following notation to match filenames within a range indication:

[:*charclass***:]**

This format instructs the system to match any single character belonging to *charclass*; the defined classes correspond to **ctype( )** subroutines as follows:

alnum
alpha
blank
cntrl
digit
graph
lower
print
punct
space
upper
xdigit

Your locale might define additional character properties, such as the following:

[:vowel:]

The preceding character class could be TRUE for **a**, **e**, **i**, **o**, **u**, or **y**. You could then use **[:vowel]** inside a **set** construction to match any vowel. Refer to **The LC_CTYPE Category** section of the **locale** file format reference page for more information.

Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > <newline> <space> <tab>

Each of the *metacharacters* previously listed has a special meaning to the shell and causes termination of a word unless quoted. A character can be *quoted* (that is, made to stand for itself) by preceding it with a \ (backslash). The pair **newline** is ignored. All characters enclosed between a pair of **‘’** (single quotes) are quoted. A single quote cannot appear within single quotes.

Inside "" (double quotes) parameter and command substitution occurs and \ quotes the characters \, **‘**, **’**, and **$**. The meaning of **$*** and **$@@@@** is identical when not quoted or when used as a parameter assignment value or as a filename. However, when used as a command argument, **’$*’** is equivalent to **’$1***d***$2***d*. . .**’**, where *d* is the first character of the **IFS** parameter, whereas **’$@@@@’** is equivalent to **’$1’ ’$2’** . . . Inside **‘‘** (grave accents) \ (backslash) quotes the characters \, **‘**, and **$**. If the grave accents occur within double quotes, then \ also quotes the **’** (single quote) character.

The special meaning of reserved words or aliases can be removed by quoting any character of the reserved word. The recognition of function names or special command names listed later cannot be altered by quoting them.

Arithmetic Evaluation

An ability to perform integer arithmetic is provided with the **let** special command. Evaluations are performed using *long* arithmetic. Constants are of the form [*base*#]*n*, where *base* is a decimal number between 2 and 36 representing the arithmetic base and *n* is a number in that base. If *base* is omitted, then base 10 is used.

An arithmetic expression uses the same syntax, precedence, and associativity of expression of the C language. All the integral operators, other than **++**, **--**, **?:**, and **,** are supported. Named parameters can be referenced by name within an arithmetic expression without using the parameter substitution syntax. When a named parameter is referenced, its value is evaluated as an arithmetic expression.

An internal integer representation of a named parameter can be specified with the **-i** option of the **typeset** special command. Arithmetic evaluation is performed on the value of each assignment to a named parameter with the **-i** attribute. If you do not specify an arithmetic base, the first assignment to the parameter determines the arithmetic base. This base is used when parameter substitution occurs.

Because many of the arithmetic operators require quoting, an alternative form of the **let** command is provided. For any command that begins with a **((**, all the characters until a matching **))** are treated as a quoted expression. More precisely, **((**...**))** is equivalent to **let** "...".

Note that **((**...**))** is a command with a return value, whereas **$((**...**))** is the way to put the string representation of the value of an arithmetic expression into the command line (that is, it is like a **$** variable).

Prompting

When used interactively, the shell prompts with the value of **PS1** before reading a command. If at any time a newline character is typed and further input is needed to complete a command, then the secondary prompt (that is, the value of **PS2**) is issued.

Conditional Expressions

A *conditional expression* is used with the **[[** compound command to test attributes of files and to compare strings. Word splitting and filename generation are not performed on the words between **[[** and **]]**. Each expression can be constructed from one or more of the following unary or binary expressions:

-a *file* **TRUE**, if *file* exists.

-b *file* **TRUE**, if *file* exists and is a block-special file.

-c *file* **TRUE**, if *file* exists and is a character-special file.

-d *file* **TRUE**, if *file* exists and is a directory.

-f *file* **TRUE**, if *file* exists and is an ordinary file.

-g *file* **TRUE**, if *file* exists and has its **setgid** bit set.

-G *file* **TRUE**, if *file* exists and its group ID matches the effective group ID of this process.

-h *file* **TRUE**, if *file* exists and is a symbolic link.

-k *file* **TRUE**, if *file* exists and has its sticky bit set.

-L *file* **TRUE**, if *file* exists and is a symbolic link.

-n *string* **TRUE**, if length of *string* is nonzero.

-o *option*
TRUE, if option named *option* is on.

-O *file* **TRUE**, if *file* exists and is owned by the effective user ID of this process.

-p *file* **TRUE**, if *file* exists and is a FIFO special file or a pipe.

-r *file* **TRUE**, if *file* exists and is readable by current process.

-s *file* **TRUE**, if *file* exists and has size greater than 0 (zero).

-S *file* **TRUE**, if *file* exists and is a socket.

-t *file_des*
TRUE, if file descriptor number *file_des* is open and associated with a terminal device.

-u *file* **TRUE**, if *file* exists and has its **setuid** bit set.

-w *file* **TRUE**, if *file* exists and is writable by current process.

-x *file* **TRUE**, if *file* exists and is executable by current process. If *file* exists and is a directory, then the current process has permission to search in the directory.

-z *string* **TRUE**, if length of *string* is 0 (zero).

file1 **-nt** *file2*
TRUE, if *file1* exists and is newer than *file2*.

file1 **-ot** *file2*
TRUE, if *file1* exists and is older than *file2*.

file1 **-ef** *file2*
TRUE, if *file1* and *file2* exist and refer to the same file.

string **=** *pattern*
TRUE, if *string* matches *pattern*.

string **!=** *pattern*
TRUE, if *string* does not match *pattern*.

string1 < *string2*
TRUE, if *string1* collates before *string2*.

string1 > *string2*
TRUE, if *string1* collates after *string2*.

expression1 **-eq** *expression2*
TRUE, if *expression1* is equal to *expression2*.

expression1 **-ne** *expression2*
TRUE, if *expression1* is not equal to *expression2*.

expression1 **-lt** *expression2*
TRUE, if *expression1* is less than *expression2*.

expression1 **-gt** *expression2*
TRUE, if *expression1* is greater than *expression2*.

expression1 **-le** *expression2*
TRUE, if *expression1* is less than or equal to *expression2*.

expression1 **-ge** *expression2*
TRUE, if *expression1* is greater than or equal to *expression2*.

A compound expression can be constructed from these primitives by using any of the following, listed in decreasing order of precedence.

(*expression***)**
TRUE, if *expression* is **TRUE**. Used to group expressions.

! *expression*
TRUE if *expression* is **FALSE**.

expression1 **&&** *expression2*
TRUE, if *expression1* and *expression2* are both **TRUE**.

expression1 || *expression2*
TRUE, if either *expression1* or *expression2* is **TRUE**.

Input/Output

Before a command is executed, you can redirect its input and output by using a special notation interpreted by the shell. The following can appear anywhere in a simple command or can precede or follow a command and are *not* passed on to the invoked command. Command and parameter substitution occurs before *word* or *digit* is used, except as noted in the following text. Filename generation occurs only if the pattern matches a single file and interpretation of spaces is not performed.

<*word* Use file *word* as standard input (file descriptor 0).

>*word* Use file *word* as standard output (file descriptor 1). If the file does not exist, it is created. If the file exists, and the **noclobber** option is on, this causes an error; otherwise, it is truncated to 0 (zero) length.

>|*word* Same as >, except that it overrides the **noclobber** option.

>>*word* Use file *word* as standard output. If the file exists, output is appended to it (by first seeking to the End-of-File); otherwise, the file is created.

<>*word* Open file *word* for reading and writing as standard input.

<<[-]*word*
The shell input is read up to a line that is the same as *word*, or to an End-of-File. No parameter substitution, command substitution, or filename generation is performed on *word*. The resulting document, called a *here document*, becomes the standard input. If any character of *word* is quoted, then no interpretation is placed upon the characters of the document; otherwise, parameter and command substitution occurs, **\newline** is ignored, and \ must be used to quote the characters \, **$**, **‘**, and the first character of *word*. If **-** is appended to <<, then all leading tabs are stripped from *word* and from the document.

<**&***digit* The standard input is duplicated from file descriptor *digit* (see **dup(2)**). The standard output is duplicated using **>&** *digit*.

<**&-** The standard input is closed. The standard output is closed using **>&-**.

<**&p** The input from the coprocess (or background process) is moved to standard input.

>**&p** The output to the coprocess is moved to standard output.

If one of the preceding redirections is preceded by a digit, then the file descriptor number referred to is that specified by the digit (instead of the default 0 or 1). For example:

```
... 2>&1
```

means file descriptor 2 is to be opened for writing as a duplicate of file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates each redirection in terms of the (*file descriptor*, *file*) association at the time of evaluation. For example:

```
... 1>fname >&1
```

first associates file descriptor 1 with file **fname**. It then associates file descriptor 2 with the file associated with file descriptor 1 (that is, **fname**). If the order of redirections is reversed, file descriptor 2 is associated with the terminal (assuming file descriptor 1 is) and then file descriptor 1 is associated with file **fname**.

If a command is followed by **&** and job control is not active, the default standard input for the command is the empty **/dev/null** file. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Environment

The *environment* is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The names must be identifiers and the values are character strings. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value and marking it **export**. Executed commands inherit the environment. If you modify the values of these parameters or create new ones, using the **export** or **typeset -x** commands, they become part of the environment. The environment used by any executed command is thus composed of any name-value pairs originally inherited by the shell, whose values can be modified by the current shell, plus any additions that must be noted in the **export** or **typeset -x** commands.

You can augment the environment for any simple command or function by prefixing it with one or more parameter assignments. A parameter assignment argument is a word of the form *identifier=value*.

Thus, the following two expressions are equivalent (as far as the execution of *command* is concerned):

TERM=450 *command argument ...*

(export TERM; TERM=450; *command argument ...*)

Functions

The **function** reserved word is used to define shell functions. Shell functions are read in and stored internally. Alias names are resolved when the function is read. Functions are executed like commands with the arguments passed as positional parameters. (See **Execution**.)

Functions execute in the same process as the caller and share all files and the present working directory with the caller. Traps caught by the caller are reset to their default action inside the function. A trap condition that is not caught or ignored by the function causes the function to terminate and the condition to be passed on to the caller. A trap on **EXIT** set inside a function is executed after the function completes in the environment of the caller. Ordinarily, variables are shared between the calling program and the function. However, the special command **typeset** used within a function defines local variables whose scope includes the current function and all functions it calls.

The special command **return** is used to return from function calls. Errors within functions return control to the caller.

Function identifiers can be listed with the **-f** or **+f** option of the **typeset** special command. The text of functions is also listed with **-f**. Functions can be undefined with the **-f** option of the **unset** special command.

Ordinarily, functions are unset when the shell executes a shell script. The **-xf** option of the **typeset** command allows a function to be exported to scripts that are executed without a separate invocation of the shell. Functions that need to be defined across separate invocations of the shell should be specified in the **ENV** file with the **-xf** option of **typeset**.

Jobs

If the **monitor** option of the **set** command is turned on, an interactive shell associates a job with each pipeline. It keeps a table of current jobs, printed by the **jobs** command, and assigns them small integer numbers. When a job is started asynchronously with **&**, the shell prints a line that looks like:

```
[1] 1234
```

This line indicates that the job, which was started asynchronously, was job number 1 and had one (top-level) process, whose process ID was 1234.

If you are running a job and want to do something else, you can enter the Suspend key sequence (normally **<Ctrl-z>**), which sends a **SIGTSTP** signal to the current job. The shell then normally indicates that the job has been stopped, and it prints another prompt. You can then manipulate the state of this job, putting it in the background with the **bg** command, or run some other commands and then eventually bring the job back into the foreground with the foreground command **fg**. The job suspension takes effect immediately, and corresponds to the Interrupt key sequence in that pending output and unread input are discarded. A special key sequence, **<Ctrl-y>**, does not generate a **SIGTSTP** signal until a program attempts to read it. (See the **read(2)** reference page for more information.) This key sequence can usefully be typed ahead when you have prepared some commands for a job that you wish to stop after it has read them.

A job being run in the background will stop if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by issuing the **stty tostop** command. If you set this **tty** option, then background jobs will stop when they try to produce output like they do when they try to read input.

There are several ways to refer to jobs in the shell. A job can be referred to by the process ID of any process of the job, or by one of the following:

%*job_number*
The job with the given number.

%*string* Any job whose command line begins with *string*.

%?*string*
Any job whose command line contains *string*.

%% Current job.

%+ Equivalent to **%%**.

%- Previous job.

This shell knows immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work.

When the monitor mode is on, each background job that is completed triggers any trap set for **CHLD**.

When you try to leave the shell while jobs are stopped or running, you are warned that `You have stopped(running) jobs.` You can use the **jobs** command to see what they are. If you do this or immediately try to exit again, the shell does not warn you a second time, and the stopped jobs are terminated.

Signals

The **SIGINT** and **SIGQUIT** signals for an invoked command are ignored if the command is followed by **&** and job **monitor** option is not active. Otherwise, signals have the values inherited by the shell from its parent (but see also the **trap** command).

Execution

Each time a command is executed, the previous substitutions are carried out. If the command name matches one of the shell built-in commands it is executed within the current shell process. Next, the command name is checked to see if it matches one of the user-defined functions. If it does, the positional parameters are saved and then reset to the arguments of the function call. When the function is completed or issues a **return**, the positional parameter list is restored and any trap set on **EXIT** within the function is executed. The value of a function is the value of the last command executed. A function is also executed in the current shell process. If a command name is not a special command or a user-defined function, a process is created and an attempt is made to execute the command via **exec**.

The **PATH** shell parameter defines the search path for the directory containing the command. Alternative directory names are separated by a **:** (colon). The default path is **/usr/bin:** (specifying **/usr/bin**, and the current directory in that order). The current directory can be specified by two or more adjacent colons, or by a colon at the beginning or end of the path list. If the command name contains a / (slash), then the search path is not used. Otherwise, each directory in the path is searched for an executable file.

If the file has execute permission but is not a directory or an **a.out** file, it is assumed to be a file containing shell commands. A subshell is spawned to read it. All nonexported aliases, functions, and named parameters are removed in this case. If the shell command file does not have read

permission, or if the **setuid** and/or **setgid** bits are set on the file, the shell executes an agent whose job it is to set up the permissions and execute the shell with the shell command file passed down as an open file. A command in parentheses is executed in a subshell without the removal of nonexported quantities.

Command Reentry

The text of the last **HISTSIZE** (default 128) commands entered from a terminal device is saved in a history file. The **$HOME/.sh_history** file is used if the **HISTFILE** variable is not set or is not writable. A shell can access the commands of all *interactive* shells that use the same named **HISTFILE**. The **fc** special command is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by giving the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to **fc**, then the value of the **FCEDIT** parameter is used. If **FCEDIT** is not defined, then **/usr/bin/ed** is used. The edited commands are printed and reexecuted upon leaving the editor. The editor name **-** (dash) is used to skip the editing phase and to reexecute the command. In this case, a substitution parameter of the form *old=new* can be used to modify the command before execution. For example, if **r** is aliased to **'fc -e -'**, then typing '**r bad=good c**' reexecutes the most recent command, which starts with the letter **c**, replacing the first occurrence of the string **bad** with the string **good**.

Inline Editing Options

Normally, each command line entered from a terminal device is simply typed followed by a newline character (**<Return>** or linefeed). If the **vi** option is active, you can edit the command line. To be in this edit mode, **set** the corresponding option. An editing option is automatically selected each time the **VISUAL** or **EDITOR** variable is assigned a value ending in the option name.

The editing features require that the terminal accept **<Return>** as carriage-return without linefeed and that a space must overwrite the current character on the screen. ADM terminal users should set the space-advance switch to **Space**. Hewlett-Packard series 2621 terminal users should set the straps to **bcGHxZ etX**.

The editing modes create the impression that the user is looking through a window at the current line. The window width is the value of **COLUMNS** if it is defined, otherwise it is 80 bytes. If the line is longer than the window width minus 2, a mark is displayed at the end of the window to notify the user. As the cursor moves and reaches the window boundaries, the window will be centered about the cursor. The mark is a **>** (right angle bracket) if the line extends on the right side of the window, a **<** (left angle bracket) if the line extends on the left side of the window, and an ***** (asterisk) if the line extends on both sides of the window.

The search commands in each edit mode provide access to the history file. Only strings are matched, not patterns, although if the leading character in the string is a **^** (circumflex), the match is restricted to begin at the first character in the line.

The vi Editing Mode

There are two typing modes. Initially, when you enter a command you are in the **input** mode. To edit, the user enters **control** mode by typing **<Esc>** (ASCII **033**) and moves the cursor to the place needing correction and then inserts or deletes characters or words as needed. Most control commands accept an optional repeat count prior to the command. When in **vi** mode on most systems, canonical processing is initially enabled and the commands are echoed again if the speed is 1200 baud or greater, if it contains any control characters, or if less than 1 second has elapsed since the prompt was printed. The Escape character terminates canonical processing for the remainder of the command and the user can then modify the command line.

This scheme of using two typing nodes has the advantages of canonical processing with the type-ahead echoing of raw mode. If the option **viraw** is also set, the terminal always has canonical processing disabled. This mode is implicit for systems that do not support two alternative End-of-Line delimiters, and can be helpful for certain terminals.

Input Edit Commands

By default the editor is in input mode.

Erase (User-defined Erase character as defined by the **stty** command, often **<Ctrl-h>** or **#**.) Deletes the previous character.

<Ctrl-w>
Deletes the previous space-separated word.

<Ctrl-d>
Terminates the shell (at the beginning of a line only).

<Ctrl-v>
Escapes the next character. Editing characters and the user's Erase or Kill characters can be entered in a command line or in a search string if preceded by a **<Ctrl-v>**. **<Ctrl-v>** removes the next character's editing features (if any).

\ Escapes the next Erase or Kill character.

Motion Edit Commands

These commands move the cursor:

[*count*]**l** Cursor forward (right) one character.

[*count*]**w** Cursor forward one word. A word is a string of characters delimited by spaces or tabs.

[*count*]**W**
Cursor to the beginning of the next word that follows a space.

[*count*]**e** Cursor to the end of the word.

[*count*]**E** Cursor to end of the current space-delimited word.

[*count*]**h** Cursor backward (left) one character.

[*count*]**b** Cursor backward one word.

[*count*]**B** Cursor to the preceding space-delimited word.

[*count*]| Cursor to the column *count*.

[*count*]**f***c* Finds the next character *c* in the current line.

[*count*]**F***c*
Finds the previous character *c* in the current line.

[*count*]**t***c* Equivalent to **f** followed by **h**.

[*count*]**T***c*
Equivalent to **F** followed by **l**.

[*count*]**;** Repeats *count* times, the last single character find command: **f**, **F**, **t**, or **T**.

[*count*]**,** Reverses the last single character find command *count* times.

0 Cursor to the start of the line.

^ Cursor to the first nonspace character in the line.

$ Cursor to the end of the line.

Search Edit Commands

These commands access your command history.

[*count*]**k** Fetches the previous command. Each time **k** is entered, the previous command back in time is accessed.

[*count*]**-** Equivalent to **k**.

[*count*]**j** Fetches the next command. Each time **j** is entered, the next command forward in time is accessed.

[*count*]**+** Equivalent to **j**.

[*count*]**G** Fetches the command number *count*. The default is the least recent **history** command.

/string Searches backward through history for a previous command containing the specified *string*. *string* is terminated by **<Return>** or a newline character. If the specified string is preceded by a ^ (circumflex), the matched line must begin with *string*. If *string* is null, the previous string is used.

?*string* Same as / (slash) except that the search is in the forward direction.

n Searches for next match of the last pattern to the / or **?** commands.

N Searches for next match of the last pattern to the / or **?** commands, but in reverse direction. Searches the command history for the *string* entered by the previous / command.

Text Modification Edit Commands

These commands modify the line.

a Enters input mode and enters text after the current character.

A Appends text to the end of the line. Equivalent to **$a**.

[*count*]**c***motion*

c[*count*]*motion*
Deletes the current character through the character to which *motion* would move the cursor, and enters input mode. If *motion* is **c**, the entire line is deleted and input mode is entered.

C Deletes the current character through the end of line, and enters input mode. Equivalent to **c$**.

S Equivalent to **cc**.

D Deletes the current character through the end of line. Equivalent to **d$**.

[*count*]**d***motion*

d[*count*]*motion*
Deletes the current character through the character to which *motion* would move. If *motion* is **d**, the entire line is deleted.

i Enters input mode and inserts text before the current character.

I Inserts text before the beginning of the line. Equivalent to **0i**.

[*count*]**P** Places the previous text modification before the cursor.

[*count*]**p** Places the previous text modification after the cursor.

R Enters input mode and replaces characters on the screen with the characters you type, overlay fashion.

[*count*]**r***c*
Replaces the *count* characters, starting at the current cursor position with *c* and advancing the cursor.

[*count*]**x** Deletes the current character.

[*count*]**X** Deletes the preceding character.

[*count*]**.** Repeats the previous text modification command.

[*count*]**~** Inverts the case of the *count* characters, starting at the current cursor position and advancing the cursor.

[*count*]_ Causes the *count* word of the previous command to be appended and input mode entered. The last word is used if *count* is omitted.

***** Causes an ***** (asterisk) to be appended to the current word and filename generation to be attempted. If no match is found, it sounds the bell. Otherwise, the word is replaced by the matching pattern and input mode is entered.

\ Filename completion. Replaces the current word with the longest common prefix of all filenames matching the current word with an * (asterisk) appended. If the match is unique, a / (slash) is appended if the file is a directory; a space is appended if the file is not a directory.

Miscellaneous vi Commands

[*count*]**y***motion*

y[*count*]*motion*
Yanks the current character through the character to which *motion* would move the cursor and puts the characters into the delete buffer. The text and cursor are unchanged.

Y Yanks from current position to the end of line. Equivalent to **y$**.

u Undoes the last text-modifying command.

U Undoes all the text-modifying commands performed on the line.

[*count*]**v** Returns the command **fc -e vi** *count* in the input buffer. If *count* is omitted, the current line is used.

<Ctrl-l> Performs a linefeed and prints the current line. Effective only in control mode.

<Ctrl-j> Executes the current line, regardless of mode (newline).

<Ctrl-m>
Executes the current line, regardless of mode (enter).

Sends the line after inserting a **#** (number sign) in front of the line. Useful for causing the current line to be inserted in the history without being executed.

= Lists the filenames that match the current word if an * (asterisk) is appended to it.

@@@@*letter*
Searches the alias list for an alias by the name *_letter* . If an alias of this name is defined, its value is inserted in the input queue for processing.

Special sh Commands

Shell built-in commands are executed by the OSS shell and run entirely within the shell process. A subshell process is not created for shell built-in commands as it is for a command that is not a shell built-in command.

The following shell built-in commands also have counterparts that are regular OSS commands having the same names:

echo.1
kill.1
pwd.1
read.1

The shell built-in command is the default. To run the regular version of a command (instead of the shell built-in version) specify the command as follows:

/bin/*command_name*

To make the regular version the default, create an alias to the regular version.

The shell built-in version and the regular version of a command may not behave the same way or have the same flags.

The shell commands described below are executed in the shell process. Input/output redirection is permitted.

DESCRIPTION

:[*argument* ...]
The command only expands arguments. It is used when a command is needed, as in the **then** condition of an **if** command, but nothing is to be done by the command.

Parameter assignment lists that precede the command remain in effect when the command completes.

I/O redirections are processed after parameter assignments.

Errors cause a script that contains the commands so marked to abort.

. *file* [*argument* ...]
Reads the complete file and executes the commands. The commands are executed in the current shell environment. The search path specified by **PATH** is used to find the directory containing *file*. Unlike normal command search, however, the file searched for by the **.** command need not be executable. If any arguments are specified, they become the positional parameters. Otherwise, the positional parameters are unchanged. If no readable file is found, a noninteractive shell aborts; an interactive shell writes a diagnostic message to standard error, but this condition is not considered a syntax error. The exit status is the exit status of the last command executed, or a 0 (zero) if no command is executed.

Parameter assignment lists that precede the command remain in effect when the command completes.

I/O redirections are processed after parameter assignments.

Errors cause a script that contains the commands so marked to abort.

add_define
Creates DEFINEs for the Guardian environment. An HP extension.

alias Creates or lists aliases.

bg Puts each specified *job* into the background.

break Exits from the enclosing **for**, **while**, **until**, or **select** loop.

cd Changes the current directory.

continue
Resumes the next iteration of the enclosing **for**, **while**, **until**, or **select** loop.

del_define
Deletes DEFINEs from the current shell process. An HP extension.

echo Sends the string given as an argument to the standard output.

eval Reads arguments as input to the shell and executes arguments as commands.

exec Executes commands specified as arguments

exit Causes the shell to exit.

export Marks names automatic export to the shell environment.

fc Lists or edits and reexecutes commands previously entered to an interactive shell.

fg Moves processes into the foreground.

getopts Checks argument for legal options.

hash Affects the way the shell remembers the locations of utilities.

history Lists the contents of the **history** file, which contains a list of previously executed commands.

info_define
Displays information about DEFINEs. An HP extension.

jobs Lists information about jobs.

kill Sends either the **TERM** signal or the specified signal to the specified jobs or processes.

let Evaluates arguments as arithmetic expressions.

print The shell output mechanism; prints arguments to standard output as described for **echo**.

pwd Prints the current working directory to standard output.

read The shell input mechanism.

readonly
The variable names given as arguments are marked read only. These names cannot be changed by subsequent assignment.

reset_define
Restores the attributes of a DEFINE to their original settings. An HP extension.

return Causes a shell function to return to the invoking script.

set Sets parameters.

set_define
Sets the values for DEFINE attributes. An HP extension.

shift Renames positional parameters.

show_define
Displays values of DEFINE attributes. An HP extension.

times Prints the accumulated user and system times for the shell and for processes run from the shell.

trap Defines variables to be read and executed when the shell receives the specified signals.

type Returns the location of commands.

typeset Sets attributes and values for shell parameters.

umask Sets the user file-creation mask to *mask*.

unalias Removes alias definions.

unset Erases values assigned to variables.

wait Waits for the specified process and reports its termination status.

whence Indicates how names would be interpreted if used as commands.

Invocation

If the shell is invoked by **exec**, and the first character of argument zero (**$0**) is **-** (dash), the shell is assumed to be a login shell and commands are read from **/etc/profile** and then from either **.profile** in the current directory or **$HOME/.profile**, if either file exists. Next, commands are read from the file named by performing parameter substitution on the value of the **ENV** environment variable, if the file exists. If the **-s** flag is not present and *argument* is present, a path search is performed on the first *argument* to determine the name of the script to execute. The script *argument* must have read permission and any **setuid** and **getgid** settings are ignored. Commands are then read, as described in the following text.

See the **FLAGS** section for a complete description of flags that can be interpreted by the shell when it is invoked.

FILES

/etc/profile	System profile.
$HOME/.profile	User profile.

NOTES

1. If a command is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will execute the original command. Use the **hash** command to correct this situation.

2. When the shell encounters the >> characters, it does not open the file in append mode; instead, the shell opens the file for writing and seeks to the end.
3. Failure (nonzero exit status) of a special command preceding a || symbol prevents the list following || from executing.
4. If a command that is a *tracked alias* is executed, and then a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell continues to **exec** the original command. Use the **-t** flag of the **alias** command to correct this situation.
5. Using the **fc** built-in command within a compound command causes the whole command to disappear from the history file.
6. The built-in **.file** command reads the whole file before any commands are executed. Therefore, the **alias** and **unalias** commands in the file do not apply to any functions defined in the file.
7. Traps are not processed while a job is waiting for a foreground process. Thus, a trap on **CHLD** is not executed until the foreground job terminates.
8. The shell displays the following progress message if it needs to retry the fork operation during an attempt at process creation:

   ```
   sh: Resource temporarily unavailable....
       will retry fork() for MAX 62 secs...
   ```

   If the indicated time passes before the fork operation is successful, the shell returns the following message:

   ```
   /bin/-sh:  sh: tdm_fork() failed with errno EAGAIN:
                  cannot fork too many processes
   ```

EXIT VALUES

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. Otherwise, the shell returns the exit status of the last command executed. (See also the **exit** command, described previously.) If the shell is being used noninteractively, execution of the shell file is abandoned. Run-time errors detected by the shell are reported by printing the command or function name and the error condition. If the line number that the error occurred on is greater than 1, the line number is also printed in **[ ]** (brackets) after the command or function name.

RELATED INFORMATION

Commands: **cd(1)**, **chmod(1)**, **echo(1)**, **env(1)**, **setacl(1)**, **sh(1)**, **stty(1)**, **test(1)**, **umask(1)**, **vi(1)**.

Functions: **exec(2)**, **fcntl(2)**, **fork(2)**, **ioctl(2)**, **lseek(2)**, **pipe(2)**, **rand(3)**, **umask(2)**, **ulimit(3)**, **wait(2)**.

Files: **locale(4)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The following commands are HP extensions to the shell built-in commands of the XPG4 Version 2 specification. They are described in detail in their own reference pages:

add_define

Creates DEFINEs for the Guardian environment.

del_define
Deletes DEFINEs from the current shell process.

info_define
Displays information about DEFINEs.

reset_define
Restores the attributes of a DEFINE to their original settings.

set_define
Sets the values for DEFINE attributes.

show_define
Displays values of DEFINE attributes.

NAME

ld - Runs the TNS/R native linker utility for position-independent code

SYNOPSIS

ld

[**-allow_duplicate_procs**]
[**-allow_missing_libs**]
[**-allow_multiple_mains**]
[**-ansistreams**]
[**-bdllsonly**]
[**-bdynamic**]
[**-bglobalized**]
[**-blocalized**]
[**-bstatic**]
[**-bsymbolic** | **-bsemi_globalized**]
[**-call_shared**]
[**-change** *attribute_name attribute_value filename3*]
[**-d** *address1*]
[**-dll** | **-shared**]
[{ **-dllname** | **-soname** } *DLL_name*]
[**-e** *symbol_name1*]
[**-export** *symbol_name2*]
[**-export_all**]
[**-export_not** *symbol_name3*]
[**-first_l** *pathname1*]
[{ **-fl** | **-obey** } *filename5*]
[**-include_whole** | **-no_include_whole**]
[{ **-l** | **-lib** } *filename6*]
[{ **-L** | **-libvol** } *pathname2*]
[**-libname** *Guardian_filename*]
[**-limit_runtime_paths**]
[**-m** | **-map**]
[**-no_optional_lib** | **-optional_lib**]
[**-no_preset**]
[**-no_reexport** | **-reexport**]
[**-nostdfiles** | **-no_stdfiles**]
[**-nostdlib** | **-no_stdlib**]
[**-noverbose** | **-no_verbose**]
[**-o** *filename7*]
[**-rld_l** *path_list1*]
[**-rld_first_l** *path_list2*]
[**-s**]
[**-set** *attribute_name attribute_value*]
[**-show_multiple_defs**]
[**-stdin**]
[**-strip** *filename9*]
[**-t** *address2*]
[**-temp_o** *filename10*]
[**-u** *symbol_name4*]
[**-ul**]
[**-unres_symbols** { **error** | **ignore** | **warn** }]
[**-verbose**]
[**-warn**]

[**-y** *symbol_name5*]
[*filename13*] ...

FLAGS

-allow_duplicate_procs

Tells **ld** to unconditionally accept multiple copies of a procedure. The only check made is that all copies of the procedure have the same procedure attributes; for example, it is acceptable if they have different sizes. The first copy of the duplicated procedure is the one that is kept. When building an executable file, no space is allocated for the unused copies.

The default action is to accept multiple copies of only procedures specifically marked as duplicatable by C++.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-allow_missing_libs

Tells **ld** not to stop processing when it cannot find an archive or a dynamic-link library (DLL) after searching for the name specified by a **-l** or **-lib** flag. Instead, a warning message is issued and processing continues.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

The default action when a needed DLL or archive cannot be found is to stop processing.

-allow_multiple_mains

Directs **ld** not to issue an error message if more than one procedure has the MAIN attribute. All main procedures are included in the output file. Only the first procedure having the MAIN attribute is listed as the main entry point in the file header.

The default action is to stop processing and report an error when more than one procedure has the MAIN attribute.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-ansistreams Specifies that C run-time library functions create files of file code 180 (C text as binary) instead of file code 101 (EDIT). The type of files created can also be set with the `ANSISTREAMS` C and C++ compiler pragma.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

See the *C/C++ Programmer's Guide* for more information.

-bdllsonly Tells **ld** to limit searches to DLLs when resolving the file names specified for the **-l** and **-lib** flags.

If a file name is qualified, **ld** searches for a DLL with that name.

If a filename is unqualified, in each search path, **ld** first searches for a DLL with the file name as specified in the **-l** or **-lib** flag. If **ld** cannot find a DLL, the file name is unqualified, and the search path is not in the Guardian file system (/G), then **ld** prefixes **lib** and suffixes **.so** to the file name and searches again. If **ld** still cannot find the DLL, it searches the path again with the same prefix but with **.srl** as the suffix. For more information on search paths, see the **Finding Libraries** subsection of this reference page under **DESCRIPTION**.

When a DLL cannot be found, **ld** issues an error message unless the **-allow_missing_libs** flag is specified.

The **-bdllsonly**, **-bdynamic**, and **-bstatic** flags are search control toggles. Multiple flags can be specified in a single **ld** invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** and **-lib** flags and search for just archive files for others. The default library search control is **-bdynamic**.

-bdynamic Directs **ld** to search for DLLs and archive files when resolving the file names specified for the **-l** and **-lib** flags.

If a file name is qualified, **ld** searches for a DLL or archive with that name.

If a filename is unqualified, in each search path, **ld** first searches for a DLL or archive with the file name as specified in the **-l** or **-lib** flag. If **ld** cannot find a DLL or archive, the file name is unqualified, and the search path is not in the Guardian file system (/G), then **ld** prefixes **lib** and suffixes **.so** to the file name and searches again. If **ld** still cannot find the DLL or archive, it searches the path again with the same prefix but with **.srl** as the suffix. If **ld** still cannot find the DLL or archive, it searches the path again with the same prefix but with **.a** as the suffix. For more information on search paths, see the **Finding Libraries** subsection of this reference page under **DESCRIPTION**.

When a DLL or archive cannot be found, **ld** issues an error message unless the **-allow_missing_libs** flag is specified.

The **-bdllsonly**, **-bdynamic**, and **-bstatic** flags are search control toggles. Multiple flags can be specified in a single **ld** invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** and **-lib** flags and search for just archive files for others. The default library search control is **-bdynamic.**

-bglobalized Directs **ld** to use the following sequence as its linker searchList when resolving the file names specified for the **-l** and **-lib** flags:

- At link time:
  1. Libraries on the libList
  2. Breadth-first transitive closure of DLLs on the libList
  3. Implicit libraries
- At load time:
  1. Libraries on the libList
  2. Breadth-first transitive closure of DLLs on the libList
  3. Loader loadList (libraries loaded by the program or libraries that caused this loadfile to be loaded; this list is built from the program's and libraries' libList and a breadth-first transitive closure of the libList-specified libraries)
  4. Implicit libraries

A filename that is either a relative OSS pathname or a Guardian filename that is not qualified is found using search path lists, as described in the **Finding Libraries** subsection of this reference page under **DESCRIPTION**.

You cannot use this flag when you use the **-blocalized**, **-bsemi_localized**, or **-bsymbolic** flag. The default action is the action for the **-blocalized** flag.

-blocalized Directs **ld** to use the following sequence as its linker searchList when resolving the filenames specified for the **-l** and **-lib** flags:

1. Loadfile itself
2. Libraries on the libList
3. Breadth-first transitive closure of re-exported libList-specified DLLs
4. Implicit libraries

This is the default **ld** action.

A filename that is either a relative OSS pathname or a Guardian filename that is not qualified is found using search path lists, as described in the **Finding Libraries** subsection of this reference page under **DESCRIPTION**.

You cannot use this flag when you use the **-bglobalized**, **-bsemi_localized**, or **-bsymbolic** flag.

-bstatic Directs **ld** to search for archive files when resolving the file names specified for the **-l** and **-lib** flags.

If a file name is qualified, **ld** searches for an archive with that name.

If a filename is unqualified, in each search path, **ld** first searches for an archive with the file name as specified in the **-l** or **-lib** flag). If **ld** cannot find an archive, the file name is unqualified, and the search path is not in the Guardian file system (/G), then **ld** prefixes **lib** and suffixes **.a** to the file name and searches again. For more information on search paths, see the **Finding Libraries** subsection of this reference page under **DESCRIPTION**.

When an archive cannot be found, **ld** issues an error message unless the **-allow_missing_libs** flag is specified.

The **-bdllsonly**, **-bdynamic**, and **-bstatic** flags are search control toggles. Multiple flags can be specified in a single **ld** invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** and **-lib** flags and search for just archive files for others. The default library search control is **-bdynamic.**

-bsymbolic | **-bsemi_globalized**
Directs **ld** to use the following sequence as its linker searchList when resolving the file names specified for the **-l** and **-lib** flags:

- At link time:
    1. Loadfile itself
    2. Libraries on the libList
    3. Breadth-first transitive closure of DLLs on the libList
    4. Implicit libraries

- At load time:
    1. Loadfile itself
    2. Libraries on the libList
    3. Breadth-first transitive closure of DLLs on the libList
    4. Loader loadList (libraries loaded by the program or libraries that caused this loadfile to be loaded; this list is built from the program's and libraries' libList and a breadth-first transitive closure of the libList-specified libraries)
    5. Implicit libraries

A filename that is either a relative OSS pathname or a Guardian filename that is not qualified is found using search path lists, as described in the **Finding Libraries** subsection of this reference page under **DESCRIPTION**.

You cannot use this flag when you use the **-bglobalized** or **-blocalized** flag. The default action is the action for the **-blocalized** flag.

-call_shared Tells **ld** to mark the linkfile specified by *filename13* as the loadfile for a program. This is the default **ld** action.

You cannot use this flag when you use the **-dll** or **-shared** flag.

-change *attribute_name attribute_value filename3*

Changes the value of the run-time attribute specified in *attribute_name* to the value specified in *attribute_value* in the existing file specified by *filename3*.

For a linkfile, you can only change the values for **FLOATTYPE**, **FLOAT_LIB_OVERRULE**, and **SYSTYPE**.

You can use the **-set** flag to set an attribute when creating a loadfile. See the **-set** flag for a description of *attribute_name* and *attribute_value*.

You cannot specify other loadfile filenames or flags other than the following with the **-change** flag:

-noverbose, **-verbose**, or **-warn**
-fl or **-obey**
-stdin

The resulting loadfile has the same **ld** timestamp as before.

-d *address1* Specifies the hexadecimal virtual address at which the data area starts. When creating a program file, the default value for *address1* is 08000000. When creating a DLL, the default value is set to the next multiple of 16384 (0x4000) bytes after the end of the text area.

The value specified for *address1* is always hexadecimal and can optionally be prefixed by 0x. The specified value is automatically rounded up to a multiple of 4096 (0x1000) bytes.

If you use this flag, you must also use the **-t** flag. If you use this flag with the **-dll** or **-shared** flag, the value of *address1* must be the next available address modulo 16384 after the text area in the DLL.

HP recommends against using this flag when creating a DLL.

-dll | **-shared** Tells **ld** to mark the loadfile specified by *filename7* as a PIC DLL. When you specify the **-dll** or **-shared** flag, the exported symbols are those exported by the **-export_all** or **-export** flags, or those marked by the compiler to be exported. Any symbols specified by the **-export_not** flag are not exported.

You cannot use this flag when you use the **-call_shared** flag. The default action is the action for the **-call_shared** flag.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

{**-dllname** | **-soname** } *DLL_name*
Tells **ld** the DLL name to store in the DLL being created. When this DLL is specified in the link step of another loadfile, the DLL name stored in this DLL is placed in the libList of the loadfile for later use by **rld** when searching for DLLs.

If the DLL being created will reside in the Guardian file system, *DLL_name* must conform to Guardian filename rules. If the DLL being created will reside in the OSS file system, *DLL_name* must conform to OSS pathname rules. To reside in either file system, *DLL_name* must be an unqualified Guardian file identifier.

If you specify both a *DLL_name* and the **-o** flag, the output loadfile filename is determined by the **-o** specification and *DLL_name* is saved in the DLL being created.

If you specify *DLL_name* but do not use the **-o** flag, the output loadfile filename uses the *DLL_name* value.

If you do not specify *DLL_name* but use the **-o** flag, the output loadfile filename is used as the DLL name stored in the DLL being created. Only the unqualified part (rightmost part) of the output filename is used.

If you omit both a *DLL_name* and the **-o** flag, the output loadfile filename and *DLL_name* in the libList both default to **a.out**. HP recommends against using this default value.

-e *symbol_name1*
Specifies a function identifier. The specified function is the entry point, that is, the point at which to begin executing the program when the program is loaded.

You should use this flag only when linking a program that will execute without standard run-time support facilities and without linking a module such as CCPPMAIN (in the Guardian file system) or **ccppmain.o** (in the OSS file system) that contains a function with the MAIN attribute. Do not use this flag for libraries.

-export *symbol_name2*
Tells **ld** to mark *symbol_name2* for export in the output loadfile in addition to those normally marked. This flag can be used with the **-export_not** flag to create sets of symbols to be exported.

symbol_name2 cannot be the same symbol as *symbol_name3*.

The default action is to export only those symbols marked by a compiler as requiring export.

-export_all Tells **ld** to mark for export in the output loadfile all symbols in the external symbol table that are not one of the following:

- **multiext**

- starting with **__sti__** (global constructors), **__std__** (global destructors), **__INIT__** (initialization functions), or **__TERM__** (termination functions)

This flag can be used with the **-export_not** flag to create a subset of symbols to be exported.

The default action when the **-ul** flag is not used is to export only those symbols marked by a compiler as requiring export.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-export_not *symbol_name3*

Tells **ld** not to mark *symbol_name3* for export in the output loadfile. This flag can be used with the **-export_all** flag to create sets of symbols to be exported.

symbol_name3 cannot be the same symbol as *symbol_name2*.

The default action is to export all symbols marked by a compiler as requiring export.

-first_l *pathname1*

Tells **ld** to use the specified pathname when searching for libraries. *pathname1* is used in library searches before the public libraries are searched.

You cannot embed spaces (blanks) in *pathname1*. A pathname can be either a relative or absolute OSS directory pathname or an unqualified, partially qualified, or fully qualified Guardian subvolume name.

The **-first_l** flag can be specified more than once in a command line or an obey file.

See the **Finding Libraries** subsection under **DESCRIPTION** for details about the effect of this flag on search order.

{**-fl** | **-obey** } *filename5*

Specifies the name of an **ld** command file containing **ld** command tokens (such as filenames and command flag specifications).

filename5 is a C text file. Tokens can be separated by spaces, tabs, or ends of lines. Tokens can contain double quotation marks (") to group items into a single string, consistent with OSS shell usage. Within the command file, two hyphens indicate a comment that extends to the end of the current line. Command files can be nested; there is no limit to the depth of nesting. Recursive nesting does not cause an error; **ld** does not read a command file invoked by itself.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-include_whole | **-no_include_whole**

Tells **ld** whether to include in the loadfile all linkable archive members of all archive libraries encountered after this flag is specified.

Specifying **-include_whole** begins this linking action. When **-no_include_whole** behavior is in effect, archive searches are controlled by the existence of undefined symbols. Archives are searched in the order specified on the command line. Symbols are marked as undefined by compilers or by the user through the **-u** flag. When an archive member is found that resolves an undefined symbol, the member´s symbols are merged into the external symbol

table for the loadfile being created. After the merge, the undefined symbol that triggered the merge is resolved (marked as defined). The same merge might resolve other undefined symbols or result in more undefined symbols.

You can stop the linking action of **-include_whole** by specifying the **-no_include_whole** flag later in the command line or obey file.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

The default setting is **-no_include_whole**.

{ **-l** | **-lib** } *filename6*

Specifies the name of a DLL or archive file to use to resolve external references from the executable file being linked. The **-l** flag must be specified in lowercase type, and the space after the flag and before *filename6* is optional. **-l** is a synonym for **-lib**.

Other flags affect how *filename6* is used. See the **Finding Libraries** subsection under **DESCRIPTION** for details.

{ **-L** | **-libvol** } *pathname2*

Specifies a pathname to search for a DLL or archive file specified by a simple filename in an **-l** or **-lib** flag. A simple filename is an OSS pathname without any directory components. The **-L** flag must be specified in uppercase type, and the space after the flag and before *pathname2* is optional. **-libvol** is a synonym for **-L** and the space before *pathname2* is required.

Other flags affect how *pathname2* is used. See the **Finding Libraries** subsection under **DESCRIPTION** for details.

ld does not verify the names of locations specified in **-L** or **-libvol** flags. If you specify the **-verbose** flag, **ld** writes to its output listing the locations where it found a DLL or archive file.

-libname *Guardian_filename*

Associates a DLL as a native user library with an executable native program file. You can associate a native user library to a program loadfile. The **-set** and **-change** flags can also associate a native user library with an executable native program.

The value specified for *Guardian_filename* cannot be the Guardian name of an OSS file.

-limit_runtime_paths

Tells **ld** to mark the loadfile so that **rld** will omit certain locations when searching for DLLs or archives to resolve symbols. See **Finding Libraries** in the **DESCRIPTION** section of this reference page for more information.

The default action is to search all locations described in **Finding Libraries**.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-m | **-map** Tells **ld** to produce a memory map of the PIC program or DLL being created.

The default behavior does not produce a memory map.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-no_optional_lib | **-optional_lib**
Specifies whether a library specified in the command line or obey file should be considered optional when creating a loadfile.

When **-no_optional_lib** behavior is in effect, any library specified in a **-l** or **-lib** flag is included in the **.liblist** section of the loadfile being created. When **-optional_lib** behavior is in effect, a specified library can be omitted from the **.liblist** section of the loadfile being created if omitting it would not affect how symbolic references are resolved.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

If a library is specified more than once, and at least one specification occurs when **-no_optional_lib** is in effect, the library is included in the **.liblist** section of the loadfile being created.

The default behavior is **-no_optional_lib**.

-no_reexport | **-reexport**
Tells **ld** whether to mark any library specified in a **-l** or **-lib** flag after this flag for reexport in its **libList** entry in the loadfile being created. Specifying **-no_reexport** leaves the library unmarked; specifying **-reexport** marks the library. Reexport is a run-time attribute that is used by **rld** to decide what DLLs it needs to load.

-no_reexport is the default action.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

-nostdfiles | **-no_stdfiles**
Specifies that C run-time library functions do not automatically open the standard input and standard output files.

-nostdlib | **-no_stdlib**
Prevents **ld** from searching the standard library locations for DLLs and archive files.

You can specify these flags as often as you want in the command line or an obey file. Each specification is processed when encountered.

-noverbose | **-no_verbose**
Prevents **ld** from writing warning and informational messages to its output listing. Only error messages and output specifically requested by other options appears in the listing.

The default value is **-no_verbose**.

You can specify the flags **-warn**, **-verbose**, **-noverbose**, and **-no_verbose** as often as you want in the command line or an obey file. The value used is the final value entered.

-o *filename7* Specifies the filename of the output loadfile.

filename7 can be the same as the input file name. When this is true and linking is successful, **ld** deletes the input file and then writes the output file. An error occurs if you do not have permission to delete the input file.

If you do not specify a **-o** flag, the default output loadfile filename depends on

whether a **-dllname** or **-soname** flag is specified. *filename7* can also become the DLL name used for the file in the libList. See the description of the **-dllname** flag in this reference page for more information.

-rld_l *path_list1*

Tells **ld** to set search paths in the loadfile for later use by the **rld** loader. *path_list1* identifies paths to be searched after using the loadfile location and before using the **rld** default locations.

path_list1 contains one or more pathname entries, separated by a colon (**:**); you cannot embed spaces (blanks) in *path_list1*. A pathname can be either an absolute OSS directory pathname or a fully qualified Guardian subvolume name.

The **-rld_l** flag can be specified more than once in a command line or an obey file. Multiple *path_list1* specifications are concatenated into a single loadfile entry.

See the **Finding Libraries** subsection under **DESCRIPTION** for details about the effect of this flag on search order.

-rld_first_l *path_list2*

Tells **ld** to set search paths in the loadfile for later use by the **rld** loader. *path_list2* identifies paths to be searched after using the location specified by **-first_l** and before using the public libraries.

path_list2 contains one or more pathname entries, separated by a colon (**:**); you cannot embed spaces (blanks) in *path_list2*. A pathname can be either an absolute OSS directory pathname or a fully qualified Guardian subvolume name.

The **-rld_first_l** flag can be specified more than once in a command line or an obey file. Multiple *path_list2* specifications are concatenated into a single loadfile entry.

See the **Finding Libraries** subsection under **DESCRIPTION** for details about the effect of this flag on search order.

-s

Removes symbol information used for linking and symbolic debugging from the output loadfile. A file stripped of all symbol information cannot be symbolically debugged with the Visual Inspect debugger.

You can use this flag only when creating a loadfile. To strip all symbol information from an existing loadfile, use the **-strip** flag.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-set *attribute_name attribute_value*

Sets the value of the run-time attribute specified in *attribute_name* to the value specified in *attribute_value* when creating a loadfile. Use the **-change** flag to change a run-time attribute in an existing loadfile.

Each *attribute_name* has a corresponding range of accepted *attribute_value*s as follows:

- **FLOAT_LIB_OVERRULE** is either **ON** or **OFF**. The default value is **OFF**. (A **FLOAT_TYPE_OVERRULE** value of **ON** is ignored for library loadfiles; this attribute only has meaning for program loadfiles.)

- **FLOATTYPE** is one of the following:

  IEEE_FLOAT
  NEUTRAL_FLOAT
  TANDEM_FLOAT

  If **FLOATTYPE** is not specified, the value used comes from the input linkfile. If **FLOATTYPE** is specified more than once, all occurrences except the final one are ignored.

- **HEAP_MAX**, **MAINSTACK_MAX**, [**PROCESS_**]**SUBTYPE**, and **SPACE_GUARANTEE** are unsigned numbers. The default value is 0 (zero).

- **HIGHPIN**, **HIGHREQUESTER**[**S**] or **HIGHREQUESTOR**[**S**], and **INSPECT** are either **ON** or **OFF**. The default value is **ON**.

- **LIBNAME** is the Guardian filename of a library file, specified as described for the **-libname** flag. The default value is none.

- **RUNNAMED** and **SAVEABEND** are either **ON** or **OFF**. The default value is **OFF**.

- **RLD_UNRESOLVED** is **ERROR**, **IGNORE**, or **WARN**. The default value is **ERROR**.

- **SYSTYPE** is either **OSS** or **GUARDIAN**. The default value is determined by the file system that contains the output file. For users of this reference page, the default value is probably **OSS**. (If the output loadfile is created in the Guardian file system through the **/G** directory, the default is **GUARDIAN**.)

See the *ld Manual* for a description of each run-time attribute.

-show_multiple_defs

Tells **ld** to produce a listing of any symbols with multiple definitions within the input linkfiles.

The default action does not display instances of multiple definitions.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-stdin Reads the contents of the standard input file at the place in the command line where the flag is specified.

You can specify this flag as often as you want in the command line or an obey file. Each specification is processed when encountered.

-strip *filename9*

Removes symbol information used for linking and symbolic debugging from an existing loadfile with the name *filename9*. A file stripped of all symbol information cannot be symbolically debugged with the Visual Inspect debugger or linked again.

You can use this flag only on an existing loadfile. To strip all symbol information when creating a loadfile, use the **-s** flag.

You cannot specify other loadfile filenames or flags other than the following with the **-strip** flag:

-noverbose, **-verbose**, or **-warn**

-fl or **-obey**
-stdin

The resulting file has the same **ld** timestamp as before.

-t *address2* Specifies the hexadecimal virtual address at which the text area starts. The default value for *address2* is:

- 70000000 for user programs
- 60000000 for a DLL

The value specified for *address2* is always hexadecimal and can optionally be prefixed by 0x. The specified value is automatically rounded up to a multiple of 4096 (0x1000) bytes.

-temp_o *filename10*
Tells **ld** to save its output loadfile work with the specified file name until it has successfully rewritten *filename7*.

filename10 can be either a simple filename, a fully qualified Guardian filename, or an absolute OSS pathname. If a simple filename is used, **ld** saves the file in the directory or subvolume where *filename7* is located.

The default action omits creation of the intermediate regular file.

-u *symbol_name4*
Tells **ld** to add *symbol_name4* as an undefined symbol. This causes **ld** to search for this symbol in any archive libraries that are specified after this flag on the command line or in an obey file.

The search constraint specified by the **-u** flag is overridden by use of the **-include_whole** flag.

-ul Creates a DLL as a native user library. Specify this flag when linking modules to create a DLL as a native user library.

When you specify the **-ul** flag, the exported symbols are those described as exported by the **-export_all** flag, unless you also use the **-export_not** flag.

You can specify these flags as often as you want in the command line or an obey file. Each specification is processed when encountered.

-unres_symbols { **error** | **ignore** | **warn** }
Tells **ld** what action to take when a needed symbol cannot be resolved:

error	Issue an error message and stop processing. This setting is ignored when you also use the **-allow_missing_libs** flag.
ignore	Ignore the missing file and continue processing. This is the default action.
warn	Issue a warning message but continue processing. This setting is ignored when you also use the **-allow_missing_libs** flag.

You can specify this flag as often as you want in the command line or an obey file. The final specification is the one used.

-verbose Directs **ld** to write error, warning, and informational messages to its output listing, along with output specified by other options.

The default value is **-no_verbose**.

You can specify the flags **-warn**, **-verbose**, **-noverbose**, and **-no_verbose** as often as you want in the command line or an obey file. The value used is the final value entered.

-warn Directs **ld** to write only error and warning messages to its output listing, along with output specified by other options.

You can specify the flags **-warn**, **-verbose**, **-noverbose**, and **-no_verbose** as often as you want in the command line or an obey file. The default value is **-no_verbose**. The value used is the final value entered.

-y *symbol_name5*
Tells **ld** to report which linkfiles define and use the symbol *symbol_name5*. The linkfiles are listed in the order encountered.

This information can be useful if a previous **ld** session produced error or warning messages about a symbol being either undefined or defined more than once.

Operands

filename13 Specifies one or more loadfiles or DLLs for the **ld** utility to link. This operand is required for all flags except the **-change** and **-strip** flags. In the OSS environment, the value specified must be a valid OSS pathname.

DESCRIPTION

The **ld** utility links one or more TNS/R native position-independent code (PIC) linkfiles to produce an executable or nonexecutable native PIC loadfile. You can also modify existing loadfiles using **ld**.

You can invoke **ld** directly or, if you are creating a C or C++ program, you can use the **c89** utility to invoke **ld** automatically for you. On the command line, the filenames are the names of input linkfiles or archives. Names of flags must be followed by spaces and are not case-sensitive, except for the **-l** and **-L** flags.

If no flags or operands are used, the **ld** command displays online help.

Saving Temporary Files

ld creates temporary working files while it processes command line or obey file information. These temporary working files are given names of the form **ZLDNF***nnn*, where:

nnn is a unique sequentially assigned decimal number, beginning with 000.

To create a final permanent file with the same name as an existing loadfile, **ld** must first remove the existing file. If **ld** processing is interrupted during the process of removing and recreating the output loadfile, all work can be lost.

The **-temp_o** flag allows you to save the completed working file as a temporary regular file with a known filename before the original loadfile is removed. The temporary file is itself removed after the final permanent loadfile is completely written.

Finding Libraries

The OSS version of **ld** searches for libraries in the following locations when resolving the values specified for the **-l** and **-lib** flags:

1. Locations specified by the current **-first_l** flag

2. Public libraries (installed by the system operator) *** ****
3. Locations specified by the current **-libvol** and **-L** flags
4. Default locations in the OSS environment:
   /lib:/usr/lib:/usr/local/lib:/G/SYSTEM/ZDLL *

   The value of the **COMP_ROOT** environment variable is added to the beginning of **/lib**, **/usr/lib**, and **/usr/local/lib**. By default, the value of **COMP_ROOT** is null in the OSS environment.
5. Default locations in the Guardian environment: $SYSTEM.ZDLL *** ****

The steps marked by an asterisk (*****) are skipped when the **-nostdlib** or **-no_stdlib** flag is in effect, and the steps marked by two asterisks (******) are skipped only when the **-bstatic** flag is used. Archive libraries encountered in steps marked by ****** are reported as errors.

ld searches in each location for libraries by file name, based upon whether the specified file name is a simple name. (A simple name is an OSS pathname without any directory components.) If the specified file name is not a simple name, **ld** tries to open the specified file.

If the specified filename is a simple name, **ld** tries to open the specified file; if it cannot, it modifies the supplied value and tries to open a file with the modified name. (Simple names specified in the Guardian file system, **/G**, are not modified; **ld** uses only the supplied value.) The prefix **lib** and the following suffixes are added to the specified name to create the modified name:

.so	To find a DLL, unless the **-bstatic** flag is in effect
.srl	To find a shared runtime library (SRL), unless the **-bstatic** flag is in effect
.a	To find an archive file, unless the **-dllsonly** flag is in effect

The OSS version of **rld** searches for DLLs and archive files in the following locations:

1. Locations specified by the current **-rld_first_l** flag
2. Public libraries (installed by the system operator)
3. Program location *****
4. Locations specified by the current **-rld_l** flag
5. Default locations:

   On OSS: **/lib:/usr/lib:/usr/local/lib:/G/SYSTEM/ZDLL**
   On Guardian: $SYSTEM.ZDLL *****

   The value of the **COMP_ROOT** environment variable is added to the beginning of **/lib**, **/usr/lib**, and **/usr/local/lib**. By default, the value of **COMP_ROOT** is null in the OSS environment.

When the **-limit_runtime_paths** flag has been used for the loadfile, the following are omitted from the search:

- The steps marked by an asterisk (*****) in the previously described search order
- Paths indicated by the TACL DEFINEs _RLD_FIRST_LIB_PATH and _RLD_LIB_PATH.

For More Information

ld is not an interactive tool like Binder. For more information on using **ld**, see the *ld Manual*. For more informaiton on run-time library use, see the *rld Manual*.

EXAMPLES

1. The following example:

   ```
   ld objecta objectb -o objectc
   ```

   links together the input linkfiles named **objecta** and **objectb** to create a loadfile named **objectc**.

2. The following example:

   ```
   ld -dll -dllname objecta -o objectb objectc
   ```

   creates a DLL named **objecta** as the loadfile named **objectb** from the linkfile named **objectc**.

3. The following example:

   ```
   ld obj1.o obj2.o -ul -o lib
   ```

   links the linkfiles named **obj1.o** and **obj2.o** together into a user library named **lib**.

4. The following example:

   ```
   ld obj3.o obj4.o -o prog -libname \$A.B.C
   ```

   links linkfiles named **obj3.o** and **obj4.o** together into a loadfile named **prog**. When **prog** runs, it has a user library with the Guardian name **$A.B.C**. The backslash (\) prevents the shell from misinterpreting the dollar sign ($).

5. The following example:

   ```
   ld /usr/lib/ccppmain.o test1.o test2.o  \
     -obey /usr/lib/libc.obey -o prog
   ```

   links the C linkfiles **test1.o** and **test2.o** to build a loadfile named **prog**. Because the program is a C program, the **ccppmain.o** library linkfile is required. The **-obey** flag directs **ld** to link in all the required hybrid SRLs.

6. The following example:

   ```
   ld obj6.o obj7.o -o prog -set systype guardian
   ```

   links the linkfiles named **obj6.o** and **obj7.o** into a loadfile named **prog** that you intend to run as a Guardian process.

7. The following example:

   ```
   ld -change highpin off exeobj
   ```

   changes the value of the **HIGHPIN** attribute in the loadfile **exeobj** to **OFF**.

NOTES

OSS filenames intended for use with **ld** should not begin with an equals (=) character. The equals character is reserved for use with MAP DEFINEs.

EXIT VALUES

The **ld** command returns one of the following values:

0 (zero)	No errors or warning conditions were detected.

1	One or more warning conditions were detected.
2	One or more general errors were detected.
3	A fatal error was detected.

RELATED INFORMATION

Commands: **c89(1)**, **nld(1)**, **noft(1)**.

Files: **float(4)**.

STANDARDS CONFORMANCE

The **ld** command is an HP extension to the Single UNIX Version 2 specification and performs functions comparable to the UNIX **ld** command.

NAME

let - Evaluates arithmetic expressions

SYNOPSIS

let *argument* ...

DESCRIPTION

The **let** command evaluates each *argument* as a separate arithmetic expression. (See **Arithmetic Evaluation** in the reference page for **sh.1** for a description of arithmetic expression evaluation.)

EXIT VALUES

The exit status is 0 (zero) if the value of the last expression is nonzero, and 1 otherwise.

EXAMPLES

The statement **let x=y+z** is equivalent to the statement **(x=y+z)**.

NOTES

The **let** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

NAME

lex - Generates a C language lexical analyzer

SYNOPSIS

lex [**-bcdfinpstvFILT8**] **-C**[**efmF**] [**-S***skeleton*] [*file* ...]

FLAGS

-b Generates backtracking information to file **lex.backtrack**. This is a list of scanner states that require backtracking and the input characters on which they backtrack. By adding rules, you can remove backtracking states. If all backtracking states are eliminated and the **-f** or **-F** flag is used, the generated scanner will run faster.

-d Makes the generated scanner run in **debug** mode. Whenever a pattern is recognized and the global **yy_lex_debug** is nonzero (which is the default value), the scanner writes to the standard error file a line of the form:

```
--accepting rule at line 53 ("the matched text")
```

The line number refers to the location of the rule in the file defining the scanner (the input to the **lex** command). Messages are also generated when the scanner backtracks, accepts the default rule, reaches the end of its input buffer (or encounters a NULL), or reaches an End-of-File.

-f Specifies *full table* (no table compression is done). The result is large but fast. This flag is equivalent to **-Cf**.

-i Instructs the **lex** command to generate a case-insensitive scanner. The case of letters given in the **lex** input patterns is ignored, and tokens in the input are matched regardless of case. The matched text given in the **yytext** variable will have the original case (as read by the scanner).

-p Generates a performance report to the standard error file. This identifies features of the **lex** input file that will cause a loss of performance in the resulting scanner.

-s Causes the default rule (that unmatched scanner input is echoed to the standard output file to be suppressed. If the scanner encounters input that does not match any of its rules, it aborts with an error.

-t Instructs the **lex** command to write the scanner it generates to the standard output file instead of to the file **lex.yy.c**.

-v Specifies that the **lex** command should write to the standard error file a summary of statistics regarding the scanner it generates.

-F Specifies that the fast scanner table representation should be used. This representation is about as fast as the full table representation (the **-f** flag), and for some sets of patterns it will be considerably smaller (and for others, larger). This flag is equivalent to the **-CF** flag.

-I Instructs the **lex** command to generate an interactive scanner; that is, a scanner that stops immediately rather than looking ahead if it knows that the currently scanned text cannot be part of a longer rule's match. Note that the **-I** flag cannot be used with full or fast tables; that is, with the **-f**, **-F**, **-Cf**, or **-CF** flags.

-L Instructs the **lex** command not to generate **#line** directives in the file **lex.yy.c**. The default action is to generate such directives so error messages in the actions will be correctly located with respect to the original **lex** input files.

-T Makes the **lex** command run in trace mode. It generates a lot of messages to the standard output file concerning the form of the input and the resultant nondeterministic and deterministic finite automata. This flag is mostly for use in maintaining the **lex** command.

-8 Instructs the **lex** command to generate an 8-bit scanner (the default scanner is a 7-bit scanner).

-C[efmF]

Controls the degree of table compression. The default setting is **-Cem**, which provides the highest degree of table compression. Faster-executing scanners can be traded off at the cost of larger tables with the following generally being true:

Slowest and smallest

-Cem
-Cm
-Ce
-C
-C{f,F}e
-C{f,F}

Fastest and largest

-C flags are not cumulative; whenever the flag is encountered, the previous **-C** settings are forgotten. The **-f** or **-F** and **-Cm** flags do not make sense together; there is no opportunity for meta-equivalence classes if the table is not being compressed. Otherwise, the flags may be freely mixed.

-C Specifies that the scanner tables should be compressed and neither equivalence classes nor meta-equivalence classes should be used.

-Ce Directs the **lex** command to construct equivalence classes; for example, sets of characters that have identical lexical properties. Equivalence classes usually give dramatic reductions in the final table/object file sizes (typically a factor of 2 to 5) and are inexpensive in terms of cost versus performance (one array look-up per character scanned).

-Cm Directs the **lex** command to construct meta-equivalence classes, which are sets of equivalence classes (or characters, if equivalence classes are not being used) that are commonly used together. Meta-equivalence classes are often a benefit when using compressed tables, but they have a moderate performance impact (one or two "if" tests and one array look-up per character scanned).

-Cf Specifies that the full scanner tables should be generated; the **lex** command should not compress the tables by taking advantage of similar transition functions for different states.

-CF Specifies that the alternative fast scanner representation should be used.

-S*skeleton*
Overrides the default skeleton file from which the **lex** command constructs its scanners. This is useful for **lex** maintenance or development.

-c Specifies table-compression options. (Obsolescent)

-n Suppresses the statistics summaries that the **-v** flag typically generates. (Obsolescent.)

DESCRIPTION

The **lex** and **flex** commands have the same functionality.

The **lex** command is a tool for generating *scanners*: programs that recognize lexical patterns in text. **lex** reads the given input files, or its standard input file if no filenames are given or if a file operand is **-** (dash), for a description of a scanner to generate. The description is in the form of pairs of regular expressions and C code, called *rules*. **lex** generates as output a C source file, named **lex.yy.c**, which defines a routine **yylex()**. This file is compiled and linked with the **-ll** library to produce an executable. When the executable is run, it scans its input and the regular expressions in its *rules* looking for the best match (longest input). When it has selected a rule it executes the associated C code, which has access to the matched input sequence (commonly referred to as a token). This process then repeats until input is exhausted.

lex treats multiple input files as one.

Syntax for lex Input

This subsection contains a description of the **lex** input files, which are normally named with a **.l** suffix. It provides a listing of the special values, macros, and functions recognized by the **lex** command.

The **lex** input file consists of three sections, separated by a line with just **%%** in it:

[*definitions*]
%%
[*rules*]
[**%%**
[*user_functions*]]

where

definitions Contains declarations to simplify the scanner specification and declarations of start states, which are explained below.

rules Describes what the scanner is to do.

user_functions Contains user-supplied functions, which are copied directly to file **lex.yy.c**.

With the exception of the first **%%** sequence, all sections are optional. The minimal scanner, **%%**, copies its input to the standard output file.

Each line in the *definitions* section can be:

name regexp Defines *name* to expand to *regexp*. *name* is a word beginning with a letter or an underscore (|*L_) followed by zero or more letters, digits, underscores, or dashes (**-**). In the regular-expression parts of the rules section, the command substitutes *regexp* wherever you refer to {*name*} (*name* within braces).

%x *state* [*state* ...] or **%s** *state* [*state* ...]
Defines names for states used in the rules section. A rule can be made conditionally active based on the current scanner state. Multiple lines defining states can appear, and each can contain multiple *state* names, separated by white space. The name of a *state* follows the same syntax as that of *regexp* names except that

dashes (**-**) are not permitted. Unlike *regexp* names, *state* names share the C **#define** namespace. In the *rules* section, states are recognized as <*state*> (*state* within angle brackets).

The **%x** directive names exclusive states. When a scanner is in an exclusive state, only rules prefixed with that state are active. Inclusive states are named with the **%s** directive.

%{ or **%}** When placed on lines by themselves, enclose C code to be passed verbatim into the global definitions of the output file. Such lines commonly include preprocessor directives and declarations of external variables and functions.

space or *tab* Appear at the beginning of lines in the definitions section that are to be passed directly into the **lex.yy.c** output file, as part of the initial global definitions.

The *rules* section follows the *definitions*, separated by a line consisting of **%%**. The rules section contains rules for matching input and taking actions, in the following format:

pattern [*action*]

pattern starts in the first column of the line and extends until the first nonescaped white space character. The command attempts to find the *pattern* that matches the longest input sequence and execute the associated *action*. If two or more *pattern*s match the same input, the one that appears first in the *rules* section is chosen. If no *action* exists, the matched input is discarded. If no *pattern* matches the input, the default action is to copy it to the standard output file.

All *action* code is placed in the **yylex()** function. Text (C code or declarations) placed at the beginning of the *rules* section is copied to the beginning of the **yylex()** function and can be used in actions. This text must begin with a space or a tab (to distinguish it from rules). In addition, any input (beginning with a space or within **%{** and **%}** delimiter lines) appearing at the beginning of the rules section before any rules are specified is written to file **lex.yy.c** after the declarations of variables for the **yylex( )** function and before the first line of code in **yylex( )**.

Elements of each rule are:

state A *pattern* can begin with a comma-separated list of *state* names enclosed by angle brackets (< *state* [**,***state*...] >). These states are entered through the **BEGIN** statement. If a *pattern* begins with a *state*, the scanner can recognize it only when in that state. The initial state is 0 (zero).

regexp A *pattern* can be a regular expression to match against the input stream. The regular expressions in the **lex** command provides a rich character-matching syntax.

The following characters, shown in order of decreasing precedence, have special meanings:

x Matches the character *x*.

"" (double quotes)
Enclose characters and treat them as literal strings. For example, **"*+"** is treated as the asterisk character followed by the plus character.

str (backslash)
If *str* is one of the characters **a**, **b**, **f**, **n**, **r**, **t**, or **v**, then represents the ANSI C interpretation (for example, **\n** is a newline). If *str* is a string of octal digits, it is interpreted as a character with octal value *str*. If *str* is a string of hexadecimal digits with a leading **x**, it is interpreted as a character with that value. Otherwise, it is interpreted literally with no

special meaning. For example, **x**yz** represents the four characters **x*****yz**.

[ ] (brackets)
Represent a character class in the enclosed range ([.-.]) or the enclosed list ([...]). The dash character (**-**) is used to define a range of characters from the ASCII value or the 8-bit class of the character that comes before the dash to the ASCII value or the 8-bit class of the character that follows the dash. For example, **[abcx-z]** matches **a**, **b**, **c**, **x**, **y**, or **z**.

The circumflex (**^**), when it appears as the first character in a character class, indicates the complement of the set of characters within that class. For example, **[^abc]** matches any character except **a**, **b**, or **c**, including special characters like newline. Similarly, **[^a-zA-Z]** is any character that is not a letter.

( ) (parentheses)
Group regular expressions. For example, **(ab)** is considered as a single regular expression.

{ } (braces)
When enclosing numbers, indicate a number of consecutive occurrences of the expression that comes before it. For example, **(ab){1,5}** indicates a match for from 1 to 5 occurrences of the string **ab**.

When enclosing a name, the name represents a regular expression defined in the *definitions* section. For example, **{digit}** will be replaced with the defined regular expression for **digit**. Note that the expansion takes place as if the definition were enclosed in parentheses.

. (dot) Matches any single character except newline.

? (question mark)
Matches zero or one of the preceding expressions. For example, **ab?c** matches both **ac** and **abc**.

***** (asterisk)
Matches zero or more of the preceding expressions. For example, **a*** is zero or more consecutive **a** characters. The utility of matching zero occurrences is more obvious in complicated expressions. For example, the expression **[A-Za-z][A-Za-z0-9]*** indicates all alphanumeric strings with a leading alphabetic character, including strings that are only one alphabetic character.

+ (plus sign)
Matches one or more of the preceding expressions. For example, **[a-z]+** is all strings of lowercase letters.

xy (concatenation)
Matches the expression *x* followed by the expression *y*.

| (vertical bar)
Matches either the preceding expression or the following expression. For example, **ab**|**cd** matches either **ab** or **cd**.

x/*y* (slash)
: Matches expression *x* only if expression *y* (trailing context) immediately follows it. For example, **ab/cd** matches the string **ab** but only if followed by **cd**. Only one trailing context is permitted per *pattern*.

^ (circumflex)
: When it appears at the beginning of the pattern, matches the beginning of a line. For example, **^abc** matches the string **abc** if it is found at the beginning of a line.

$ (dollar sign)
: When it appears at the end of a pattern, matches the end of a line. It is equivalent to **/\n**. For example, **abc$** matches the string **abc** if it is found at the end of a line.

<<EOF>>
: Matches an End-of-File.

<*x*> (angle bracket)
: Identifies a state name (see earlier description of *state*) and can appear only at the beginning of a pattern. For example, **<done><<EOF>>** matches an End-of-File, but only if it is in the state **done**.

In addition, the following rules apply for bracket expressions:

Equivalence class expressions
: These represent the set of collating elements in an equivalence class and are enclosed within bracket-equal delimiters (**[= =]**). An equivalence class generally is designed to deal with primary-secondary sorting; that is, for languages like French that define groups of characters as sorting to the same primary location, and then have a tie-breaking, secondary sort. For example, if **a, à** (a accent grave), and **â** (a circumflex) belong to the same equivalence class, then **[[=a=]b]**, **[[=à=]b]**, and **[[=â=]b]** are each equivalent to **[aàâb]**.

  NOTE: If you are viewing this reference page online using the **man** command, the special characters are not displayed. See this reference page in the *Open System Services Shell and Utilities Reference Manual*.

Character class expressions
: These represent the set of characters in the current locale belonging to the named ctype class. These are expressed as a ctype class name enclosed in bracket-colon delimiters **([: :])**.

  In the C or OSS locale, the following character class expressions are supported: [**:alpha:**], [**:upper:**], [**:lower:**], [**:digit:**], [**:alnum:**], [**:xdigit:**], [**:space:**], [**:print:**], [**:punct:**], [**:graph:**], and [**:cntrl:**].

  Other locales may define additional character classes.

Letters and digits never have special meanings. A character such as ^ or -, which has a special meaning in particular contexts, refers simply to itself when found outside that context. Spaces and tabs must be escaped to appear in a regular expression; otherwise they indicate the end of the expression.

action Each *pattern* in a rule has a corresponding *action*, which can be any arbitrary C statement. The pattern ends at the first nonescaped white space character; the remainder of the line is its *action*. If the action is empty, then when the pattern is matched, the input that matched it is discarded.

If the action contains a **{**, then the action scans till the balancing **}** is found, and the action may cross multiple lines. Using a **return** statement in an action returns from **yylex()**.

An action consisting solely of a vertical bar (|) means *same as the action for the next rule*.

lex variable that can be used within actions are:

yytext Is a string (**char ***) containing the current matched input. It cannot be modified.

yyleng Is the length (**int**) of the current matched input. It cannot be modified.

yyin Is a stream (**FILE ***) that the **lex** command reads from the standard input file by default. It can be changed, but because of the buffering **lex** uses, changing the stream makes sense only before scanning begins. Once scanning terminates because an End-of-File was found, **void yyrestart (FILE ***_new_file_**)** can be called to point **yyin** at a new input file. Alternatively, **yyin** can be changed whenever a new or different buffer is selected (see **yy_switch_to_buffer()**).

yyout Is a stream (**FILE ***) to which **ECHO** output is written (the standard output file by default). It can be changed by the user.

YY_CURRENT_BUFFER
Returns the current buffer (**YY_BUFFER_STATE**) used for scanner input.

lex macros and functions that can be used within actions are:

ECHO Copies the **yytext** variable to the scanner's output.

BEGIN *state*
Changes the scanner state to be *state*. This affects which rules are active. The *state* must be defined in a **%s** or **%x** definition. The initial state of the scanner is **INITIAL** or 0 (zero).

REJECT Directs the scanner to proceed immediately to the *next best* pattern that matches the input (which may be a prefix of the current match). The **yytext** and **yyleng** variables are reset appropriately. Note that **REJECT** is a particularly expensive feature in terms of scanner performance; if it is used in any of the scanner's actions, it slows down all the scanner's pattern matching operations. **REJECT** cannot be used if the command is invoked with either the **-f** or **-F** flag.

yymore( ) Indicates that the next matched text should be appended to the currently matched text in the **yytext** variable (rather than replace it).

yyless(*n*) Returns all but the first *n* characters of the current token back to the input stream, where they are rescanned when the scanner looks for the next match. The **yytext** and **yyleng** variables are adjusted accordingly.

yywrap( ) Returns 0 (zero) if there is more input to scan or 1 if there is not. The default **yywrap( )** always returns 1. It is implemented as a macro.

yyterminate( )
Can be used instead of a return statement in an action. It terminates the scanner and returns a 0 (zero) to the scanner's caller.

yyterminate() is automatically called when an End-of-File is encountered. It is a macro and can be redefined.

yy_create_buffer(*file, size*)
Returns a **YY_BUFFER_STATE** handle to a new input buffer large enough to accommodate *size* characters and associated with the given *file*. When in doubt, use **YY_BUF_SIZE** for the size.

yy_switch_to_buffer(*new_buffer*)
Switches the scanner's processing to scan for tokens from the given buffer, which must be a **YY_BUFFER_STATE**.

yy_delete_buffer(*buffer*)
Deletes the given buffer.

YY_NEW_FILE
Enables scanning to continue after the **yyin** variable has been assigned a new file to process.

YY_DECL Controls how the scanning function, **yylex( )**, is declared. By default, it is **int yylex( )** or, if prototypes are being used, **int yylex(void)**. This definition can be changed by redefining the **YY_DECL** macro. This macro is expanded immediately before the **{...}** (braces) that delimit the scanner function body.

YY_INPUT(*buf,result,max_size***)**
Controls scanner input. By default, **YY_INPUT** reads from the file-pointer **yyin** variable. Its action is to place up to *max_size* characters in the character array *buf* and return in the integer variable *result* either the number of characters read or the constant **YY_NULL** to indicate EOF. Following is a sample redefinition of **YY_INPUT**, in the definitions section of the input file:

```
%{
#undef YY_INPUT
#define YY_INPUT(buf,result,max_size)\
   {\
       int c = getchar();\
       result = (c == EOF) ? YY_NULL : (buf[0] = c, 1);\
   }
%}
```

When the scanner receives an End-of-File indication from **YY_INPUT**, it checks the **yywrap( )** function. If **yywrap( )** returns

zero, it is assumed that the **yyin** has been set up to point to another input file, and scanning continues. If it returns a nonzero value, then the scanner terminates, returning zero to its caller.

YY_USER_ACTION
Can be redefined to provide an action that is always executed prior to the matched pattern's action.

YY _USER_INIT
Can be redefined to provide an action that is always executed before the first scan.

YY_BREAK
Is used in the scanner to separate different actions. By default, it is simply a *break*, but it can be redefined if necessary.

The *user_functions* section consists of complete C functions, which are passed directly into the **lex.y.cc** output file (the effect is similar to defining the functions in separate **.c** files and linking them with **lex.y.cc**). This section is separated from the *rules* section by the **%%** delimiter.

Comments, in C syntax, can appear anywhere in the *user_functions* or *definitions* sections. In the *rules* section, comments can be embedded within actions. Empty lines or lines consisting of white space are ignored.

The following macros are not normally called explicitly within an action, but they are used internally by the **lex** command to handle the input and output streams.

input() Reads the next character from the input stream. You cannot redefine **input()**.

output() Writes the next character to the output stream.

unput(*c*) Puts the character *c* back into the input stream. It will be the next character scanned. You cannot redefine **unput()**.

libl.a contains default functions to support testing or quick use of a **lex** program without the **yacc** command; these functions can be linked in through **-ll**. They can also be provided by the user.

main() A simple wrapper that simply calls **setlocale()** and then calls the **yylex()** function.

yywrap() The function called when the scanner reaches the end of an input stream. The default definition simply returns 1, which causes the scanner in turn to return 0 (zero).

EXAMPLES

1. The following command processes the file **lexcommands** to produce the scanner file **lex.yy.c**:

   lex lexcommands

   This is then compiled and linked by the command:

   cc -oscanner lex.yy.c -ll

   to produce a program **scanner**.

2. The **scanner** program converts uppercase to lowercase letters, removes spaces at the end of a line, and replaces multiple spaces with single spaces. The **lexcommands** file contains:

```
%%
[A-Z]   putchar(tolower(yytext[0]));
[ ]+$
[ ]+  putchar(' ');
```

FILES

flex.skel Is the skeleton scanner.

lex.yy.c Is the generated scanner C source.

lex.backtrack Contains backtracking information generated from the **-b** flag.

NOTES

- Some trailing context patterns cannot be properly matched and generate warning messages:

  ```
  Dangerous trailing context
  ```

  These are patterns where the ending of the first part of the rule matches the beginning of the second part, such as **zx*/xy***, where the **x*** matches the **x** at the beginning of the trailing context.

- For some trailing context rules, parts that are actually fixed length are not recognized as such, leading to the previously mentioned performance loss. In particular, patterns using **{***n***}** (such as **test{3}**) are always considered variable length.

  Combining trailing context with the special | (vertical bar) action can result in fixed trailing context being turned into the more expensive variable trailing context. This happens in the following example:

  ```
  %%
  abc|
  xyz/def
  ```

- Use of the **unput( )** macro invalidates the contents of the **yytext** and **yyleng** variables within the current **lex** action.

- Use of the **unput( )** macro to push back more text than was matched can result in the pushed-back text matching a beginning-of-line (^) rule even though it did not come at the beginning of the line.

- Pattern matching of NULLs is substantially slower than matching other characters.

- The **lex** command does not generate correct **#line** directives for code internal to the scanner; thus, bugs in **flex.skel** yield invalid line numbers.

- Due to both buffering of input and read-ahead, you cannot intermix calls to **<stdio.h>** routines, such as **getchar( )**, with **lex** rules and expect it to work. Call **input( )** instead.

- The total table entries listed by the **-v** flag excludes the number of table entries needed to determine what rule was matched. The number of entries is equal to the number of deterministic finite-state automaton (DFA) states if the scanner does not use **REJECT**, and is somewhat greater than the number of states if it does.

- **REJECT** cannot be used with the **-f** or **-F** flag.

RELATED INFORMATION

Commands: **awk(1)**, **flex(1)**, **sed(1)**, **yacc(1)**.

Files: **locale(4).**

NAME

line - Reads one line from the standard input file and copies it to standard output file

SYNOPSIS

line

DESCRIPTION

The **line** command copies one line up to and including a newline character from the standard input file and writes it to the standard output file. The **line** command always writes at least a newline character.

Use this command within a shell command file to read from your terminal.

EXAMPLES

To read a line entered from the keyboard and append it to a file, enter:

```
echo 'Enter comments for the log:'
echo ': \c'
line >>log
```

This shell procedure displays the message:

```
Enter comments for the log:
```

It then reads a line of text entered from the keyboard and adds it to the end of the file **log**. The **echo ': \c'** command displays a **:** (colon) prompt. See the **echo** command for information about the **\c** escape sequence.

NOTES

Applications should use the **read** command instead of the **line** command.

EXIT VALUES

The **line** command returns a value of 1 when it detects an end-of-file; otherwise, the **line** command returns a value of 0 (zero).

RELATED INFORMATION

Commands: **echo(1)**, **ksh(1)**, **sh(1)**.

Functions: **read(2)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification.

NAME

ln - Links files

SYNOPSIS

ln [**-f**] [**-s**] *source_file target_file*

ln [**-f**] [**-s**] *source_file* [...] *target_directory*

FLAGS

-f Forces removal of existing target pathnames to allow specified links.

-s Creates symbolic links.

DESCRIPTION

The **ln** command links a single file *source_file* to file *target_file* or links one or more files to the same filenames in another existing directory (*target_directory*). If more than two arguments are specified and the final argument is not a directory, an error results.

By default, **ln** makes hard links. A hard link can only be created when all of the following are in the same OSS fileset:

- *source_file*
- *target_file* or *target_directory*
- All directories used for pathname resolution of any of the above

When the **-s** flag is used, **ln** makes symbolic links. A symbolic link can be created when *source_file* and *target_file* or *target_directory* are in different OSS filesets.

If either of the following is true and the **-f** flag is not specified:

- *target_file* exists
- *target_directory* contains a file with the same name as a *source_file* specification

the **ln** command writes a diagnostic message to the standard error file, does nothing further with the current source file, and goes on to any remaining source files.

Operands

source_file Specifies the absolute or relative pathname of the file to which the link should be made.

The file is found through normal pathname resolution rules. When a component of the pathname refers to a symbolic link rather than a directory, the pathname contained in the symbolic link is resolved. If the pathname in the symbolic link starts with a slash (/) character, the symbolic link pathname is resolved relative to the root directory of the process. If the pathname in the symbolic link does not start with a slash (/) character, the symbolic link pathname is resolved relative to the directory that contains the symbolic link.

If the file is not in the same OSS fileset as *target_file* or *target_directory*, a hard link cannot be created. If resolution of the pathname for *source_file* includes symbolic links, all of the directories traversed must be in the same OSS fileset as *target_file* or *target_directory*; otherwise, a hard link cannot be created.

target_file Specifies the relative or absolute pathname to assign to the link.

The resolved pathname for a hard link stores the inode information necessary to provide access to *source_file* only within the same OSS fileset. The resolved pathname for a symbolic link stores the filename for later resolution, regardless of whether *source_file* is in the same OSS fileset.

The value specified for *target_file* is resolved using normal pathname resolution rules. *target_file* file must be in the same OSS fileset as *source_file* if you are trying to create a hard link. If resolution of the pathname for *target_file* includes symbolic links, all of the directories traversed must be in the same OSS fileset as *source_file*; otherwise, a hard link cannot be created.

target_directory

Specifies the relative or absolute filename of a directory in which links for the specified *source_file* values should be created. *target_directory* must be in the same OSS fileset as *source_file* if you are trying to create hard links.

The value specified for *target_directory* is resolved using normal pathname resolution rules. If resolution includes symbolic links, all of the directories traversed must be in the same OSS fileset as *source_file*; otherwise, a hard link cannot be created.

EXAMPLES

1. To create a hard link to a file, enter:

   ln chap1 intro

   This links file **chap1** to the new filename **intro**, if **intro** does not already exist.

2. To create the link in Example 1 if **intro** already exists, enter:

   ln -f chap1 intro

   Now **chap1** and **intro** are two filenames that refer to the same file. Any changes made to one also appear in the other. If one name is deleted with the **rm** command, the file is not actually deleted but remains under the other name.

3. To create a hard link for the file named **index** to the same name in another directory that has the filename **manual**, enter:

   ln index manual

   This links file **index** to the new name **manual/index**.

   Note that **intro** in Example 1 is the name of a file; **manual** in Example 2 is a directory that already exists.

4. To create hard links for several files to names in another directory in the same OSS fileset, enter:

   ln chap2 jim/chap3 /u/manual

   This links file **chap2** to the new name **/u/manual/chap2** and file **jim/chap3** to **/u/manual/chap3**.

5. To use the **ln** command with pattern-matching characters, enter:

   ln manual/* .

   This links all files in the directory **manual** into the current directory (**.**), giving them the same names they have in **manual**. Note that you must type a space between the ***** (asterisk) and the **.** (dot).

6. To create a symbolic link to the final component of a pathname, enter:

   ln -s /a/b/c/d/e

   This creates a link, **e**, in the current directory to the file **/a/b/c/d/e**.

RELATED INFORMATION

Commands: **cp(1)**, **mv(1)**, **rm(1)**.

STANDARDS CONFORMANCE

The **-s** flag is an extension to the Single UNIX Specification, Version 2.

NAME

locale - Writes information about locales

SYNOPSIS

locale

[**-a** | **-m**]

locale

[**-c**] [**-k**]
name ...

FLAGS

-a Writes information about all available public locales

-c Writes the names of the specified locale categories

-k Writes the names and values of specified locale keywords

-m Writes the names of all available character map files

Operands

name Specifies a locale category or locale keyword

DESCRIPTION

With no flags or arguments, the **locale** utility writes to the standard output file the name and values of all the current locale environment variables, such as **LANG** and **LC_COLLATE**.

With the **-a** or **-m** flags, the **locale** utility displays information about available locales and character maps on your system.

- If the **-a** flag is specified, the **locale** utility writes the name of all available public locales. These are locales available to any application.
- If the **-m** flag is specified, the **locale** utility writes a list of the names of all available character-mapping files.

The **locale** utility with the *name* argument displays information about locale categories and keywords in the current locale. For example, it could display information about the **decimal_point** keyword in the **LC_NUMERIC** category or information about all keywords in the **LC_NUMERIC** category. The *name* argument can either be a locale category or a keyword from a category.

The **-c** and **-k** flags determine the information displayed by the **locale** utility as follows:

Table 5–1. Controlling locale Utility Output

Flags Set	Information Displayed
None	Value of keyword specified by the *name* parameter or values of all keywords in the category specified by the *name* parameter.
-c	Name of category containing the keyword specified by the *name* parameter or the name of the category specified by the *name* parameter, followed by value of locale keywords.
-k	Name and value of locale keywords.
-ck	Name of category, followed by name and value of locale keywords.

The following table lists the locale categories and the locale keywords that can be used in the *name* argument. There are no locale keywords for **LC_COLLATE** and **LC_CTYPE**.

Table 5–2. Categories and Keywords for the locale Utility

Category	Associated Keywords
LC_COLLATE	
LC_CTYPE	
CHARMAP	**charmap** **code_set_name** **mb_cur_max** **mb_cur_min**
LC_MESSAGES	**yesexpr** **noexpr** **yesstr** **nostr**
LC_MONETARY	**int_curr_symbol** **currency_symbol** **mon_decimal_point** **mon_grouping** **mon_thousands_sep** **positive_sign** **negative_sign** **int_frac_digits** **frac_digits** **p_cs_precedes** **p_sep_by_space** **n_cs_precedes** **n_sep_by_space** **p_sign_posn** **n_sign_posn** **debit_sign** **credit_sign**

	left_parenthesis
	right_parenthesis
LC_NUMERIC	**decimal_point**
	thousands_sep
	grouping
LC_TIME	**alt_digits**
	abday
	day
	abmon
	mon
	d_t_fmt
	d_fmt
	t_fmt
	t_fmt_ampm
	am_pm
	era
	era_d_fmt
	era_t_fmt
	era_d_t_fmt
	era_year

If several *name* arguments are specified, the **locale** utility processes them in order.

Environment Variables

The following environment variables affect the execution of **locale**: **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **NLSPATH**.

EXAMPLES

1. If you set the **LANG** environment variable to **fr_FR.ISO8859-1** and the **LC_MONETARY** environment variable to **fr_CA.ISO8859-1**, the **locale** utility entered without flags produces the following output:

   locale

   ```
   LANG=fr_FR.ISO8859-1
   LC_COLLATE="fr_FR.ISO8859-1"
   LC_CTYPE="fr_FR.ISO8859-1"
   LC_MONETARY="fr_CA.ISO8859-1"
   LC_NUMERIC="fr_FR.ISO8859-1"
   LC_TIME="fr_FR.ISO8859-1"
   LC_MESSAGES="fr_FR.ISO8859-1"
   LC_ALL=
   ```

   Note, however, that when setting the locale environment variables, some values imply values for other locale variables. For example, if **LC_ALL** is set to **en_US.ISO8859-1**, it implies **LC_COLLATE=en_US.ISO8859-1**, even if the **LC_COLLATE** environment variable is set to another locale.

2. To use the **locale** utility to retrieve the value of the **decimal_point** delimiter for the current locale, assuming the current locale is C/POSIX, enter:

   locale -ck decimal_point

   ```
   LC_NUMERIC
   decimal_point="."
   ```

DIAGNOSTICS

The **locale** utility generates these errors:

```
locale: unrecognized keyword, '\%s', in argument list.\n
usage: locale [-amck] keyword ...\n
```

RELATED INFORMATION

Files: **locale(4)**.

NAME

logger - Makes entries in the system log

SYNOPSIS

logger [**-f** *file*] [**-i**] [**-p** *priority*] [**-t** *tag*] [*string ...*]

The **logger** command makes the specified entries in the system log file.

FLAGS

-f *file* Logs all lines in *file*.

-i Logs the process ID (PID) of the **logger** process with each line.

-p *priority*
Enters the message with the specified *priority*. You can specify *priority* as a name or a numeric value. You can also enter a facility/priority pair, separated by a **.** (dot) character. (See **Facilities** and **Priorities** for information about valid values.)

-t *tag* Precedes each entry in the log with *tag*.

DESCRIPTION

The **logger** program allows information logging for later use by a system administrator or programmer in determining why noninteractive utilities have failed.

Facilities

The **logger** command provides a program and shell script interface to the **syslog( )** subroutine. The file in which entries are made depends on the current system log configuration; see the reference page for **syslog( )** for more information.

You can specify the message to be used for entries on the command line or with the **-f** *file* flag, which specifies that each line in *file* be logged as an entry, with the *string* argument. The *string* argument consists of one or more character strings separated by spaces. If you do not specify *string* or the **-f** flag, the **logger** command reads the standard input.

The specific facility names that can be entered as the facility portion of the *priority* argument to the **-p** flag appear in the following list. The corresponding numeric values appear in parentheses.

kern Kernel messages (**0**).

user Random user-level messages (**8**).

mail Mail system (**16**).

daemon System daemons (**24**).

auth Security/authorization messages (**32**).

syslog Messages generated internally by the **syslogd** subroutine (**40**).

lpr Line printer subsystem (**48**).

news Network news subsystem (**56**).

uucp UUCP subsystem (**64**).

cron Clock daemon (**72**).

Priorities

The specific priority names that can be entered as the priority portion of the *priority* argument to the **-p** flag appear in the following list. The corresponding numeric values appear in parentheses.

emerg The system is unusable (**0**).

alert Action must be taken immediately (**1**).

crit Critical conditions (**2**).

err Error conditions (**3**).

warning Warning conditions (**4**).

notice Normal but significant condition (**5**).

info Informational (**6**).

debug Debug-level messages (**7**).

EXAMPLES

1. To specify the **debug** priority with a priority name, enter:

   logger -p debug my message

2. To specify the debug priority with a priority number, enter:

   logger -p 7 my message

3. To specify both the **user** facility and the **debug** priority, enter:

   logger -p user.debug my message

4. To specify the same facility/priority pair using numeric values, enter:

   logger -p 8.7 my message

5. You can also combine alphabetic and numeric specifications:

   logger -p user.7 my message

RELATED INFORMATION

Functions: **openlog(3)**, **syslog(3)**.

NAME

logname - Displays user login name

SYNOPSIS

logname

DESCRIPTION

The **logname** command writes to the standard output file the name you used to log in to the system. This name is returned by the **getlogin( )** function. Under conditions where **getlogin( )** would fail, the **logname** command writes a diagnostic message to the standard error file and exits with a nonzero exit value.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXIT VALUES

The **logname** command returns the following values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **env(1)**.

Functions: **getlogin(2)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

lp - Sends files to a printer

SYNOPSIS

lp

[**-c**]
[**-d** *dest*]
[**-n** *copies*]
[**-s**]
[**-t** *title*]
[**-W save**]
[**-W pri=***priority*]
[**-W spl=***spooler_name*]
[*file ...*]

FLAGS

Flags for the **lp** command can appear in any order and can be mixed with filenames.

-c Copies the input files to the spooling area rather than directly to the printer device. This is the default action.

-d *dest* Specifies *dest* as the spooler destination for the job. *dest* can be specified as an alias that maps to a Guardian spooler location name. Spooler location aliases can be located in the **printcap** file in the **/etc** directory. The **lp** command checks the **printcap** file to resolve name mapping. If the destination name does not map to a physical printer device, the files to be printed are left in the spooling area for further disposition. If the destination name maps to an invalid printer location, **lp** exits with an error.

The destination name can be set with the environment variables **LPDEST** and **PRINTER**. The **-d** flag has precedence over **LPDEST**, which has precedence over **PRINTER**. If the **-d** and **-W spl** flags are not specified and none of the **LPDEST**, **PRINTER**, or **GUARDIAN_PRINTER** environment variables are set, the system default destination in the **printcap** file is used. The **lp** command exits with an error if no valid destination can be found.

-n *copies* Specifies the number of copies to be printed of each file listed in the **lp** request. *copies* is a number in the range 1 through 32767. When multiple files are to be printed, all copies of one file are printed before printing of the next file is started. The default value of *copies* is 1.

-s Suppresses messages from the **lp** command.

-t *title* Specifies a report name for the print job, where the argument *title* is a string of up to 16 alphanumeric characters and blank spaces. The report name is printed on the banner page of the output. The default report name is the user's Guardian logon name.

HP Extensions

-W save Saves the print job in the spooler after it has been printed.

-W pri=*priority*

Specifies the print request priority. The argument *priority* is defined as an integer 0 through 8, with 8 being the highest priority. The default value is 4. Normally, requests with higher priorities are printed before requests with lower priorities.

-W spl=*spooler_name*

Specifies the Guardian spooler location, which can be a maximum of 40 characters. Specify the *spooler_name* parameter as:

[*node***.]\$***collector***[.***group***[.***destination***]]**

In which:

- *node* is the system name
- **$***collector* is the spooler collector name
- *group* and *destination* are part of the spooler location

You can set the Guardian spooler name with the **GUARDIAN_PRINTER** environment variable. The **-W spl** flag has precedence over the **LPDEST** and **PRINTER** environment variables, which have precedence over the **GUARDIAN_PRINTER** environment variable. If the **-d** and **-W spl** flags are not specified and none of the **LPDEST**, **PRINTER**, or **GUARDIAN_PRINTER** environment variables are set, the system default destination in the **printcap** file is used. If no valid destination can be found, the **lp** command exits with an error.

DESCRIPTION

The **lp** command sends the specified files and associated information (collectively called a request) to a line printer for printing.

The **lp** command copies input files to an output printer device through the Guardian spooler subsystem. The system default destination is a printer device or a Guardian spooler location. The system default destination is configured by the system operator at system startup. If the system default destination becomes unavailable or if a valid destination cannot be found, the **lp** command exits with a an error.

Files are always first copied to the spooling area. From there they are sent to the printer. Depending on printer availability, files specified in a print request may not actually be sent to a printer from the spooling area until after the **lp** command successfully exits. When a job is sent to the printer device, each print job has exclusive access to the printer device and cannot be interrupted.

Operands

file The pathname of a file to be printed. If no *file* operands are specified, or if a *file* operand is specified as a **-** (hyphen), the standard input file is used. Guardian files are preceded with the **/G** naming convention. The **lp** command must have read permission to the file or it will return an error.

Standard Input Files

The standard input files can be OSS text files (filecode 180), Guardian EDIT files (filecode 101), compiler listing files (filecode 129), and PostScript files (filecode 0). Using input files other than the standard input file may produce unexpected results.

Standard input is used only if no *file* argument is specified or if *file* is specified as a **-** (hyphen).

Environment Variables

GUARDIAN_PRINTER

Specifies the Guardian spooler name. The **-W spl** flag has precedence over the **LPDEST** and **PRINTER** environment variables, which have precedence over the **GUARDIAN_PRINTER** environment variable. If the **-d** and **-W spl** flags are not specified, the **LPDEST** and **PRINTER** environment variables are not set, and **GUARDIAN_PRINTER** contains a value that is not a valid Guardian spooler name, the **lp** command exits with an error.

LANG Provides a default value for the internationalization variables that are unset or null. If the **LANG** variable is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the **lp** command behaves as if none of the variables have been defined.

LC_ALL
When set with a nonempty string, overrides the values of all other internationalization variables.

LC_CTYPE
Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multibyte characters in arguments and input files).

LC_MESSAGES
Determines the locale to be used to affect the format and contents of diagnostic messages written to the standard error file and informative messages written to the standard output file.

LC_TIME
Determines the format and contents of date and time strings displayed in the **lp** command banner page.

LPDEST
Names the output device or destination. If **LPDEST** is not set, the **PRINTER** environment variable is used. The **-d** variable takes precedence over **LPDEST**. If **-d** is not specified and **LPDEST** contains a value that is not a valid destination, the **lp** command exits with an error.

NLSPATH
Determines the location of message catalogs for processing the **LC_MESSAGES** variable.

PRINTER
Names the output device or destination. If the **-d** and **-W spl** flags are not specified and none of the **LPDEST**, **PRINTER**, or **GUARDIAN_PRINTER** environment variables are set, the system default destination in the **printcap** file is used. The **-d** and the **-W spl** flags and the **LPDEST** environment variable have precedence over the **PRINTER** environment variable. If the **-d** and **-W spl** flags are not specified, the **LPDEST** environment variable is not set, and the **PRINTER** environment variable contains a value that is not a valid device or destination, the **lp** command exits with an error.

Standard Output

The **lp** command associates a unique ID number from 1 through 4095 with each request and writes it to the standard output file. The message includes the request ID and the output destination. This request ID can be used to cancel (see the **cancel(1)** reference page) the request or to find its status (see the **lpstat(1)** reference page).

HP Extensions

The **lp** command establishes a level-3 spooling session with a spooler collector.

Spooler Destination

The spooler routing name consists of the spooler collection process name (names returned from the SPOOLCOM COLLECT command) followed by the spooler location name (names returned from the SPOOLCOM LOC command, for example, $S.#TITAN3). The spooler location names are the logical destinations of the print jobs. They may or may not be mapped to the real print devices. If a print device is associated with a specific location, the job request is eventually printed. Otherwise, the job request is left in the spooler area.

The Guardian spooler system uses the spooler location names to specify certain output attributes such as double-sided printing and landscape printing. These names are specified in the spooler configuration files. The user must select the appropriate spooler location for each print job.

Search Alogrithm

The search algorithm for the printer locations is as follows. The printer destination specification in the **-d** flag or the Guardian spooler name specification in the **-W spl** flag is used whenever it is specified. If both the **-d** and the **-W spl** flags are specified, the last printer location that is specified is used. Otherwise, the printer location is first the value set for the environment variable **LPDEST**, and then, if **LPDEST** is not set, the value set for the **PRINTER** environment variable is used. If the **PRINTER** environment variable is also not set, the value set for the **GUARDIAN_PRINTER** is used. When none of the **LPDEST**, **PRINTER**, or **GUARDIAN_PRINTER** environment variables are set, a system default location value (#DEFAULT), stored in the **printcap** database file, is used. Most HP NonStop server spooler systems are normally configured with a #DEFAULT location so jobs can always be routed to a printer device.

printcap Database

Aliases for the spooler location names are stored in the **printcap** database file. The **lp** command searches **printcap** everytime it needs to resolve the alias to its respective spooler location. A system default location should always be specified in the **printcap** file and should appear as the first entry in the file. **printcap** has the following format:

```
<alias name>        <spooler location name>
```

The system is supplied with a default **printcap** file with aliases for all the devices that are available to the system. You can create your own **printcap** file as needed. If you create your own **printcap** file, the **lp** command searches your own **printcap** file first and then the system default **printcap** file in the **/etc** file.

Preprocessing File Data

The **lp** command does not preprocess data in Guardian EDIT files, compiler listing files, and PostScript files. **lp** performs limited preprocessing of the control characters in OSS ASCII text files. For example, line feed control characters signify the end of a record and form feed control characters signify a page eject. Because **lp** has limited capability to handle all of the control characters correctly (for example, tabs and backspaces), use the **pr** command to preprocess the file data before invoking **lp**. **lp** translates unrecognizable control characters to blank characters.

EXAMPLES

1. To send an OSS text file to a printer using the destination-searching algorithm, enter:

   cat file | lp

   If the operation is successful, the exit status 0 is returned and a message,
   `stdin is routed to` *printer_name* `with request id` *id_number*
   is written to the standard output file.

2. To print multiple files with the destination and the heading for the banner specified, enter:

   lp -d hplp -t "myprint" file1 /G/system/vol/subvol/file

If the operation is successful, the exit status 0 is returned and the following messages are written to the standard output file:

```
file 1 is routed to hplp with request id: 1
/G/system/vol/subvol/file is routed to hplp with request id: 2
```

The header for the banner is "myprint."

EXIT VALUES

The following exit values are returned:

0	All input files were processed successfully.
>0	No output device was available, or an error occurred.

RELATED INFORMATION

Commands: **cancel(1)**, **lpstat(1)**.

STANDARDS CONFORMANCE

Spooling is a Guardian operation.

Extensions have been added to the **lp** command in order to support the Guardian environment.

NAME

lpstat - Displays line printer and print job status information

SYNOPSIS

lpstat [**-drst**] [**-a**[*list*]] [**-o**[*list*]] [**-p**[*list*]] [**-u**[*list*]] [**-v**[*list*]] [*ID ...*]

FLAGS

The flags can be specified in any order. Specifying no flags displays all of the information associated with the first printer alias (usually named **default**) in the user's **printcap** file, or, if no printcap file exists, in the **/etc/printcap** file.

Items specified in a *list* argument to a flag can be a series of items separated by commas or a list of items separated by commas or one or more blank spaces and enclosed in quotation marks.

-a [*list*] Displays the job acceptance status of the printer devices that are configured for the system.

The *list* argument is a list of printer aliases. If no value is specified for *list*, the information displayed is for the first printer alias (usually named **default**) in the user's **printcap** file or, if no printcap file exists, in the **/etc/printcap** file.

The display includes the device names, the current state of each device, the associated print processes, and other device attributes such as form name, selection algorithm, and header messages.

-d Displays the default spooler destination for output requests.

The information displayed is for the first printer alias (usually named **default**) in the user's **printcap** file or, if no printcap file exists, in the **/etc/printcap** file.

-o [*list*] Displays the status of all job requests in the print queue.

The *list* argument is a list of printer aliases. If no value is specified for *list*, the information displayed is for the first printer alias (usually named **default**) in the user's **printcap** file or, if no printcap file exists, in the **/etc/printcap** file.

-p [*list*] Displays the status of spooler locations.

The *list* argument is a list of printer aliases. If no value is specified for *list*, the information displayed is for the first printer alias (usually named **default**) in the user's **printcap** file or, if no printcap file exists, in the **/etc/printcap** file.

-r Displays the status of the spooler collector and supervisor.

The information displayed is for the first printer alias (usually named **default**) in the user's **printcap** file or, if no printcap file exists, in the **/etc/printcap** file.

-s Displays a status summary of the spooler supervisor, collector process, and spooler devices.

The output is the collective output of the **-v**, **-r**, and **-d** flags.

-t Displays all status information.

The output is the collective output of the **-s** and **-o** flags for all locations in the **printcap** or **/etc/printcap** file.

-u [*list*] Displays the status of output requests for users.

The *list* argument is a list of login names (NonStop Kernel user names or aliases for NonStop Kernel user names). If no value is specified for *list*, the login name of the user is used.

-v [*list*] Displays the names of spooler locations and associated devices.

The *list* argument is a list of printer aliases. If no value is specified for *list*, the information displayed is for the first printer alias (usually named **default**) in the user's **printcap** file or, if no printcap file exists, in the **/etc/printcap** file.

DESCRIPTION

The **lpstat** command displays information about the current status of the online accessible printer devices, related processes, and the status of spooler job requests.

Jobs can be in one of four states: OPEN, READY, PRINT, or HOLD. When a job request is in the process of being copied to the spooling area, it is in the OPEN state. When the job reaches the spooler queue it is in the READY state. When a printer accepts the job from the spooler queue, the job is in the PRINT state. If errors prevent a job from being copied to the spooler, the job is in the HOLD state.

Status information displayed by the **lpstat** command can also be displayed by the Guardian spooler utilities PERUSE and SPOOLCOM. These two Guardian utilities can be used to display status information on jobs that are initiated using the **lp** command.

Use on Guardian Objects

The **lpstat** command displays status information on jobs that are initiated in the Guardian environment.

Arguments

ID A job request identification number returned by the **lp** command. *ID* can be a value from 1 through 4095.

Environment Variables

LANG Provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the corresponding value from the implementation-specific default locale is used. If any of the internationalization variables contain an invalid setting, the **lpstat** command behaves as if none of the variables have been defined.

LC_ALL

When set with a nonempty string, overrides the values of all other internationalization variables.

LC_CTYPE

Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multibyte characters in arguments and input files).

LC_MESSAGES

Determines the locale to be used to affect the format and contents of diagnostic messages written to the standard error file and informative messages written to the standard output file.

LC_TIME

Determines the format and contents of date and time strings displayed in the **lpstat** command banner page.

NLSPATH

Determines the location of message catalogs for processing the **LC_MESSAGES** variable.

TZ Determines the time zone to be used with date and time strings.

EXAMPLES

1. To display the default destination, enter:

   lpstat -d

   If the operation is successful, the destination name and a zero exit status are returned.

2. To display the status of all the jobs sent to the printer with the alias **hplp** by users with the login names **rose**, **bill**, and **raj**, enter:

   lpstat -u rose, bill, raj -p hplp

EXIT VALUES

The following exit values are returned:

0 Completion was successful.

>0 An error occurred.

RELATED INFORMATION

Commands: **lp(1)**.

STANDARDS CONFORMANCE

The **lpstat** command complies with the XPG4 Version 2 specification with extensions, except for the following feature:

- The **-c** flag is not supported because the Guardian environment does not recognize printer classes. If the **-c** flag is used, the **lpstat** command issues the following message:

  `printer classes not supported`.

NAME

ls - Lists and generates statistics for files

SYNOPSIS

ls [**-W NOG**] [**-W NOE**] [**-abcCdfFgilLmnopPqrRstux1**] [*file* | *directory*] ...

ls -W guardian [**/G/**[*volume*[*/subvolume*[*/file_identifier*]]]] ...

ls -W gfinfo [**/G/**[*volume*[*/subvolume*[*/file_identifier*]]]] ...

FLAGS

-a Lists all entries in the directory, including the entries that begin with a **.** (dot).

-b Displays nonprintable characters in octal notation. For example, a file named `a^Ab` is displayed as `a\0016`.

-c Uses the time of last property change, mode change, and so on for sorting (when used with the **-t** flag) or for displaying (when used with the **-l, -g, -n, -o,** or **-u** flags).

-C Sorts output vertically in a multicolumn format. This is the default action when output is sent to a terminal.

-d Displays only the information for the directory that is named, rather than for its contents. This is useful with the **-l** flag to get the status of a directory.

-f This flag turns off the **-l**, **-t**, **-s**, and **-r** flags and turns on the **-a** flag; the flag uses the order in which entries appear in the directory.

-F Puts a **/** (slash) after each filename if the file is a directory, an ***** (asterisk) after each filename if the file can be executed, an **@** (at sign) for a symbolic link, and a **|** (vertical bar) for a FIFO file.

-g Displays the same information as the **-l** flag, except for the owner, which is not displayed.

-i Displays the inode number in the first column of the report for each file.

-l Displays the mode, number of links, owner, group, size, time of last modification for each file, and pathname. If the file is a special file, the size field instead contains the major and minor device numbers. If the file is a symbolic link, the pathname of the linked-to file is also printed preceded by ->. The attributes of the symbolic link are displayed. The **-n** flag overrides the **-l** flag.

-L Lists the file or directory the link references rather than the link itself, if the argument is a symbolic link. The **-n** flag overrides the **-l** flag.

-m Uses stream output format (a comma-separated series).

-n Displays the same information as the **-l** flag, except that it displays the user and the group IDs instead of the usernames and group names.

-o Displays the same information as the **-l** flag, except for the group, which is not displayed. The **-n** flag overrides the **-o** flag.

-p Puts a slash after each filename if that file is a directory.

-P (J06.11 or later J-series RVUs or H06.22 or later H-series RVUs only) Must be used with the **-l**, **-n**, **-g**, or **-o** flag. If the file has at least one file privilege set, a **#** appears after the permissions of the file. If the file also has optional access control list (ACL) entries, the **+** (plus sign) is not shown.

-q Displays nonprintable characters in filenames as a ? (question mark) character if output is sent to a terminal (the default destination).

-r Reverses the order of the sort, giving reverse collation or the oldest first, as appropriate.

-R Lists all subdirectories recursively.

-s Gives space used in 512-byte units (including indirect blocks) for each entry.

-t Sorts by time of last modification (latest first) instead of by name, before sorting the operands by the collating sequence.

-u Uses the time of the last access instead of the time of the last modification for sorting (when used with the **-t** flag) or for displaying (when used with the **-l** flag). The **-u** flag has no effect unless used with either the **-t** or **-l** flag or both.

-x Sorts output horizontally in a multicolumn format.

-1 Forces an output format of one entry per line; this is the default format when output is not directed to a terminal.

When the following mutually exclusive flags are specified, the last flag specified on the command line takes effect:

- **-C** and **-l** (ell)
- **-C** and **-1** (one)
- **-m** and **-l** (ell)
- **-x** and **-l** (ell)
- **-c** and **-u**

HP Extensions

-W guardian [/**G**[*/volume*[*/subvolume*[*/file_identifier*]]]] ...

Specifies a Guardian pathname. The **-W guardian** flag cannot be specified with any of the other flags for the **ls** command. The specified pathname must either identify a disk file directly or point to a Guardian subvolume. If a disk file is specified in the format /**G**/*volume/subvolume/file_identifier*, its file code is displayed. If the format used is **/G/***volume/subvolume*, the file codes for all of the disk files within the specified subvolume are displayed.

-W gfinfo [/**G**[*/volume*[*/subvolume*[*/file_identifier*]]]] ...

Specifies a Guardian pathname. The **-W gfinfo** flag cannot be specified with any of the other flags for the **ls** command. The specified pathname must either identify a disk file directly or point to a Guardian subvolume. If a disk file is specified in the format /**G**/*volume/subvolume/file_identifier*, its file type, file code, last open time, modification time, create time, permission, format, primary extents, secondary extents, and Safeguard status are displayed. If the format used is **/G/***volume/subvolume*, the identical information described above for an individual disk file is displayed for all of the disk files within the specified subvolume.

This flag is supported on systems running J06.13 or later J-series RVUs or H06.24 or later H-series RVUs only.

-W NOG

Specifies that the **/G** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not **/**,

/E, or **/E**/*system* or when recursion does not occur.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root and the recursive flag (**-R**) is used. This flag is ignored when the initial directory is not root or when recursion does not occur.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

The **-P** flag is also an HP extension.

DESCRIPTION

The **ls** command writes to the standard output file the contents of each specified directory or the name of each specified file, along with any other information you ask for with flags. If you do not specify a file or a directory, **ls** displays the contents of the current directory. Objects whose names begin with a period (.) are normally not displayed except with the **-a** flag.

By default, the **ls** command displays all information in collated order by filename. The collating sequence is determined by the **LC_COLLATE** environment variable.

There are three main ways to format the output:

- List entries in multiple columns by specifying either the **-C** or **-x** flag. **-1** is the default format, when output is sent to a terminal.
- List one entry per line.
- List entries in a comma-separated series by specifying the **-m** flag.

The **ls** command attempts to determine the number of byte positions in the output line. If **ls** cannot get this information, it uses a default value of 80. Note that columns may not be smaller than 20 bytes or larger than 400 bytes.

Access Control Lists (ACLs)

If a file has optional ACL entries:

- The file group permission bits displayed by the **ls** command are the class ACL entry permission bits instead of the file group ACL entry permission bits.
- The output of the command when the **-l**, **-n**, **-g**, or **-o** flags are used displays a plus (+) sign after the permissions of the file. However, the plus (+) sign is not displayed if one of these conditions is true:
  - — The **ls** command is executed remotely from a system that does not support OSS ACLs.
  - — You use the **-P** flag and the file has at least one file privilege set. In this case, **#** is displayed instead of **+**.

To list the contents of an ACL, use the **getacl** command.

For more information about ACLs, see the **acl(5)** reference page.

Environment Variables

The following environment variables affect the execution of the **ls** command:

LC_COLLATE
Determines the collating sequence.

LC_TIME Controls the format of the date and time.

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root and a recursive operation occurs in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE Omit the **/E** directory.

NOG Omit the **/G** directory.

NOG:NOE Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

Modes

The mode displayed with the **-l** flag is interpreted by the first character, as follows:

b Block special file

c Character special file

d Directory

l Symbolic link

p First-in-first-out (FIFO) special file

s **AF_UNIX** local socket

- Ordinary file

Permissions

The second through tenth characters in the permissions code are divided into three sets of three characters each. The first set of three characters shows the owner's permission. The next set of three characters shows the permission of the other users in the group. The last set of three characters shows the permission of everyone else. The three characters in each set show read, write and execute permission of the file. Execute permission of a directory lets you search a directory for a specified file.

Permissions are indicated as follows:

r Read

w Write

x Execute or search (directories)

- No access

The group-execute permission character is **s** if the file has set-group-ID mode. The user-execute permission character is **s** if the file has set-user-ID mode. See the **chmod** command for the meaning of this mode. The indications of set-ID bit of the mode are capitalized (**S**) if the corresponding execute permission is not set.

When the sizes of the files in a directory are listed, the **ls** command displays a total count in 512-byte units, including indirect blocks.

Use on Guardian Objects

For each pathname specified with the **ls -W guardian** command that names a /**G***/volume/subvolume/file_identifier*, **ls** writes the name of the Guardian disk file and its file code attribute to the standard output file. For each operand that names a /**G***/volume/subvolume*, **ls** writes the names of all Guardian disk files that are contained within that subvolume, along with their associated file codes.

If you invoke the **ls -W guardian** command without specifying a pathname and your current working directory is not within **/G**, a warning message is written to the standard error file and **ls** exits in error. If your current working directory is within **/G**, the file codes for all of the files in the current directory are displayed.

When you invoke the **ls** command with the **-W guardian** flag and specify a valid Guardian pathname, the following message is displayed:

```
%s:,<Guardian pathname in form/G/volume/subvolume>
```

For all files contained within the specified Guardian pathname (/**G***/volume/subvolume*) and for all specified disk filenames, the following message is displayed:

```
%s %d,<file_identifier>, <filecode>
```

For each pathname specified with the **ls -W gfinfo** command that names a /**G***/volume/subvolume/file_identifier*, **ls** writes the name of the Guardian disk file and its file type, file code, last open time, modification time, create time, permission, format, primary extents, secondary extents, and Safeguard status attributes to the standard output file. For each operand that names a /**G***/volume/subvolume*, **ls** writes the names of all Guardian disk files that are contained within that subvolume, along with the identical information described above for an individual disk file for all of the disk files within the subvolume.

If you invoke the **ls -W gfinfo** command without specifying a pathname and your current working directory is not within **/G**, a warning message is written to the standard error file and **ls** exits in error. If your current working directory is within **/G**, the file type, file code, last open time, modification time, create time, permission, format, primary extents, secondary extents, and Safeguard status attributes for all of the files in the current directory are displayed.

For all files contained within the specified Guardian pathname (/**G***/volume/subvolume*) and for all specified disk filenames, the following message is displayed:

```
%s %c %d %s %s %s %s %d %d %d %s <file_identifier>, <filetype>,
<filecode> , <lastopen_time>, <modification_time>, <create_time>,
<file_permission>, <file_format> , <primary_extents>,
<secondary_extents>, <safeguard_status>
```

EXAMPLES

1. To list all files in the current directory, enter:

   ls -a

   This command lists all files, including . (dot), .. (dot dot), and other files whose names begin with a dot.

2. To display detailed information, enter:

   ls -l chap1 .profile

   This command displays a long listing with detailed information about the files **chap1** and **.profile**.

3. To display detailed information about a directory, enter:

   ls -d -l . manual manual/chap1

   This command displays a long listing for the directories **.** (dot) and **manual** and for the file **manual/chap1**. Without the **-d** flag, this command lists the files in **.** (dot) and **manual** instead of providing detailed information about the directories themselves.

4. To list the files in the current directory in order of modification time, enter:

   ls -l -t

   This command displays a long listing of the files that were modified most recently, followed by the older files.

5. To list one or more individual files using the **-W guardian** flag, enter, for example:

   ls -W guardian /G/osf/kill/mykill /G/osf/kill/test

   The following is displayed:

   ```
   /G/osf/kill/mykill              100
   /G/osf/kill/test                100
   ```

6. To list all the files in a subvolume using the **-W guardian** flag, enter, for example:

   cd /G/osf/kill
   ls -W guardian

   The following is displayed:

   ```
   bindkil         101
   makekil         101
   mykill          100
   test            100
   testc           101
   ```

7. To list a specific file in one subvolume and all the files in another subvolume using the **-W guardian** flag, enter, for example:

   ls -W guardian /G/osf/rose/rosec /G/osf/kill

   The following is displayed:

   ```
   /G/osf/rose/rosec       101
   /G/osf/kill:
   bindkil                 101
   makekil                 101
   mykill                  100
   testc                   101
   test                    100
   ```

8. To recursively list all subdirectories in the OSS file system on the local node, enter:

   ls -W NOG -W NOE -R /

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

RELATED INFORMATION

Commands: **chmod(1)**, **find(1)**, **getacl(1)**, **ln(1)**, **stty(1)**.

Files: **locale(4)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The **UTILSGE** environment variable and the **-P**, **-W NOG**, and **-W NOE** flags are HP extensions to the XPG4 Version 2 specification.

Section 6. User Commands (m - o)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letters **m** through **o**.

NAME

make - Maintains program dependencies

SYNOPSIS

make [**-einpqrst**] [**-f** *makefile*] ... [**-k** | **-S**]
[*string1*=[*string2*]] ... [*target_name* ...]

FLAGS

-e Specifies that environmental variables override macro assignments within makefiles.

-f *makefile* Specifies a makefile to read instead of the default makefile. If *makefile* is **-** (dash), the standard input file is read. Multiple makefiles can be specified, and they are read in the order specified.

-i Ignores nonzero exit of shell commands in the makefile. This flag is equivalent to specifying **-** (dash) before each command line in the makefile.

-k Continues processing after errors are encountered, but only on those targets that do not depend on the target whose creation caused the error.

-n Displays the commands that would have been executed, but does not actually execute them. If the lines have **+** (plus sign) prefixes, the commands are executed.

-p Displays all the macro definitions and target descriptions.

-q Does not execute any commands, but returns exit value 0 (zero) if the specified targets are up-to-date and 1 otherwise. If the lines have **+** (plus sign) prefixes, the commands are executed.

-r Does not use the built-in rules specified in the system makefile.

-S Terminates the **make** process if errors are encountered during updates. This is the default behavior and is the opposite of the **-k** option.

-s Does not echo any commands as they are executed. This flag is equivalent to specifying **@** before each command line in the makefile.

-t Creates a target or updates its modification time to make it appear up-to-date, instead of rebuilding a target as specified in the makefile. The target command lines are typically not executed, unless the target command lines have **+** (plus sign) prefixes, in which case they are executed.

string1=[*string2*]
Defines a macro, as described under **Using Macros**, later in this reference page. The value specified as *string1* is the macro name. The value specified as *string2* is the macro definition.

target_name Specifies a target rule or a special target name, as described under **Target Rules** and **Special Targets**, later in this reference page.

If no *target_name* is specified on the command line, the **make** utility uses the first target defined in the makefile and builds that target.

DESCRIPTION

The **make** program is designed to simplify the maintenance of other programs. Its input is a list of specifications of the files that programs and other files depend upon. By default, the following files are tried in sequence to provide this list of specifications: **./makefile** and **./Makefile**.

There are four different types of lines in a makefile: file dependency specifications, shell commands, variable assignments, and comments.

In general, command lines in a makefile can be continued from one line to the next by ending them with a \ (backslash). The trailing newline character and initial white space on the following line are compressed into a single space.

File Dependency Specifications

Dependency lines consist of one or more targets, an operator, and zero or more sources (prerequisites). Dependency lines create a relationship where the targets depend on the sources and are usually created from them.

The exact relationship between the target and the source is determined by the operator that separates them. The operators are as follows:

: A target is considered out-of-date if its modification time is less than those of any of its sources. Sources for a target accumulate over dependency lines when this operator is used. The target is removed if **make** is interrupted unless the target has the **.PRECIOUS** attribute.

:: If no sources are specified, the target is always re-created. Otherwise, a target is considered out-of-date if any of its sources were modified more recently than the target. Sources for a target do not accumulate over dependency lines when this operator is used. The target will not be removed if **make** is interrupted.

File dependency specifications have two types of rules, inference and target, as follows:

inference rules	Have one target with no **/** (slash) and a minimum of one **.** (period). These rules specify how a target is to be made up-to-date.
target rules	Can have more than one target. These rules specify how to build the target.

Makefile Execution

The **make** command executes the commands in the makefile line by line. As **make** executes each command, it writes the command to the standard output file (unless otherwise directed; for example, by the **-s** option). A makefile must have a tab in front of the commands on each line.

When a command is executed through **make**, it uses the **make** program's execution environment. This includes any macros from the command line to **make**, and any environment variables specified in the **MAKEFLAGS** variable (refer to **Variable Assignments**, later in this reference page). The **make** utility's environment variables overwrite any variables of the same name in the existing environment.

Target Rules

Target rules have the following format:

target [*target...*] **:** [*prerequisite...*] [**;***command*]
<**Tab**>*command*
...

Multiple targets and prerequisites are separated by spaces (note that the list of prerequisites can be empty). Any text that follows the **;** (semicolon) and all of the subsequent lines that begin with a tab character are considered commands to be used to update the target. A new target entry is started when a new line does not begin with a tab character or **#** (number sign). The following

section, **Special Targets**, lists the special sources, or prerequisites, and targets for a makefile.

Special Targets

Special targets must not be included with other targets; that is, they must be the only target specified. These control the operation of the **make** command.

The supported special target names are:

.DEFAULT
This is used as the rule for any target (that was used only as a source) that **make** cannot create in any other way. Only the shell script is used. The < (left angle bracket) variable of a target that inherits **.DEFAULT**'s commands is set to the target's own name.

.IGNORE
Prerequisites of this target are targets themselves; this causes errors from commands associated with them to be ignored. If no prerequisites are specified, this is the equivalent of specifying the **-i** flag.

.POSIX This keyword currently has no effect on **make** behavior, because it is the only behavior supported. However, you should use this as the first noncommand line of a makefile if you want to avoid possible conflicts with future enhancements.

.PRECIOUS
Prerequisites of this target are targets themselves. **.PRECIOUS** prevents the target from being removed. If no sources are specified, the **.PRECIOUS** attribute is applied to every target in the file. Normally, when **make** is interrupted (for example, with **SIGHUP**, **SIGTERM**, **SIGINT**, or **SIGQUIT** signals), it removes any partially made targets. If **make** was invoked with the **-n**, **-p**, or **-q** flags, however, the target is considered to have the **.PRECIOUS** attribute.

.SILENT
Prerequisites of the target are targets themselves; this prevents commands associated with the target from being written to the standard output file before they are executed. If no sources are specified, the **.SILENT** attribute is applied to every command in the file.

.SUFFIXES
Prerequisites of the target are appended to the list of known suffixes. If no suffixes are specified, any previously specified suffixes are deleted. These suffixes are used by inference rules, as described in the **Inference Rules** subsection. To change the order of suffixes, you need to specify an empty **.SUFFIXES** entry, then a new list of **.SUFFIXES** entries. Makefiles must not associate commands with **.SUFFIXES**.

Inference Rules

The **make** command has a default set of inference rules, which you can supplement, or overwrite, with additional inference rule definitions in the makefile. Inference rules consist of target suffixes and commands. From the suffixes, **make** infers the prerequisites, and from both the suffixes and their prerequisites, **make** command can infer how to make a target up-to-date. Inference rules have the following format:

rule:
<**Tab**>*command*
...

where *rule* has one of the following forms:

.*s1* A single-suffix inference rule. This rule describes how to build a target that is appended with one of the single suffixes.

.s1.s2 A double-suffix inference rule. This rule describes how to build a target that is appended with ***.s2*** with a prerequisite that is appended with ***.s1***.

s1 and *s2* are suffixes defined as prerequisites of the special target, **.SUFFIXES**. The inference rules use the suffixes in the order in which they are specified in **.SUFFIXES**. A new inference rule is started when a new line does not begin with a <Tab> or **#** (number sign).

If *rule* is empty, for example:

*rule***: ;**

execution has no effect; **make** recognizes that the suffix exists, but takes no actions when targets are out-of-date.

Libraries

A target or prerequisite can also be a member of an archive library, and it is treated as such if there are parentheses in the name. For example, *library***(***name***)** indicates that *name* is a member of the archive library *library*. To update a member of a library from a particular file, you can use the format ***.s1*.a**, where a file with the ***.s1*** suffix is used to update a member of the archive library. The **.a** refers to an archive library.

Using Macros

Macro definitions are defined in the format:

*string1***=[***string2*]

The macro named *string1* is defined as having the value of *string2*. The value of *string2* includes all characters (or no characters) after the equals sign (=) until a comment character (#) or an unescaped newline character occurs. Blanks before or after the equals sign are ignored.

The forms *string1***[:***subst1***=[***subst2***]])** or *string1***[:***subst1***=[***subst2***]]}** can be used to replace all occurrences of *subst1* with *subst2* when the macro substitution is performed. The *subst1* value is recognized when it is a suffix at the end of a word in *string1*, where "word" is defined as a string delimited by one of the following:

- The beginning of a line
- A blank
- An unescaped newline character

Macros can appear throughout the makefile, as follows:

- If a macro appears in a target line, then it is evaluated when the target line is read.
- If a macro appears in a command line, then it is evaluated when the command is executed.
- If a macro appears in a macro definition line, it is evaluated when the new macro itself appears in a rule or command.

If a macro has no definition, it evaluates to NULL. A new macro definition overwrites an existing macro of the same name. Macros assignments can come from the following, in the listed order:

1. Default inference rules
2. Contents of the environment
3. Makefiles

4. Command lines

Note, however, that the **-e** option causes environment variables to override those defined in the makefile.

The **SHELL** macro is special. It is set by **make** to the pathname of the shell command interpreter (**/bin/sh**). However, if it is redefined in the makefile, or on the command line, then this default setting is overridden. Note that this macro does not affect, and is not affected by, the **SHELL** environment variable.

Shell Commands

Each target can have associated with it a series of shell commands, normally used to create the target. Each of the commands in this script **must** be preceded by a tab character. While any target can appear on a dependency line, only one of these dependencies can be followed by a creation script, unless the **::** operator is used.

If the first one or two characters of the command line are **@**, **-**, **+**, the command is treated specially, as follows:

@	Prevents the command from being echoed before it is executed.
-	Causes any nonzero exit status of the command line to be ignored.
+	Causes a command line to be executed, even though the **-n**, **-q**, or **-t** flags are specified.

Variable Assignments

Variables in **make** are much like variables in the shell and, by tradition, consist of all uppercase letters. The = operator assigns values to variables. Any previous variable is then overridden.

Any white space before the assigned value is removed; if the value is being appended, a single space is inserted between the previous contents of the variable and the appended value.

Variables are expanded by surrounding the variable name with either **{}** (braces) or **()** (parentheses) and preceding it with a **$** (dollar sign). If the variable name contains only a single letter, the surrounding braces or parentheses are not required. This shorter form is not recommended.

Variable substitution occurs at two distinct times, depending on where the variable is being used. Variables in dependency lines are expanded as the line is read. Variables in shell commands are expanded when the shell command is executed.

The four different classes of variables (in order of increasing precedence) are:

Environment variables
: Variables defined as part of the **make** program's environment.

Global variables
: Variables defined in the makefile or in included makefiles.

Command line variables
: Variables defined as part of the command line.

Local variables
: Variables that are defined specific to a certain target. The local variables are as follows:

$<	Represents either the full name of a source that made a target out-of-date (inference rule), or the full name of a target (**.DEFAULT** rule).

$* Represents the filename section of a source that made a target out-of-date (in an inference rule) without a suffix.

$@ Represents the full target name of the current target, or the archive filename part of the library archive target.

$? Represents the list of sources causing a target to be out-of-date (inference and target rules).

$% Represents a library member in a target rule if the target is a member of the archive library.

You can also use these local variables appended with **D** or **F**, where

D Indicates that the local variable applies to the directory part of the name. This is the pathname prefix without a trailing / (slash). For current directories, **D** is a **.** (period).

F Indicates that the local variable applies to the filename part of the name.

The **$?** local variable can represent a list of sources. When used with **D** or **F**, the local variable can represent a list of directory and filename parts, respectively.

make converts the expression **$$** to a single dollar sign (**$**).

When **make** encounters a line beginning with the word **include** followed by another word that is the name of a makefile (for example, **include depend**), **make** attempts to open that file and process its contents as if the contents appeared where the include line occurs. This behavior occurs only if the first noncomment line of the first makefile read by **make** is not the **.POSIX** target; otherwise, a syntax error occurs.

Comments

Comments begin with a **#** (number sign), anywhere but in a shell command line, and continue to the end of the line.

Environment Variables

The **make** command supports the following environment variables:

LANG Determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the setting of **LANG** or any other **LC_** environment variable.

LC_CTYPE Determines the locale for the interpretation of sequences of bytes of text data as characters; for example, single-byte characters versus multibyte characters in arguments.

LC_MESSAGES
Determines the language in which messages should be written.

MAKEFLAGS
Contains any flags that might be specified on the **make** utility's command line. Anything specified on the **make** utility's command line is appended to the **MAKEFLAGS** variable, which is then entered into the environment for all programs that **make** executes. Note that the operation of the **-f** and **-p** flags in the **MAKEFLAGS** variable are undefined. Command line options have precedence

over the **-f** and **-p** flags in this variable.

EXAMPLES

1. To compile, link, and run a TNS/R program using a non-PIC library and the files **mainstr.c**, **mystrng.c**, and **mystrng.h**, use the following **makefile**:

```
TOOLS = /G/SYSTEM/SYSTEM
CFLAGS = -Woptimize=0 -g -Werrors=5 -Wextensions \
        -I /G/SYSTEM/ZSYSDEFS -I ${TOOLS} -c
LDFLAGS = -obey /G/SYSTEM/SYSTEM/libcobey -L .

all: revstr
revstr : mainstr.o mystrng.o
        nld -o $@ ${LDFLAGS} mainstr.o mystrng.o \
        ${TOOLS}/crtlmain
mainstr.o : mainstr.c
        c89 -o $@ $? ${CFLAGS}
mystrng.c : mystrng.h
        touch mystrng.c
mystrng.o : mystrng.c
        c89 -o $@ $? ${CFLAGS}
clean:
        rm *.o revstr *.dll
run: revstr
        ./revstr abc 45678
```

2. To compile, link, and run a TNS/R program using a PIC DLL and the files **mainstr.c**, **mystrng.c**, and **mystrng.h**, use the following **makefile**:

```
TOOLS = /G/SYSTEM/SYSTEM
CFLAGS_NON_PIC = -Woptimize=0 -g -Werrors=5 -Wextensions \
        -I /G/SYSTEM/ZSYSDEFS -I ${TOOLS} -c
CFLAGS= ${CFLAGS_NON_PIC} -Wcall_shared
LDFLAGS = -obey /G/SYSTEM/SYSTEM/libcobey -L .
LDFLAGS_DLL = ${LDFLAGS} -shared

all: revstr
revstr : mainstr.o mystrng.dll
        ld -o $@ ${LDFLAGS} mainstr.o -l mystrng.dll \
        ${TOOLS}/ccppmain
mainstr.o : mainstr.c
        c89 -o $@ $? ${CFLAGS}
mystrng.c : mystrng.h
        touch mystrng.c
mystrng.dll : mystrng.o
        ld -o $@ ${LDFLAGS_DLL} mystrng.o \
        -export MyStr_Version -export StrRev
mystrng.o : mystrng.c
        c89 -o $@ $? ${CFLAGS}
clean:
        rm *.o revstr *.dll
run: revstr
        ./revstr abc 45678
```

3. To compile, link, and run a TNS/R program using different DLLs and the files **mainstr.c**, **mystrng.c**, and **mystrng.h**, use the following **makefile**:

```
TOOLS = /G/SYSTEM/SYSTEM
CFLAGS_NON_PIC = -Woptimize=0 -g -Werrors=5 -Wextensions \
        -I /G/SYSTEM/ZSYSDEFS -I ${TOOLS} -c
CFLAGS= ${CFLAGS_NON_PIC} -Wcall_shared
LDFLAGS = -obey /G/SYSTEM/SYSTEM/libcobey -L .
LDFLAGS_DLL = ${LDFLAGS} -shared

all: mystrng.dll
mystrng.c : mystrng.h
        touch mystrng.c
mystrng.dll : mystrng.o
        ld -o $@ ${LDFLAGS_DLL} mystrng.o \
        -export MyStr_Version -export StrRev
mystrng.o : mystrng.c
        c89 -o $@ $? ${CFLAGS}
clean:
        rm *.o revstr *.dll
        unset _RLD_FIRST_LIB_PATH
run: revstr
        export _RLD_FIRST_LIB_PATH=/home/les/dll3; \
        ../dll2/revstr abc 45678
```

4. To compile, link, and run a TNS/R program that dynamically adds a DLL and uses the files **hello.c**, **main.c**, and **hello.h**, use the following **makefile**:

```
TOOLS = /G/SYSTEM/SYSTEM
CFLAGS_NON_PIC = -Woptimize=0 -g -Werrors=5 -Wextensions \
        -I /G/SYSTEM/ZSYSDEFS -I ${TOOLS}
CFLAGS= ${CFLAGS_NON_PIC} -Wcall_shared
LDFLAGS = -obey /G/SYSTEM/SYSTEM/libcobey -L .
LDFLAGS_DLL = ${LDFLAGS} -shared

all: main.exe hello.dll
main.exe: main.c
        c89 -o $@ $? ${CFLAGS} -l zrldsrl
hello.dll : hello.o
        ld -o $@ ${LDFLAGS_DLL} hello.o -export hello
hello.o : hello.c
        c89 -o $@ $? ${CFLAGS} -c
clean:
        rm *.o revstr *.dll
```

5. To compile, link, and run a TNS/R program that dynamically adds a DLL, uses the **qsort** utility, and uses the files **sort.c**, **main.c**, and **sort.h**, use the following **makefile**:

```
TOOLS = /G/SYSTEM/SYSTEM
CFLAGS_NON_PIC = -Woptimize=0 -g -Werrors=5 -Wextensions \
        -I /G/SYSTEM/ZSYSDEFS -I ${TOOLS}
CFLAGS= ${CFLAGS_NON_PIC} -Wcall_shared
LDFLAGS = -obey /G/SYSTEM/SYSTEM/libcobey -L .
LDFLAGS_DLL = ${LDFLAGS} -shared

all: main.exe sort.dll
```

```
main.exe: main.c
        c89 -o $@ $? ${CFLAGS} -l zrldsrl
sort.dll : sort.o
        ld -o $@ ${LDFLAGS_DLL} sort.o -export CompareInts
sort.o : sort.c
        c89 -o $@ $? ${CFLAGS} -c
clean:
        rm *.o revstr *.dll
```

FILES

/usr/share/mk/posix.mk	Default POSIX rules for the **make** utility.
makefile	List of dependencies.
Makefile	List of dependencies.

EXIT VALUES

The **make** command exits with one of the following values:

0 (zero)	To indicate successful completion.
1	To indicate that the target was not up-to-date when the **-q** flag was specified.
> 1	To indicate an error occurred.

RELATED INFORMATION

Commands: **sh(1)**.

Miscellaneous: **environ(5)**.

STANDARDS CONFORMANCE

The OSS implementation conforms to the XPG4 Version 2 specification with the following exceptions:

- The OSS implementation does not support the extended description features of the XPG4 Version 2 specification.
- The following extensions are supported:
  - The **include** feature for nested makefiles.

NAME

man - Displays reference page information

SYNOPSIS

man [**-c**] [-] [**-M** *pathname*] [*section*] *title* ...

man [**-M** *pathname*] **-f** | **-k** *keyword* ...

FLAGS

-c Does not pipe output through **more**.

-f *keyword* ... Displays descriptions of all commands, calls, functions, or special filenames matching the specified keyword. Locates reference pages by function (same as the **whatis** command).

This option requires the existence of a **whatis** keyword database file. The information returned represents a match of the keyword against information to the left of the hyphen in all entries within such a database file.

-k *keyword* ... Displays descriptions of all commands, calls, functions, or special file names that contain the keyword in their name or description. Locates reference pages by keyword (same as the **apropos** command).

This flag requires the existence of a **whatis** keyword database file. The information returned represents a match of the keyword against information to the right of the hyphen in all entries within such a database file.

-M *pathname* Specifies an alternative search path. *pathname* is a colon-separated list of directories in which **man** expects to find the standard subdirectories containing reference pages or a **whatis** keyword database file.

If this flag is omitted, the **man** command uses the values in effect for the **MANPATH** environment variable to locate searchable subdirectories or a **whatis** keyword database file. When the **MANPATH** environment variable is not defined, the **man** command searches directories as described in **Environment Variables** in this reference page.

- Does not pipe output through **more**. (Same as **-c**.)

Operands

title Specifies the name of the command, function call, file, or other topic whose reference page is to be displayed.

section Specifies the subdirectories within the search path that the **man** command should restrict its search to.

section is either a section number from 1 through 8 or one of the letters **CLFlnop**. If you specify **C** or **1**, **man** searches in the sections **Cnlpo1**.

The following *section* values have a pre-defined meaning:

C Command

l Local

o Old

n New

The other *section* values correspond to a logical section identified by that letter or number within the online reference information.

DESCRIPTION

The **man** program provides online access to the system's reference pages.

Reference pages must be located within a known directory structure. A directory structure is known when it is specified by the **-M** option, by a value in the **MANPATH** environment variable, or by default.

Subdirectories within each known directory structure are searched in a specific order for a requested reference page. Only the first matching reference page is returned by the command.

The **man** command searches subdirectories in the following order:

- Subdirectories named **man***n* are searched first for **nroff** source files to format. The **man** command attempts to call the **/bin/nroff** formatter to process the first file it finds whose file name matches the *title* specified in the command line. Note that for H06.26 and earlier H-series RVUs and J06.15 and earlier J-series RVUs, the **/bin/nroff** formatter is not provided with the OSS shell.

  The value used for *n* is the value specified for *section* in the command. If no value is specified for *section*, then the **man** command searches the subdirectories using values for *n* in the following order:

  ```
  C, L, F, n, l, p, o, 1, 2, 3, 4, 5, 6, 7, 8
  ```

  The **man** command formats and displays the first reference page source file it finds whose name matches the *title* specified in the command line.

- When no matching file exists in the subdirectories named **man***n*, the **man** command searches the subdirectories named **cat***n*.

  The value used for *n* is the value specified for *section* in the command. If no value is specified for *section*, then the **man** command searches the subdirectories using values for *n* in the following order:

  ```
  C, L, F, n, l, p, o, 1, 2, 3, 4, 5, 6, 7, 8
  ```

  The **man** command displays the first formatted reference page it finds whose file name matches the *title* specified in the command line.

The basic OSS product includes formatted reference pages in the following subdirectories:

cat1	Commands available to the general interactive user.
cat2	Application program interface functions reserved on some UNIX systems to use by privileged programs (this restriction does not exist on OSS systems).
cat3	Application program interface functions available to all C and C++ language programs.
cat4	Application program interface header files and other file formats.
cat5	Miscellaneous topics.
cat7	Special file information.
cat8	Commands intended for system administrators.

The **cat6** directory is empty (it is traditionally reserved for game information).

These subdirectories exist in the following directory structures:

/usr/share/man
Contains the basic OSS product set reference pages.

/nonnative/usr/share/man
Contains reference information for the TNS C compiler (G-series RVUs only).

Independent products can either create additional directory structures for reference pages or add their reference page files to one of these sets. If the **man** command does not display a reference page for a recently installed product, contact your site administrator for the pathname of its reference page directory structure.

The **-M** flag is used if you have reference pages in a directory other than either the default locations or the directories in **MANPATH**.

If the standard output is a tty, and the **-c** flag is not provided, **man** uses **more**, or the pager provided by the **PAGER** environment variable, to display the reference page.

Environment Variables

The **man** command supports the **MANPATH** environment variable in the following ways:

- If **MANPATH** is not defined and the **-M** flag is not used, then directories are searched in the following order:

  /usr/share/man/man*
  /usr/local/man/man*
  /usr/share/man/cat*
  /usr/local/man/cat*

  This ordering can cause the content of **/usr/share/man/cat*** to become unavailable when source files are installed in either of the **man*** subdirectories.

- If **MANPATH** is defined with multiple values, then the first directory in which a match is found ends the search for reference pages to display.

FILES

/nonnative/usr/share/man/whatis
The **whatis** keyword database for the reference pages in **/nonnative/usr/share/man** (G-series RVUs only).

/usr/local/man A directory structure reserved for local site reference pages. This directory might not exist on your system.

/usr/share/man/whatis The **whatis** keyword database for the OSS reference pages in **/usr/share/man**.

RELATED INFORMATION

Commands: **apropos(1)**, **more(1)**, **merge_whatis(8)**, **whatis(1)**.

STANDARDS CONFORMANCE

The following are HP extensions to the Single UNIX Specification, Version 2:

- All command line flags other than **-k**
- Support for the **MANPATH** environment variable

NAME

mkcatdefs - Preprocesses a message source file

SYNOPSIS

mkcatdefs *catname source_file ...* [**-h**]

FLAGS

-h Suppresses the generation of a **_msg.h** file. This flag must be the last argument to the **mkcatdefs** command.

Operands

catname is the name of a message catalog

source_file is a text file you create to hold messages printed by your program

DESCRIPTION

The **mkcatdefs** utility preprocesses a message source file to change symbolic identifiers into numeric constants. The **mkcatdefs** utility' standard output is a set of commands suitable for passing to the **gencat** utility, which creates a new message catalogue. In addition, **mkcatdefs** is used in producing a file named *catname***_msg.h** containing definition statements that equate your symbolic identifiers with set numbers and message ID numbers assigned by **mkcatdefs**. The *catname***_msg.h** file is required in your application program if you use symbolic identifiers.

See the explanation of the **gencat** utility for a description of input format for commands. The only difference between **gencat** and **mkcatdefs** is that **gencat** requires a number to identify each message, while **mkcatdefs** accepts either a number or a symbolic name. The **mkcatdefs** program can send message source, data with numbers instead of symbolic identifiers to standard output.

A symbolic name is converted to a number in the **mkcatdefs** output. Each set and message in a program must have a unique number or symbolic name. Symbolic identifiers can contain letters, digits, and underscores. The first character cannot be a digit or an underscore. You cannot use a symbolic name for a set in the **$delset** command.

The **mkcatdefs** program is designed to create new message catalogs, not to change existing ones incrementally. Thus, its first operation on each set is to delete it, in case the catalog contains a set with that number. The sets specified in *source_file* are assigned numbers in ascending order, starting at 1. Within each set, messages are also assigned numbers in ascending order, starting at 1. If you assign a message to a number in your *source_file*, **mkcatdefs** continues its ascending series with that number.

EXAMPLES

1. The following example shows a message source file with symbolic message identifiers and quoted messages:

```
$quote " Use a double quotation mark to delimit message text
$set MSFAC         Message Facility - symbolic identifiers
SYM_FORM "Symbolic identifiers can contain only letters \
and digits and the _ (underscore character)\n"
5 "You can mix symbolic identifiers and numbers \n"
$quote
MSG_H Remember to include the "_msg.h" file in your program\n
```

   In this example, the **$quote** command sets the quote character to " (double quote), then disables it before the last message, which contains double quotes.

   When you process the file with **mkcatdefs**, the modified source is written to standard output.

2. Assume that the preceding file is named **symb.src**. It can be processed with **mkcatdefs** as follows:

   $ mkcatdefs symb symb.src >symb.msg

   The following source is created:

```
$quote "Use a double quotation mark to delimit message text
$delset 1
$set 1
1  "Symbolic identifiers can contain only letters \
and digits and the _ (underscore character)\n"
5 "You can mix symbolic identifiers and numbers \n"
$quote
6  Remember to include the "_msg.h" file in your program\n
```

   Note that the assigned message numbers are noncontiguous because the source contained a specific number. The **mkcatdefs** utility always assigns the previous number plus 1 to a symbolic identifier.

   The generated **symb_msg.h** file is:

```
#ifndef _H_SYMB_MSG
#define _H_SYMB_MSG
#include <limits.h>
#include <nl_types.h>
#define MF_SYMB "symb.cat"

/* The following was generated from symb.src. */

/* definitions for set MSFAC */
#define MSFAC 1

#define SYM_FORM 1
#define MSG_H 6
#endif
```

   Note that **mkcatdefs** also created a symbol **MF_SYMB** by adding **MF_** to the *catname* using uppercase letters. In this example, **mkcatdefs** used the name of the generated catalog (**symb.cat**) and generated this symbol for use with **catopen**.

   Because this file includes **limits.h** and **nl_types.h**, you do not need to include them in your application program. (**nl_types** defines special data types required by the message facility routines.)

NOTES

Symbolic references are not defined by XPG4, but are an OSS extension, allowing a convenient input source for both the message catalog and the program's header file.

DIAGNOSTICS

The **mkcatdefs** utility generates these error messages:

```
Usage: mkcatdefs catname msg_file [msg_file...] [-h]\n
```

The following message is displayed if *catname* is greater than **FILENAME_MAX** - 7 characters:

```
mkcatdefs: catname too long\n
```

The following message is displayed if any of the input files cannot be opened:

```
mkcatdefs: Cannot open \%s\n
mkcatdefs: catname contains invalid character\n
Usage: mkcatdefs SymbolName SourceFile[...SourceFile] [-h]\n
```

The following messages pertain to the **.msg** file:

```
mkcatdefs: There were write errors on file \%s\n
mkcatdefs: Errors found: no \%s created\n
mkcatdefs: \%s created\n
mkcatdefs: No symbolic identifiers; no \%s created\n
mkcatdefs: no \%s created\n
```

The following message is displayed if a message line is greater than **NL_TEXTMAX** + 128 characters:

```
mkcatdefs: The message text is too long [\%d]:\n\t\%s\n
```

The following message is displayed if the set identifier is greater than 64:

```
mkcatdefs: The set identifier is too long [\%d]:\n\t\%s\n
```

The following message is displayed if an invalid character is used:

```
mkcatdefs: The symbolic set or message identifier is not
valid:\n\t\%s\n
mkcatdefs: A set number or identifier is missing:\n\t\%s\n
mkcatdefs: set # \%d already assigned or sets not in ascending
sequence\n
mkcatdefs: name \%s used more than once\n
mkcatdefs: Only message text can use ´\´ to continue: \%s\n
mkcatdefs: The symbolic identifier is too long [\%d]:\n\t\%s\n
mkcatdefs: sourcefile contains invalid character:\n\t\%s\n
```

The following message is displayed if the set number is zero:

```
mkcatdefs: \%s is an invalid identifier\n
mkcatdefs: message id \%s already assigned to identifier\n
mkcatdefs: source messages not in ascending sequence\n
```

The following message pertains to the **.h** file:

```
mkcatdefs: There were read errors on file \%s\n
```

RELATED INFORMATION

Commands: **gencat(1)**, **runcat(1)**.

Functions: **catclose(3)**, **catgets(3)**, **catopen(3)**.

NAME

mkdir - Makes a directory

SYNOPSIS

mkdir [**-m** *mode*] [**-p**] *directory* ...

The **mkdir** command creates new directories with read, write, and execute permissions based upon the permissions established by the **umask** setting.

FLAGS

-m *mode* Sets the file permissions to *mode*, after creating the specified directory. The *mode* argument can be either an absolute mode string or a symbolic mode string as defined for **chmod**. In the latter case, use only 9 protection bits in the *mode* argument, because the set-user-ID, set-group-ID, and sticky bits are ignored.

In symbolic mode strings, the operation characters **+** and **-** are interpreted relative to an assumed initial mode of **a=rwx**; A **+** adds permissions to the default mode, whereas a **-** deletes permissions from the default mode.

-p Creates intermediate directories as necessary; otherwise, the full pathname prefix to *directory* must already exist. Note that **mkdir** requires write permission in the parent directory for users other than root.

For each *directory* argument that does not name an existing directory, effects equivalent to those caused by the following command occur:

mkdir -p -m $(umask -S),u+wx $(dirname *directory***) &&**
mkdir [-m *mode***]** *directory*

[-m *mode***]** represents the flag supplied to the original invocation of **mkdir**, if any.

Each component of *directory* that does not name an existing directory is created with mode 777, modified by the current file mode creation mask (**umask**). The equivalent of **chmod u+wx** is performed on each component to ensure that **mkdir** can create lower directories regardless of the setting of **umask**.

For systems running G06.30 and later G-series RVUs, each component of *directory* that names an existing directory is ignored without error. For systems running other RVUs, each component of *directory* that names an existing directory is ignored without error, except for the last component. If an intermediate pathname component exists, but permissions are set to prevent writing or searching, **mkdir** fails and returns an error message.

The *mode* argument does not apply to any intermediate directories created when the **-p** flag is specified.

DESCRIPTION

The **mkdir** command also creates the standard entries **.** (dot), for the directory itself, and **..** (dot dot), for its parent.

The value of the bitwise inclusive OR of **S_IRWXU**, **S_IRWXG**, and **S_IRWXO** is used as the *mode* argument. (If the **-m** flag is specified, the *mode* argument overrides this default.)

Access Control Lists (ACLs)

If the parent directory has an ACL that contains default ACL entries, the new directory inherits ACL entries and permissions as described in the **acl(5)** reference page.

EXAMPLES

1. To create a new directory called **test**, enter:

   mkdir test

2. To set file permissions for new directory **test** in absolute mode, enter:

   mkdir -m 444 test

3. To set file permissions for new directory **test** in symbolic mode, enter:

   mkdir -m+rw test

NOTES

To make a new directory, you must have write permission in the parent directory.

RELATED INFORMATION

Commands: **chmod**(1), **rm**(1), **rmdir**(1), **setacl(1)**, **sh**(1).

Functions: **mkdir**(2).

Miscellaneous topics: **acl(5)**.

NAME

mkfifo - Makes FIFO special files

SYNOPSIS

mkfifo [**-m** *mode*] *file* ...

The **mkfifo** utility creates FIFO special files in the order specified.

FLAGS

-m *mode* Sets the file permission bits of the new file to the specified *mode* value, after creating the FIFO special file. The *mode* argument is a symbolic mode string (see **chmod**), in which the *operator* characters **+** (plus sign) and **-** (minus) are interpreted relative to the default file mode for that file type. The **+** character adds permissions to the default mode, and **-** deletes permissions from the default mode.

The default *mode* is **a=rw** (permissions of **rw-rw-rw**).

DESCRIPTION

For each *file* argument, **mkfifo** performs actions equivalent to the **mkfifo()** call with the following arguments:

1. The *file* argument is used as the *pathname* argument.
2. The value of the bitwise inclusive OR of **S_IRUSR**, **S_IWUSR**, **S_IRGRP**, **S_IWGRP**, **S_IROTH**, and **S_IWOTH** is used as the *mode* argument.

Access Control Lists (ACLs)

If the parent directory has an ACL that contains default ACL entries, the new directory inherits ACL entries and permissions as described in the **acl(5)** reference page.

EXAMPLES

1. To create a FIFO special file with permissions **prw-r--r--**, enter:

   mkfifo
   -m 644 /tmp/myfifo

   The command creates the **/tmp/myfifo** file with read/write permissions for the owner and read permission for the group and for others.

2. To create a FIFO special file using the **-** (minus) operand to set permissions of **prw-r-----**, enter:

   mkfifo
   -m g-w,o-rw /tmp/fifo2

   The command creates the **/tmp/fifo2** file, removing write permission for the group and all permissions for others.

EXIT VALUES

The **mkfifo** utility exits with one of the following values:

0 Indicates that all the specified FIFO special files were created successfully.

>0 Indicates that an error occurred.

RELATED INFORMATION

Commands: **mknod**(8), **setacl(1)**.

Functions: **acl(2)**, **chmod**(2), **mkdir**(2), **mkfifo**(3), **mknod**(2), **umask**(2).

Miscellaneous topics: **acl(5)**.

NAME

more - Displays a file one screenful at a time

SYNOPSIS

Current syntax

more [**-cdeiNsuvz**] [**-n** *number*] [**-p** *command*] [**-t** *tagstring*] [**-W** *option*] [**-x** *tabs*] [**-***number*] [*file ...*]

Obsolescent syntax

more [**-cdeiNsuvz**] [**-***number*] [**+***command*] [**-t** *tagstring*] [**-W** *option*] [**-x** *tabs*] [**-***number*] [*file ...*]

The **more** command invokes a filter that allows examination of continuous text, one screenful at a time, on a soft-copy terminal.

FLAGS

-c Redraws each line of the screen in turn, from the top of the screen to the bottom, instead of scrolling the screen, if a screen is to be written that has no lines in common with the current screen, or **more** is writing its first screen. In addition, if **more** is writing its first screen, the screen will be cleared.

-d Prompts you to continue, quit, or obtain help after each screenful of text.

-e Exits immediately after writing the last line of the last file in the argument list.

-i Performs pattern matching in searches without regard to case.

-n *number*

-*number* (Obsolescent)
Sets the number of lines in the display window to *number*, a positive decimal integer. The default is one line less than the number of lines displayed by the terminal; on a screen that displays 24 lines, the default is 23. The **-n** flag overrides any values obtained from the environment.

-N Suppresses line numbers. The default (to display line numbers) can cause **more** to run more slowly in some cases, especially with very large input files. Using line numbers means that the line number is displayed in the = subcommand and the command passes the current line number to the editor (if it is **vi**).

-p *command*

+*command* (Obsolescent)
Executes the **more** command initially in the *command* argument for each file examined. If the command is a positioning command, such as a line number or a regular expression search, sets the current position to represent the final results of the command, without writing any intermediate lines of the file. For example, the following two commands are equivalent and cause the display to start with the current position at line 1000, bypassing the lines that **j** would write and scroll off the screen if it had been issued during the file examination:

```
more -p 1000j file
more -p 1000G file
```

If the positioning command is unsuccessful, the first line in the file is the current position.

-s Squeezes multiple empty lines from the output, producing only one empty line. Especially helpful when viewing **nroff** output, this flag maximizes the amount of useful information present on the screen.

-t *tagstring*
Writes the screenful of the file containing the tag named by the *tagstring* argument. The specified tag appears in the current position. If both **-p** and **-t** are specified, **more** processes **-t** first. (The *tagstring* argument specifies a file created with the **ctags** utility. OSS does not support the **ctags** utility, but **more** does support **ctags** files that have been copied to the OSS environment from another system.)

-u Treats **<Backspace>** as a printable control character, suppressing backspacing and the special handling that produces underlined or standout-mode text on some terminal types. Also, does not ignore a carriage-return character at the end of a line.

-v Does not display nonprinting characters graphically. Without this flag, all non-ASCII and control characters (except **<Tab>**, **<Backspace>**, and **<Return>**) are displayed visibly in the form **^X** for **<Ctrl-x>**, or **M-x** for non-ASCII character **x**.

-W *option*
Provides optional extensions to the **more** command. Currently, the following two options are supported:

notite Prevents **more** from sending the terminal initialization string (the **ti termcap** or the **smcup terminfo** capability) before displaying the file. This argument also prevents **more** from sending the terminal de-initialization string (the **te termcap** or the **rmcup terminfo** capability) before exiting.

tite Causes **more** to send the initialization and de-initialization strings. This is the default.

The preceding options control whether **more** sends the control codes described, which for certain terminals (such as certain xterms) cause **more** to switch to an alternative screen. This causes the file you were viewing to vanish from your screen when you exit.

This is also something you could set in your login file with the **MORE** environment variable.

-x *tabs* Sets the tabstops every *tabs* position. The default value for the *tabs* argument is 8.

-z Same as if the **-v** flag is not given, but in addition, **<Backspace>** is displayed as **^H**, **<Return>** as **^M**, and **<Tab>** as **^I**.

DESCRIPTION

The **more** command normally pauses after each screenful, printing the filename at the bottom of the screen. You can then enter a carriage-return to display one more line, or press **<Space>** to display another screenful. Other possibilities are described under **SUBCOMMANDS**.

The **more** command looks in the **terminfo** database to determine terminal characteristics, and to determine the default window size. On a terminal capable of displaying 24 lines, the default window size for **more** is 23 lines.

The **more** command looks in the **MORE** environment variable to preset any desired flags. The **MORE** variable thus sets a string containing flags and arguments, preceded with **-** (dash) characters and space-separated as on the command line. Any command-line flags or arguments are processed after those in the **MORE** variable, as if the command line were as follows:

more $MORE *flags arguments*

For example, assume that you prefer to view files using the **-c** and **-w** flags. The **sh** command sequence **MORE='-cw' ; export MORE** would cause all invocations of **more**, including invocations by programs such as **man** to use this mode. Normally, you place the command sequence that sets up the **MORE** environment variable in the **.profile** file or the **.kshrc** file. Note that the

string you set the **MORE** environment variable to must begin with a **-** (dash).

If **more** is reading from a file, rather than a pipe, then a percentage is displayed along with the filename. This gives the fraction of the file (in characters, not lines) that was displayed so far.

If the standard output is not a terminal, then **more** processes like **cat**. The compact viewing format produced by the **-s** flag can also be used in this case.

SUBCOMMANDS

The **more** command provides the following subcommands that you can type when **more** pauses. These commands are designed to be similar to the commands supported by the **vi** editor. (*i* is an optional integer argument, defaulting to 1.) Regular expressions (as referred to here) are described under **grep**.

*i***<Return>**

*i***j**

*i***<Ctrl-e>**

*i***<Space>**

Scrolls forward *i* lines. The default *i* for **<Space>** is one screenful; for **j** and **<Return>** it is one line. The entire *i* lines are written, even if *i* is more than the screen size. At End-of-File, **<Return>** causes **more** to continue with the next file in the list, or exits if the current file is the last file in the list.

d

*i***<Ctrl-d>**

Scrolls forward *i* lines, with a default of one-half of the screen size. If *i* is specified, it becomes the new default for subsequent **d** and **u** subcommands.

*i***u**

*i***<Ctrl-u>**

Scrolls backward *i* lines, with a default of one-half of the screen size. If *i* is specified, it becomes the new default for subsequent **d** and **u** subcommands. Note that if your line kill character is **<Ctrl-u>**, then you must use the **u** command to scroll backward.

*i***k**

*i***<Ctrl-y>**

Scrolls backward *i* lines, with a default of one line. The entire *i* lines are written, even if *i* is more than the screen size.

*i***z** Displays *i* more lines and sets the new window (screenful) size to *i*.

*i***g** Goes to line *i* in the file, with a default of 1 (beginning of file). Scrolls or rewrites the screen so that that line is at the current position. If *i* is not specified, then **more** displays the last screenful in the file. Instead of exiting (or going on to the next file) after showing the last line of the file, **more** displays the filename prompt. This gives you an opportunity to scroll or page backward through the file.

*i***G** Goes to line *i* in the file, with a default of the end of the file. If *i* is not specified, scrolls or rewrites the screen so that the last line in the file is at the bottom of the screen. If *i* is specified, scrolls or rewrites the screen so that that line is at the current position.

*i***s** Skips forward *i* lines, with a default of one line, and writes the next screenful beginning at that point. If *i* would cause the current position to be such that less than one screenful would be written, the last screenful in the file is written.

*i***f**

*i***<Ctrl-f>**
Moves forward *i* lines, with a default of one screenful. At End-of-File, **more** continues with the next file in the list, or exits if the current file is the last file in the list.

*i***b**

*i***<Ctrl-b>**
Moves backward *i* lines, with a default of one screenful (see the **-n** flag). If *i* is more than the screen size, only the final screenful is written.

q, **Q**

ZZ Exits from **more.**

=

<Ctrl-g>
Writes the name of the file currently being examined, the number relative to the total number of files there are to examine, the current line number, the current byte number, and the total bytes to write, and what percentage of the file precedes the current position. If **more** is reading from standard input, or the file is shorter than a single screen, some of these items need not be written. All of these items reference the first byte of the line after the last line written.

v Invokes an editor to edit the current file being examined. The name of the editor is taken from the **$EDITOR** environment variable and defaults to **vi**. If **$EDITOR** represents either **vi** or **ex**, the editor is invoked with options such that the current editor line is the physical line corresponding to the current position in **more** at the time of invocation. For example, either **ex** or **vi** is invoked by specifying the editor name and following that with **-c** *linenumber*.

When the editor exits, **more** resumes on the current file by rewriting the screen with the current line as the current position.

h Displays a description of all the **more** subcommands.

i/[**!**]*expression*
Searches forward in the file for the *i*th line containing the regular expression *expression*. The default value for *i* is 1. If the search is successful, the screen is modified so that the searched-for line is in the current position. The null regular expression (/**<Return>**) repeats the search using the previous regular expression. If the **!** (exclamation point) character is included, the lines for searching are those that do not contain *expression*.

If there are less than *i* occurrences of *expression*, and the input is a file rather than a pipe, then the position in the file remains unchanged.

You can use Erase and Kill characters to edit the regular expression, which must be terminated by pressing **<Return>** (with no trailing / character). Erasing back past the first column cancels the search command.

*i***?**[**!**]*expression*
Same as /, but searches backward in the file for the *i*th line containing the regular expression *expression*.

*i***n** Repeats the previous search for the *i*th line (default 1) containing the last *expression* (or not containing the last *expression*, if the previous search was **/!** or **?!**).

*i***N** Repeats the search in the opposite direction of the previous search for the *i*th line (default 1) containing the last *expression* (or not containing the last *expression*, if the previous search was **/!** or **?!**).

' (single quotes)
Returns to the position from which the last large movement subcommand was executed ("large movement" is defined as any movement of more than a screenful of lines). If no such movements have been made, returns to the beginning of the file.

!*command* or **:!***command*
Invokes a shell with *command*. The **%** (percent sign) and **!** (exclamation point) characters in *command* are replaced with the current filename and the previous shell command, respectively. If there is no current filename, **%** is not expanded. The sequences **\%** and **\!** are replaced by **%** and **!**, respectively.

:e [*file*] Examines a new file. If a filename is not specified, the "current" file (see the **:n** and **:p** subcommands) from the list of files in the command line is re-examined. The filename is subject to the process of shell word expansions. If *file* is a **#** (number sign) character, the previously examined file is re-examined.

*i***:n** Examines the next file. If *i* is specified, examines the *i*th next file specified in the command line.

*i***:p** Examines the previous file. If *i* is specified, examines the *i*th previous file given in the command line. If this command is given during display of a file, **more** returns to the beginning of the file. If **more** is not reading from a file, the bell is rung and nothing else happens.

:f Displays the current filename and line number (same as =).

:t *tagstring*
Goes to the supplied tag string and scrolls or rewrites the screen with that line in the current position. For more information, see the **-t** flag.

:q, **:Q** Exits from **more** (same as **q** or **Q**).

m*letter* Marks the current position with the specified letter, where *letter* represents the name of one of the lowercase letters of the portable character set.

'*letter* Returns to the position that was previously marked with the specified letter, making that line the current position.

r

<Ctrl-l> Redraws the screen.

R Redraws the screen, discarding any buffered input. If the current file is nonseekable, buffered input is not discarded and the **R** subcommand is equivalent to the **r** subcommand.

The commands take effect immediately; it is not necessary to type a carriage-return. Up to the time when the command character itself is given, you can enter the line Kill character to cancel the numerical argument being formed. In addition, you can enter the Erase character to redisplay the prompt.

The terminal is set to **noecho** mode by this program so that the output can be continuous. Thus, subcommands you enter do not show on your terminal, except for the / (slash), **?** (question mark), and **!** (exclamation point) commands. In addition, the value of *i* (if it is not the default) is shown at the bottom of the screen preceded by a **:** (colon).

EXAMPLES

1. To examine each file starting with its last screenful, enter:

   more -p G file1 file2

2. To examine each file starting with line 100 in the current position (usually the third line, so line 98 would be the first line written), enter:

   more -p 100 file1 file2

3. To examine each file starting with the first line containing the string **100** in the current position, enter:

   more -p /100 file1 file2

RELATED INFORMATION

Commands: **cat**(1), **grep**(1), **man**(1), **sh**(1).

Files: **terminfo**(4).

NAME

mv - Moves files and directories

SYNOPSIS

mv [**-i** | **f**] *file1 file2*

mv [**-i** | **f**] *file1 ... directory*

mv [**-W NOG**] [**-W NOE**] [**-i** | **f**] *directory1 ... destination_directory*

The **mv** command moves files from one directory to another or renames files and directories.

FLAGS

-f Overrides the **-i** flag and any mode restrictions. If both **-f** and **-i** are specified (for example, because an alias includes one of them) whichever appears last overrides the other.

-i Prompts you with the name of the file followed by a question mark whenever a move is to supercede an existing file. If the answer begins with **y**, or the locale's equivalent of a **y**, the move continues. Any other reply prevents the move from occurring. If both **-f** and **-i** are specified (for example, because an alias includes one of them) whichever appears last overrides the other.

HP Extensions

-W NOG Specifies that the **/G** directory should be omitted when the initial directory is root. This flag is ignored when the initial directory is not **/**, **/E**, or **/E/***system*.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root. This flag is ignored when the initial directory is not root.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

If you move a file to a new directory, **mv** retains the original filename. When you move a file, all other links to the file remain intact.

In the second form, one or more files are moved to *directory* with their original filenames. In the third form, one or more directories are moved to the destination directory with their original names.

Note that **mv** does not move a file onto itself.

When you use **mv** to rename a file, the target file can be either a new filename or a new directory pathname. If moving the file would overwrite an existing file, the existing file is overwritten unless you specify the **-i** flag, in which case you will be prompted. If moving the file would overwrite an existing file that does not have write permission set and if standard input is a tty, **mv** displays the permission code of the file to be overwritten and reads one line from standard input. If the line begins with **y**, or the locale's equivalent of a **y**, the move takes place and the file is overwritten. If not, **mv** does nothing with the file.

When you use **mv** to move a directory into an existing directory, the directory and its contents are added under the existing directory.

If the access permissions of the *destination_directory* or the existing destination file (*file2*) forbid writing, the **mv** command asks the issuer of the command for permission to overwrite the file:

1. The **mv** command prints the mode (see "Access Control Lists (ACLs)," in this reference page, and the **chmod(1)** reference page) followed by the first letters of the words *yes* and *no* in the language of the current locale, to standard output, prompting for a response.

2. The **mv** command reads one line from the standard input.
3. If the response is the first letter of the word *yes* in the language of the current locale, the operation occurs. Otherwise, the operation does not occur, and the command proceeds to the next source file, if any.

If a **mv** operation fails, **mv** generally writes a diagnostic message to standard error, does nothing more with the current source file, and goes on to process any remaining source files.

If the copying or removal of a file is prematurely terminated by a signal or error, **mv** might leave a partial copy of the file at either the source or the target pathname. The **mv** program will not modify both the source and target pathnames simultaneously; therefore, program termination at any point always leaves either the source file or the target file complete.

Access Control Lists (ACLs)

If optional ACL entries are associated with the new file (*file2*), the **mv** command displays a plus sign (+) after the access mode when asking for permission to overwrite the file.

When a file or directory is renamed, all the ACL entries for the file or directory are retained.

This table describes the impact of ACLs on the permissions used when a new file or directory is created by the **mv** command. Cases not included in the table represent impossible situations.

mv Command Supports ACLs	Source Fileset Supports ACLs	Source File/Dir Has Opt. ACLs	Dest. Filesset Supports ACLs	Impact of ACLs on Permissions of New File/Dir
Y/N	Y/N	N	N	None
Y/N	Y/N	N	Y	See Note 2
Y/N	Y	Y	N	See Notes 1 and 3
N	Y	Y	Y	See Notes 1 and 2
Y	Y	Y	Y	See Note 4

Note 1: The optional ACLs for the source file or directory are not copied to the destination file or directory.

Note 2: If the destination parent directory has default ACL entries, those default ACL entries are inherited by the new file or directory (see the **acl(5)** reference page).

Note 3: The file permissions are copied to the destination file or directory and the class entry permissions in the ACL are used for the destination file or directory group permissions.

Note 4: All ACL entries are retained by the new file or directory.

For G-series RVUs, H06.19 and earlier H-series RVUs, or J06.08 and earlier J-series RVUs, the OSS Network File System (NFS) cannot access OSS objects that have OSS ACLs that contain optional ACL entries.

For J06.09 and later J-series RVUs and H06.20 and later H-series RVUs, access by the OSS Network File System (NFS) to OSS objects that have OSS ACLs that contain optional ACL entries can be allowed, depending upon the NFSPERMMAP attribute value for the fileset that contains the object, however:

- The **mv command does not move any ACLs associated with the object.**

- The permissions used when an object is created by the **mv** command depend on the value of the NFSPERMMAP attribute for the fileset on the system that contained the original file.

For more information about NFS and ACLs, see the **acl(5)** reference page.

Environment Variables

The following environment variables affect the execution of the **mv** command:

LC_MESSAGES
Determines the locale's equivalent of **y** or **n** (for yes/no queries).

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root and a recursive operation occurs in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE Omit the **/E** directory.

NOG Omit the **/G** directory.

NOG:NOE Omit both the **/G** and **/E** directories.

The effect of assigning a value to **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

EXAMPLES

1. To rename a file, enter:

   mv file1 file2

   This renames **file1** to **file2**. If a file named **file2** already exists, its old contents are replaced with those of **file1**.

2. To move a directory, enter:

   mv dir1 dir2

   This moves **dir1** to **dir2**. It moves all files and directories under **dir1** to the directory named **dir2**, if the second directory exists. Otherwise, the directory **dir1** gets renamed **dir2**.

3. To move a file to another directory and give it a new name, enter:

   mv file1 dir1/file2

   This moves **file1** to **dir1/file2**. The name **file1** is removed from the current directory, and the same file appears as **file2** in the directory **dir1**.

4. To move a file to another directory, keeping the same name, enter:

   mv file1 dir1

   This moves **file1** to **dir1/file1**.

5. To move several files into another directory, enter:

   mv file1 dir1/file2 /u/dir2

   This moves **file1** to **/u/dir2/file1** and **dir1/file2** to **/u/dir2/file2**.

6. To use **mv** with pattern-matching characters, enter:

   mv dir1/* .

   This moves all files in the directory **dir1** into the current directory (.), giving them the same names they had in **dir1**. This also empties **dir1**. Note that you must type a space between the ***** (asterisk) and the **.** (dot).

7. To move all OSS files on the local node to a remote node, enter:

   mv -W NOG -W NOE / /E/*node*

   where *node* is the Expand node name of the target remote node.

NOTES

If the source is on a different file system than the destination, **mv** must copy the source to the destination's file system and then delete the source. In this case, if the current user is not privileged enough to change ownership to the actual user, then the user ID becomes that of the current user, but the mode and times are not changed. The effect is equivalent to the following command syntax:

rm -f *destination* **&& cp -pR** *source destination* **&& rm -rf** *source*

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

For H06.23 and later H-series RVUs, J06.12 and later J-series RVUs, the H06.22 RVU with the T9194H01^AFA SPR installed, or the J06.11 RVU with the T9194J01^AEZ SPR installed, a remote user can move an SQL object that is in the OSS file system even if the file is present in the destination directory.

For H06.22 (without the T9194H01^AFA SPR installed) and earlier H-series RVUs, and J06.11 (without the T9194J01^AEZ SPR installed) and earlier J-series RVUs, if a remote user attempts to move an SQL object that is in the OSS file system and the file is present in the destination directory, the **mv** command fails with error "Guardian or User Defined Error 197". To move this type of file on these RVUs, a local user must execute the **mv** command.

File-Label and SQL Catalog Table Inconsistencies

When you issue the **mv** command to move an OSS SQL program file and an SQL program with the same name already exists in the destination directory, if the command fails due to certain failure scenarios such as a CPU failure, disk failure, or application outage, this failure can cause an inconsistency between the destination SQL program file label already existing and the corresponding SQL catalog. The program can lose its SQL properties, but have SQL catalog entries present. In this case, the following conditions occur:

- The program does not run after losing its SQL properties.
- The SQL catalog tables have stray entries for this program file. The program exists, but its just like a normal Enscribe file.
- If any DDL command is issued on a table on which the program depends, the program obtains partial SQL properties and remains invalid.

CAUTIONS

The **mv** command may overwrite existing files unless the **-i** flag is specified to prompt you first.

RELATED INFORMATION

Commands: **chmod(1)**, **cp(1)**, **ln(1)**, **rm(1)**, **rmdir(1)**.

Functions: **rename(2)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The **-W NOG** and **-W NOE** flags and the **UTILSGE** environment variable are HP extensions to the XPG4 Version 2 specification.

NAME

nawk - Manipulates text and matches patterns in files

SYNOPSIS

nawk -f *program* [**-F***character*] [*file* ...]

nawk [**-F***character*] *statement* ... [*file* ...]

FLAGS

-F*character*

Uses *character* as the field separator character (a space by default).

-f *program*

Searches for the patterns and performs the actions found in the file *program*.

DESCRIPTION

The **nawk** command provides a flexible text-manipulation language suitable for simple report generation. It is a more powerful tool for text manipulation than either **sed** or **grep**.

The **nawk** command:

- Performs convenient numeric processing.
- Allows variables within actions.
- Allows general selection of patterns.
- Allows control flow in the actions.
- Does not require any compiling of programs.

Pattern-matching and action *statements* can be specified either on the command line or in a program file. In either case, **nawk** first reads all matching and action statements, then reads a line of input and compares it to each specified pattern. If the line matches a specified pattern, **nawk** performs the specified actions and writes the result to standard output. When it has compared the current input line to all patterns, it reads the next line.

The **nawk** command reads input files in the order stated on the command line. If you specify a filename as a **-** (dash) or do not specify a filename, **nawk** reads standard input.

Enclose pattern-action statements on the command line in **’’** (single quotes) to protect them from interpretation by the shell. Consecutive pattern-action statements on the same command line must be separated by a **;** (semicolon), within one set of quotes. Consecutive pattern-action statements in a **nawk** program file must appear on separate lines.

You can assign values to variables on the **nawk** command line as follows:

variable=value

The awk command treats input lines as fields separated by spaces, tabs, or a field separator you set with the **FS** variable. (Consecutive spaces are recognized as a single separator.) Fields are referenced as **$1**, **$2**, and so on. **$0** refers to the entire line.

Pattern-Action Statements

Pattern-action statements follow the form:

pattern {*action*}

If a *pattern* lacks a corresponding *action*, **nawk** writes the entire line that contains the pattern to standard output. If an *action* lacks a corresponding *pattern*, **nawk** applies the action to every line.

Actions

An action is a sequence of statements that follow C language syntax. These statements can include:

if (*expression***)** *statement* [**else** *statement*]

while (*expression***)** *statement*

for (*expression***;***expression***;***expression***)** *statement*

for (*variable* **in** *array***)** *statement*

break

continue

{ [*statement ...*] **}**

variable=expression

print [*expression_list*] [>*file*] [| *command*]

printf *format*[**,***expression_list*] [>*file* | >>*file* | | *command*]

next

exit [*expression*]

delete *array* [*expression*]

Statements can end with a semicolon, a newline character, or the right brace enclosing the action.

Expressions can have string or numeric values and are built using the operators **+**, **-**, , /, **%**, and ^ (exponentiation), a space for string concatenation, and the C operators **++**, **--**, **+=**, **-=**, , **/=**, **%=**, **^=**, ***=**, >, >=, <, <=, ==, **!=**, and **?:**.

Because the actions process fields, input white space is not preserved in the output.

The *file* and *command* arguments can be literal names or expressions enclosed in parentheses. Identical string values in different statements refer to the same open file.

The **print** statement writes its arguments to standard output (or to a *file* if > *file* or >> *file* is present), separated by the current output field separator and terminated by the current output record separator.

The **printf** statement writes its arguments to standard output (or to a file if >*file* or >>*file* is present, or to a pipe if | *command* is present), separated by the current output field separator, and terminated by the output record separator. *file* and *command* can be literal names or parenthesized expressions. Identical string values in different statements denote the same open file. You can redirect the output into a file using the **print ... >** *file* or **printf (...) >** *file* statements. The **printf** statement formats its expression list according to the format of the **printf( )** subroutine.

Variables

Variables can be scalars, array elements (denoted **x[i])**, or fields.

Variable names can consist of uppercase and lowercase alphabetic letters, the underscore character, the digits (**0** to **9**), and extended characters. Variable names cannot begin with a digit.

Variables are initialized to the null string. Array subscripts can be any string; they do not have to be numeric. This approach allows for a form of associative memory. Enclose string constants in expressions in "" (double quotes). Multiple subscripts such as [*i*,*j*,*k*] are permitted; the constituents are concatenated and separated by the value of **SUBSEP** (see the description in the following list).

There are several variables with special meaning to **nawk. They include:**

ARGC Argument count, assignable.

ARGV Argument array, assignable; nonnull members are interpreted as filenames.

FS Input field separator (default is a space). If it is a space, then any number of spaces and tabs can separate fields.

NF The number of fields in the current input line (record), with a limit of 99.

NR The number of the current input line (record).

FNR The number of the current input line (record) in the current file.

FILENAME
The name of the current input file.

RS Input record separator (default is a newline character).

OFS The output field separator (default is a space).

ORS The output record separator (default is a newline character).

OFMT The output format for numbers (default **% .6g**).

SUBSEP
Separates multiple subscripts (default is 031).

Functions

Functions are defined at the position of a pattern-action statement, as follows:

function foo(a, b, c) { ... ; return x }

Arguments are passed by value if scalar and by reference if array name; functions can be called recursively. Arguments are local to the function; all other variables are global.

There are several built-in functions that can be used in **nawk** actions. (For information about regular expressions as referred to in this subsection, see the **grep(1)** reference page.)

length(*argument***)**
Returns the length, in characters, of *argument*, or of the entire line if there is no *argument*.

blength(*argument***)**
Returns the length, in bytes, of *argument*, or of the entire line if there is no *argument*.

close(*argument***)**
Closes the file or pipe expression. Note that you must enclose a filename in double quotes when redirecting output with the **nawk** command; otherwise, it is treated as a **nawk** variable. For example:

```
print "Hello" > "/tmp/junk"
close ("/tmp/junk")
```

exp(*number***)**
Takes the exponential of its argument.

rand Returns a random number on (0, 1).

srand(*number***)**
Sets seed for **rand**. The default is the time of day.

log(*number***)**
Takes the base e logarithm of its argument.

sqrt(*number***)**
Takes the square root of its argument.

int(*number***)**
Takes the integer part of its argument.

substr(*string***,***position***,***number***)**
Returns the substring *number* characters long of *string*, beginning at *position*.

index(*string***,***string2***)**
Returns the position in *string* where *string2* occurs, or **0** (zero) if it does not occur.

match(*string***,***regular_expression***)**
Returns the position in *string* where *regular_expression* occurs, or **0** (zero) if it does not occur. The **RSTART** and **RLENGTH** built-in variables are set to the position and length, in bytes, of the matched string.

split(*string***,***a***,**[*regular_expression*]**)**
Splits *string* into array elements *a*[**1**], *a*[**2**], . . ., *a*[*number*], and returns *number*. The separation is done with the specified regular expression or with the **FS** field separator if *regular_expression* is not given.

sub(*regular_expression***,***string2***,**[*string*]**)**
Substitutes *string2* for the first occurrence of the regular expression *regular_expression* in *string*. If *string* is not given, the entire line is used.

gsub(*regular_expression***,***string2***,**[*string*]**)**
Same as **sub** except that all occurrences of the regular expression are replaced; both **sub** and **gsub** return the number of replacements.

sprintf(*fmt***,***expression1***,**
*expression2***, ...)**" Formats the expressions according to the **printf** format string *fmt* and returns the resulting string.

system(*command***)**
Executes *command* and returns its exit status.

The **getline** function sets **$0** to the next input record from the current input file; **getline** < *file* sets **$0** to the next record from *file*. **getline x** sets variable *x* instead. Finally, *command* **|** **getline** pipes the output of *command* into **getline**. Each call of **getline** returns the next line of output

from *command*. In all cases, *getline* returns **1** for a successful input, **0** (zero) for End-of-File, and **-1** for an error.

Patterns

Patterns are arbitrary Boolean combinations of patterns and relational expressions (the **!**, |, and **&** operators and parentheses for grouping). You must start and end regular expressions with slashes. You can use regular expressions as described for the **grep** command, including the following special characters:

+ One or more occurrences of the pattern.

? Zero or one occurrence of the pattern.

| Either of two statements.

( ) Grouping of expressions.

Isolated regular expressions in a pattern apply to the entire line. Regular expressions can occur in relational expressions. Any string (constant or variable) can be used as a regular expression, except in the position of an isolated regular expression in a pattern.

If two patterns are separated by a comma, the action is performed on all lines between an occurrence of the first pattern and the next occurrence of the second.

Regular expressions can contain extended (multibyte) characters with one exception: range constructs in character class specifications using brackets cannot contain multibyte extended characters. Individual instances of extended (multibyte) characters can appear within brackets; however, extended characters are treated as separate one-byte characters.

Inclusion of extended characters in ranges is determined by the collating sequence as defined by the current locale. The wild-card characters **.**, **+**, and **?** match characters and character strings, not bytes.

There are two types of relational expressions you can use. The first type has the form:

expression match_operator pattern

where *match_operator* is either: **~** (for *contains*) or **!~** (for *does not contain*).

The second type has the form:

expression relational_operator expression

where *relational_operator* is any of the six C relational operators: <, >, <=, >=, ==, and **!=**. A conditional can be an arithmetic expression, a relational expression, or a Boolean combination of these expressions.

You can use the **BEGIN** and **END** special patterns to capture control before the first and after the last input line is read, respectively. **BEGIN** must be the first pattern; **END** must be the last. **BEGIN** and **END** do not combine with other patterns.

You have two ways to designate a character other than white space to separate fields. You can use the **-F***character* flag on the command line, or you can start *program* with the following sequence:

BEGIN { FS = *c* }

Either action changes the field separator to *c*.

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, add **0** (zero) to it. To force it to be treated as a string, append a null string ("").

EXAMPLES

1. To display the lines of a file longer than 72 bytes, enter:

   nawk 'length >72' chapter1

   This command selects each line of the file **chapter1** that is longer than **72** bytes. **nawk** then writes these lines to standard output because no action is specified.

2. To display all lines between the words **start** and **stop**, enter:

   nawk '/start/,/stop/' chapter1

3. To run a **nawk** program (**sum2.awk**) that processes a file (**chapter1**), enter:

   nawk -f sum2.awk chapter1

4. To print the first two fields of a file named filename in reverse order, enter:

   nawk '{ print $2, $1 }' filename

5. The following **nawk** program prints the first two fields of the input file in reverse order, with input fields separated by a comma and a space, then adds up the first column and prints the sum and average:

```
BEGIN   { FS = ",[ ]*|[ ]+" }
        { print $2, $1}
        { s += $1 }
END     { print "sum is", s, "average is", s/NR }
```

RELATED INFORMATION

Commands: **awk(1)**, **grep(1)**, **sed(1)**.

Functions: **printf(3)**.

Files: **locale(4)**.

NAME

newgrp - Changes the shell process to a new group

SYNOPSIS

newgrp [-] [*group*]

FLAGS

- Changes the login environment as well as the primary group identification.

DESCRIPTION

The **newgrp** command changes the primary group identification of the current shell process to *group*. Both the user's real group ID and effective group ID are changed. Any active user-generated shell is terminated.

The user remains logged in and the current directory is unchanged. Execution of the **newgrp** command always replaces the current shell with a new shell, even when the command terminates with an error (such as `unknown group`).

The program used as the new shell is chosen as follows:

1. The program file specified in the security database as the INITIAL-PROGRAM attribute of the user ID is run.
2. If the INITIAL-PROGRAM attribute is not specified, the program file specified as the most recently exported value for the **SHELL** environment variable is run.

If you specify a **-** (dash), **newgrp** also changes the login environment of the new shell process to the login environment of the new group.

Operands

group Specifies the group name of the group to which the primary group identification of the shell should be changed.

If you do not specify a *group*, **newgrp** changes the primary group identification to the default group specified for the current user in the system security database.

Only a user with appropriate privileges can change the primary group identification of the shell to a group to which that user does not belong.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **NLSPATH**, and **SHELL** environment variables.

EXIT VALUES

When the **newgrp** command successfully creates a new shell execution environment, it returns the exit value of the old shell execution environment. This exit value does not necessarily indicate that the primary group identification was successfully changed.

When the **newgrp** command fails, it returns the following values:

>0 An error occurred.

RELATED INFORMATION

Commands: **sh(1)**.

STANDARDS CONFORMANCE

This command conforms to the obsolescent alternative within the XPG4 Version 2 specification. The **-l** flag is not supported.

NAME

nice - Runs a command at a different priority

SYNOPSIS

nice [**-n** *priority*] *command* [*argument* ...]

FLAGS

-n *priority*

Specifies how the system scheduling priority of the executed utility is to be adjusted. The *priority* argument is a positive or negative decimal integer that changes the **nice** value used when determining scheduling priority. Positive *priority* values cause a lower or unchanged system scheduling priority. Negative *priority* values require appropriate privileges and cause a higher or unchanged system scheduling priority.

DESCRIPTION

The **nice** command lets you run the command specified by *command* at a different priority. The *argument* operand passes a command flag, operand, or other argument to the specified command.

For users without appropriate privileges, the value of *priority* can be in the range **1** through **19**, with **19** being the lowest priority. The default value of *priority* is **10**.

For users with appropriate privileges, the value of *priority* can be in the ranges **1** through **19** or **-1** through **-19**. The highest possible priority is **-19**.

When you have appropriate privileges, you can run commands at a higher priority by specifying *priority* as a negative number; for example, **-10**.

If you lack appropriate privileges to affect command priority, **nice** is still invoked, but the change in priority you specify does not take effect.

Refer to the **nice(2)** reference page for a description of the relationship among the **nice** value for a process, the OSS scheduling priority for the process, and the Guardian priority for the process.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **NLSPATH**, and **PATH** environment variables.

EXAMPLES

1. To run the **c89** utility as a low-priority command in the background, enter:

   ```
   nice c89 -c *.c &
   ```

   This command executes the command **c89 -c *.c** at a low priority in the background. Your terminal is free so that you can run other commands while **c89** is running. Refer to the **sh(1)** reference page for details about starting background processes with **&** (ampersand).

2. To specify a very low priority for the **c89** utility, enter:

   ```
   nice -n 15 c89 -c *.c &
   ```

   This command executes the command **c89 -c *.c** in the background at a priority that is even lower than the default priority set by **nice**.

EXIT VALUES

The **nice** command returns the following exit values:

1-125	An error occurred in the **nice** utility.
126	The specified command was found but could not be invoked.

127 The specified command could not be found.

RELATED INFORMATION

Commands: **nohup(1)**, **sh(1)**.

Functions: **nice(2)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

nl - Numbers lines in a file

SYNOPSIS

nl

[**-b** *type*]
[**-d** *delimiter1*[*delimiter2*]]
[**-f** *type*]
[**-h** *type*]
[**-i** *number*]
[**-l** *number*]
[**-n** *format*]
[**-p**]
[**-s** [*separator*]]
[**-v** *number*]
[**-w** *number*]
[*file*]

FLAGS

Use the following flags to change the default settings. If a particular flag is not specified, **nl** uses its default value.

-b *type* Specifies which logical page body section lines to number. The recognized values for *type* are:

a	Numbers all lines
t	Does not number blank lines (the default value for **-b**)
n	Does not number any lines
p*pattern*	Numbers only those lines containing the basic regular expression specified by *pattern*

-d *delimiter1*[*delimiter2*]

Uses *delimiter1* and *delimiter2* as the delimiter characters for the start of a logical page section. The two default characters are **\:** (backslash followed by a colon). You can specify either one or two characters after the **-d** flag. If you enter only one character after **-d**, the second delimiter character remains the default (**:**). If you want to use a backslash as a delimiter, you must enter two backslashes (\\).

-f *type* Specifies which logical page footer lines to number. The values recognized for *type* are the same as those for **-b** *type*. The default value for *type* in the **-f** flag is **n** (no lines numbered).

-h *type* Specifies which logical page header lines to number. The values recognized for *type* are the same as those for **-b** *type*. The default value for *type* in the **-h** flag is **n** (no lines numbered).

-i *number*

Increments logical page line numbers by *number*. The default value of *number* is 1.

-l *number*

Counts *number* blank lines as 1. For example, **-l 3** numbers only the third adjacent blank (if the appropriate **-h a**, **-b a**, or **-f a** flag is set). The default value of *number* is 1.

-n *format*
Specifies *format* as the line-numbering format. Recognized formats are:

ln Left justified, leading zeros are suppressed

rn Right justified, leading zeros are suppressed (the default value)

rz Right justified, leading zeros are kept

-p Ignores logical page delimiters (does not restart numbering).

-s[*separator*]
Separates text from line numbers with the *separator* string. The default value of *separator* is a tab character.

-v *number*
Sets the initial logical page line number to *number*. The default value of *number* is 1.

-w *number*
Specifies *number* as the number of digits in the line number. The default value of *number* is 6.

DESCRIPTION

The **nl** command reads *file* (the standard input file by default), numbers the lines in the input, and writes the numbered lines to the standard output file.

In the output, **nl** numbers the lines on the left, according to the flags you specify on the command line.

The input text must be written in logical pages. Each logical page has a header, a body, and a footer section (sections can be empty). Unless you use the **-p** flag, **nl** resets the line numbers at the start of each logical page. You can set line-numbering flags independently for the header, body, and footer sections (for example, no numbering of header and footer lines while numbering text lines only in the body).

Signal the start of logical page sections with lines in *file* that contain nothing but the following delimiter characters:

Line Contents	**Start of**
\:\:\:	Header
\:\:	Body
\:	Footer

If *file* contains none of these lines, then the entire file is considered the body.

You can name only one file on the command line. You can list the flags and the filename in any order.

Unless otherwise specified, **nl** assumes the text being read is in a single logical page body.

EXAMPLES

1. To number only the nonblank lines in the file **chap1**, enter:

   nl chap1

   This displays a numbered listing of **chap1**, numbering only the nonblank lines in the body sections by default. If **chap1** contains no **\:\:\:**, **\:\:**, or **\:** delimiters, then the entire file is considered the body.

2. To number all lines in **chap1**, enter:

   nl -b a chap1

   This command numbers all the lines in the body sections, including blank lines. This form of the **nl** command is adequate for most uses.

3. To number the lines in **chap1** and specify a different line-number format, enter:

   nl -i 10 -n rz -s :: -v 10 -w 4 chap1

   This command numbers the lines of **chap1**, starting with 10 (**-v 10**) and incrementing by 10 (**-i 10**). It displays four digits for each number (**-w 4**), right-justified including leading zeros (**-n rz**). The line numbers are separated from the text by two colons (**-s ::**). For example, if **chap1** contains the following text:

   ```
   A not-so-important note to remember:

   You can't kill time without injuring eternity.
   ```

   then the numbered listing is as follows:

   ```
   0010::A not-so-important note to remember:

   0020::You can't kill time without injuring eternity.
   ```

   Note that the blank line was not numbered. To number the blank line too, use the **-b a** flag as shown in Example 2.

RELATED INFORMATION

Commands: **cat(1)**, **pr(1)**.

Files: **locale(4)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification.

NAME

nld - Creates a non-PIC executable object file (loadfile) from one or more relinkable non-PIC object files (linkfiles)

SYNOPSIS

nld

[**-allow_duplicate_procs**]
[**-allow_missing_libs**]
[**-allow_multiple_mains**]
[**-ansistreams**]
[**-bdynamic**]
[**-bstatic**]
[**-change** *attribute-name attribute-value filename*]
[**-e** *name*]
[**-elf_check** *filename*]
[**-export** *symbol_name*]
[**-import** *symbol_name* { *filename* | *=srlname* }]
[{ **-l** | **-lib** } *filename*]
[{ **-L** | **-libvol** } *pathname*]
[**-libname** *Guardian_filename*]
[**-nostdfiles**]
[**-nostdlib**]
[**-noverbose**]
[**-o** *filename*]
[{ **-obey** | **-fl** } *filename*]
[**-r**]
[**-rename** *old-name new-name*]
[**-s**]
[**-set** *attribute-name attribute-value*]
[**-stdin**]
[**-strip** *filename*]
[**-ul**]
[**-verbose**]
[**-x**]
[**-y** *symbol*]
[*obj-filename*] ...

FLAGS

-allow_duplicate_procs

Tells **nld** to unconditionally accept multiple copies of a procedure, rather than to allow only procedures specifically marked as duplicatable by C++. The only check made is that all copies of the procedure have the same procedure attributes; it is an acceptable condition, for example, if they have different sizes. The first copy of the duplicated procedure is the one that is kept. When building an executable file, no space is allocated for the unused copies.

-allow_missing_libs

Tells **nld** not to stop processing the input file when it cannot find an archive or a shared run-time library (SRL) after searching for the name specified by a **-l** or **-lib** flag. Instead, a warning message is issued and processing continues.

-allow_multiple_mains
Tells **nld** that it is not to issue an error message if more than one procedure has the MAIN attribute. All main procedures are included in the output file. Only the first procedure having the MAIN attribute is listed as the main entry point in the file header.

-ansistreams
Specifies that C run-time library functions create files of type 180 (C binary) instead of type 101 (EDIT). The type of files created can also be set with the `ANSISTREAMS` C and C++ compiler pragma. Refer to the *C/C++ Programmer's Guide* for details.

-bdynamic
Directs **nld** to search for SRLs and archive files when resolving **-l** and **-lib** flags. This is the **nld** default action.

nld first searches for an SRL. If an SRL cannot be found, **nld** then searches for an archive file. **nld** issues an error if neither an SRL nor archive file can be found.

This flag can be disabled by the **-bstatic** flag. Multiple **-bdynamic** and **-bstatic** flags can be specified in a single **nld** invocation. Thus, it is possible to search for both SRLs and archive files for some **-l** and **-lib** flags and to search for just archive files for others.

-bstatic Directs **nld** to to search for archive files when resolving **-l** and **-lib** flags. **nld** does not search for SRLs. **nld** issues an error if an archive file cannot be found.

This flag can be disabled by the **-bdynamic** flag. Multiple **-bdynamic** and **-bstatic** flags can be specified in a single **nld** invocation. Thus, it is possible to search for both SRLs and archive files for some **-l** and **-lib** flags and to search for just archive files for others.

-change *attribute-name attribute-value filename*
Changes the value of the run-time attribute specified in *attribute-name* to the value specified in *attribute-value* in the existing relinkable or executable object file specified by *filename*. See the **-set** flag for a description of *attribute-name* and *attribute-value*.

You cannot specify other flags or object filenames with the **-change** flag. The resulting object file has the same **nld** timestamp as before. The **-set** flag can be used to set an attribute when creating a relinkable or executable object file.

-e *name* Specifies a function identifier. The specified function is the point at which to begin executing the program when the program is loaded.

This flag should be used only when linking a program that will execute without standard run-time support facilities and without linking a module such as CRTLMAIN that contains a function with the MAIN attribute. This flag should not be used for libraries.

-elf_check *filename*
Tells **nld** to check the specified executable file for corruptions that might have occurred when:

- Unresolved references in the data portion of the file that were not on a 4-byte-aligned boundary were resolved.
- The PC version of NLD changed the file header during an SRL fixup operation.

This is a read-only operation.

-export *symbol_name*
Specifies that the named symbol is exported by the SRL being created.

-import *symbol_name* { *filename* | *=srlname* }
Tells **nld** to import the named unresolved symbol from the SRL specified by either its filename or its SRL name. Searches for the symbol within the SRL are governed by the specification for the **-libvol** flag, using the rules for **-lib** flags.

This flag cannot be specified when the **-r** flag is specified.

{ **-l** | **-lib** } *filename*
Specifies the name of an SRL or archive file to use to resolve external references from the executable file being linked. The **-l** flag must be specified in lowercase type, and the space after the flag and before *filename* is optional.

A simple name is an OSS pathname without any directory components. If *filename* is not a simple name, **nld** searches the specified location. If *filename* is a simple name, **nld** adds the string **lib** to the beginning of the name. It also adds either **.a** or **.srl** to the end of the name, depending on whether **-bdynamic** or **-bstatic** is set. Files with the suffix **.a** are archive files, and files with the suffix **.srl** are SRLs. Simple filenames in the Guardian file system, **/G**, are not modified.

nld searches for files specified with simple names in **-l** and **-lib** flags in locations specified in each **-L** and **-libvol** flag, in the order specified to **nld**, before searching any of the standard library locations. If you specify the **-nostdlib** flag, **nld** does not search the standard library locations. See the **Standard Library Locations** subsection under **DESCRIPTION** for details.

{ **-L** | **-libvol** } *pathname*
Specifies a pathname to search for an SRL or archive file specified by a simple filename in an **-l** or **-lib** flag. A simple filename is an OSS pathname without any directory components. The **-L** flag must be specified in uppercase type, and the space after the flag and before *pathname* is optional. **-libvol** is a synonym for **-L**.

nld searches for files specified with simple names in **-l** and **-lib** flags in locations specified in each **-L** and **-libvol** flag, in the order specified to **nld**, before searching any of the standard library locations. If you specify the **-nostdlib** flag, **nld** does not search the standard library locations. See the **Standard Library Locations** subsection under **DESCRIPTION** for details.

nld does not verify the names of locations specified in **-L** or **-libvol** flags. If you specify the **-verbose** flag, **nld** writes to its output listing the locations where it found an SRL or archive file.

-libname *Guardian_filename*
Associates a TNS/R native user library with an executable native program file. You can associate a native user library to an executable object file but not to a relinkable object file. The **-set** and **-change** flags can also associate a native user library with an executable native program.

The value specified for *Guardian_filename* must use uppercase characters and cannot be the Guardian name of an OSS file.

-nostdfiles
Specifies that C run-time library functions do not automatically open the standard input and standard output files.

-nostdlib
Prevents **nld** from searching the standard library locations for SRLs and archive files.

-noverbose
Prevents **nld** from writing warning and informational messages to its output listing. The **-verbose** flag directs **nld** to write warning and informational messages to its output listing. The default value is **-noverbose.**

-o *filename*
Specifies the filename of the output object file.

filename can be the same as the input file name. When this is true and linking is successful, **nld** deletes the input file and then writes the output file. An error occurs if you do not have permission to delete the input file.

If you do not specify a **-o** flag, the default output filename is **a.out**.

{ **-obey** | **-fl** } *filename*
Specifies the name of an **nld** command file containing **nld** command tokens (such as filenames and command flag specifications).

filename is a C text file. Tokens can be separated by spaces, tabs, or ends of lines. Within the command file, two hyphens indicate a comment that extends to the end of the current line. Command files can be nested and there is no limit to the depth of nesting. Recursive nesting does not cause an error; **nld** does not read the same command file more than once.

-r Directs **nld** to create a relinkable object file for later use as an **nld** input file, instead of creating an executable object file. **nld** creates an executable object file by default.

-rename *old-name new-name*
Changes the symbol name of an externally visible procedure or data item. *old-name* is the name of the procedure or data item to rename. *new-name* is the new name to give the procedure or data item. See the *nld and noft Manual* for details.

-s Removes symbol information used for linking and symbolic debugging from the output executable object file. A file stripped of all symbol information cannot be symbolically debugged with the Inspect debugger or linked again by **nld**.

You can use this flag only when creating an executable object file. If you specify both the **-r** and **-s** flags, the **-s** flag is ignored. To strip all symbol information from an existing executable object file, use the **-strip** flag. You can strip only the symbol information used for symbolic debugging with the **-x** flag.

-set *attribute-name attribute-value*
Sets the value of the run-time attribute specified in *attribute-name* to the value specified in *attribute-value* when creating a relinkable or executable object file. Use the **-change** flag to change a run-time attribute in an existing relinkable or executable object file.

Each *attribute-name* has a corresponding range of accepted *attribute-value*s as follows:

- **FLOATTYPE** is one of the following:

  IEEE_FLOAT
  NEUTRAL_FLOAT
  TANDEM_FLOAT

  If **FLOATTYPE** is specified more than once, all occurrences except the final one are ignored.

- **HEAP_MAX**, **MAINSTACK_MAX**, **PFS**[**SIZE**], [**PROCESS_**]**SUBTYPE**, and **SPACE_GUARANTEE** are numbers.
- **FLOAT_LIB_OVERRULE**, **HIGHPIN**, **HIGHREQUESTER**[**S**] | **HIGHREQUESTOR**[**S**], **INSPECT**, **RUNNAMED**, and **SAVEABEND** are either **ON** or **OFF**.
- **LIBNAME** is the Guardian filename of a library file, specified as described for the **-libname** flag.
- **SYSTYPE** is either **OSS** or **GUARDIAN**.

The default values of run-time attributes are as follows:

- **FLOAT_LIB_OVERRULE** is **OFF**.
- **FLOATTYPE** is derived from the floating-point data type of the corresponding input object files.
- **HEAP_MAX** is 0.
- **HIGHPIN** is **ON**.
- **HIGHREQUESTER**[**S**] | **HIGHREQUESTOR**[**S**] is **ON**.
- **INSPECT** is **ON**.
- **LIBNAME** is none.
- **MAINSTACK_MAX** is 0.
- **PFS**[**SIZE**] is 0.
- [**PROCESS_**]**SUBTYPE** is 0.
- **RUNNAMED** is **OFF**.
- **SAVEABEND** is **OFF**.
- **SPACE_GUARANTEE** is 0.
- **SYSTYPE** is **OSS**.

Refer to the *nld and noft Manual* for a description of each run-time attribute.

-stdin Reads the contents of the standard input file at the place in the command line where the flag is specified.

-strip *filename*

Removes symbol information used for linking and symbolic debugging from an existing executable object file with the name *filename*. A file stripped of all symbol information cannot be symbolically debugged with the Inspect debugger or linked again by **nld**.

You can use this flag only on an existing executable object file. To strip all symbol information when creating an executable object file, use the **-s** flag. You can strip only the symbol information used for symbolic debugging with the **-x** flag. You cannot specify any other flags or object filenames with the **-strip** flag. The resulting file has the same **nld** timestamp as before.

-ul Creates a native user library. Specify this flag when linking modules to create a native user library.

When the **-ul** flag is specified, all functions are exported unless the **-export** option is also used.

-verbose Directs **nld** to write warning and informational messages to its output listing. The default value is **-noverbose**.

-x Removes symbol information used for symbolic debugging from the output file. This action often decreases the size of an object file. The file cannot be symbolically debugged with the Inspect debugger, but enough information remains so the object file can be used as **nld** input again. This flag is often used with the **-r** flag.

If you specify only one input filename and both the **-x** and **-r** flags, and if you specify the same filename again for the output file with the **-o** flag, you can partially strip a file in place. The resulting file has a new **nld** timestamp. The resulting object file is not necessarily smaller than the original file.

-y *symbol*

Identifies which object files define and use the symbol *symbol*. If the **-verbose** flag is specified, **nld** writes to its output listing information to identify which object files define and use the specified symbol. This information can be useful if a previous **nld** session produced error or warning messages about a symbol being either undefined or defined more than once.

obj_filename

Specifies one or more object files for the **nld** utility to link. This operand is required for all flags except the **-change** and **-strip** flags.

DESCRIPTION

The **nld** utility links one or more TNS/R native object files to produce an executable or nonexecutable native object file in a non-position-independent code (non-PIC) form, in contrast with the **ld** utility. You can also modify existing executable files using **nld**.

You can invoke **nld** directly, or:

- If you are creating a TNS/R native C or C++ program with non-position-independent (nonPIC) code, you can use the **c89** utility to invoke **nld** automatically
- If you are creating a TNS/R native COBOL program with nonPIC code, you can use the **nmcobol** utility to invoke **nld** automatically

On the command line, the filenames are the names of input object files, archives, or SRLs. Names of flags must be followed by spaces and are not case-sensitive, except for the **-l** and **-L** flags.

Standard Library Locations

The OSS version of **nld** searches for SRLs and archive files in the following standard library locations:

- The directory with the current version of the operating system image (the active **/G/system/sys***nn* directory)
- The **/lib** directory

- The **/usr/lib** directory
- The **/usr/local/lib** directory

The value of the **COMP_ROOT** environment variable is added to the beginning of **/lib**, **/usr/lib**, and **/usr/local/lib**. By default, the value of **COMP_ROOT** is null in the OSS environment.

For More Information

nld is not an interactive tool like Binder. For more information on using **nld** and details on mapping Binder commands to **nld** commands, refer to the *nld and noft Manual*.

EXAMPLES

1. The following example:

   ```
   nld objecta objectb -o objectc
   ```

   links together the input object files named **objecta** and **objectb** to create an executable file named **objectc**.

2. The following example:

   ```
   nld obj1.o obj2.o -ul -o lib
   ```

   links the object files named **obj1.o** and **obj2.o** together into a user library named **lib**.

3. The following example:

   ```
   nld obj3.o obj4.o -o prog -libname \$A.B.C
   ```

   links object files named **obj3.o** and **obj4.o** together into a program named **prog** and runs it as a user library with the Guardian name **$A.B.C**. The backslash (\) is necessary to prevent the shell from misinterpreting the dollar sign ($).

4. The following example:

   ```
   nld /usr/lib/crtlmain.o test1.o test2.o  \
     -obey /usr/lib/libc.obey -o prog
   ```

   links the C object files **test1.o** and **test2.o** to build a program named **prog**. Because the program is a C program, the **crtlmain.o** library object file is required. The **-obey** flag directs **nld** to link in all the required SRLs.

5. The following example:

   ```
   nld obj6.o obj7.o -o prog -set systype guardian
   ```

   links the object files named **obj6.o** and **obj7.o** into a program named **prog** that you intend to run as a Guardian process.

6. The following example:

   ```
   nld -change highpin off exeobj
   ```

   changes the value of the **HIGHPIN** attribute in the executable object file **exeobj** to **OFF**.

NOTES

Some NLD options are not available as command flags in the OSS environment. For example, specifying the **-NS_extent_size** or **-NS_max_extents** flag for the **nld** command on an OSS shell command line causes an error.

OSS filenames intended for use with **nld** should not begin with an equals (=) character. The equals character is reserved for use with MAP DEFINEs.

RELATED INFORMATION

Commands: **c89(1)**, **nmcobol(1)**, **eld(1)**, **enoft(1)**, **ld(1)**, **noft(1)**.

Files: **float(4)**.

STANDARDS CONFORMANCE

The **nld** command is an HP extension to the XPG4 Version 2 specification and performs functions comparable to the UNIX **ld** command.

NAME

nm - Displays the name list of a linkfile, loadfile, or other object file

SYNOPSIS

nm
[**-A**]
[**-e** | **-g** | **-u**]
[**-f**]
[**-o**]
[**-P**]
[**-t** *format*]
[**-v**]
[**-x**]
[*file*] ...

FLAGS

You can specify the following flags in any combination, but some flags will override others.

-A Writes the full pathname or library name of a named object on each line.

-e Displays only external (global) definitions and static symbol information.

-f Produces full output and writes redundant symbols that are normally suppressed. In the HP implementation, this flag has no effect, but it is retained for XPG4 compatibility.

-g Writes only external (global) symbol information.

-o Displays a symbol's value and size as an octal rather than a decimal number (same as the **-to** flag).

-P Writes information in a portable output format as specified in **Standard Output** under **DESCRIPTION**.

-t *format* Writes each numeric value in the specified format. The format depends on the single character used as the *format* argument:

- **d** The offset value is written in decimal (this is the default format).
- **o** The offset value is written in octal.
- **x** The offset value is written in hexadecimal.

-u Displays only undefined symbols.

-v Sorts external symbols by value instead of alphabetically before displaying them.

-x Displays a symbol's value and size as a hexadecimal rather than a decimal number (same as the **-tx** flag).

DESCRIPTION

The **nm** command writes the name list of each specified file to the standard output file.

The **nm** command displays symbolic information appearing in the linkfile, loadfile, object file, executable file, or library named by the *file* operand. If no symbolic information is available for a valid input file, **nm** reports that fact but does not consider it an error condition.

The default base for numeric values is decimal.

Operands

The *file* operand can be a single linkfile, loadfile, object file, an executable file, or an archive library. If you do not specify a *file* operand, the symbols in the **a.out** file are listed by default. An input file must have a format that is the same as those produced by the Binder, **ld**, **eld**, **nld**, or **ar** utility to be used for linking.

Environment Variables

This utility supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

Standard Output

If symbolic information is present in the input files, the **nm** command writes, for each file or for each member of an archive, the following information to the standard output file:

- Library member or object name, if the **-A** flag is present
- Symbol name
- Symbol type for TNS and accelerated object files:

  D Global data symbol

  d Local data symbol

  E External function

  T Global text symbol

  t Local text symbol

  U Undefined symbol

- Symbol type for native linkfiles or loadfiles:

  A External absolute

  a Local absolute

  B External zeroed data (TNS/R) or Global zeroed data (TNS/E)

  b Local zeroed data

  C Common

  D Global data symbol (TNS/R) or Global data (TNS/E)

  d Local data symbol (TNS/R) or local data (TNS/E)

  E External function

  e HP entry vector (TNS/R only)

  F Fini section or HP resident text (TNS/R) or Global function section (TNS/E)

  f Local function (TNS/E only)

  G External small initialized data (TNS/R) or Global small intialized data (TNS/E)

  g Local small initialized data

I	Init section (TNS/R only)
K	HP NonStop Kernel gateway (TNS/R only)
k	HP user gateway (TNS/R only)
N	Nil storage class (TNS/R only)
P	Procedure section
R	External read-only data (TNS/R) or Global read-only data (TNS/E)
r	Local read-only data
S	External small zeroed data (TNS/R) or Global small zeroed data (TNS/E)
s	Local small zeroed data
T	Global text symbol
t	Local text symbol
U	Undefined symbol
V	External small undefined symbol (TNS/R) or Global pointer (GP) value (TNS/E)
W	Stack unwinding section (TNS/E only)
X	Exception data (TNS/R only)
0	Text data (TNS/R only)
$	Section within an object file (TNS/E only)

- Value of the symbol

If the **-P** flag is specified, the information is displayed using the following portable format. The three versions of the portable format differ depending on whether **-t d**, **-t o**, or **-t x** is specified.

- If **-t d** is specified, the following is displayed:

  `"%s%s %s %d %d\n",` *library/object-name*, *name*, *type*, *value*

- If **-t o** is specified, the following is displayed:

  `"%s%s %s %o %o\n",` *library/object-name*, *name*, *type*, *value*

- If **-t x** is specified, the following is displayed:

  `"%s%s %s %x %x\n",` *library/object-name*, *name*, *type*, *value*

In all cases, *library/object-name* is formatted as follows:

- If the **-A** flag is not specified, *library/object-name* is an empty string.
- If the **-A** flag is specified and the corresponding *file* operand does not name a library, *library/object-name* is:

  `"%s: ",` *file*

- If the **-A** flag is not specified and if either more than one file is specified or only one file that names a library is specified, **nm** writes a line identifying the object containing the

following symbols before the lines containing those symbols.

If the *file* operand does not name a library, the format of the line is:

`"%s: ",` *file*

If the corresponding *file* operand names a library, the format of the line is:

`"%s[%s]:\n",` *file*, *object-file*

EXAMPLES

1. To list the external global symbols of the linkfile **a.out**, enter:

   ```
   nm -g a.out
   ```

2. To display symbol sizes and values as hexadecimal values and then sort the symbols by value, enter:

   ```
   nm -xv a.out
   ```

3. To show undefined symbols from a file `xyz`, enter:

   ```
   nm -u xyz
   ```

4. To use the output format with the object name on every line, enter:

   ```
   nm -PA foo.o
   ```

FILES

a.out The default input file.

EXIT VALUES

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

RELATED INFORMATION

Commands: **ar(1)**, **make(1)**, **strip(1)**.

Files: **ar(4)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions, except that the **-f** flag is ignored.

NAME

nmcobol - Compiles TNS/R native COBOL85 programs

SYNOPSIS

nmcobol

[**-c** | **-Wnolink**]
[**-g**]
[**-L** *directory*] ...
[**-l** *library*] ...
[**-O** [*optlevel*]]
[**-o** *outfile*]
[**-s**]
[**-Wansistreams**]
[**-Wcall_shared** | **-Wnon_shared** | **-Wshared**]
[**-WBdllsonly** | **-WBdynamic** | **-WBstatic**]
[**-Wcobol**="*args*"]
[**-Wcolumns**=*n*]
[**-Wcopylib**=*pathname*]
[**-Wdryrun**]
[**-Werrors**=*n*]
[**-Wheap**=*n*[**b** | **w** | **p**]
[**-Whelp** | **-Wusage**]
[**-Whighpin**={**on** | **off**}]
[**-Whighrequesters**={**on** | **off**}]
[**-W**[**no**]**include_whole**]
[**-W**[**no**]**innerlist**]
[**-W**[**no**]**inspect**]
[**-Wld**="*args*"]
[**-Wld_obey**="*pathname*"]
[**-Wlines**=*n*]
[**-W**[**no**]**list**]
[**-W**[**no**]**map**]
[**-WmoduleCatalog**="*catalog_spec*"]
[**-WmoduleGroup**[="[*group_spec*]"]]
[**-WmoduleSchema**="*schema_spec*"]
[**-WmoduleTableSet**[="[*tableset_spec*]"]]
[**-WmoduleVersion**[="[*version_spec*]"]]
[**-Wmxcmp**[="*args*"]]
[**-Wmxcmp_add**="*args*"]
[**-Wmxcmp_files**="*file*[**,**...]"]
[**-Wmxcmp_querydefault**="*attr_name*=*attr_value*[**,**...]"]
[**-Wnld**="*args*"]
[**-Wnld_obey**="*pathname*"]
[**-Wnostdlib**]
[**-Woptimize**=*n*]
[**-W**[**no**]**optional_lib**]
[**-Wr**]
[**-W**[**no**]**reexport**]
[**-Wrunnamed**]
[**-W**[**no**]**saveabend**]
[**-Wsavetemps**]
[**-Wsettog**=*n*[**,** *n*] ...]
[**-Wsql**[="*args*"]]
[**-Wsqlcomp**[="*args*"]]

[**-Wsqlmx**[="*arg*[**,**...]"]
[**-Wsqlmxadd=**"*args*"]
[**-W**[**no**]**suppress**]
[**-Wsyntax**]
[**-Wsystype=**{**guardian** | **oss**}]
[**-Wtimestamp=***value*]
[**-Wu=**"*symbol_name*"]
[**-Wv**]
[**-Wverbose**]
[**-Ww**]
[**-Wx**]
operand ...

FLAGS

-c | **-Wnolink** Performs compilation of the specified source files but suppresses the linking phase. This flag does not delete any object files that are produced.

For source files of the form *file***.cbl**, creates object files with names of the form *file***.o** on the current directory.

Use this flag when an SQL compiler is to be invoked without having to rebuild the executable file.

-g Produces symbols information for symbolic debugging in the object or executable files. This is equivalent to specifying the SYMBOLS and INSPECT directives to the NMCOBOL compiler.

-L *directory* Changes the algorithm for searching the libraries named in the **-l** flags to look in the directory named by the *directory* pathname before looking in the default directories **/lib**, **/usr/lib**, and **/usr/local/lib**. Directories named in **-L** options are searched in the order specified.

The order of specifying the **-l** and **-L** flags is significant. If the **-L** flag is specified, it should be specified before specifying any of the following flags, to affect the processing of **-l** flags related to those flags:

-WBdllsonly, **-WBdynamic**, or **-WBstatic**
-Wshared

-l *library* Specifies the filename of a library file to be used for linking. This flag can be specified more than once in a command line and is normally used following specification of **-WBdllsonly**, **-WBdynamic**, **-WBstatic**, or **-Wshared**.

In static linking mode, specifying this flag instructs the linker to search for the library named **lib***library***.a**. In dynamic linking mode, specifying this flag instructs the linker to search for the library named **lib***library***.srl** or **lib***library***.so**; if **lib***library***.srl** or **lib***library***.so** is not found, use **lib***library***.a**.

The position of **-l** *library* operands within a list of flags affects the order in which libraries are searched.

The order of specifying the **-l** and **-L** flags is also significant. If the **-L** flag is specified, it should be specified before specifying any of the following flags, to affect the processing of **-l** flags related to those flags:

-WBdllsonly, **-WBdynamic**, or **-WBstatic**
-Wshared

-O [*optlevel*] Specifies the optimization level to be used for the program file using one of the following values:

0 Specifies an OPTIMIZE 0 NMCOBOL compiler directive

1 Specifies an OPTIMIZE 1 NMCOBOL compiler directive

2 or no *optlevel* value
Specifies an OPTIMIZE 2 NMCOBOL compiler directive

If a **-O** flag is not specified, an OPTIMIZE 1 NMCOBOL compiler directive is specified.

-o *outfile* Uses the pathname *outfile* instead of the default pathname **a.out** for the executable file produced.

-s Strips symbolic and other information not required for proper execution from object and executable files. If both the **-g** and **-s** flags are used, symbolic information is kept in the object files but is stripped from the executable file. Do not specify the **-s** and **-Wsql** flags in the same **nmcobol** invocation.

-Wansistreams
Generates a program that opens text files as file code 180 files instead of file code 101 (EDIT) files when a program is compiled for the Guardian environment and includes C or C++ modules compiled with the **c89** flag **-Wsystype=guardian**. (By default Guardian C or C++ modules open text files as file code 101 files.) This flag is ignored if **-Wsystype=oss** is specified. OSS C or C++ modules can open text files only as file code 180 files.

-WBdllsonly | **-WBdynamic** | **-WBstatic**
Specifies the type of linking to be performed:

-WBdllsonly Specifies that the **ld** linker should limit searches to position-independent code (PIC) files that are dynamic-link libraries (DLLs) when resolving the file names specified for the **-l** and **-L** flags.

If a file name is qualified, **ld** searches for a DLL with that name.

If a filename is unqualified, in each search path, **ld** first searches for a DLL with the file name as specified in the **-l** or **-L** flag. If **ld** cannot find a DLL, the file name is unqualified, and the search path is not in the Guardian file system (/G), then **ld** prefixes **lib** and suffixes **.so** to the file name and searches again. If **ld** still cannot find the DLL, it searches the path again with the same prefix but with **.srl** as the suffix. For more information on search paths, see the **Finding Libraries** subsection of the **ld(1)** reference page under **DESCRIPTION**.

When a DLL cannot be found, **ld** issues an error message unless its **-allow_missing_libs** flag is specified.

The **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** flags are search control toggles. Multiple flags can be specified in a single **ld** invocation; the behavior specified remains in effect until another flag in the set is specified. Thus, you can search for both DLLs and archive files for some **-l** and **-L** flags and search for just archive files for others. The default library search control is **-WBdynamic**.

-WBdynamic Specifies that the linker utility should use dynamic linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Dynamic linking is in effect until a **-WBstatic** flag is specified. **-WBdynamic** is the default setting. Refer to the **Differences Between Dynamic and Static Linking** subsection for details.

-WBstatic Specifies that the linker utility should use static linking when searching for libraries specified in subsequent operands of the form **-l** *library*. Static linking is in effect until a **-WBdynamic** flag is specified. **-WBdynamic**, not **-WBstatic**, is the default setting. Refer to the **Differences Between Dynamic and Static Linking** subsection for details.

You cannot use these flags if you use the **-c** or **-Wnolink** flag.

-Wcall_shared | **-Wnon_shared** | **-Wshared**
Specifies the kind of linked file that should be created:

-Wcall_shared Directs the compiler to create a position-independent code (PIC) program loadfile using the **ld** linker. If you also specify the **-c** or **-Wnolink** flag, the file created is a PIC linkfile instead.

When you use this flag, you cannot use the following flags:

-Wnld or **-Wnld_obey**

-Wnon_shared
Directs the compiler to create non-PIC object files using the **nld** linker. This is the default behavior.

When you use this flag, you cannot use the following flags:

-Wld or **-Wld_obey**

-Wshared Directs the compiler to create a PIC dynamic-link library (DLL) using the **ld** linker.

When you use this flag, you cannot use the following flags:

-Wnld or **-Wnld_obey**

-Wcobol="*args***"**
Passes to the NMCOBOL compiler the directives in the argument string enclosed in quotation marks. This string follows any directives generated by other flags. If you repeat this flag, arguments are passed to the compiler in the order specified.

-Wcolumns=*n* Sets the maximum number of columns for an input file to *n*, where *n* is a number in the range 12 through 32767. If *n* is greater than 132, 132 is used. The compiler ignores text in columns beyond *n*.

-Wcopylib="*pathname***"**
Specifies *pathname* as the source file to use as the default COPY library for any COPY statement in the source program that does not specify a library. If you repeat this flag, the last file specified is the default COPY library. The default is to look for a file called COPYLIB in the current working directory.

-Wdryrun Verifies the syntax and semantics of the flags and operands that were specified and enables the **-Wv** flag. No compilation system components are run.

-Werrors=*n* Stops compiling when *n* errors have been encountered.

-Wheap=*n*[**b** | **w** | **p**]
Specifies the value that the linker should use for the HEAP_MAX attribute of the output file. *n* can be any positive value that gives a size valid for the NonStop server node on which the file is used. The size can be specified in units of:

b	Bytes; this is the default unit
w	Words
p	Pages

-Whelp | **-Wusage**
Displays information on how to run the **nmcobol** utility. No compilation system components are run.

-Whighpin={**on** | **off** }
Directs the linker to set the HIGHPIN attribute to **on** or **off** in the output object files. This attribute specifies whether the object file will run at a high PIN or a low PIN.

If the program is compiled for execution in the Guardian environment, the default setting is **-Whighpin=off**. If the program is compiled for execution in the OSS environment, the default setting is **-Whighpin=on**. This flag is set only if an executable object file is produced.

-Whighrequesters={**on** | **off** }
Directs the linker to set the HIGHREQUESTERS attribute to **on** or **off** in the output object file. This attribute specifies whether the object file supports requests from requesters running at a high PIN.

The object file must contain a C or C++ **main( )** function. If the C or C++ module was compiled with the **c89 -Wsystype=guardian** flag set, the default setting is **-Whighrequesters=off**. If the C or C++ module was compiled with the **c89 -Wsystype=oss** flag set, the default setting is **-Whighrequesters=on**. This flag is set only if an executable object file is produced.

-W[**no**]**include_whole**
Tells the **ld** linker whether to include in the loadfile all linkable archive members of all archive libraries encountered after this flag is specified.

Specifying **-Winclude_whole** begins this linking action. When **-Wnoinclude_whole** behavior is in effect, archive searches are controlled by the existence of undefined symbols. Archives are searched in the order specified on the command line. Symbols are marked as undefined by compilers or by the user through the **-Wu** flag or the **ld** linker **-u** flag. When an archive member is found that resolves an undefined symbol, the member´s symbols are merged into the external symbol table for the loadfile being created. After the merge, the undefined symbol that triggered the merge is resolved (marked as defined). The same merge might resolve other undefined symbols or result in more undefined symbols.

You can stop the linking action of **-Winclude_whole** by specifying the **-Wnoinclude_whole** flag later in the command line or an obey file.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

The default setting is **-Wnoinclude_whole**.

-W[**no**]**innerlist**
Enables [disables] the generation of instruction code mnemonics in the compiler listing immediately following each corresponding statement. This flag works only if the **-Wnosuppress** flag is specified. The default is **-Wnoinnerlist**.

-W[**no**]**inspect** Designates [does not designate] the Inspect debugger as the default debugger for the output object file. Use this flag with the **-g** flag. The default setting is **-Wnoinspect**. This flag is set only if an executable object file is produced.

-Wld="*args*" Passes to the **ld** utility the directives in the argument string enclosed in quotation marks after any other arguments are passed. If you repeat this flag, arguments are passed to the **ld** utility in the order specified.

You can only use this flag when you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when linking is suppressed.

-Wld_obey="*pathname*"
Passes *pathname* (a file of **ld** utility commands) to the **ld** utility.

You can only use this flag when you use one of the following flags:

-Wcall_shared or **-Wshared**

This flag is ignored when linking is suppressed.

-Wlines=*n* Sets the maximum number of lines on a listing page to *n*, if a listing is generated. *n* is a number in the range 10 through 32767.

-W[**no**]**list** Temporarily enables [disables] the generation of the compiler listing. This flag works only if the **-Wnosuppress** flag is specified. The default is **-Wlist**.

-W[**no**]**map** Temporarily enables [disables] the generation of identifier maps in the compiler listing. This flag works only if the **-Wnosuppress** flag is specified. The default is **-Wnomap**.

-WmoduleCatalog="*catalog_spec*"
Specifies a NonStop SQL/MX module catalog name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a catalog name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor release 2.0 and newer.

-WmoduleGroup[="*group_spec*"]
Specifies a string for a module group specification to use as a prefix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a group name. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-WmoduleSchema="*schema_spec***"**
Specifies a NonStop SQL/MX module schema name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a schema name. The string cannot contain more than 128 characters.

This flag is valid only for preprocessor release 2.0 and newer.

-WmoduleTableSet[**="**[*tableset_spec*]**"**]
Specifies a string for a tableset specification to use as the first suffix to the module name. The specified string is used only if the input file does not contain an SQL/MX module directive or its module directive does not specify a tableset name. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-WmoduleVersion[**="**[*version_spec*]**"**]
Specifies a string for a tableset specification to use as the second suffix to the externally qualified module name that is written to the module file. The string cannot contain more than 31 characters.

This flag is valid only for preprocessor release 1.8 and newer.

-Wnostdlib Suppresses the searching of the standard library directories to locate libraries for any C or C++ modules in the program. Refer to the **Standard Library Directories** subsection of the **c89(1)** reference page for details.

-Wmxcmp[**="***args***"**]
Invokes the NonStop SQL/MX compiler after the NonStop SQL/MX preprocessor is invoked.

If a value is supplied for *args*, it must be one of the following:

warn Directs the NonStop SQL/MX compiler to generate a warning rather than an error if a table does not exist at compilation time.

verbose Directs the NonStop SQL/MX compiler to display summary information as well as error and warning messages.

If the **-Wmxcmp** flag is specified more than once, only the last occurrence has an effect. If the **-Wmxcmp** flag is specified without the **-Wsqlmx** flag, and if a file specified for *operand* has a name of the form *file***.m**, that file is passed to the NonStop SQL/MX compiler.

If the **-Wmxcmp** flag is specified, you cannot use the **-Wsql** or **-Wsqlcomp** flag. The **-Wmxcmp** flag is ignored when a flag, such as **-Wsyntax**, that prevents compilation is specified.

-Wmxcmp_add="*args***"**
Specifies a string to pass to the SQL/MX compiler without validation or change. If more than one value is specified, they must be separated by commas without any white space.

-Wmxcmp_files="*file***[,...]"**
Passes MDF files specified to **mxcmp** in release 1 module management mode. Passes all specified files without the **.m** extension to **mxCompileUserModule** in release 2 module management mode.

-Wmxcmp_querydefault="*attr_name=attr_value*[**,**...]"
Specifies attribute settings (CONTROL QUERY DEFAULT settings) to pass to the NonStop SQL/MX compiler. These attribute settings override any corresponding entries in the **SYSTEM_DEFAULTS** table.

-Wnld="*args*" Passes to the **nld** utility the directives in the argument string enclosed in quotation marks. If you repeat this flag, arguments are passed to the **nld** utility in the order specified.

You cannot use this flag if you use any of the following flags:

-Wcall_shared or **-Wshared**

-Wnld_obey="*pathname*"
Passes *pathname* (a file of **nld** utility commands) to the **nld** utility.

You cannot use this flag if you use any of the following flags:

-Wcall_shared or **-Wshared**

-Woptimize=*n* Sets the optimization level to *n*. *n* is 0, 1, or 2. The NMCOBOL compiler handles 2 as if it were 1. The default is 1.

-W[**no**]**optional_lib**
Indicates whether a library specified in the command stream should be considered optional when the **ld** linker creates a loadfile.

When **-Wnooptional_lib** behavior is in effect, any library specified in a **-l** or **-lib** flag is included in the **.liblist** section of the loadfile being created. When **-Woptional_lib** behavior is in effect, a specified library can be omitted from the **.liblist** section of the loadfile being created if omitting it would not affect how symbolic references are resolved.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

If a library is specified more than once, and at least one specification occurs when **-Wnooptional_lib** is in effect, the library is included in the **.liblist** section of the loadfile being created.

The default behavior is **-Wnooptional_lib**.

-Wr Produces a nonexecutable non-PIC object file. The object file can be used as input to the **nld** utility. If this flag is not specified and a linker is invoked, the object file is executable.

You should not specify this flag for a PIC file.

-W[**no**]**reexport**
Tells the **ld** linker whether to mark any library specified in an **-l** or **-L** flag after this flag for reexport in its **libList** entry in the loadfile being created. Specifying **-Wnoreexport** leaves the library unmarked; specifying **-Wreexport** marks the library. Reexport is a run-time attribute that is used by the **rld** loader to decide what DLLs it needs to load.

-Wnoreexport is the default action.

These flags can be specified as many times as needed in the command stream. Providing either flag overrides the current setting, so that the linker actions can be controlled on a library-by-library basis.

-Wrunnamed Directs the linker to set the RUNNAMED ON attribute in the current object file. This attribute specifies that the object file runs as a named process. The default is RUNNAMED OFF.

-W[**no**]**saveabend**
Specifies that a saveabend file is [not] created if the program terminates abnormally. The default is **-Wnosaveabend**.

-Wsavetemps Saves all temporary and intermediate files created by compilation system components. Use the **-Wv** flag to display the filenames.

-Wsettog=*n*[**,** *n*] ...
Specifies a numeric toggle in the range 1 through 15 that is defined only during the NonStop SQL/MX preprocessing step. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about the NonStop *SQL/MX* **-d** *toggle* option.

All **-Wsettog** values that are supplied to **nmcobol** are automatically passed as **-d** options to the NonStop SQL/MX preprocessor. The **-d** options control the processing of ?IF directives by the preprocessor; the options do not pass ?SETTOG directives to the COBOL compiler.

This flag is ignored unless the **-Wsqlmx** flag is also specified. This flag can be specified more than once.

-Wsql[="*args*"]
Enables NonStop SQL/MP support when processing COBOL85 source files, using the arguments in the argument string enclosed in quotation marks. Refer to the NonStop *SQL/MP Programming Manual for COBOL85* for a description of the arguments that can be passed to the NonStop SQL/MP compiler. If no errors occur, **-Wsql** also runs the SQLCOMP compiler after the link step.

If you specify the **-Wsql** flag, you cannot use the **-s**, **-Wmxcmp**, or **-Wsqlmx** flag.

-Wsqlcomp[="*args*"]
Runs the NonStop SQL/MP SQLCOMP compiler after the link step, using the arguments specified in *args*.

If you specify the **-Wsqlcomp** flag, you cannot use the **-Wmxcmp** or **-Wsqlmx** flag.

-Wsqlmx[="*arg*[**,**...]"]
Invokes the NonStop SQL/MX **mxsqlco** preprocessor, using the arguments given by *args*. If an *arg* value is specified, it must be one of the following; if more than one value is specified, they must be separated by commas without any white space:

ansi_format Directs the preprocessor to assume ANSI fixed format for the source file that it reads.

double_quotes Directs the preprocessor to accept SQL string literals delimited by double quotes in addition to literals delimited by single quotes.

listing Directs the preprocessor to write its diagnostic messages to a file named *file***.eL**, where *file* is the name of the primary source file.

preprocess_only
Directs the preprocessor to suppress all steps after preprocessing.

This option is valid only for preprocessor release 2.0 and newer.

refrain_r2 Directs the SQL/MX preprocessor to use only the rules and features that apply to preprocessors prior to release 2.0. The default action is to use only the rules and features that apply to preprocessors beginning with release 2.0.

This option is valid only for preprocessor release 2.0 and newer.

If you specify the **-Wsqlmx** flag, you cannot use the **-Wsql** or **-Wsqlcomp** flag.

-Wsqlmxadd="*args***"**
Specifies a string to pass to the SQL/MX preprocessor without validation or change. If more than one value is specified, they must be separated by commas without any white space.

-W[**no**]**suppress**
Disables [enables] the generation of the compiler listing. The compiler listing is written to standard output. The default is **-Wsuppress**.

-Wsyntax Checks the syntax of the source program, but does not generate any code.

-Wsystype={**guardian** | **oss** }
Specifies the target execution environment. This flag selects definitions used during compilation, program startup code, default libraries, and system routines used during linking. The default setting is **-Wsystype=oss**. (To run files compiled for a Guardian target execution environment, you must set the file code to 700 with a `FUP ALTER` *filename*`, CODE 700` command from a TACL prompt.)

-Wtimestamp=*value*
Provides a creation timestamp for the NonStop SQL/MX preprocessor that is written to the two output files created by the preprocessor. See the HP NonStop *SQL/MX Programming Manual for C and COBOL* for details about the formats allowed for *value*. If this flag is specified more than once, only the last occurrence has an effect. Note that **nmcobol** does not check that *value* is valid; it relies on the NonStop SQL/MX preprocessor to validate this argument.

This flag is ignored unless the **-Wsqlmx** flag is also specified.

-Wu="*symbol_name***"**
Tells the **ld** linker to add *symbol_name* as an undefined symbol. This causes **ld** to search for this symbol in any archive libraries that are specified after this flag on the command line or in an obey file.

The search constraint specified by the **-Wu** flag is overridden by use of the **-Winclude_whole** flag.

-Wv Echoes to the standard error file the command line as each component of the compilation system is run.

-Wverbose Displays detailed information from the NMCOBOL compiler and linker utility.

-Ww Suppresses the printing of compiler warning messages.

-Wx Strips part of the symbol table from the output object file, but keeps information necessary for the object file to be used as input to the linker utility. This flag is typically used with the **-Wr** flag.

Do not include a space before or after the = (equal sign).

Quotation marks around string values in flags are optional but recommended to avoid errors caused by shell substitutions or deletions.

DESCRIPTION

The **nmcobol** utility is the interface to the NMCOBOL compilation system; it accepts source code conforming to the ISO COBOL85 standard. The system consists of an NMCOBOL compiler and a linker utility (**nld** or **ld**), with additional program components supporting SQL preprocessing (the SCI library for NonStop SQL/MP and **mxsqlco** for NonStop SQL/MX) and SQL compilation (**sqlcomp** for NonStop SQL/MP and **mxcmp** or **mxCompileUserModule** for NonStop SQL/MX).

nmcobol performs simple validation of the flags and operands on its command line and, depending on those items, invokes components of the language compilation system. **nmcobol** does not verify the existence of files it passes to compilation system components. It does verify that *operand* identifies valid files to pass to compilation system components. **nmcobol** and the components it runs issue messages to the standard error file.

nmcobol performs the following steps:

1. If the **-Wsqlmx** flag is specified, invokes the NonStop SQL/MX preprocessor to preprocess any COBOL source files that contain embedded NonStop SQL/MX statements to create either of the following:
   - COBOL source files with module definitions (using the release 2 module management method)
   - COBOL-only source files and module definition files (MDFs) (using the release 1 module management method)
2. Compiles any specified COBOL source files or source files produced by Step 1 into object files.
3. If the **-Wmxcmp** flag is specified, invokes the NonStop SQL/MX compiler to compile any module definitions or MDFs.
4. Links the object files together with any libraries specified on the command line. This step occurs if no flags that prevent linking (such as **-c** or **-Wnolink**) are specified and if the source files are compiled without errors.
5. If the **-Wsqlcomp** flag is specified, invokes the NonStop SQL/MP compiler to process any embedded NonStop SQL/MP statements in files created by Step 1 or specified in the command.
6. Writes an executable object file, dynamic link library (DLL), or shared run-time library (SRL) specified by the **-o** flag (if present) or to the file **a.out**.

The files specified in the *operand* list are operated on by the appropriate program components of the compilation system, depending on the command line flags and the type of file operands.

If the **-c** flag is specified, then for all pathname operands of the form *file***.cbl**, the files **$(***basename pathname***.c).o** are created as the result of successful compilation.

If **-c** is not specified, the object files created after successful compilation are combined by the link operation into a program file, dynamic-link library (DLL), or user library. When linking is performed and either the **-Wsqlmx** or **-Wmxcmp** flag is specified, the list of libraries searched

automatically includes **zclisrl**. Object files created are not deleted after successful generation of the executable program file.

The executable file is created according to OSS file creation rules, except that the file permissions are set to **S_IRWXO | S_IRWXG | S_IRWXU** and the bits specified by the **umask** value of the process are cleared.

HP Extensions

The **-W** flags are specific to HP for supporting the HP compilation environment. The argument strings within these flags are passed to the program components unchanged, along with default argument strings and argument strings corresponding to **nmcobol** command line flags meaningful to the program components. Do not specify conflicting instructions in **-W** flag argument strings or **nmcobol** command line flags. The results of conflicting instructions are undefined.

Operands

An *operand* is a pathname. At least one pathname must be specified. The following operands are supported:

file.**a**	A library of object files typically produced by the **ar** command, and passed directly to the linker utility.
file.**cbl**	A COBOL85 language source file to be compiled and optionally linked. Embedded NonStop SQL/MP information might be present.
file.**cob**	A COBOL85 language source file to be compiled and optionally linked. Embedded NonStop SQL/MP information might be present.
file.**ECBL**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**ecbl**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**ECOB**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**ecob**	A COBOL85 language source file that contains embedded NonStop SQL/MX statements to be compiled and optionally linked.
file.**m**	A module definition file (MDF) containing NonStop SQL/MX information for a corresponding COBOL source file.
file.**o**	An object file produced by a previous NMCOBOL compilation, to be passed directly to the linker utility.
file.**srl**	A shared run-time library passed directly to the **nld** utility. The shared run-time library is used by the **nld** utility to resolve external references.

Input Files

Input files are one or more of the following:

- A text file containing a COBOL85 language source program
- An object file in the format produced by the command **nmcobol -c** or the command **c89 -c**
- A library of object files in the format produced by archiving zero or more object files using the **ar** command

- A library of object files produced by the **nld** utility or the **ld** utility
- An executable file produced by the **nld** utility or the **ld** utility

When **-Wsqlmx** is specified, **nmcobol** uses the source file extension to determine whether a file requires preprocessing and the names of the source files created by the NonStop SQL/MX preprocessor. The name of the source file created is the name of the primary source file with the following transformation to the file extension:

- Each source file with the extension **.ecbl**, **.ECBL**, **.ECOB**, or **.ecob** is given to the **mxsqlco** program for preprocessing. The resulting source files have the extensions **.cbl** and **.m**, where the file named *file***.cbl** contains the COBOL source to be compiled and the file named *file***.m** contains the corresponding module definition file (MDF).
- Source files with the extensions **.cbl** or **.cob** are not given to the **mxsqlco** program; these files are assumed to contain no embedded SQL statements.

Files created by the NonStop SQL/MX preprocessor overwrite any existing files with the same name in the current working directory.

Output Files

Output files are object files, executable files, log files, NonStop SQL/MX module definition files created by the NonStop SQL/MX preprocessor, or all four. Log files have names of the form *file***.eL**. Module definition files have names of the form *file***.m**.

Standard Output

The standard output file is a text file that contains the compiler listing, if generated.

Standard Error

The standard error file is used for diagnostic and informational messages. If more than one file operand is specified, then for each such file, `"%s: \n",`*file* might be written. These messages precede the processing of each input file.

Environment Variables

The following environment variables affect the execution of **nmcobol**. The **nmcobol** utility and its program components do not support locale variables.

AS1 Determines the pathname of the **as1** assembler component of the NMCOBOL compiler. **/usr/lib/as1** is the default location for the OSS environment.

COMP_ROOT
Changes the default pathnames for the **nmcobol** compilation system components. In the OSS environment, the string specified in **COMP_ROOT** is added to the beginning of the default pathnames. If a component's environment variable is set explicitly, the **COMP_ROOT** environment variable does not modify the component's environment variable.

LD Determines the pathname of the **ld** utility invoked by **nmcobol**. **/usr/bin/ld** is the default location for the OSS environment.

MXCMP Determines the pathname of the NonStop SQL/MX release 1 compiler. **/G/system/system/mxcmp** is the default.

MXCMPUM Determines the pathname of the NonStop SQL/MX release 2 compiler. **/usr/tandem/sqlmx/bin/mxCompileUserModule** is the default.

MXSQLCO Determines the pathname of the NonStop SQL/MX preprocessor, **mxsqlco**. **/usr/tandem/sqlmx/bin/mxsqlco** is the default.

NLD Determines the pathname of the **nld** utility invoked by **nmcobol**. **/usr/bin/nld** is the default location for the OSS environment.

SQLCOMP Determines the pathname of the NonStop SQL/MP compiler invoked by **nmcobol**. By default, the program **sqlcomp** in the directory **/G/system/system** is used.

SQLCIO Determines the pathname of the object file for the NonStop SQL/MX application program interface to the NMCOBOL compiler. **/usr/tandem/lib/sqlci.o** is the default.

SQLMX_PREPROCESSOR_VERSION
Indicates the preprocessor rules and features to be used. Specifying the value 800 causes rules and features associated with release 1.8 to be used; the **mxcmp** compiler is used and only MDF files and annotated source files are produced, while rules and features associated with release 2.0 and later are ignored. Specifying a value of 1200 or larger or not specifying a value causes rules and features associated with release 2.0 and later to be used; the **mxCompileUserModule** compiler is used and annotated source files that contain embedded module definitions are produced instead of MDF files, while restrictions associated with release 1.8 or earlier are ignored.

TMPDIR Determines the pathname that overrides the default directory for temporary files created by **nmcobol** and components it invokes. By default, temporary files are stored in the **/tmp** directory. If **TMPDIR** is set to a directory that does not exist or is not writable, **nmcobol** uses the default directory as described on the **tempnam(3)** reference page.

UGEN Determines the pathname of the **ugen** assembler component of the **nmcobol** compiler. **/usr/lib/ugen** is the default.

Processes
With the exception of the COBOLFE process, which is invoked as a Guardian process, all components are invoked as OSS processes.

Standard Libraries
The following libraries are available for COBOL85 programs in the OSS environment.

-l cob Contains COBOL library and utility routines described in the *COBOL Manual for TNS and TNS/R Programs*.

-l cre Contains C run-time library routines.

-l cli Contains SQL/MX support routines.

In the absence of flags that inhibit invocation of a linker utility, such as **-c** or **-Wnolink**, **nmcobol** passes **-l cob** and **-l cre** operands to the linker utility, causing the COBOL and CRE libraries to be searched. If the **-Wsqlmx** or **-Wmxcmp** flags are present, **nmcobol** also passes a **-l cli** operand to the linker utility, causing the SQL/MX support library to be searched.

If you want the libraries to be searched in a specific order or you want linking options to be processed in a specific order, you should invoke the linker using the **ld** or **nld** command from the OSS

shell and not use **nmcobol** to do the linking.

Differences Between Static and Dynamic Linking

The **-WBdllsonly** and **-WBdynamic** operands specify dynamic linking. The **-WBstatic** operand specifies static linking.

In dynamic linking, the **nld** utility first searches for a shared run-time library (SRL) or the **ld** utility first searches for a dynamic-link library (DLL). If an SRL or DLL cannot be found, the linker utility searches for an archive file. If no archive file can be found, an error is issued.

In static linking, the linker utility searches for an archive file but does not search for an SRL or DLL. If the archive file cannot be found, an error is issued.

Dynamic and static linking are not exact opposites. Dynamic linking accepts either an SRL, DLL, or an archive file, but static linking accepts only an archive file.

Unlike **nmcobol** flags, multiple **-WBdllsonly**, **-WBdynamic**, and **-WBstatic** operands can be specified in a single **nmcobol** invocation; thus, it is possible to perform dynamic linking for some **-l** operands and static linking for others.

-WBdllsonly, **-WBdynamic** and **-WBstatic** operands specified to **nmcobol** are temporarily overridden by linking arguments specified in the **-Wld**, **-Wnld**, **-Wld_obey**, or **-Wnld_obey** flags.

The **nld** utility performs dynamic linking by default. The **ld** utility can be used instead. Refer to the **ld(1)** or **nld(1)** reference page for more information.

Using the c89 and nmcobol Utilities

OSS COBOL85 programs can contain COBOL85 modules and C modules. Compile COBOL85 modules using the **nmcobol** utility and C modules using the **c89** utility. To produce a program containing COBOL85 and C modules, first compile all the modules written in either COBOL85 or C. You can also link these modules together or with other libraries at this time, but do not SQL-compile the modules. After you have compiled all the modules of one language, compile the modules written in the other language, specifying any necessary linking or SQL-compiling options.

For example, to produce an executable object file made up of COBOL85 modules **cobol1.cbl** and **cobol2.cbl** and C modules **c1.c** and **c2.c**, you can first run the C compiler using the **c89** utility with:

```
c89 -c -o cprog.o c1.c c2.c
```

This command directs **c89** to compile the two modules but not link them. The output object file is **cprog.o**.

You can then invoke the **nmcobol** utility to compile the two COBOL85 modules and link the NMCOBOL compiler output with the previously produced C object file and the standard C library to produce the executable object **myprog** with:

```
nmcobol -o myprog cprog.o cobol1.cbl cobol2.cbl
```

Refer to the *C/C++ Programmer's Guide* and the *Open System Services Programmer's Guide* for details on writing and compiling C programs in the OSS environment.

EXAMPLES

1. The command

   ```
   nmcobol test1.cbl
   ```

   compiles the source file **test1.cbl** and links the object file into a program file **a.out**.

2. The command

   ```
   nmcobol -c test1.cbl
   ```

   compiles the source file **test1.cbl** into an object file **test1.o**.

3. The command

   ```
   nmcobol -g -o test2 x.cbl y.cbl z.cbl
   ```

   compiles source files **x.cbl**, **y.cbl**, and **z.cbl** and links the object files into a program file **test2**. Symbolic information is generated by the compiler and retained by the linker utility for debugging.

4. The command

   ```
   nmcobol -o xyz -Wsql x.o y.o z.o
   ```

   links the object files **x.o**, **y.o**, and **z.o** into a program file **xyz**. The NonStop SQL/MP compiler, **sqlcomp**, is then invoked to compile **xyz**.

5. The command

   ```
   nmcobol -Wnolink -Wsql="catalog \$abc.def" xyz
   ```

   invokes the NonStop SQL/MP compiler, **sqlcomp**, on program file **xyz** without going through the linking process. In addition to the input filename **xyz**, the catalog option is passed to the NonStop SQL/MP compiler.

6. The command

   ```
   nmcobol -o testprog -L .  -L /usr/test/lib testprog.cbl -l tdm
   ```

   compiles the COBOL85 language source program **testprog.cbl** and links the object file with the library specified in the **-l** operand. It also links the object file with a shared run-time library, if found. If a shared run-time library is not found, it uses the standard C run-time library. The **nld** utility produces a program file named **testprog**.

   By default, dynamic linking is selected. **nmcobol** searches directories for the library **tdm** specified by the **-l** flag in the following order and selects the first copy found:

   libtdm.so in the current directory (**-L .**)

   libtdm.a in the current directory (**-L .**)

   libtdm.so in **/usr/test/lib** (**-L /usr/test/lib**)

   libtdm.a in **/usr/test/lib** (**-L /usr/test/lib**)

   libtdm.so in **/lib** (by default)

   libtdm.a in **/lib** (by default)

   libtdm.so in **/nonnative/usr/lib** (by default)

   libtdm.a in **/nonnative/usr/lib** (by default)

   libtdm.so in **/usr/lib** (by default)

   libtdm.a in **/usr/lib** (by default)

   libtdm.so in **/usr/local/lib** (by default)

   libtdm.a in **/usr/local/lib** (by default)

7. The command

   ```
   nmcobol -Wsqlmx -Wmxcmp -o sqlprog.exe sqlprog.ecbl sqlprog.m
   ```

   when using the release 1 module management method, processes the single COBOL module named **sqlprog.ecbl** containing embedded NonStop SQL/MX statements as follows:

   a. The NonStop SQL/MX preprocessor is invoked to process the source file. The preprocessor creates the files **sqlprog.cbl** and **sqlprog.m**. The file **sqlprog.cbl** is the COBOL-only equivalent of **sqlprog.ecbl**; that is, the preprocessor translates all embedded NonStop SQL/MX statements to the appropriate COBOL code. The file **sqlprog.m** is the corresponding module definition file.

   b. If no errors occurred in Step a, the NMCOBOL compiler processes the file **sqlprog.cbl** to create the file **sqlprog.o**.

   c. If no errors occurred in Step b, the NonStop SQL/MX compiler is invoked to process the module definition file **sqlprog.m**.

   d. If no errors occurred in Step c, **nld** is invoked to link the file **sqlprog.o** with the standard COBOL library and produces the executable file **sqlprog.exe**.

8. The command

   ```
   nmcobol -Wmxcmp -Wmxcmp_files="test1.m,test1.o"
   ```

   SQL-compiles the MDF file **test1.m** using the NonStop SQL/MX **mxcmp** compiler and processes the file **test1.o** using the NonStop SQL/MX **mxCompileUserModule** without also linking it.

9. The command

   ```
   nmcobol -c -Wsqlmx file1.ecob file2.ecob file3.ecob
   ```

   preprocesses the three specified files and also compiles them, but does not link the resulting object files. If no errors are detected during either preprocessing or compilation, the following files are created: **file1.m**, **file1.cob**, **file2.m**, **file2.cob**, **file3.m**, **file3.cob**, **file1.o**, **file2.o**, and **file3.o**.

10. The command

    ```
    nmcobol -c -Wsqlmx file1.cbl file2.ecbl file3.ecob file4.cob
    ```

    mixes COBOL source files with and without embedded NonStop SQL/MX statements. All files are compiled but not linked. When using the release 1 module management method, if no errors are detected during either preprocessing or compilation, the following files are created: **file2.m**, **file2.cob**, **file3.m**, **file3.cob**, **file1.o**, **file2.o**, **file3.o**, **file4.o**.

DIAGNOSTICS

If **nmcobol** encounters a compilation error that prevents an object file from being created, it writes a diagnostic message to the standard error file and continues to compile other source code operands; however, it does not perform program linking and returns a nonzero exit status. If the linking is unsuccessful, **nmcobol** writes a diagnostic message to the standard error file and returns a nonzero exit status.

EXIT VALUES

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

RELATED INFORMATION

Commands: **ar(1)**, **c89(1)**, **cobol(1)**, **ecobol(1)**, **ld(1)**, **nld(1)**, **strip(1)**.

Functions: **tempnam(3)**.

STANDARDS CONFORMANCE

The **nmcobol** utility is an HP extension to the XPG4 Version 2 specification.

NAME

noft - Reads and displays information from TNS/R native object files

SYNOPSIS

noft

[**-CD** *pathname*]
[**-COMMENT** [*text*]]
[- [**DUMPADDRESS** | **DA**] *address-spec*]
[{ **-DUMPOFFSET** | **-DO** } *address-spec*]
[{ **-DUMPPROC** | **-DP** } { *proc-spec* | *proc-num* }
[*address-spec*]]
[**-DYNSTR2**]
[**-ENV**]
[{ **-FILE** | **-F** } *filename*]
[**-HELP** [**ALL** | *flag* | *help-topic*]]
[**-LAYOUT**]
[**-LIBLIST**]
[{ **-LISTATTRIBUTE** | **-LA** }]
[{ **-LISTCOMPILERS** | **-LC** }
[{ **BRIEF B** } | { **DETAIL** | **D** }]]
[{ **-LISTOPTIMIZE** | **-LO** } { **0** | **1** | **2** | ***** }]
[{ **-LISTPROC** | **-LP** } { *proc-spec* | *proc-num*|***** }
[{ **SUBPROC** | **SP** } | { **NOSUBPROC** | **NSP** }]
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]]
[{ **-LISTSOURCE** | **-LS** } *source-spec*
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]]
[{ **-LISTSRLEXPORTS** | **-LLE** }
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]]
[{ **-LISTSRLFIXUPS** | **-LLF** }
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]]
[{ **-LISTSRLINFO** | **-LLI** }
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]]
[{ **-LISTUNREFERENCED** | **-LUR** }
{ { **PROC** | **P** } | { **DATA** | **D** } | ***** }
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]]
[{ **-LISTUNRESOLVED** | **-LU** }
{ { **PROC** | **P** } | { **DATA** | **D** } | ***** }
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }
| { **EXCLUDE** | **E** }]]
[-[**SET**] **LOG** { *filename* | **OFF** | **?** }]
[**-OBEY** *filename*]
[-[**SET**] **OUT** { *filename* | **OFF** | **?** }]
[**-RESET** { *set-param* | ***** }]
[**-SET** *set-param*]
[**-SHOW** [*****]]
[{ **-XREFPROC** | **-XP** } { *proc-spec* | *proc-num* | ***** }
[**CALLEDBY** | **CALLS** | **BOTH**]
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]]
...
object-filename

FLAGS

-CD *pathname* Changes the current working directory **noft** uses to search for relative pathnames to the specified pathname.

-COMMENT [*text*]
Allows comments in **noft** command files. Comments are not displayed in output.

-[{**DUMPADDRESS** | **DA** }] *address-spec*
Displays code and data from a virtual address inside an object file's memory space. *address-spec* is the following:

start-address [*range-specifier*] [**IN** *format-specifier*]

start-address
Specifies the starting virtual addresses in hexadecimal format.

range-specifier
Specifies the amount of information to display. *range-specifier* is one of the following:

TO *end-address*
Is an ending virtual address in hexadecimal format.

FOR *number* **BYTES**
Is the number of bytes to display.

FOR *number* **WORDS**
Is the number of words to display.

FOR * Displays information to the end of the code or data section. If the **-DUMPADDRESS** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **FOR *** specifier is used.

IN *format-specifier*
Specifies how the information is to be formatted. *format-specifier* is one of the following:

ASCII | **A** Displays portions of the object file in ASCII format.

DECIMAL | **D**
Displays portions of the object file in decimal format.

HEX | **H** Displays portions of the object file in hexadecimal format.

ICODE | **IC** Displays portions of the object file in disassembled program code. This is the default format.

INNERLIST | **IN**
Displays portions of the object file in disassembled code and displays the source code interspersed with the assembler. This option can be used only for dumping text.

OCTAL | **O** Displays portions of the object file in octal format.

{**-DUMPOFFSET** | **-DO** } *address-spec*
Displays code and data from a physical offset within an object file. *address-spec* is identical to that of the **-DUMPADDRESS** flag, except that the addresses are physical offsets within the file instead of virtual addresses.

{**-DUMPPROC** | **-DP** } { *proc-spec* | *proc-num* } [*address-spec*]
Displays the contents of a procedure or part of a procedure.

proc-spec
Specifies the procedure name. Procedure names are case-sensitive in C and C++ but not in pTAL. *proc-spec* is one of the following:

proc-name
Limits the scope to the specified procedure and subprocedures. If *proc-name* is not completely specified, **noft** resolves the name and lists conforming procedure names with numbers.

proc-name.subproc-name
Limits the scope to the specified subprocedure. If *proc-name* or *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

subproc-name
Limits the scope to the specified subprocedure. If *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

proc-num
Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

address-spec
Specifies addresses in a form that is identical to that of the **-DUMPADDRESS** flag, except that the addresses are virtual offsets within the file instead of virtual addresses.

-DYNSTR2 Displays the dynamic string information pointed to by the library list and the **rld** loader search lists. In other words, **-DYNSTR2** displays the libraries specified by the **ld -lib** flag and the paths specified by the **ld -rld_l** and **-rld_first_l** flags when this loadfile was built.

-ENV Displays the current settings of **noft** environment parameters. The **-SHOW** flag also displays these parameters and additional parameters.

{**-FILE** | **-F** } *filename*
Specifies the name of the target object file you want to use with **noft**. A subsequent **-FILE** flag closes the current object file and opens the specified object file.

-HELP [**ALL** | *flag* | *help-topic*]
Displays descriptions and syntax for **noft** flags and operands.

ALL Displays a single line description of each **noft** flag. This information can be directed to an output file or log file.

flag Displays information about the specified flag, including syntax.

help-topic Specifies one of the following **noft** topics for which you want detailed information:

```
CD                 COMMENT
DUMPADDRESS        DUMPOFFSET
DUMPPROC           DYNSTR2
ENV                EXIT
FC                 FILE
HELP               HISTORY
LAYOUT             LIBLIST
LISTATTRIBUTE      LISTCOMPILERS
LISTOPTIMIZE       LISTPROC
LISTSOURCE         LISTSRLINFO
LISTSRLEXPORTS     LISTSRLFIXUPS
LISTUNRESOLVED     LISTUNREFERENCED
LOG                OBEY
OUT                RESET
SET CASE           SET FORMAT
SET HISTORYBUFFER
SET HISTORYWINDOW
SET LINES          SET LOG
SET OUT            SET SCOPEPROC
SET SCOPESOURCE    SET SORT
SHOW               XREFPROC
all                commands
command_line       object_files
procedures         shortcuts
source_files
```

-LAYOUT Lists the parts of an object file in order by file offset.

-LIBLIST Displays the library list of a loadfile.

-LISTATTRIBUTE | **-LA**
Lists process-specific information associated with an object file.

{**-LISTCOMPILERS** | **-LC** } [{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Lists version information about the native compiler components and linker utility used to create an object file. **BRIEF** provides minimal information and **DETAIL** provides detailed information.

{**-LISTOPTIMIZE** | **-LO** } { **0** | **1** | **2** | ***** }
Lists procedures based on their optimization level.

0, **1**, or **2** List procedures with an optimization level corresponding to the specified number.

***** List procedures sorted by optimization level. If the **-LISTOPTIMIZE** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

{**-LISTPROC** | **-LP** } { *proc-spec* | *proc-num* | ***** }
[{ **SUBPROC** | **SP** } | { **NOSUBPROC** | **NSP** }]
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Lists procedures and their subprocedures.

proc-spec Specifies the procedure name. Procedure names are case sensitive in C and C++ but not in pTAL. *proc-spec* is one of the following:

proc-name
Limits the scope to the specified procedure and subprocedures. If *proc-name* is not completely specified, **noft** resolves the name and lists conforming procedure names with numbers.

proc-name.subproc-name
Limits the scope to the specified subprocedure. If *proc-name* or *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

subproc-name
Limits the scope to the specified subprocedure. If *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

proc-num Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

***** Specifies all procedures in the current scope. If the **-LISTPROC** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

SUBPROC | **SP** Lists the subprocedures along with procedures. If procedure P contains subprocedure S, a **-LISTPROC P SUBPROC** flag lists S, because it is contained within P.

NOSUBPROC | **NSP**
Does not list subprocedures along with procedures. If procedure P contains subprocedure S, a **-LISTPROC P NOSUBPROC** flag lists only P and not S, even though S is contained within P.

BRIEF | **B** Provides minimal information about procedures and subprocedures. This is the default value.

DETAIL | **D** Provides detailed information about procedures and subprocedures.

{**-LISTSOURCE | -LS** } *source-spec*
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Lists the source files in the object file. If only one procedure is dumped, then the **-LISTSOURCE** flag dumps the entry for the source file containing the procedure.

source-spec Has one of the following forms:

source-name Specifies the name of the procedure for which you want information.

source-number Specifies the number of the procedure for which you want information. This number is determined by the order of procedures in the object files's procedure table.

***** Specifies that you want information for all procedures. If the **-LISTSOURCE** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

BRIEF | **B** Displays minimal information about the procedures. This is the default value.

DETAIL | **D** Displays detailed information about the procedures.

{**-LISTSRLEXPORTS** | **-LLE** } [{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Lists all symbols exported by a shared run-time library (SRL).

BRIEF | **B** Displays only exported names and numbers. This is the default value.

DETAIL | **D** Displays more information about each library, such as address and size in decimal.

{**-LISTSRLFIXUPS** | **-LLF** } [{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Lists the names of shared run-time libraries (SRLs) to which the unresolved symbols in a foreign client object file have been "fixed up".

BRIEF | **B** Displays only unresolved symbols and library names. This is the default value.

DETAIL | **D** Displays more information about each library, such as number and fixup address.

{**-LISTSRLINFO** | **-LLI** } [{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Lists the shared run-time libraries (SRLs) linked into an object file.

BRIEF | **B** Displays minimal information about SRLs. This is the default value.

DETAIL | **D** Displays detailed information about the SRLs.

{**-LISTUNREFERENCED** | **-LUR** } { { **PROC** | **P** } | { **DATA** | **D** } | ***** }
[{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Lists the undefined and unreferenced symbols in an object file. These symbols must be linked before the object file can be executed.

PROC | **P** Displays unresolved procedures.

DATA | **D** Displays unresolved data items.

***** Displays all unresolved items. If the **-LISTUNREFERENCED** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

BRIEF | **B** Displays only symbol names and numbers. This is the default format.

DETAIL | **D** Displays detailed information such as the type of each symbol. For C++ functions, **DETAIL** provides the "demangled" (original) names as well as the "mangled" iinternal equivalents.

{**-LISTUNRESOLVED** | **-LU** } { { **PROC** | **P** } | { **DATA** | **D** } | ***** }
[{ **BRIEF** | **B** } | { **DETAIL** | **D** } | **EXCLUDE** | **E**]
Lists the undefined names in an object file. These references must be resolved before the file can be executed.

PROC | **P** Displays unresolved procedures.

DATA | **D** Displays unresolved data items.

***** Displays unresolved procedures and data items. If the **-LISTUNRESOLVED** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

BRIEF | **B** Displays only symbol names and numbers. This is the default format.

DETAIL | **D** Displays detailed name information. This option is not available for position-independent code (PIC) files.

EXCLUDE | **E**
Excludes common operating system function names so that function calls that will not be linked can be identified before running the program. Millicode calls and calls to functions within shared run-time libraries (SRLs) are omitted.

This option is not available for PIC files.

-[**SET**] **LOG** { *filename* | **OFF** | **?** }
Writes a copy of the current session's input and output to a file.

filename Identifies the file to receive the copy of the command lines and output. If the file does not exist, **noft** creates it.

OFF Closes the current log file and stops all logging.

? Displays the name of the current log file. If the **-SET LOG** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

-OBEY *filename*

Directs **noft** to read command lines from the file specified in *filename*.

-[SET] OUT { *filename* | **OFF** | **?** }

Directs the input and output listings to a specified file.

filename Specifies the name of the file. The **-OUT** flag can be specified as an option to any listing or dumping flag. In this case, the specified file is opened just for one flag and then closed.

OFF Turns off redirection to a file and reverts to the original output file.

? Displays the current output filename or, if that file does not exist, indicates the standard output file. If the **-SET OUT** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

-RESET { *set-param* | ***** }

Resets one or more of the target object file parameters previously specified with the **-SET** flag to their default values.

set-param Is one of the following target object file parameters:

CASE
FORMAT
HISTORYBUFFER
HISTORYWINDOW
LINES
LOG
OUT
SCOPEPROC
SCOPESOURCE
SORT

***** Specifies that all target object file parameters are reset to their default values. If the **-RESET** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

Refer to the description of the **-SET** flag for descriptions of these object file parameters.

-SET *set-param*

Sets a **noft** target object file parameter to the specified value. The **-SET** flag can be abbreviated by combining it with letters that abbreviate *set-param* values; these abbreviated flags are shown in the following list of valid *set-param* values:

{**CASE** | **-SC** } { **ON** | **OFF** | **?** }

Specifies the case sensitivity of the **noft** environment. If the first procedure (not necessarily the MAIN procedure) in the target

object file is written in C or C++, the default value is case sensitivity. If the first procedure is written in pTAL, the default is no case sensitivity.

ON Turns on case sensitivity in the **noft** environment.

OFF Turns off case sensitivity in the **noft** environment. If turned off, some files and procedures written in C and C++ are unavailable.

? Returns the current case-sensitivity setting. If the **-SET CASE** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

{**FORMAT** | **-SF** } { { **ASCII** | **A** } | { **DECIMAL** | **D** }
| { **HEX** | **H**} | { **ICODE** | **IC** }
| { **INNERLIST** | **IN** } | { **OCTAL** | **O** } | **?** }
Specifies the format used to display the object file.

ASCII | **A** Displays portions of the object file in ASCII format.

DECIMAL | **D**
Displays portions of the object file in decimal format.

HEX | **H** Displays portions of the object file in hexadecimal format.

ICODE | **IC** Displays portions of the object file in disassembled program code.

INNERLIST | **IN**
Displays portions of the object file in disassembled program code and displays the source code interspersed with the assembler. This option can be used only for text dumps.

OCTAL | **O** Displays portions of the object file in octal format.

? Returns the current format setting. If the **-SET FORMAT** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

{**HISTORYBUFFER | -SHB** } [*number* | **?**]
Specifies the number of command lines in memory that are to be available to the **!**, **FC**, or **HISTORY** subcommands. The default value is 50 command lines.

number Specifies the number of command lines in the history buffer. If *number* is greater than the current buffer size, **noft** is unable to retrieve command lines that have already left the history buffer. If *number* is smaller than the current buffer size, the command lines lost from the

buffer are not retrievable.

? Returns the current history buffer size. If the **-SET HISTORYBUFFER** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

The **HISTORYBUFFER** setting is only meaningful when **noft** is used interactively.

{**HISTORYWINDOW** | **-SHW** } [*number* | **?**]
Specifies the number of command lines displayed with the **HISTORY** subcommand. The default value is 10 command lines.

number Specifies the number of command lines displayed with the **HISTORY** subcommand.

? Returns the current history window size. If the **-SET HISTORYWINDOW** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

The **HISTORYWINDOW** setting is only meaningful when **noft** is used interactively.

LINES [*number*]
Specifies the number of lines of output to display before pausing so that an area of output does not scroll out of the terminal or emulator display memory. A single line of output from **noft** can result in multiple lines of output on a screen, so more lines than are specified by *number* might be displayed.

The default value for *number* is 0 (zero). A zero value causes output to continue until all results are displayed.

LOG { *filename* | **OFF** | **?** }
Writes a copy of the current session's input and output to a file. The file created by the **-LOG** flag appends an existing file with the same name.

filename Identifies a file to receive the copy of the command lines and output. If the file does not exist, **noft** creates it.

OFF Closes the current log file and stops all logging.

? Displays the name of the current log file. If the **-SET LOG** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

OUT { *filename* | **OFF** | **?** }
Directs the input and output listings to a specified file.

filename Specifies the name of the file.

OFF Turns off redirection to a file and reverts to the original output file.

? Displays the name of the current log file. If the **-SET OUT** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

{ **SCOPEPROC** | **-SSP** } { *proc-spec* | *proc-num* | ***** | **?** }

Narrows the scope to a single procedure or subprocedure. This is helpful when trying to find unique items within a procedure or subprocedure and when trying to limit output to a range within a single scope.

proc-spec

Specifies the procedure name. Procedure names are case-sensitive in C and C++ but not in pTAL. *proc-spec* is one of the following:

proc-name

Limits the scope to the specified procedure and subprocedures. If *proc-name* is not completely specified, **noft** resolves the name and lists conforming procedure names with numbers.

proc-name.subproc-name

Limits the scope to the specified subprocedure. If *proc-name* or *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

subproc-name

Limits the scope to the specified subprocedure. If *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

proc-num

Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

***** Eliminates any scope limitations and allows you to view the entire object file. If the **-SET SCOPEPROC** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

? Returns the current procedure in scope. If the **-SET SCOPEPROC** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

{ **SCOPESOURCE** | **-SSS** } { *filename* | *file-number* | ***** | **?** }
Narrows the scope to a single source file, which is helpful when trying to find unique items within a source file, as well as limiting the output to a range within the designated scope.

filename Narrows the scope to a single named source file. If *filename* does not uniquely identify a source file, **noft** provides additional help.

file-number Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

***** Eliminates any scope limitations present and opens selections to the entire object file. If the **-SET SCOPESOURCE** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

? Returns the current source file in scope, if any. If the **-SET SCOPESOURCE** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

{ **SORT** | **-ST** } { { **ALPHA** | **A** }
| { **LOC** | **L** } | { **NONE** | **N** } | **?** }
Specifies the sorting order of the output.

ALPHA | **A**
Sorts **noft** output in alphabetic order.

LOC | **L** Sorts **noft** output in virtual address order.

NONE | **N**
Sorts **noft** output in numeric order determined in the relevant table. The default value is **NONE**.

? Returns the current sorting order. If the **-SET SORT** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the **?** specifier is used.

-SHOW [*****] Displays the current values of the **noft** program environment parameters and the target object file parameters. This flag is a superset of the **-ENV** flag.

If the **-SHOW** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

{**-XREFPROC** | **-XP** } { *proc-spec* | *proc-num* | ***** }
[**CALLEDBY** | **CALLS** | **BOTH**] [{ **BRIEF** | **B** } | { **DETAIL** | **D** }]
Displays a cross-reference listing of procedures.

proc-spec Specifies the procedure name. Procedure names are case-sensitive in C and C++ but not in pTAL. *proc-spec* is one of the following:

proc-name
Limits the scope to the specified procedure and subprocedures. If *proc-name* is not completely specified, **noft** resolves the name and lists conforming procedure names with numbers.

proc-name.subproc-name
Limits the scope to the specified subprocedure. If *proc-name* or *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

subproc-name
Limits the scope to the specified subprocedure. If *subproc-name* is not completely specified, **noft** resolves the name and lists conforming subprocedure names with numbers.

proc-num Specifies the procedure number. This number specifies the ordering in the object file's procedure table. Use the **-LISTPROC** flag to list each procedure number.

***** Specifies all procedures in the current scope. If the **-XREFPROC** flag is used on an OSS shell command line, the flag specification must be enclosed in quotation marks when the ***** specifier is used.

CALLEDBY Lists each procedure and the procedures that call it. A scope setting restricts the procedures that are the children of the given procedure.

CALLS Lists each procedure and the procedures that it calls. A scope setting restricts the procedures that are the parents of the given procedure.

BOTH Lists the information for both **CALLEDBY** and **CALLS**.

BRIEF | **B** Lists the called or calling procedures referenced by the indicated procedures.

DETAIL | **D** Lists the called or calling procedures referenced by the indicated procedures and the addresses where the calls are made. If the target object file is non-PIC, references to functions that might be used for calls (for example, function pointers passed as parameters) are identified by the word "reference".

The virtual addresses of the call sites are shown with the **DETAIL** option specified. A **SORT** setting affects both lists and sublists of procedures.

object-filename
Specifies the target object file.

DESCRIPTION

The **noft** utility reads and displays information from TNS/R native object files. **noft** enables you to:

- Determine the optimization level of procedures in a file.
- Display object code with corresponding source code.
- List SRL references in an object file.
- List object file attributes.

These **noft** capabilities are useful when developing and debugging programs.

The **noft** utility can be used from the command line or interactively to examine object files. To use **noft** interactively, enter the **noft** command without specifying any flags; you can then specify the flags interactively as subcommands in the manner described in the **SUBCOMMANDS** subsection of this reference page.

To use **noft** from a command file, capture the flags listed in the **FLAGS** subsection of this reference page or the subcommands listed in the **SUBCOMMANDS** subsection of this reference page. Capture one flag or subcommand per line in the command file and then specify the command file as the standard input file to the **noft** command.

In either instance, you can display the following object file components with **noft**:

- Various file headers
- Program text and data
- Symbol table and component parts
- Run-time procedure table and relocation tables

For complete information on using **noft**, refer to the *nld and noft Manual*.

SUBCOMMANDS

noft supports all flags listed in the **SYNOPSIS** section as subcommands for interactive use. Such subcommands consist of the flag without the prefixed dash (-).

The following subcommands are also supported for interactive use. These subcommands can be entered as OSS shell command line flags when prefixed by a dash (-) but are only meaningful when used interactively:

break key on keyboard
Interrupts the processing of the current subcommand as soon as possible without corrupting any **noft** internal tables. The **noft** utility resumes operation with the next subcommand line.

! [*history-number*]
Allows you to execute a previously executed subcommand line determined by the specified *history-number* value. If no *history-number* value is specified, the previous subcommand line is executed.

EXIT | **E** Stops the **noft** process.

FC [*history-number* | *-history-offset* | *text*]
Allows you to edit or repeat a previously executed subcommand line.

history-number Specifies the number of a previously entered subcommand line. The default value is the previously entered subcommand line.

-history-offset Specifies a negative offset from the current subcommand line. The flag entered before the **FC** subcommand is -1.

text Is a string of characters.

{ **HISTORY** | **H** } [*number*]
Displays previously entered subcommand lines. *number* specifies the number of subcommand lines to be displayed. The default value is 10 subcommand lines.

QUIT | **Q** Stops the **noft** utility.

EXAMPLES

1. To find the names of procedures in a source file named **sample.c**:

   noft -FILE sample.o -SET SCOPESOURCE sample.c
   "-LISTPROC *"
   or
   `noft -F sample.o -SSS sample.c "-LP *"`

2. To find all the procedures that are called by source file **sample.c**:

   noft -FILE sample.o -SET SCOPESOURCE sample.c
   "-XREFPROC * CALLEDBY"
   or
   `noft -F sample.o -SSS sample.c "-XP * CALLEDBY"`

3. To look at the optimization levels for source file **sample.c**:

   noft -FILE sample.o -SET SCOPESOURCE sample.c
   "-LISTOPTIMIZE *"
   or
   `noft -F sample.o -SSS sample.c "-LO *"`

4. To look at the optimization level for a single procedure:

   noft -FILE sample.o -SET SCOPEPROC *procedure-name*
   `"-LISTOPTIMIZE *"`
   or
   `noft -F sample.o -SSP` *procedure-name* `"-LO *"`
   or
   `noft -FILE sample.o -LISTPROC` *procedure-name* `DETAIL`
   or
   `noft -F sample.o -LP` *procedure-name* `D`

5. To look at source file numbers for **sample.o**:

   noft -FILE sample.o "-LISTSOURCE *"

6. To look at procedure numbers:

   noft -F sample.o "-LP *"

7. To see the instructions for a procedure:

   noft -FILE sample.o
   -DUMPPROC *procedure-name* `IN ICODE`
   or
   `noft -F sample.o -DP` *procedure-name* `IN IC`

8. To look at a particular 20 words referenced by one of those instructions in octal:

   noft -FILE sample.o
   -DUMPADDRESS 0x00000390 FOR 20 WORDS IN OCTAL
   or
   `noft -F sample.o -0x00000390 FOR 20 IN O`

9. To look at the first 30 bytes in an object file in ASCII:

   noft -FILE sample.o
   -DUMPOFFSET 0x0 FOR 30 BYTES IN ASCII
   or
   `noft -F sample.o -DO 0x0 FOR 30 B IN A`

10. To find out if all the SRLs referenced by this object were resolved correctly:

    noft -FILE sample.o -LISTSRLINFO DETAIL

11. To see all the data items external to the object file that need to be linked in and where they are used in alphabetic order:

    noft -FILE sample.o -SET SORT ALPHA
    -LISTUNRESOLVED DATA DETAIL
    or
    `noft -F sample.o -ST A -LU DATA D`

RELATED INFORMATION

Commands: **eld(1)**, **enoft(1)**, **ld(1)**, **nld(1)**.

STANDARDS CONFORMANCE

The **noft** command is an HP extension to the XPG4 Version 2 specification.

NAME

nohup - Runs a utility ignoring hangups

SYNOPSIS

nohup *utility* [*argument* ...]

DESCRIPTION

The **nohup** command runs *utility* with arguments supplied as *argument* values, ignoring all hangup signals. You can use this command to run programs in the background after logging out of the system. To run a **nohup** command in the background, add an & (ampersand) to the end of the command. When *utility* is invoked, the **SIGHUP** signal is ignored.

If **nohup** output is redirected to a terminal or is not redirected at all, the output is appended to the file **nohup.out**. If the file is created, the permission bits are set to **S_IRUSR** and **S_IWUSR**. If **nohup.out** is not writable in the current directory, the output is redirected to **$HOME/nohup.out**. If neither file can be created or opened for appending, *utility* is not invoked.

The **nohup** command accepts just one utility as an argument. To apply **nohup** to a pipeline or list of commands, enter the pipeline or list in a shell script file. Then run **sh** as *utility* using the following format:

nohup sh -c *file*

You can also assign the shell file execute permission and run it as the utility in the form:

nohup *file*

EXAMPLES

1. To leave a command running after logging out (**sh** only), enter:

   nohup find / -print &

   Shortly after you enter this, the following is displayed:

   ```
   670
   $ Sending output to nohup.out
   ```

   670 is the process ID of the command you just put in the background. The $ (dollar sign) is the shell prompt.

   The following is a message informing you that the output from the **find** command is in the file **nohup.out**.

   ```
   Sending output...
   ```

   You can log out after you see these messages, even if the **find** command has not finished yet.

2. To do the same, but redirect the standard output to a different file, enter:

   nohup find / -print >filenames &

   This runs the **find** command and stores its output in a file named **filenames**. Now only the process ID and prompt are displayed.

   Wait for a second or two before logging out, because the **nohup** command takes a moment to start the utility you specified. If you log out too quickly, *utility* may not run at all. Once *utility* has started, logging out will not affect it.

3. To run more than one utility, use a shell script. For example, if you include the following in a shell script:

   comm -23 fi4 fi5 | comm -23 - fi6 | join -a1 - fi7 > comm.out

   and rename it **ncomm**, you can run **nohup** for all of the utilities in **ncomm** by entering:

   nohup sh ncomm

   If you assign **ncomm** Execute permission, you can obtain the same results by issuing the command:

   nohup ncomm

   To run this command in the background, enter:

   nohup ncomm &

FILES

nohup.out Standard output and standard error file for **nohup**..

NOTES

The term *utility*, rather than the term *command*, is used to describe the argument to **nohup** because shell compound commands, pipelines, special built-in programs, and so on, cannot be used directly. In addition, *utility* includes user application programs and shell scripts, not just the standard utilities.

EXIT VALUES

The **nohup** command returns the following exit values:

126 The specified utility was found, but could not be invoked.

127 The error occurred in the **nohup** utility or the specified utility could not be found.

The exit status is otherwise that of the utility.

RELATED INFORMATION

Commands: **sh**(1).

Functions: **sigaction**(2).

NAME

od - Writes the contents of a file to the standard output file

SYNOPSIS

od

[**-v**]
[**-A** *address_base*]
[**-j** *skip*]
[**-N** *count*]
[**-t** *type_string*...]
[*file* ...]

od

[**-abBcCdDefFhHiIlLoOpPvxX**]
[**-A** *address_base*]
[**-j** *skip*]
[**-N** *count*]
[**-t** *type_string*...]
[**-s**][*number*]
[**-w**][*number*]
[*file* ...]
[**+**]
[*offset*]
[**.**][**b** | **B**]
[*label*][**.**][**b** | **B**]

FLAGS

Format characters are as follows:

-a Displays bytes as characters and displays them with their ASCII names. If the **p** character is also given, bytes with even parity are underlined. The **P** character causes bytes with odd parity to be underlined. Otherwise, parity is ignored.

-A *address_base*
Specifies the input offset base with the single-character *address_base* argument. The characters **d**, **o**, and **x** specify that the offset base be written in decimal, octal, or hexadecimal, respectively. The character **n** specifies that the offset not be written at all.

-b Displays bytes as octal values. This flag is equivalent to **-t o1**.

-B Displays short words as octal values. This flag is equivalent to **-o** and **-t o2**.

-c Displays bytes as characters using the current setting of the **LC_CTYPE** variable. This flag is equivalent to **-t -c**. The following nongraphic characters appear as C escape sequences:

\0 Null

\a Alarm (or bell)

\b Backspace

\f Formfeed

\n Newline character

\r Enter

\t Tab

\v Vertical tab

Other nongraphic characters appear as 3-digit octal numbers. Bytes with the parity bit set are displayed in octal.

-C Displays any extended characters as standard printable ASCII characters using the appropriate character escape string.

-d Displays short words as unsigned decimal values. This flag is equivalent to **-t u2**.

-D Displays long words as unsigned decimal values.

-e Displays long words as double-precision, floating-point. (Same as **-F**.)

-f Displays long words as single-precision, floating-point.

-F Displays long words as double-precision, floating-point.

-h Displays short words as unsigned hexadecimal values.

-H Displays long words as unsigned hexadecimal values.

-i Displays short words as signed decimal values.

-I, **-l**, **-L** Display long words as signed decimal values. (The three flags are identical.)

-j *skip* Jumps over (reading or seeking) *skip* bytes from the beginning of the concatenated input files. If the input is not at least *skip* bytes long, **od** writes a diagnostic message to standard error and returns a nonzero exit value.

The *skip* argument is interpreted as a decimal number by default. If you include a leading offset of **0x** or **0X**, *skip* is interpreted as a hexadecimal number. A leading offset of **0** (zero) causes *skip* to be interpreted as an octal number.

If you append the character **b**, **k**, or **m** to *skip*, the number is interpreted as a multiple of 512, 1024, or 1,048,576 bytes, respectively.

-N *count* Causes **od** to format no more than *count* bytes of input.

The *count* argument is interpreted as a decimal number by default. If you include a leading offset of **0x** or **0X**, *count* is interpreted as a hexadecimal number. A leading offset of **0** (zero) causes *count* to be interpreted as an octal number. If there are not *count* bytes of input available (after successfully skipping bytes as specified by **-j**), **od** formats the available input.

-o Displays short words as octal values. This flag is equivalent to **-t o2**. This is the default.

-O Displays long words as unsigned octal values.

-p Indicates even parity on **-a** conversion.

-P Indicates odd parity on **-a** conversion.

-s[*number*]
Looks for strings of ASCII graphic characters, terminated with a null byte. *number* specifies the minimum length string to be recognized. By default, the minimum length is 3 characters. Allowable characters are those between blank (040) and tilde (0176), as well as backspace, tab, linefeed, formfeed, and carriage-return (010 through 015, except 013).

-t *type_string*...
Specifies one or more output types. The *type_string* argument is a string that specifies the types to be used when writing the input data. *type_string* consists of the following type specification characters:

a Named character

c Character

d Signed decimal

f Floating point

o Octal

u Unsigned decimal

x Hexadecimal

The type specification characters **d**, **f**, **o**, **u**, and **x** can be followed by an optional unsigned decimal integer that specifies the number of bytes to be transformed by each instance of the output type.

The type specification character **f** can be followed by one of the following optional characters, which indicate the type of the item to which the conversion should be applied.

F **float**

D **double**

L **long double**

The type specification characters **d**, **o**, **u**, and **x** can be followed by one of the following optional characters, which indicate the type of the item to which the conversion should be applied:

C **char**

I **int**

L **long**

S **short**

You can concatenate multiple types within the same *type_string* argument and you can specify multiple **-t** arguments. **od** writes the output lines for each type specified in the order in which you entered the type specification characters.

-v Shows all data. By default, display lines that are identical to the previous line are not output (except for the byte offsets), but are indicated with an ***** (asterisk) in column 1.

-w[*number*]
Specifies the number of input bytes to be interpreted and displayed on each output line. If **-w** is not specified, 16 bytes are read for each display line. If *number* is not specified, it defaults to 32.

-x Displays short words as unsigned hexadecimal values. (Same as **-h** and **-t x2**.)

-X Displays long words as unsigned hexadecimal values. (Same as **-H**.)

An uppercase format character implies the long or double-precision form of the object.

You can specify multiple types by using multiple **-bcdostx** flags. Output lines are written for each type specified in the order in which the types are specified.

DESCRIPTION

The **od** command reads *file* (standard input by default), and writes the information stored in *file* to standard output using the format specified by the first flag. If you do not specify the first flag, the **-o** flag is the default.

When **od** reads standard input, the *offset* and *label* parameters must be preceded by a + (plus sign).

The *offset* argument specifies the point in the file at which the output starts. The *offset* argument is interpreted as octal bytes. If a . (dot) is added to *offset*, it is interpreted in decimal. If *offset* begins with **x** or **0x**, it is interpreted in hexadecimal. If **b** (**B**) is appended, the offset is interpreted as a block count, where a block is 512 (1024) bytes.

The output continues until the end of the file. If the file argument is omitted and none of the **-A**, **-j**, **-N**, or **-t** flags is specified, the *offset* argument must be preceded by a **+** (plus sign) character.

If the first character of *file* is a **+** (plus sign) or the first character of the first file argument is numeric, no more than two arguments are given, and none of the **-A**, **-j**, **-N** or **-t** flags is specified, the argument is assumed to be an offset.

The *label* argument is interpreted as a pseudoaddress for the first byte displayed. It is shown in parentheses following the file offset. It is intended to be used with core images to indicate the real memory address. The syntax for *label* is identical to that for *offset*.

EXAMPLES

1. To display a file in octal word format, a page at a time, enter:

   od a.out | more

2. To translate a file into several formats at once, enter:

   od -cx a.out >a.xcd

   This writes **a.out** in hexadecimal format (the **-x** flag) into the file **a.xcd**, giving also the ASCII character equivalent, if any, of each byte (the **-c** flag).

3. To start in the middle of a file, enter:

   od -bcx a.out +100.

   This displays **a.out** in octal-byte, character, and hexadecimal formats, starting from the 100th byte. The . (dot) after the offset makes it a decimal number. Without the **.** (dot), the dump starts from the 64th (100 octal) byte.

NOTES

Compatibility Note

The **-i** flag displays short words as signed decimal values. The **-i** flag used to be **-s** in System V.

RELATED INFORMATION

Files: **locale**(4).

NAME

osh - Runs a process in the OSS environment from the Guardian environment

SYNOPSIS

osh [*option ...*] [*redirection ...*] [*operands*]

FLAGS

Operands used with the **osh** command must follow **osh** option specifications. Redirection can appear anywhere on the command line.

Options

All filename and pathname arguments used with **osh** options must be specified using OSS pathname syntax. In the current release, filenames and pathnames within the **/E** directory are not supported by the OSS file system. However, **osh** supports specification of the **/E** directory for filenames and pathnames in options such as **-prog** *pathname*. These specifications allow the user to identify Guardian environment objects on other HP NonStop server nodes.

The specification of **osh** options ends when **osh** finds one of the following:

- The end of the input line
- Any operand; that is, any specification that does not start with a minus (-) or plus (+) and is not a redirection specification
- The -- option
- The **-p** option

The **osh** options cannot be grouped after a single minus (-) or plus (+). That is, **-debuginspect on** is not a valid option specification.

The **osh** options can be specified in any order; **osh** processes options from left to right. When you specify an **osh** option more than once, the rightmost specification on the command line determines the value used.

-? | -help Displays usage information for the **osh** command. If this option is specified, all remaining **osh** options and operands on the command line are ignored.

-c *command* Submits *command* to the OSS child process as a single argument, prefixed by **-c**.

If the *command* string contains more than one item, the entire string must be enclosed in quotation marks ("). The *command* string can be any valid OSS shell command, OSS utility name, or script file pathname and can include parameters to be processed by the named command, utility, or script.

-cpu *n* Specifies the processor (0-15) in which the child process is to run. The default processor is the processor in which **osh** is running.

-debug Starts a Guardian environment debugging tool at the first executable instruction of the child process. The default action is to start the child process without a debugging tool active.

Refer to the appropriate debugger manual for additional information.

-defmode on | -defmode off

Specifies the Guardian DEFINE mode for the child process.

If you use **-defmode on**, all Guardian DEFINE values of the process executing **osh** are inherited by the child process.

If you use **-defmode off**, only Guardian =_DEFAULTS DEFINE values are inherited by the child process.

The default mode is the DEFINE mode in effect for **osh**.

Refer to the *TACL Reference Manual* for additional information about Guardian DEFINEs.

-extswap *pathname*

Specifies a Guardian swap file or swap volume for the extended data segment of the child process. This option is no longer used but is retained for compatability with older versions of the command. The *pathname* must be specified in OSS pathname syntax.

The default action is to use the swap volume specified in the =_DEFAULTS DEFINE values for the child process. If that volume is not available, the operating system chooses a swap volume.

Refer to the *Guardian Programmer's Guide* for additional information about swap files.

-gpri *n*

Assigns the initial Guardian execution priority *n* (1-199) to the child process. The value 1 is the lowest priority; the value 199 is the highest priority.

The default priority is the priority used for **osh**.

-highpin on | -highpin off

Specifies whether the child process can run with a Guardian process identification number (PIN) greater than 255.

Specifying **-highpin on** means that the child process can run with a PIN greater than 255. Specifying **-highpin off** means the child process must run with a PIN between 0 and 254.

The default value is **-highpin on**, unless an off value for the Guardian process attribute is inherited from **osh**.

Refer to the *TACL Reference Manual* for additional information about high and low PINs.

-inspect on | -inspect off | -inspect saveabend

Indicates the Inspect mode to be used for the child process.

Specifying **-inspect on** causes the child process to enter the currently defined symbolic debugger when the **-debug** option is specified or if a debug event occurs.

Specifying **-inspect off** causes the child process to enter the default debugger when the **-debug** option is specified or if a debug event occurs.

Specifying **-inspect saveabend** causes creation of a saveabend file (process snapshot file or core file) when the process terminates abnormally.

The default Inspect mode is the mode specified in the program file of the child process.

Refer to the appropriate debugger manual for additional information about debug events and debugger use.

-jobid 0 | -jobid -1

Controls the job ID to be assigned to the child process.

Specifying **-jobid 0** prevents the child process from running as part of a batch job (the **osh** process cannot function as a batch job ancestor, so no other value than **-jobid -1** is supported).

Specifying **-jobid -1** causes the child process to inherit its job ID (if any) from **osh**.

The default value is **-jobid -1**.

-lib *pathname* Specifies the OSS pathname of a user library file in the Guardian file system to be used by the child process; the program file for the child process is modified to point to the specified library.

Use of this option requires write access to the program file for the child process. The library file must be in the Guardian file system on the same HP node as the program file for the child process.

This option is needed only when a child process requires a user library and an alternate is needed. The default action is to run the child process with no modification to its user library usage.

+lib Specifies that the child process is to run without any user library file; the program file for the child process is modified so that it does not point to a library.

Use of this option requires write access to the program file for the child process.

This option is needed only when a child process has used a user library that is no longer needed. The default action is to run the child process with no modification to its user library usage.

-ls Specifies that the child process is a login shell. The last pathname in the *argv***[0]** value passed to the child process is prefixed by a minus (**-**), indicating that the **/etc/profile** and **$HOME/.profile** files should be run.

Refer to the rules later in this reference page under **Process Environment** for more information on the source of the values used for the initial login and current working directories.

If the **osh** command is used without specifying either the **-p** or **-prog** option, then the **-ls** option is the default option and causes execution of the **/bin/sh** file as a login shell.

+ls Specifies that the child process is not a login shell (the value passed to the child process in *argv***[0]** is not changed).

The **osh** command sets the current working directory for the child process to the directory specified by the current value of the PWD PARAM. If the PWD PARAM is not specified, **osh** command sets the current working directory for the child process to the current Guardian file system default volume and subvolume (as indicated in the =_DEFAULTS DEFINE). Refer to the rules later in this reference page under **Process Environment** for more information on the source of the value used for this directory.

If the **osh** command is used with either the **-p** or **-prog** option, then the **+ls** option is the default option.

-name {**/G/** | **/E/***nodename***/G/**}*processname*
Starts the child process as a named process using the specified name; *processname* must conform to Guardian process name rules for length.

If **/G/** or **/E/***nodename***/G/** is omitted, the full filename of the process is resolved using Guardian environment rules. Refer to the RUN command description in the *TACL Reference Manual* for the rules affecting process name length and resolution.

The rules for mapping between Guardian filenames and OSS pathnames mean that *processname* cannot begin with the dollar sign (**$**) used in the Guardian environment. Similarly, *nodename* cannot begin with the backslash (\) character used in

the Guardian environment.

If only **-name /G** is specified, the operating system creates a unique four-character process name.

The default action is to use the process name attribute for the program file of the child process.

+name Starts the child process as an unnamed process. This specification is ignored if the Guardian RUNNAMED process attribute is set in the program file for the child process.

If you specify neither the **-name** nor the **+name** option, the default behavior is **+name**.

-nowait Exits without waiting for the child process to terminate. The default action is to wait for the child process to terminate.

-osstty Starts a copy of the OSSTTY process and redirects any OSS standard input, standard output, or standard error file through the process, to or from Guardian file system objects, as specified by the redirection operators or the RUN options in the same command. OSSTTY is started only if at least one of the objects specified would otherwise not be accessible to the OSS program started by this command. If **-osstty** is the only option specified and no redirection is requested in the command, OSSTTY does not start.

OSSTTY starts as a single-use process without a backup copy, on the same processor as the **osh** process; **osh** ignores any server copies of OSSTTY that might already be running. The instance of OSSTTY is given a 4-character process name generated by the system and terminates as soon as the OSS application launched by this command terminates.

All OSSTTY error messages are suppressed; the **osh** process issues its own error messages instead. Output sent to standard files is not prefixed with information to identify its source (the **-osstty** flag does not use the OSSTTY **-prefixpid** option).

Any OSS process started with a valid user ID can read or write data to a standard file through the OSSTTY copy started by this command.

The standard error file cannot be redirected to a Guardian EDIT file.

When the **-p** or **-c** flag specifies an application program that cannot inherit OSS standard files and you use the **-osstty** flag, standard file data is exchanged with one or more of the following, as required:

standard input data
/G/*abcd***/#stdin**

standard output data
/G/*abcd***/#stdout**

standard error data
/G/*abcd***/#stderr**

where *abcd* represents the 4-character process name generated for the copy of OSSTTY.

The use of redirection specifications affects this determination; see **Redirection** in this reference page.

If the OSS process uses a Guardian process for its standard error file, OSSTTY uses the process specified for the HOMETERM of that process for its standard

error file; otherwise, OSSTTY uses the HOMETERM associated with the **osh** process.

-p *pathname* Runs the program specified by *pathname* as a child process. This option is an alternate syntax for the **-prog** option followed by the -- option (described later in this reference page). For example, **osh -p** *pathname operands* is equivalent to specifying **osh -prog** *pathname* **--** *operands*.

No other **osh** option can be specified after this option on the **osh** command line.

-phd Specifies that the child process uses the initial login directory of the effective (PAID) user instead of the initial login directory of the login user (the default action). For more information, see "Process Environment." This flag is available on systems running G06.31 and later G-series RVUs and H06.11 and later H-series RVUs.

-pfs *n* Specifies the size of the operating system process file segment (PFS) for the child process in 2048-byte virtual memory pages (64-512). The default size is determined from the program file executed as the child process.

Refer to the *TACL Reference Manual* for additional information about PFS pages.

-pmsg on | -pmsg off

Specifies whether the **osh** command displays the child process OSS process ID (pid) and exit status information on its Guardian standard output file.

Specifying **-pmsg on** means the information is displayed. Specifying **-pmsg off** means the information is not displayed.

The default option is **-pmsg off**.

-prog *pathname*

Runs the specified program file as the child process. If this option is specified:

- The **osh** command uses the Guardian environment PATH PARAM to resolve (expand) the *pathname* argument. If the Guardian environment PATH PARAM is not defined, then **osh** searches the **/bin** directory for a file identified by the *pathname* value. If execution fails because the file is not in executable format and the specified filename is not a directory, the file is assumed to be a shell script and an OSS shell process is spawned to execute it. If the file is not found, an error diagnostic is printed and **osh** terminates.
- The **osh** command uses the **+ls** option instead of the **-ls** option as its default option.

If you do not specify the **-prog** *pathname* option, the default action is to run the OSS **/bin/sh** file as the child process and as a login shell (refer to the **-ls** option).

-swap *pathname*

Specifies the name of a Guardian swap file or swap volume for the data segment of the child process. This option is no longer used and is provided for compatibility with previous versions of the command. If specified, the name must be

- in OSS pathname syntax
- valid for an existing file

but is otherwise ignored. The operating system chooses a swap volume.

-term *pathname*

Specifies the filename of a Guardian terminal device to be used as the home terminal of the child process. The name must be specified in OSS pathname syntax.

The default action is to use the home terminal of the **osh** process.

Refer to the *TACL Reference Manual* for additional information about Guardian terminal device names.

-- Specifies that there are no more options on the **osh** command line. Any information following this option is either processed as redirection specifications or passed to the child process as operands.

Redirection

The **osh** command initially routes the input, output, and error file information of the child process to its own standard input, output, and error files. A change in routing is called redirection.

Files that can be redirected include OSS regular files, odd-unstructured Guardian Enscribe format 1 files (file code 180), and Guardian terminal simulation processes such as OSSTTY or Telserv windows. Other Guardian processes cannot be specified as the source or destination of child process redirection; this restriction applies to such processes as spoolers and home terminal emulators (for example, $VHS cannot receive standard error output unless OSSTTY is used as an intermediate process). Guardian EDIT files (file code 101) can be used for redirection only when the type of redirection allows them to be opened for read-only access.

You can redirect information routed to standard files by embedding either or both of the following in the **osh** command line:

- Standard Guardian command RUN options (specifying an IN, OUT, or TERM parameter with a Guardian file name between slashes in the command line)
- POSIX.2 standard file redirection specifications

When a RUN option is used by itself, the Guardian file-system object it specifies is used for the corresponding standard file of both the **osh** process and the OSS application that it launches. The following values are valid:

IN *filename1* specifies where to find standard input file data. You can specify a Guardian terminal simulation process (such as OSSTTY or a Telserv window), an odd-unstructured Enscribe format 1 (file code 180) file, or an EDIT file (file code 101) using a Guardian filename.

OUT *filename2*

specifies where to write standard output file data. You can specify a Guardian terminal simulation process (such as OSSTTY or a Telserv window) or an odd-unstructured Enscribe format 1 (file code 180) file using a Guardian filename.

TERM *filename3*

specifies where to write standard error file data. You can specify a Guardian terminal simulation process (such as OSSTTY or a Telserv window) using a Guardian filename. If you omit this option, the HOMETERM for your terminal session is used.

When a POSIX.2 redirection specification is used by itself, it applies to the standard file of the OSS application being launched by the **osh** command.

When both a RUN option and a POSIX.2 redirection specification for the same standard file are used, the **osh** process determines from the file in the POSIX.2 redirection specification whether a copy of OSSTTY should be started automatically to redirect the standard file data appropriately.

osh POSIX.2 redirection specifications are a subset of those defined for a POSIX-conformant shell. The following rules apply:

- All pathnames specified must use OSS pathname syntax. Relative pathnames are resolved (expanded) using the initial working directory specified in the OSS environment variable **PWD** passed to the child process.
- Any files that are the target of POSIX.2 redirection must be capable of being opened by the OSS file system. See the **open(2)** reference page either online or in the *Open System Services System Calls Reference Manual* for the kinds of files accessible.
- POSIX.2 redirection specifications can appear anywhere in the command line. The specifications (and redirection operators within them) are processed in the order they appear, from left to right.

osh supports only file descriptors 0 through 9. Leading zeros are allowed.

In the following descriptions, if the file descriptor number is omitted and the first character of the redirection operator is <, the redirection refers to the standard input file (file descriptor 0). If the first character of the redirection operator is >, the redirection refers to the standard output file (file descriptor 1). You must specify file descriptor 2 to refer to the standard error file.

Redirecting Input

[*n*]<*pathname* Redirection of input opens the named file for read access on file descriptor *n*. The standard input file (file descriptor 0) is used if *n* is not specified.

Redirecting Output

[*n*]>*pathname* or [*n*]>|*pathname*
Redirection of output opens the named file or pipe for write access on file descriptor *n*. The standard output file (file descriptor 1) is used if *n* is not specified.

If the named file does not exist, it is created. If the named file does exist, it is truncated to zero length.

Appending Redirected Output

[*n*]>>*pathname* Redirection of output opens the named file for append access on file descriptor *n*. The standard output file (file descriptor 1) is used if *n* is not specified.

If the named file does not exist, it is created.

Redirecting Input and Output

[*n*]<>*pathname* Redirection of both input and output opens the named file for both read and write access on file descriptor *n*. The standard input file (file descriptor 0) is used if *n* is not specified.

If the named file does not exist, it is created.

Duplicating Input

[*n*]<&*fd* Duplicates input file descriptors. If the *fd* argument consists of one or more digits, the file descriptor *n* is made a copy of file descriptor *fd*. If *fd* is -, file descriptor *n* is closed. If *n* is not

specified, the standard input file (file descriptor 0) is used.

Duplicating Output

[*n*]>&*fd* Duplicates output file descriptors. If the *fd* argument consists of one or more digits, the file descriptor *n* is made a copy of file descriptor *fd*. If *fd* is -, file descriptor *n* is closed. If *n* is not specified, the standard output file (file descriptor 1) is used.

The order of redirections is significant. For example, the sequence

osh -c "ls -a" >lsout 2>&1

directs both standard output and standard error to the file **lsout**, while the sequence

osh -c "ls -a" 2>&1 >lsout

directs only standard output to the file **lsout**, because the standard error file was duplicated as the standard output file before the standard output file was redirected.

Operands

All operands that the **osh** command does not interpret as *option* or *redirection* arguments are passed to the child process as positional arguments in *argv***[1]** through *argv***[***n*]. **osh** does not expand operands or interpret special characters, except that double quotes can be used to group a set of operands into a single argument.

The character string that constitutes a single operand depends on the parsing done to the command line. An **osh** command line can be parsed up to four times before being executed.

The parsers that might interpret the line are:

TACL
The C run-time initializer
The **osh** process
The OSS shell

Each of these parsers applies different rules for special characters and uses different characters as escape sequences to bypass parsing. A command line that contains special characters must be entered with the escape sequences appropriate for each of these parsers.

For example, suppose a command line is entered through the Guardian environment TACL command interpreter. The TACL process can interpret or expand operands on the command line before passing the line to the **osh** command to interpret. The settings used for TACL features can affect the outcome of the command. In the following **osh** command, the OSS shell pipe symbol reaches the shell for execution if the user has #INFORMAT PLAIN in effect:

osh -c "grep '#'include *.c | sort | unique | more"

However, the pipe symbol cannot reach the shell for execution if the user has #INFORMAT TACL in effect because the | character has a programming function within TACL. With #INFORMAT TACL, the appropriate command line becomes:

osh -c "grep '#'include *.c | sort | unique | more"

In both examples, the double quotes (") prevent the C run-time initializer from dividing the *command* parameter value into pieces that the **osh** command passes on to the child process (the OSS shell) as multiple operands. The single quotes (') prevent the OSS shell from interpreting all characters of the operand following the number sign (#) as a comment.

You can pass 980 bytes of operands to the child process.

DESCRIPTION

The **osh** command executes an OSS program or shell script from the Guardian environment. **osh** is a Guardian process that spawns an OSS process within the same HP node. **osh** allows you to specify the environment and initial process attributes of the child process; it also allows redirection of data for the initially open files of the child process.

For the H06.23 or later H-series RVUs, or J06.12 RVUs or later J-series RVUs, or systems that have installed SPR T8628H01_AAU, the file mode creation mask (umask) of the spawned process is set to 0022. For the H06.25 or later H-series RVUs, or J06.14 or later J-series RVUs, or systems that have installed SPR T8628H01_AAV or later SPRs, the file mode creation mask (umask) of the spawned process is set using the Guardian DEFINE =OSSˆUMASK. If DEFINE =OSSˆUMASK is not set or the value is invalid, **osh** uses the default value of 0022.

EXAMPLES

1. Running the OSS **/bin/sh** file (a Korn shell) interactively:

   osh

2. Running an interactive shell script:

   osh script1

3. Running a noninteractive script:

   osh < script2

4. Reading this reference page:

   osh -c "man osh"

5. Running the OSS shell **ls** command with output to an OSS file:

   osh ls -a >/dirlist.txt

6. Using OSS pipes and regular expressions in a shell command (note the use of single quote marks (apostrophes) to prevent the OSS shell from interpreting the number sign (#) character):

   osh -c "grep '#'include *.c | sort | uniq | more"

7. Using an OSS process directly without a shell:

   osh -p cp ./myprog /usr/bin/myprog

8. Running a server program that has no controlling terminal:

   osh -nowait -p /etc/portmap <- >- 2<>logfile

9. Preventing the TACL command interpreter from interpreting an absolute pathname as a RUN option:

   osh -- /usr/scripts/myscript

10. Running the **osh** process programmatically from a Guardian environment C program:

```
void main( void )
{
  short myprocesshandle[ 10 ];
  char myterm[ 64 ];
  short mytermlen;
  char cmd[ 255 ];

  /* Get the home terminal name */
  PROCESSHANDLE_NULLIT_( myprocesshandle );
  PROCESS_GETINFO_( myprocesshandle ,
              /* filename */, /* filenamemaxlen */,
              /* filenamelen */, /* pri */, /* mom */,
              myterm, ( short )( sizeof ( myterm ) - 1),
              &mytermlen );
  myterm[ mytermlen ] = '\0';

  /* Build the command line */
  sprintf ( cmd, "osh / IN %s, OUT %s / -c \"%s \"",
        myterm, myterm, shcmd );

  system ( cmd );
  exit ( 0 );
}
```

where *shcmd* is any valid OSS shell command.

11. Passing text and multiple shell commands using a TACL macro as a substitute for a UNIX Here-document:

```
?TACL MACRO
#FRAME
==
== First, create a Guardian file named msg
== containing text.  To do this,
== add the TACL variable to be used, then
== place character strings in it.
==
#PUSH msg
#APPEND msg **********
#APPEND msg * Hello world! *
#APPEND msg **********
[ #IF [#FILEINFO/EXISTENCE/ msg] |THEN|
        SINK [#PURGE msg] ]
==
== Write the variable to the Guardian file
==
VARTOFILE msg msg
==
== Next, create a Guardian file named cmds
== containing OSS shell commands.  To do this,
== add the TACL variable to be used, then
== place character strings in it.
==
#PUSH cmds
#APPEND cmds echo 'cp msg /tmp/msg'
#APPEND cmds cp msg /tmp/msg
#APPEND cmds echo 'cat /tmp/msg'
#APPEND cmds cat /tmp/msg
#APPEND cmds echo 'rm /tmp/msg'
#APPEND cmds rm /tmp/msg
[ #IF [#FILEINFO/EXISTENCE/ cmds] |THEN|
        SINK [#PURGE cmds] ]
==
== Write the variable to the Guardian file
==
VARTOFILE cmds cmds
==
== Now, run a shell to process the commands
== in the Guardian file cmds
==
osh +ls cmds
==
== Remove no longer needed Guardian files
==
SINK [#PURGE cmds]
SINK [#PURGE msg]
#UNFRAME
```

12. Using the output from an OSS command in the TACL variable VAR1:

```
#SET #INFORMAT TACL
OBEY GNMTOVAR
#PUSH VAR1
GNAMETOVAR /usr/donl/printcap VAR1
OUTVAR VAR1
```

```
\BOSTON.$XPG.ZYQ00000.Z0000M5L
```

where the file GNMTOVAR contains the following code:

```
== This macro has the following runtime syntax:
==
== GNAMETOVAR <OSS_pathname> <TACL_variablename>
==
== where <OSS_pathname> is represented as %1% and
==     <TACL_variablename> is represented as %2%
==
[ #DEF GNAMETOVAR MACRO |BODY|
==
== Add temporary TACL variables to be used
==
        #PUSH cmds tmpfile1 tmpfile2
==
== Create a Guardian temporary file named tmpfile1
== to hold the output; a second file would be needed
== if there was more than one line of output (see below):
==
        #SET/TYPE DELTA/cmds 0,Z-4K
        [ #LOOP |DO|
                #SET tmpfile1 zz[ #DELTA/COMMANDS cmds/
                        [#TIMESTAMP] ]
                |UNTIL| NOT
                        [ #FILEINFO/EXISTENCE/ tmpfile1 ]
        ]
        CREATE [tmpfile1]
==
== Redirect the output of the OSS shell
== gname command to the Guardian temporary file
==
        osh -p gname -s %1% > [tmpfile1]
==
== Read the temporary file into the TACL variable and
== delete the trailing newline character; if the output
== will contain more than one line,  convert the file
== to an EDIT file using CTOEDIT [tmpfile1] [tmpfile2]
== and omit the #CHARDEL line:
==
        FILETOVAR [tmpfile1] %2%
        #CHARDEL %2% [ #CHARCOUNT %2% ] FOR 1
        SINK [ #PURGE [tmpfile1] ]
==
== Remove no longer needed TACL variables
==
```

```
  #POP cmds tmpfile1 tmpfile2
]
```

13. Redirecting the standard output file from an OSS shell session to the $VHS virtual home terminal subsystem for later postprocessing:

    OSH / OUT $VHS / -osstty

14. Overriding the default file mode creation mask (umask) used by **osh** with a value of 023 using DEFINE =OSS^UMASK. Note that DEFINE =OSS^UMASK must be set before running **osh**, or **osh** uses the default value of 0022:

    add DEFINE =OSS^UMASK, class MAP, file #023

FILES

/bin/sh Contains the default OSS shell to be executed.

NOTES

Many of the **osh** command options correspond to the options of the TACL RUN or RUND command that are usually called RUN options in the Guardian environment. Except as described under **Redirection** in this reference page, RUN options can be specified as part of the **osh** command but apply to the **osh** process itself, not to the child process created by the call to **osh**.

If you specify a RUN option value for **osh** but do not specify the corresponding **osh** option for the child process, the RUN option value you specify can be used as the default value for that attribute of the child process. However, if you specify both a RUN option value for **osh** and the corresponding **osh** option for the child process, the value specified for the RUN option has no effect on the value used for the child process.

For example:

osh / CPU 5 / -- /usr/scripts/myscript

runs both **osh** and **myscript** in processor 5, but

osh / CPU 5 / -cpu 3 -- /usr/scripts/myscript

runs **osh** in processor 5 and **myscript** in processor 3.

Use in Programs

When the **osh** command is run using the **system**() function, the TACL RUN options IN and OUT must be specified.

Guardian processes typically open a Guardian environment disk file for output by requesting protected or exclusive access. This practice can conflict with use of a file by an OSS process.

For example:

osh -c "ls -al"

fails if it is invoked within a program and the output from the program is redirected to a Guardian disk file. Because the output file is still open by the Guardian process executing the command file, the OSS shell cannot open the file and terminates abnormally.

Guardian Environment Variables

The following Guardian environment variables affect the execution of the **osh** command.

ASSIGNs

STDERR — Names the Guardian file to be used by **osh** as its Guardian standard error file. The default file is the hometerm file for the TACL session.

No Guardian environment ASSIGN values are passed to the child process.

PARAMs

HOME — If this parameter is defined, **osh** passes the value to the child process as its OSS environment variable **HOME**. If the Guardian HOME PARAM is not defined, then **osh** sets the OSS environment variable **HOME** according to the rules indicated under **Process Environment** later in this reference page.

LOGNAME — If this parameter is defined, **osh** passes the value to the child process as its OSS environment variable **LOGNAME**. If the Guardian LOGNAME PARAM is not defined, then **osh** sets the OSS environment variable **LOGNAME** to the user's login name.

The **LOGNAME** value has no effect on the determination of the **HOME** or **PWD** environment variable value passed to the child process. Refer to the rules indicated under **Process Environment** later in this reference page.

PATH — If this parameter is defined, **osh** interprets the value as a list of OSS directories, separated by colons. These directories are searched when resolving a relative OSS pathname for the program specified with the **-p** or **-prog** option.

If the Guardian PATH PARAM is not defined, **osh** uses the OSS **/bin** directory to resolve relative pathnames.

PWD — If this parameter is defined, **osh** passes the value to the child process as its OSS environment variable **PWD**. If the Guardian PWD PARAM is not defined, then **osh** sets the OSS environment variable **PWD** according to the rules indicated under **Process Environment** later in this reference page.

All Guardian PARAMs are converted into OSS environment variables for the child process. Circumflex (^) and hyphen (-) characters within a PARAM name are converted to underscore (_) characters in the equivalent OSS environment variable name.

A single PARAM name and value can contain up to 255 bytes of character information for one environment variable. Up to 1024 bytes of PARAM names and values are supported.

DEFINEs

=_DEFAULTS — Provides the default values for the current Guardian volume and subvolume names.

If the **-defmode on** option is used, all Guardian DEFINEs inherited by **osh** are

inherited by the child process. Up to 256K bytes of DEFINEs can be inherited. The actual maximum depends on the size of the PFS for the child process.

If the **-defmode off** option is used, only the Guardian =_DEFAULTS DEFINE values inherited by **osh** are inherited by the child process.

Process Environment

The child process is a session leader and its OSS parent process ID is set to 1. Its Guardian mom, or its Guardian ancestor if it is a named process, is the **osh** process that created it.

osh child processes have an initial signal mask in which all signals are defaulted.

The value of the OSS environment variable **HOME** passed to the child process is the first applicable value from the following list:

1. The content of the Guardian environment HOME PARAM variable, if one is defined.
2. If the **-phd** option was specified to **osh**, the initial login directory of the current effective user ID (PAID) of the parent process as configured on the HP node that runs the child process.
3. If the **-phd** option was not specified to **osh**, the initial login directory of the parent process as configured on the HP node that runs the child process.

   The initial login directory is determined by looking up the user ID that corresponds to the login name of the parent process in the authentication database of the node on which the child process runs. If that user ID matches the real user ID of the child process, then the initial login directory configured for the user ID of the parent process on the target node is used as the **HOME** environment variable value. If the user ID of the parent process does not match the real user ID of the child process, the initial login directory configured for the real user ID of the child process is used as the **HOME** environment variable value.

The value of the OSS environment variable **PWD** passed to the child process is the first applicable value from the following list:

1. The content of the Guardian environment PWD PARAM variable, if one is defined.
2. If a logon shell is being created and the **-phd** option was specfied to **osh**, the initial login directory of the current effective user ID (PAID) of the parent process as configured on the HP node that runs the child process.
3. If a login shell is being created and the **-phd** option was not specified to **osh**, the initial login directory of the parent process as configured on the HP node that runs the child process.

   The initial login directory is determined by looking up the user ID that corresponds to the login name of the parent process in the authentication database of the node on which the child process runs. If that user ID matches the real user ID of the child process, then the initial login directory configured for the user ID of the parent process on the target node is used as the **PWD** environment variable value. If the user ID of the parent process does not match the real user ID of the child process, the initial login directory configured for the real user ID of the child process is used as the **PWD** environment variable value.
4. The current working directory of the parent process (if a login shell is not being created).

The creation of the child process fails when an initial working directory and current working directory cannot be identified.

DIAGNOSTICS

Error diagnostics are written to the Guardian STDERR file of the **osh** process. All **osh** error messages are prefixed with `osh[`*n*`]:`, where *n* is a unique message number. The following messages can appear:

`osh[1]: standard files must be local to` *nodename*
: One of the standard input, output, or error files is not local to the HP node on which **osh** is running.

  Make sure that the IN, OUT, and either TERM or STDERR files are on your local HP node. Reenter the command with new specifications if needed.

`osh[2]: invalid I/O redirection syntax`
: The command line contains an unrecognized redirection specification.

  Check for typographical errors and reenter a corrected command line.

`osh[3]: here-documents are not supported`
: Your input included a UNIX **here**-document. **osh** does not support **here**-documents.

  Copy the **here**-document into a file and use the file as input. Or use **osh** to start an interactive OSS shell and reissue your **here** document as part of the appropriate shell command.

`osh[4]: only fds 0-9 are supported with I/O redirection`
: You entered a redirection specification containing a file descriptor outside of the supported range (0 through 9).

  Reissue the command using a file descriptor within the supported range.

`osh[5]: unable to get login name, using derived name` *username*
: The **getlogin**() function returned a NULL string. **osh** substituted the indicated name from your effective user ID.

  This is a warning message, indicating that **osh** was started from a process without a valid OSS login name.

`osh[6]: unable to get default volume, DEFINEINFO error` *n*
: Guardian file-system error *n* was returned from a call to the Guardian DEFINEINFO procedure that attempted to get the information for the =_DEFAULTS DEFINE. A system software problem might exist; the =_DEFAULTS DEFINE should exist for all user processes.

  Refer to the DEFINEINFO procedure description in the *Guardian Procedure Calls Reference Manual* for the meaning of the error returned. Check the current value of your =_DEFAULTS DEFINE by using the TACL INFO DEFINE command.

`osh[7]: the` *option_name* `option must be followed by` *permissible_values*
: You specified the indicated option with an unrecognized value.

  Check for typographical errors and reenter a corrected command line.

`osh[8]: root fileset is not mounted`
: The OSS root fileset is not mounted on your local HP node. The root fileset must be mounted to execute an **osh** command.

  Contact your system administrator and ask that the root fileset be mounted.

`osh[9]: unable to chdir to` *pathname*`, error` *n*`:` *strerror(n)*

osh could not set the current working directory for the child process. The **chdir**() function call returned the indicated error number. The meaning of that error number as returned by the **strerror**() function is displayed. The indicated *pathname* might not exist, or you might not have search permission (read and execute access) for it.

Check the value used for the OSS environment variable **PWD** if the child process was not to be a login shell. Set the Guardian PWD PARAM environment variable to a new value if necessary and reenter the command.

`osh[10]: <stdin|stdout|stderr|fd`*n*`> file` *pathname* `is not OSS-openable`

The file identified by *pathname* cannot be opened by an OSS environment **open**() function call. The file must be one of the following for an OSS application to open it:

- An unstructured disk file, a FIFO, or a Telserv window
- For **stdin** when the **-osstty** flag is not used, an EDIT file
- For **stdin** when the **-osstty** flag is used, a Guardian process or an EDIT file
- For **stdout** when the **-osstty** flag is used, a Guardian process or an EDIT file
- For **stderr** when the **-osstty** flag is used, a Guardian process

If **-osstty** has been specified, OSSTTY was not launched. The OSS application is not running.

Reenter the command but redirect the information for the indicated file to a file that can be opened within the OSS environment.

`osh[11]: unable to open child process, error` *n*`:` *strerror(n)*

The **osh** command detected an error while trying to open the child process and send the child process a sequence of Guardian environment startup messages. The Guardian file-system error *n* was returned. The meaning of that error number as returned by the **strerror**() function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions. A typical error is 201, because the child process terminated as soon as it started. If this is the case, use the **-inspect saveabend** option to create a saveabend file or the **-debug** option to start the child process within a debugging tool.

`osh[12]: Unable to run` *pathname*`, error (`[*errno*`,]` *n*[`,`*m*`]):` *explanation*

osh could not start the child process for the program file *pathname*. If the error occurred in the OSS environment, the *errno* value returned is shown. The Guardian process-creation error *n* was returned. The associated error detail *m* value is also returned. An explanation of the error is given.

Refer to the following for an explanation of the error and possible error-specific recovery actions:

- The display from the TACL command ERROR *errno*

- The process-creation error *n* and *m* descriptions in the *Guardian Procedure Errors and Messages Manual*

If the value of *n* is 111 or 112 and the value of *m* is 0, 1, or 2, the underlying PROCESS_SPAWN_ procedure call was unable to open the standard files of the OSS application. Check if any of the standard files specified is a Guardian process or if the standard output file is an EDIT file. If either condition exists, reissue the command with the **-osstty** flag specified.

If **osh** is unable to stop the OSSTTY process, this warning message is displayed:

```
WARNING: unable to stop OSSTTY process process-name,
PROCESS_STOP_() failed with error: error
```

Stop the OSSTTY process manually. For more information about *error*, see the PROCESS_STOP_ procedure in the *Guardian Procedure Calls Reference Manual*.

`osh[13]: Unable to read from` *filename*, error *n*`:` *strerror(n)*

The **osh** command detected an error while trying to read the indicated file. The Guardian file-system error *n* was returned. The meaning of that error number as returned by the **strerror**() function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and possible error-specific recovery actions.

`osh[14]: Unable to get process information,` *procedure* `error` *n*

The **osh** process received the Guardian file-system error *n* when trying to determine its own Guardian process attributes. The **osh** process uses many of its own Guardian process attributes (such as processor number and home terminal name) as default values when creating the child process.

The Guardian procedure call indicated by *procedure* encountered the problem. This message occurs when one of the following conditions is true:

- The version of **osh** used is not compatible with the version of the operating system.
- The NonStop Kernel message system does not have enough resources to provide the information.
- A coding error exists within this version of **osh**.

Recovery action depends on the error returned. Refer to the description of *procedure* in the *Guardian Procedure Calls Reference Manual* for more detailed information about the error when returned by that procedure.

`osh[15]: unrecognized option:` *option_name*

You specified an option that **osh** does not recognize.

Check for typographical errors and reenter a corrected command line.

`osh[16]: unable to open $RECEIVE, error` *n*`:` *strerror(n)*

Guardian file-system error *n* was returned when **osh** attempted to open its own Guardian $RECEIVE file. The meaning of that error number as returned by the **strerror**() function is displayed.

The recovery action depends on the Guardian file-system error number. Refer to the *Guardian Procedure Calls Reference Manual* for an explanation of the error

and suggested error-specific recovery actions.

`osh[17]: internal error -` *description*
The **osh** process has detected the situation described by *description*. This error should also create a saveabend file for the **osh** process.

Report this problem to your service provider. Give the service provider a copy of the saveabend file and describe the conditions necessary to reproduce the problem.

`osh[18]: unable to allocate` *nbytes* `for` *purpose*
The **osh** process could not allocate the indicated amount of memory from its heap. This probably indicates that **osh** could not allocate a Guardian file-system extent on the extended segment file used for the heap.

If this problem occurs because of disk allocation failure, run **osh** using an extended swap volume or extended segment file that has more free space.

`osh[19]: unable to send` *msgtype* `message, error` *n*`:` *strerror(n)*
The **osh** process could not send a Guardian-environment startup, ASSIGN, or PARAM message to its child process. The Guardian file-system error *n* was returned. The meaning of that error number as returned by the **strerror**() function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

procedure `error` *n* `on` *file*`:` *strerror(n)*
The indicated procedure returned Guardian file-system error *n* for the named file. The meaning of that error number as returned by the **strerror**() function is displayed.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

`osh[21]: Unable to redirect standard files via OSSTTY (`*a*`,` *b*`,` *c*`):` *explanation*
The meaning of this message depends upon the value returned for *a*:

1 The value of *explanation* is: `unable to launch OSSTTY`. The value returned as *b* is the Guardian file-system error returned by the attempt. The value returned as *c* is the related detail error value.

The **osh** process did not launch the OSS application.

Refer to the process-creation error descriptions in the *Guardian Procedure Errors and Messages Manual* for an explanation of the error value and suggested error-specific recovery actions.

2 The value of *explanation* is: `unable to communicate with OSSTTY`. The value returned as *b* is the Guardian file-system error returned by the attempt. No value is returned as *c*.

OSSTTY started but stopped immediately. The **osh** process did not launch the OSS application.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions.

3 The value of *explanation* is: `OSSTTY terminated, unable to run OSS application`. The value returned as *b* is the Guardian file-system error returned by the attempt to create the OSS application process. The value returned as *c* is the file descriptor for the OSSTTY process involved in the failure.

The **osh** process did not launch the OSS application.

Refer to the process-creation errors in the *Guardian Procedure Errors and Messages Manual* for an explanation of the error and suggested error-specific recovery actions. Check the following for the Guardian objects corresponding to the standard files:

- The Guardian file or process exists
- The Guardian file is not corrupted
- The Guardian file has the correct access permissions to be used in this command
- The Guardian file is not opened by another process

Check event logs for any OSSTTY Event Management Service (EMS) messages.

4 The value of *explanation* is: `WARNING: OSSTTY terminated... OSS application running.` The value returned as *b* is the completion code of the OSSTTY process. No value is returned for *c*. If an external process stopped OSSTTY, the process name and user ID associated with that process is written to the **osh** process standard error file.

OSSTTY stopped after the OSS application process was created. The OSS application process might continue to run, depending on how it is coded. However, neither it nor any OSS process it spawns can redirect their standard files to Guardian file-system objects.

Refer to the *Guardian Procedure Errors and Messages Manual* for an explanation of the completion code. Check event logs for any OSSTTY Event Management Service (EMS) messages.

`osh[22]:` *stdfile* `is a non-existent file/process.`

The specification for the standard file represented by *stdfile* refers to a nonexistent file or process for OSSTTY redirection.

The value returned for *stdfile* is one of the following:

STDERR The standard error file

STDIN The standard input file

STDOUT The standard output file

Neither OSSTTY nor the OSS application are running.

Either create the missing file or launch the required process, then reissue the command.

EXIT VALUES

If the **osh** command successfully starts the child process, **osh** returns the completion code 0. Otherwise, **osh** returns one of the following completion codes:

3	The **osh** process terminated abnormally.
2	The child process terminated abnormally.
1	Warnings occurred when the child process started.

The completion code returned by the child process is returned as the termination status of the **osh** process. For example:

126	Indicates the command was found but is not executable.
127	Indicates the command was not found.

RELATED INFORMATION

Commands: **gtacl(1)**.

Functions: **getlogin(2)**, **open(2)**, **tdm_execve(2)**, **tdm_execvep(2)**, **strerror(3)**, **system(3)**.

Files: **core(4)**, **errno(5)**.

STANDARDS CONFORMANCE

The **osh** command as a mechanism for starting a shell is an HP extension to the XPG4 Version 2 specification.

Section 7. User Commands (p - r)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letters **p** through **r**.

NAME

pack - Compresses files

SYNOPSIS

pack [**-f**] [-] *file* ...

FLAGS

- Displays statistics about the input files. The statistics are calculated from a Huffman minimum redundancy code tree built on a byte-by-byte basis. Repeating the - (dash) on the command line toggles this function.

-f Forces compaction of input files.

DESCRIPTION

The **pack** command stores the specified file in a compressed form. The input file is replaced by a packed file with a name derived from the original filename (*file***.z**). The **pack** command tries to preserve the access modes, access and modification dates, and owner from the original file, but it can do so only if you have the appropriate privileges (see the **chmod(1)** reference page); otherwise, **pack** compresses the file and assigns your owner and group ID to the new file.

Directories cannot be compressed.

If **pack** cannot create a smaller file, it stops processing and reports that it is unable to save space, unless you specify **-f**. (The **-f** flag forces packing to occur even if the files cannot benefit from packing.) A failure to save space generally happens with small files or files with uniform character distribution.

The amount of space saved depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each **.z** file, you will generally not be able to save space with files smaller than three blocks. Typically, text files are reduced 25 to 40 percent. Object files, which use a larger character set and have a more uniform distribution of characters, show only a 10-percent reduction when packed.

Packing is not done if any of the following conditions are true:

- The file is already packed.
- The file has hard links.
- The file is a directory.
- The file cannot be opened.
- No storage blocks are saved by packing.
- A file called *file***.z** already exists.
- The **.z** file cannot be created.
- An I/O error occurs during processing.
- The filename is more than **NAME_MAX**-2 bytes long.
- The file is empty and the **-f** option has not been specified.

If the file has an access control list (ACL), the ACL is preserved when the file is packed. For more information about ACLs, see the **acl(5)** reference page.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

1. To compress files, enter:

   pack chap1 chap2

   This command compresses the files **chap1** and **chap2**, replacing them with files named **chap1.z** and **chap2.z**. The **pack** command displays the percent decrease in size for each file.

2. To display statistics about the amount of compression done, enter:

   pack - chap1 - chap2

   This command compresses the files **chap1** and **chap2** and displays statistics about **chap1** but not about **chap2**. The first **-** (dash) turns on the statistic display, and the second **-** (dash) turns it off.

NOTES

If you try to use **pack** on a very small file, you might receive the following message:

`pack` *filename*`: No saving -- file unchanged`

EXIT VALUES

The **pack** command returns the following values:

0 (zero)	The command completed successfully and all files were packed.
>0	An error occurred because some of the files could not be packed. The number returned is the number of files that the **pack** command could not pack.

RELATED INFORMATION

Commands: **cat(1)**, **compress(1)**, **uncompress(1)**, **unpack(1)**, **zcat(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

paste - Joins lines from one or more files

SYNOPSIS

paste [**-d** *list*] [**-s**] *file ...*

The **paste** command reads input files, joins corresponding lines, and writes the result to standard output. It also reads standard input if you specify a **-** (dash) instead of a filename.

FLAGS

-d *list* Replaces the delimiter that separates lines in the output (tab by default) with one or more characters from *list*. If *list* contains more than one character, then the characters are repeated in order until the end of the output. In parallel merging, the lines from the last file always end with a newline character instead of one from *list*.

The following special characters can be used in *list*:

\n	Newline character
\t	Tab
\\	Backslash
\0	Empty string (not a null character)
c	An extended character

You must quote characters that have special meaning to the shell.

When the **-s** flag is specified with **-d**, the last newline character in a file is preserved, and the delimiter is reset to the first element of *list* after each file is processed. If **-s** is not specified with **-d**, the newline characters in the last file specified are preserved, and the delimiter is reset to the first element of *list* each time a line is processed.

-s Merges all lines from each input file into one line of output (serial merging). With this flag, **paste** works through one entire file before starting on the next. When it finishes merging the lines in one file, it forces a newline and then merges the lines in the next input file, continuing in the same way through the remaining input files, one at a time. A tab separates the input lines unless you use the **-d** flag. Regardless of the *list*, the last character of the output is a newline character.

DESCRIPTION

Without a flag, or with the **-d** flag, **paste** treats each file as a column and joins them horizontally with a tab character by default (parallel merging).

With the **-s** flag, **paste** combines all lines of each input file into one output line (serial merging). These lines are joined with the tab character by default.

Output lines can be of arbitrary length.

If an End-of-File condition is detected on one or more input files, but not all input files, **paste** behaves as though empty lines were read from the file(s) on which End-of-File was detected, unless the **-s** flag is specified.

Note that the output of **pr -t -m** is similar to that of **paste**, but it creates extra spaces, tabs, and lines for an enhanced page layout.

EXAMPLES

1. To paste several columns of data together, enter:

   paste names places dates > npd

   This creates a file named **npd** that contains the data from **names** in one column, **places** in another, and **dates** in a third. The columns are separated by tab characters.

   npd then contains:

```
names           places          dates

rachel          New York        28 February
jerzy           Warsaw          27 April
mata            Nairobi         21 June
michel          Boca Raton      27 July
segui           Managua         18 November
```

   A tab character separates the name, place, and date on each line.

2. To separate the columns with a character other than a tab (**sh** only), enter:

   paste -d "!@" names places dates > npd

   This alternates **!** and **@** as the column separators. If **names**, **places**, and **dates** are the same as in Example 1, then **npd** contains:

```
rachel!New York@28 February
jerzy!Warsaw@27 April
mata!Nairobi@21 June
michel!Boca Raton@27 July
segui!Managua@18 November
```

3. To display the standard input in multiple columns, enter:

   ls | paste - - - -

   This lists the current directory in four columns. Each **-** (dash) tells **paste** to create a column containing data read from the standard input. The first line is put in the first column, the second line in the second column, ... and then the fifth line in the first column, and so on.

   This is equivalent to the following:

   ls | paste -d "\t\t\n" -s -

   The preceding command line fills the columns across the page with subsequent lines from the standard input. The **-d "\t\t\n"** defines the character to insert after each column: a tab character (**\t**) after the first three columns, and a newline character (**\n**) after the fourth. Without the **-d** flag, **paste -s -** displays all of the input as one line with a tab between each column.

4. To merge the lines of the file **names** above into one output line, enter:

   paste -s names

   This results in the following:

```
rachel  jerzy   mata    michel  segui
```

RELATED INFORMATION

Commands: **cut**(1), **grep**(1), **pr**(1).

Files: **locale**(4).

NAME

patch - Applies changes to files

SYNOPSIS

patch
[**-bflNRsSv**]
[**-c** | **-n**]
[**-B** *prefix*]
[**-d** *directory*]
[**-D** *define*]
[**-i** *patchfile*]
[**-o** *outfile*]
[**-p** [*number*]]
[**-r** *rejectfile*]
[*original_file*]
[[**+** *flags*] [*original_file*] . . .]

FLAGS

-b Saves a copy of the original contents of each modified file, before the differences are applied, in a file of the same name with the suffix **.orig** appended. If the file already exists, it is overwritten; if multiple patches are applied to the same file, the **.orig** file is written only for the first patch. If **-o** *outfile* is also specified, *patchfile***.orig** is not created, but if *outfile* already exists, *outfile***.orig** is created.

-B *prefix* Specifies a prefix to the backup filename.

-c Interprets the patch file as a context **diff** script (the output of **diff** when the **-c** or **-C** flag is specified).

-d *directory* Changes the current directory to *directory* before processing.

-D *define* Uses the C preprocessor **#ifdef** *define* **#endif** construct to mark changes. The *define* argument is used as the differentiating symbol.

-f Suppresses queries to the user. To suppress commentary, use the **-s** flag.

-i *patchfile* Reads the patch information from the file named by *patchfile*, rather than from the standard input file.

-l Causes any sequence of spaces and tabs (white space) in the **diff** script to match any sequence of spaces in the input file. Other characters are matched exactly.

-n Interprets the script as a normal **diff** script.

-N Ignores patches where the differences have already been applied to the file; by default, already applied patches are rejected. (See the **-R** flag.)

-o *outfile* Instead of modifying the files (specified by the *patchfile* argument or the **diff** listings) directly, writes a copy of the file referenced by each patch, with the appropriate differences applied, to *outfile*. Multiple patches for a single file are applied to the intermediate versions of the file created by any previous patches, and result in multiple, concatenated versions of the file written to *outfile*.

-p [*number*] Sets the pathname strip count, which controls how pathnames found in the patch file are treated. This flag is useful if you keep your files in a different directory than that specified by the patch. The strip count *number* specifies how many slashes to strip from the front of a pathname. Any intervening directory names are also stripped.

If *number* is omitted or 0, then the pathname is not modified. If the **-p** flag is omitted, all slashes and directory names preceding the filename are stripped.

For example, if the filename in the patch file was **/u/howard/src/blurfl/blurfl.c**, entering **-p** or **-p0** leaves the entire pathname unmodified. Entering **-p1** results in **u/howard/src/blurfl/blurfl.c** without the leading slash. Entering **-p4** results in **blurfl/blurfl.c**. Omitting **-p** from the **patch** command line results in **blurfl.c**.

The **patch** command looks for the resulting pathname in either the current directory or the directory specified by the **-d** flag.

-r *rejectfile* Specifies the filename of the reject file. By default, the reject file has the same name as the output file, with the suffix **.rej** appended.

-R Reverses the sense of the patch script; that is, **patch** assumes that the **diff** script was created by comparing the new version to the old version. The **patch** command tries to reverse each portion of the script before applying it. Rejected differences are saved in swapped format.

If this flag is not specified, then until a portion of the patch file is successfully applied, **patch** tries to apply each portion in its reversed sense as well as in its normal sense. If the attempt is successful, you are prompted to determine whether **-R** should be set.

Note that this method cannot detect a reversed patch if it is a normal **diff** script and if the first command is an append (that is, it should have been a delete): appends always succeed because a **NULL** context matches anywhere. However, most patches add or change lines rather than delete them, so most reversed normal **diff** scripts begin with a delete, which will fail, triggering the heuristic.

-s Patches silently unless an error occurs.

-S Ignores a patch from the patch file, but continues looking for the next patch in the file. For example

patch -S + -S + <patchfile

ignores the first and second patches in **patchfile**.

-v Prints out the revision header and patch level.

+ *flags* [*original_file*]
Specifies flags (and possibly another original filename) for the second and subsequent patches. The argument list for each patch must be preceded with **+** (plus sign). (Note that the argument list for a second or subsequent patch may not specify a new patch file.)

Operands

original_file Specifies the file to be patched.

DESCRIPTION

The **patch** command takes a patch file that contains either of the forms of difference listing produced by the **diff** program (normal or context) and applies those differences to an original file, producing a patched version.

By default, the patched version of a file replaces the original. The original file can be backed up to the same name with the extension **.orig** by specifying the **-b** flag. You can also specify where you want the output to go with the **-o** flag. If the **-i** *patchfile* flag is not specified, or if *patchfile* is **-** (dash), the patch is read from the standard input file.

The **patch** command attempts to determine the type of the **diff** script, unless it is overruled by a **-c** or **-n** flag, which specify context **diff**s and normal **diff**s, respectively.

The patch file must contain zero or more lines of header information followed by one or more patches. Each patch must contain zero or more lines of filename identification in the format produced by **diff -c**, and one or more sets of **diff** output, which are customarily called "hunks."

The **patch** command tries to skip any leading text, apply the **diff**, and then skip any trailing text. Therefore, you could feed an article or message containing a **diff** listing to **patch**, and it will work. If the entire **diff** is indented by a consistent amount, **patch** takes this into account.

If *original_file* is not specified, **patch** tries to determine the name of the file to edit from the leading text. In the header of a context **diff**, **patch** searches for filenames in lines beginning with ******* (the name of the file from which the patches arose) or **---** (the name of the file to which the patches should be applied), and selects the shortest name of an existing file. If there is an **Index:** line in the leading text, **patch** tries to use the filename from that line. The context **diff** header takes precedence over an **Index:** line. If no filename can be determined from the leading text, **patch** asks you for the name of the file to patch.

If the original file cannot be found but a suitable SCCS or RCS file is available, **patch** attempts to get or check out the SCCS or RCS file.

Additionally, if the leading text contains a **Prereq:** line, **patch** takes the first word from the prerequisites line (normally a version number) and searches the input file for that word. If the word is not found, **patch** asks for confirmation before proceeding.

If the patch file contains more than one patch, **patch** tries to apply each patch as if it came from a separate patch file. In this case, the name of the file to patch must be determined for each **diff** listing, and the header text before each **diff** listing is examined for information such as filenames and revision level. You can give flags (and another original filename) for the second and subsequent patches by separating the corresponding argument lists with a **+**.

For each hunk, **patch** calculates the location to apply the hunk using the line number mentioned for the hunk, plus or minus any offset used in applying the previous hunk. If that calculation does not give the correct place to apply the hunk, **patch** scans both forward and backward for a set of lines matching the context given in the hunk. The **patch** command searches for a location where all lines of the context match.

If **patch** cannot find a place to install that hunk of the patch, it places the hunk in a reject file (normally, a file with the same name as the output file plus the suffix **.rej**). The rejected hunk is written in the form of a context **diff** script, regardless of the format of the patch file. If the input was a normal **diff** or **ed**-style script, the reject file might contain **diff**s with zero lines of context. The line numbers on the hunks in the reject file might be different from the line numbers in the patch file; the reject file line numbers reflect the approximate locations for the failed hunks in the new file rather than the old file.

As each hunk is completed, **patch** tells you whether the hunk succeeded or failed, and on which line in the new file **patch** assumed the hunk should go. If this line does not have the line number specified in the **diff**, **patch** tells you the offset. A single large offset might be an indication that a hunk was installed in the wrong place.

NOTES

Note the following if you are going to be sending out patches:

- HP recommends that you keep a **patchlevel.h** file that is patched to increment the patch level as the first **diff** in the patch file you send out. If you put a **Prereq:** line in with the patch, users will not be able to apply patches out of order without some warning.

- Make sure you specify the filenames correctly, either in a context **diff** header or with an **Index:** line. If you are patching something in a subdirectory, be sure to tell the patch user to specify a **-p** flag as needed.
- You can create a file by using a **diff** script that compares a null file to the file you want to create. This works only if the file you want to create does not already exist in the target directory.
- Take care not to send out reversed patches, because this makes users wonder whether they have already applied the patch.
- While you might be able to put many **diff** scripts into one file, it is advisable to group related patches into separate files.
- The **patch** command can detect bad line numbers in a normal **diff** script only when it finds a **change** or a **delete** command.
- The results of the **patch** command are guaranteed to be correct only when the patch is applied to exactly the same version of the file from which the patch was generated.
- If the code has been duplicated (for example, with **#ifdef OLDCODE** ... **#else** ... **#endif**), **patch** is incapable of patching both versions, and, if **patch** works at all, it will likely patch the wrong version and tell you that it succeeded.
- If you apply a patch you have already applied, **patch** assumes it is a reversed patch and offers to undo it.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **NLSPATH**, and **LC_TIME** environment variables.

DIAGNOSTICS

The message `Hmm...` indicates that there is unprocessed text in the patch file and that **patch** is attempting to determine whether there is a patch in that text and, if there is, what kind of patch it is.

EXIT VALUES

The **patch** command returns the following values:

0 (zero)	The command completed successfully.
1	At least one reject file was created.
>1	An error occurred.

When applying a set of patches in a loop, you should check this exit status after each call to **patch**, so that you do not apply a later patch to a partially patched file.

RELATED INFORMATION

Commands: **diff(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

pathchk - Checks pathnames

SYNOPSIS

pathchk [**-p**] *pathname* ...

The **pathchk** command checks that one or more pathnames are valid (that is, they can be used to access or create a file without causing syntax errors) and portable (that is, no filename truncation will result).

FLAGS

-p Performs pathname checks based on POSIX portability standards. An error message is sent if any of the following conditions are true:

- The byte length of the full pathname is longer than allowed by POSIX standards (**_POSIX_PATH_MAX**).
- The byte length of a component is longer than allowed by POSIX standards (**_POSIX_NAME_MAX**).
- A character in any component is not in the portable filename character set.

DESCRIPTION

By default, the **pathchk** command checks each component of each pathname specified by the *pathname* argument based on the underlying file system. If the **-p** flag is not specified, **pathchk** sends an error message if any of the following conditions are true:

- The byte length of the full pathname is longer than allowed by the system (**PATH_MAX** bytes).
- The byte length of a component is longer than allowed by the system (**NAME_MAX** bytes).
- Search permission is not allowed for a component.
- A character in any component is not valid in its containing directory.

It is not considered an error if one or more components of a pathname do not exist, as long as a file matching the pathname specified by the *pathname* argument could be created without violating any of the preceding criteria.

EXAMPLES

1. To check the validity and portability of the **/u/bob/work/tempfiles** pathname, enter:

   pathchk /u/bob/work/tempfiles

2. To check the validity and portability of the **/u/bob/temp** pathname for POSIX standards, enter:

   pathchk -p /u/bob/temp

EXIT VALUES

The **pathchk** command returns the following exit values:

0 All pathname operands passed the checks.

>0 An error occurred.

NAME

pax - Extracts (reads), writes, and lists archive files, and copies files and directory hierarchies

SYNOPSIS

List Members of Archived Files

pax [**-cdnv**] [**-f** *archive*]
[**-s** *replstr*] [**-W** *options*] [*pattern* ...] [*pattern* ...]

Read (Extract) Archived Files

pax -r [**-cdiknuv**] [**-f** *archive*] [**-p** *string*] ...
[**-s** *replstr*] ... [**-W** *options*] [*pattern* ...]

Write a File Archive

pax -w [**-aAdituvX**] [**-b** *blocksize*] [**-f** *archive*] [**-L**]
[**-s** *replstr*] ... [**-x** *format*] [**-W** *options*] [*pattern* ...] [*file* ...] [*file* ...]

Copy Files

pax -r -w [**-Adiklntuv X**] [**-p** *string*] ...
[**-s** *replstr*] ... [**-W** *options*] [*file* ...] *directory*

FLAGS

-a Appends files to the end of the archive. The **-a** flag is used only when writing disk files. It is not supported for tape archives.

-A Suppresses warning messages about optional access control list (ACL) entries. Because the **pax** utility does not archive optional ACL entries, a warning message is printed for each file that has optional ACL entries. However, if **pax** is executed remotely from a system that does not support OSS ACLs, no warnings are printed.

-b *blocksize*

Specifies a positive decimal integer of bytes to be the block size for output. The block size cannot exceed 32256 bytes for archives on disk or 28672 bytes for archives on tape. Block size is automatically determined on input.

blocksize is specified as a series of digits (0-9) followed by the optional letter **b** or **k**. If **b** is specified, the block-size value is multiplied by 512. If **k** is specified, the block-size value is multiplied by 1024. For example, if **10b** is specified as the value for *blocksize*, it is translated into a block-size value of 5120 bytes (10 * 512).

The default block size for the **cpio** archive format is **10b** (5120 bytes). The block size of the last group of blocks is always set to the maximum size.

The default block size for the **ustar** archive format is **10k** (10240 bytes). The *blocksize* argument is specified as a multiple of 512 bytes.

-c Selects all file or archive members that do not match the *pattern* or *file* operands.

-d Specifies that directories being copied or archived, and archived directories being extracted, match only the directory or archive that is explicitly named. Information for intermediate subdirectories in an archive is not stored, and files are not extracted unless the required directories already exist.

-f *archive*

Specifies the pathname of the input or output archive. The value specified for *archive* overrides the default standard input file when in list or read mode and overrides the default standard output file when in write mode.

Guardian tape-device names can be specified with the **/G** naming convention (**/G/tape**). **pax** does not support labeled tapes, so only unlabeled tapes should be used

for archives. Refer to the **NOTES** subsection of this reference page for considerations when using unlabeled tapes.

If the **-a** option is also specified, and the archive file is written to a disk volume, files are appended to the end of the archive.

-i Renames files or archives interactively. For each archive member that matches the *pattern* operand or *file* operand, a prompt is written to the terminal associated with the **pax** process. The prompt contains the name of the file or archive member that is to be renamed. Users' responses are also read from the terminal.

If the user's response to the prompt is empty, the file or archive member is skipped. If the response is a single period (dot), the file or archive member is processed with no modification to its name. Otherwise, the name of the file or archive member is replaced with the contents of the response.

The **pax** utility immediately terminates with a nonzero exit value if an end-of-file is encountered when reading a response. The **pax** utility terminates with a nonzero exit value if an interrupt signal is received, or if the terminal cannot be opened for reading and writing.

-k Prevents **pax** from writing over existing files (in read and copy modes).

-l Links files. In copy mode, hard links are made between the source and destination file hierarchies whenever possible.

-L Archives the file hierarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. The default action is to archive the symbolic link itself.

-n Selects the first archive member that matches each *pattern* operand. No more than one archive member is matched for each pattern (although members of type directory will still match the file hierarchy rooted at that file).

-p *string* Specifies one or more file characteristic options (privileges). The *string* argument must be a string specifying the file characteristics to be retained or discarded on extraction, subject to the permissions of the invoking process. Otherwise, the attribute is determined as part of the normal file creation action.

The *string* can consist of any combination of the following options:

a Do not preserve file access times.

e Preserve the user ID, group ID, file mode bits, file access times, and file modification times when the user executing the **pax** command has a logon ID with appropriate privileges.

m Do not preserve file modification times.

o Preserve the user ID and group ID when the user executing the **pax** command has a logon ID with appropriate privileges.

p Preserve the file mode bits.

Multiple **-p** options are allowed in one command. If an option in *string* duplicates or conflicts with another option in *string*, the option given last takes precedence.

If neither the **-e** nor the **-o** option is specified, or if the user ID and group ID are not retained, **pax** does not set the **S_ISUID** and **S_ISGID** bits of the access permission. If the retention of any of these items fails, **pax** writes a diagnostic message to the

standard error file. Failure to retain any of the items affects the exit value but does not cause the extracted file to be deleted.

-r Reads an archive file from the standard input file.

-s *replstr* Modifies file-member or archive-member names specified by the *pattern* or *file* operands according to the substitution expression *replstr* and using the syntax of the **ed** command. The substitution expression has the following format:

-s /*old*/*new*/[**gp**]

where (as in the **ed** command) *old* is a basic regular expression and *new* is the replacement string to be inserted in place of strings that match the regular expression. The string *new* can contain an **&** (ampersand), *n* (where *n* is a digit), back references, or subexpression matching. A **g** following *new/* replaces all matches. A **p** following *new/* causes successful substitutions to be written to the standard error file.

Any nonnull character can be used as a delimiter (/ shown here). Multiple **-s** flag expressions can be specified in one command. Multiple expressions are applied in the order specified, terminating with the first successful substitution. File-member or archive-member names that are replaced by an empty string are ignored when reading and writing archives.

-t Prevents the access times of the archived files from being changed when the files are read by **pax**.

-u Ignores files that are older (that have a less recent file modification time) than a preexisting file or archive member with the same name.

When extracting files from an archive, an archive member with the same name as a file in the file system is extracted if the archive member is newer than the file.

When writing files to an archive, an archive member with the same name as a file in the file system is superseded if the file is newer than the archive member.

When copying files to a destination directory, a file in the destination hierarchy is replaced by a file in the source hierarchy or by a link to a file in the source hierarchy if the file in the source hierarchy is newer.

-v In list mode, **-v** produces a table of contents (see **Standard Output** under **DESCRIPTION**). Otherwise, **-v** writes the archive member pathnames to the standard error file (see **Standard Error** under **DESCRIPTION**).

-w Writes files to the standard output file in the specified archive format.

-x *format*

Specifies the output archive format. The **pax** command recognizes the following formats:

cpio
Extended **cpio** interchange format.

ustar
Extended **tar** interchange format.

If this option is omitted, **ustar** format is used.

Any attempt to append an archive file in a format that is different from the existing format causes **pax** to exit immediately with a nonzero exit value.

-X Prevents **pax** from descending into directories that have a different device ID when traversing the file hierarchy specified by a pathname.

HP Extensions

-W clobber

Allows the matching files from an archive to be restored to a Guardian target and to overwrite any existing Guardian target file with the same name. The files are restored as unstructured files and Guardian file attributes are not preserved.

Be sure you understand the problems this can create (refer to the subsection **Guardian Filename Transformation** under **DESCRIPTION** before using this flag).

-W NOG

Specifies that the Guardian namespace (**/G**) should not be accessible to **pax**. This flag is effective only if the specified file is in the local (**/G**) or remote (**/E/***node*/G) Guardian namespace. This flag is ignored if you specify a wildcard character in or as the target of the **pax** command because wildcard characters are not recursive operations.

-W NOE Specifies that the Expand namespace (**/E**) should not be accessible to **pax**. This flag is effective only if the specified file is in the Expand (**/E**) namespace. This flag is ignored if you specify a wildcard character in or as the target of the **pax** command because wildcard characters are not recursive operations.

-W noprompt

Specifies that **pax** should not prompt the user to load a tape. **pax** waits indefinitely until the tape gets mounted. The **-W noprompt** flag has precedence over the **-W wait** flag.

-W norewind

Specifies that the tape be left on the tape drive without being rewound and unloaded after **pax** has finished writing to or reading from it. The tape drive is closed. The default action is for the tape to be rewound.

The **pax** utility cannot append files on unlabeled tapes. Refer to the **NOTES** subsection of this reference page for more information.

The **-W nounload** flag is assumed when the **-W norewind** flag is specified. The **-W unload** and **-W norewind** flags cannot be specified together.

-W [**unload** | **nounload**]

Specifies whether the tape is to be unloaded or left on the tape drive after it has been rewound. The default action is **-W unload**.

-W [**wait** | **nowait**]

Specifies which of the following **pax** is to do if a tape is not mounted on the tape drive or the tape drive goes off line:

- Issue a prompt and wait for you to mount a tape
- Exit immediately, without waiting

The default action is **-W nowait**.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

The **pax** utility reads and writes member files of archive files; writes lists of the member files of archives; and copies directory hierarchies.

The **-r** and **-w** flags specify the archive operation performed by **pax**.

The name of the archive file can be specified with the **-f** flag. The archive file can be a disk file or tape file on a mounted tape. If the **-f** flag is not specified, in read mode the archive file is assumed to be the standard input file and in write mode the archive file is assumed to be the standard output file. If the standard input file is a disk file, the file should be a text file with one pathname per line and no leading or trailing blanks.

Operands

directory The destination directory pathname for copy mode.

file The pathname of a file to be copied or archived.

pattern A pattern matching one or more pathnames of archive members. The default action, if no pattern is specified, is to select all members in the archive.

Flag Interaction and Processing Order

The flags that operate on the names of files or archive members (**-c**, **-i**, **-n**, **-s**, **-u**, and **-v**) interact as follows.

When extracting files (**-r** flag), archive members are selected, using the modified names, according to the user-specified pattern arguments as modified by the **-c**, **-n**, and **-u** flags. Then, any **-s** and **-i** flags modify, in that order, the names of the selected files. The **-v** flag writes the names resulting from these modifications.

When writing files to an archive file (**-w** flag), or when copying files, the files are selected according to the user-specified pathnames as modified by the **-n** and **-u** flags. Then, any **-s** and **-i** flags modify, in that order, the names resulting from these modifications. The **-v** flag writes the names resulting from these modifications.

If both the **-u** and **-n** flags are specified, **pax** does not consider a file selected unless it is newer than the file to which it is compared.

Modes

The action taken by **pax** depends on the presence of the **-r** and **-w** flags. Four combinations of these two flags are possible. The combinations are referred to as list, read, write, and copy modes. These modes correspond to the four forms of the command shown in **SYNOPSIS**.

List Mode

When neither the **-r** nor the **-w** flag is specified, the **pax** command writes the names of the members of the archive file read from the standard input file, with pathnames matching the specified patterns, to the standard output file. If a named file is a directory, the file hierarchy contained in the directory is also written.

You can specify the **pax** command without the **-r** or **-w** flags using the **-c**, **-d**, **-f**, **-n**, **-s**, and **-v** flags and with the *pattern* operand.

If neither the **-r** nor **-w** flags are present, **pax** lists the contents of the specified archive, one file per line. **pax** lists hard link pathnames as follows:

pathname == linkname

pax lists symbolic link pathnames as follows:

pathname -> linkname

In both of the preceding cases, *pathname* is the name of the file that is being extracted and *linkname* is the name of a file that appeared earlier in the archive.

If intermediate directories are necessary to extract an archive member, **pax** creates the directories with access permissions set as the bitwise inclusive OR of the values of **S_IRWXU**, **S_IRWXG**, and **S_IRWXO**, modified by the current file mode creation mask (umask).

Read Mode

When the **-r** flag is specified but the **-w** flag is not, **pax** extracts the members of an archive file read from the standard input file, and with pathnames matching the *pattern* operand if one is specified. If an extracted file is a directory, the file hierarchy contained in the directory is also extracted. The extracted files are created relative to the current file hierarchy. The **-r** flag can be specified with the **-c**, **-d**, **-f**, **-n**, **-s**, and **-v** flags and a *pattern* operand.

The access and modification times of the extracted files are the same as the archived files. The access permissions of the extracted files remain as archived unless affected by the user's default file creation mode. The **S_ISUID** and **S_ISGID** bits of the extracted files are cleared.

If intermediate directories are necessary to extract an archive member, the **pax** command creates the directories with access permissions set as the bitwise inclusive OR of the values of **S_IRWXU**, **S_IRWXG**, and **S_IRWXO**, modified by the current file mode creation mask (umask).

If the selected archive format supports the specification of linked files, it is an error if these files cannot be linked when the archive is extracted. **pax** informs you of the error and continues processing. Both the **ustar** and **cpio** formats support hard-linked files.

The ownership, access and modification times, and file mode of the restored files are discussed in the description of the **-p** flag.

Write Mode

When the **-w** flag is specified and the **-r** flag is not, **pax** writes the contents of the files specified by the *file* operands to the standard output file in an archive format. If no *file* operands are specified, a list of files to copy, one per line, is read from the standard input file. When the *file* operand specifies a directory, all of the files contained in the directory are written. The **-w** flag can be specified with the **-b**, **-d**, **-f**, **-i**, **-s**, **-t**, **-u**, **-v**, **-x**, and **-X** flags and with *file* operands.

If **-w** is specified but no files are specified, the standard input file is used. If neither **-f** nor **-w** is specified, the standard input file must be an archive file.

Copy Mode

When both the **-r** and **-w** flags are specified, **pax** copies the files specified by the *file* operand to the destination directory specified by the *directory* operand. If no file arguments are specified, a list of files to copy, one per line, is read from the standard input file. If a specified file is a directory, the file hierarchy contained in the directory is also copied. The **-r** and **-w** flags can be specified with the **-d**, **-i**, **-k**, **-l**, **-p**, **-n**, **-s**, **-t**, **-u**, **-v**, and **-X** flags and with the *file* operands. A *directory* operand must be specified.

Copied files are the same as if they were written to an archive file and subsequently extracted, except that there might be hard links between the original and the copied files.

For filesets that support OSS access control lists (ACLs), this command also copies any ACL entries associated with the file, so that the copied file has the same ACL entries as the source file. If the parent directory has an ACL that contains default ACL entries, the new directory inherits ACL entries and permissions as described in the **acl(5)** reference page.

Standard Input

The input file is named by the *archive* argument of the **-f** flag.. If the *archive* argument maps to a disk file, the input file must must be formatted in either **cpio** or **ustar** data interchange format. If the archive is read from a Guardian tape device, the tape file mounted on that device must be formatted in either **cpio** or **ustar** data interchange format.

The file **/dev/tty** is used to write tape mount messages.

In write mode, the standard input file is used only if no *file* operand is specified. The standard input file must be a text file containing a list of pathnames, one per line, without leading or trailing blank characters. In list and read modes, the standard input file must be an archive file. In all other cases, the standard input file is not used.

Standard Output

In write mode, the archive is written to the standard output file if **-f** is not supplied.

In list mode, the table of contents of the selected archive members is written to the standard output file as:

`%s\n,` *pathname*

If the **-v** flag is also specified in either write or list mode, the output has the following format:

`%s\n,` *ls -l listing*

where *ls -l listing* is the format used by the **ls** utility when the **-l** flag is used in that command.

Standard Error

When **-v** is specified in the read, write, or copy modes, **pax** writes the pathnames it processes to the standard error file using the following format:

`%s\n,` *pathname*

The pathname is written as soon as processing starts on the file or archive member and is sent to the standard error file.

When **-s** is specified in the read, write, or copy modes, and the replacement string has a trailing p, substitutions are written to the standard error file in the following format:

`%s>>%s\n,` *original-pathname,new-pathname*

Diagnostic messages are written to the standard error file.

Use on Guardian Objects

Unless otherwise noted, **pax** makes no distinction between Guardian and OSS files and treats them the same way.

Guardian files can be specified with the **/G** pathname convention. On output, the Guardian files are copied but their file attributes are not preserved. On input, files can be restored to the Guardian target, but the existing Guardian files are only overwritten if the **-W clobber** flag is specified. The file is restored as an unstructured Guardian file having the file code 180. Only Guardian files that are supported by the OSS function calls (**open( )**, **read( )**, **write( )**) are processed.

If the underlying function calls fail to operate on **/G** files, **pax** sends the error back to the caller together with a diagnostic message and, if possible, continues to process the other files.

Guardian Tape Devices

pax uses Guardian tape devices to read and write tape archives. Guardian tape devices are controlled by the Guardian tape process executing in the Guardian environment and do not behave the same as UNIX devices. The interaction between the tape process and tape device is transparent to the **pax** user.

If the **-W wait** flag is set, **pax** first issues a mount request to the Guardian tape process and then prints the following tape mount message to the terminal (**/dev/tty**):

```
Device not ready or tape is not mounted?
```

If **/dev/tty** is not available, **pax** does all of the following:

- Writes a diagnostic message to the standard output file

- Generates an EMS event message to the local collector process
- Waits indefinitely for the tape to be mounted

If the **-W wait** and **-W noprompt** flags are not specified, **pax** exits with the following message:

```
Device not ready, offline, or tape not mounted.
```

if the tape has not already been properly mounted on the drive. **pax** remains in a wait state until an unlabeled tape has been mounted on the tape drive correctly. Use the Guardian utility MEDIACOM. This utility can be invoked from the HP Tandem Advanced Command Language (TACL) command intrpreter or from the OSS shell through the **gtacl** command.

If errors occur that are related to the device or the mounted tape during the tape mount process, diagnostic messages are issued. You have a choice of correcting the errors and remounting the tape or canceling the tape mount request through MEDIACOM.

To access remote tape drives on a system that runs a release preceding D43, do the following:

1. Set the environment variable **GUARD_REMOTE_TAPE** to the remote tape devicename.
2. Invoke **pax** without specifying the **-f** flag.

Guardian Filename Transformation

Because of the syntactic differences between Guardian filenames and OSS filenames, the following behaviors can occur when an archive member is restored to a Guardian system.

A Guardian filename that is generated by the underlying OSS function calls for the file might contain illegal Guardian filename characters. As a result, the archive member cannot be created on the Guardian target and the restore fails.

In the name conversion process, OSS filenames that are longer than eight characters are truncated to the first valid eight characters. For example, an OSS filename like **abcde.fghi** is converted to the Guardian name ABCDEFGH. This can cause confusion and make identification of files difficult. Filenames that are similiar might be converted to the same filename. This results in the file overwriting a previously restored file.

Environment Variables

The following environment variables affect the execution of the **pax** command:

GUARD_REMOTE_TAPE
Specifies the Guardian device name of a tape device.

LANG Provides a default value for the internationalization variables that are NULL. If **LANG** is unset or NULL, the corresponding value from the implementation-specific default locale is used. If internationalization variables contain invalid settings, **pax** behaves as though none of the variables had been defined.

LC_ALL
When given a valid setting, overrides the values of all the other internationalization variables.

LC_COLLATE
Determines the locale for the behavior of ranges, equivalence classes and multi-character collating elements used in the pattern-matching expressions for the *pattern* operand, the basic regular expression for the **-s** flag, and the extended regular expression defined for the **yesexpr** locale keyword in the **LC_MESSAGES** variable.

LC_CTYPE
Determines the locale for the interpretation of bytes of text data as characters (for example, single-byte as opposed to multibyte characters in arguments).

LC_MESSAGES
Determines the locale for the processing of affirmative responses that should be used to affect the format and contents of diagnostic messages written to the standard error file.

LC_TIME
Determines the format and contents of the date and time strings to be displayed.

NLSPATH
Determines the location of message catalogs for processing **LC_MESSAGES**.

UTILSGE
Specifies that HP extensions to the / (root) directory should be omitted when the initial directory is the root directory and a recursive operation occurs in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE	Omit the **/E** directory.
NOG	Omit the **/G** directory.
NOG:NOE	Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command. This environment variable is ignored if you specify a wildcard character in or as the target of the **pax** command because wildcard characters are not recursive operations.

EXAMPLES

The following examples create or read archives in **ustar** format.

1. To copy the file hierarchy of the current directory to the tape mounted on Guardian tape device $TAPE using the blocking factor of 5120 bytes, enter:

   ```
   pax -w -f /G/tape -b 10b .
   ```

2. To copy the directory **olddir** to the directory **newdir**, enter:

   ```
   mkdir newdir
   pax -rw olddir newdir
   ```

3. To read the archive **a.pax**, with all files rooted in **/usr** in the archive extracted relative to the current directory, enter:

   ```
   pax -r -s ',^//*usr//*,,' -f a.pax
   ```

4. To restore files from a tape mounted on $TAPE to the Guardian target $VOL.SUBVOL, extracting only the **.c** files and overwriting any existing Guardian files with the same name, enter:

   ```
   cd /G/vol/subvol
   pax -r -f /G/tape -W clobber *.c
   ```

5. To archive all OSS files on the local node into the file named **paxfile**, enter:

   ```
   pax -wvf paxfile -W NOG -W NOE /
   ```

NOTE: If you specify a wildcard character, such as *****, instead of **/** in this command, the **-W NOG** and **-W NOE** flags are ignored.

6. If the **UTILSGE** environment variable is not set, to archive all files on the local node, including files in **/E** and **/G**, into the file named **paxfile**, enter:

   pax -wvf paxfile /

7. To archive all files on the local node, including files in **/E** and **/G** (ignoring the value of the **UTILSGE** environment variable) into the file named **paxfile**, enter:

   pax -wvf paxfile *

8. To extract and restore all OSS files in an archive named **paxfile** but skip files archived from **/G**, enter:

   pax -rvf paxfile -W NOG

CAUTIONS

Because of industry standards and interoperability goals, **pax** does not support the archival of files larger than 8 gigabytes.

DIAGNOSTICS

A diagnostic message is written to the standard error file and a nonzero exit value is returned (but processing continues) when **pax** cannot create a file or a link when reading an archive or when **pax** cannot preserve the user ID, group ID, or file mode bits when the **-p** flag is specified.

If the extraction of a file from an archive is prematurely terminated by a signal or error, **pax** might have only partially extracted the file or, if the **-n** flag was not specified, might have extracted a file that has the same name as that specified by the user but that is not the file the user wanted. In addition, the file modes of extracted directories might have incorrect modification and access times.

When appropriate privileges are required to set one of the access mode bits and if the user restoring the files from the archive does not have the appropriate privileges, the mode bits for which the user does not have privileges are ignored.

EXIT VALUES

The following exit values are returned by **pax**:

0 All files were processed successfully.

> 0 An error occurred.

NOTES

1. The **pax** command can fail with the error message `Name too long` when an attempt is made to archive a file with a filename longer than 100 characters. This message is displayed when the default USTAR format is used to create an archive. The command fails because the default USTAR format does not support filenames longer than 100 characters, in conformance with the 1990 edition of the POSIX Standard IEEE 1003.1. A practical workaround is to use the **pax** command with the **-x cpio** flag, because the **cpio** archive format supports filenames longer than 100 characters.

2. Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product. However, these flags are ignored if you specify a wildcard character in or as the target of the **pax** command.

3. The **pax** utility cannot append a file to an unlabeled tape. Each successive write to such a tape begins at the beginning of the tape.

   For example, if you issue the following commands from the shell:

   find xlog -print | pax -wv -f /G/TAPE -W norewind
   find xlog.bsm -print | pax -wv -f /G/TAPE -W norewind

   Then physically unload the tape, reload the tape, and enter:

   pax -rv -f /G/TAPE -W norewind

   You will see that the tape contains only the last file archived by the two commands above.

   To archive more than one file on an unlabeled tape, you must enter all of the commands within the same subshell. For example:

   (find xlog -print; find xlog.bsm -print) | pax -wv -f /G/tape

   The above command causes all the files printed by both the **find** commands to be put on tape because the **find** commands are executed in a subshell.

RELATED INFORMATION

Commands: **cp(1)**, **cpio(1)**, **ed(1)**, **pinstall(1)**, **tar(1)**.

Files: **cpio(4)**, **tar(4)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The **-W** flags and the **GUARD_REMOTE_TAPE** and **UTILSGE** environment variables are HP extensions to the XPG4 Version 2 specification.

NAME

pinstall - Extracts files from a **pax** (**ustar**) format archive file and copies them to the OSS file system

SYNOPSIS

pinstall [**-r**] [**-cdiknuv**] [**-f** *archive*] [**-p** *string*] ...
[**-s** *replstr*] ... [**-W** *options*] [*pattern*] ...

FLAGS

-c Matches all file or archive members except those specified by the *pattern* operand.

-d Does not create intermediate directories unless they are specifically named in the archive file. Similiarly, files are not extracted unless the required directories already exist.

-f *archive*

Specifies the OSS pathname of the input archive. The *archive* argument can be:

- The pathname for a Guardian tape device (such as **/G/tape** for the Guardian device named \NODE1.$TAPE on \NODE1)
- The pathname for a Guardian file (such as **/G/system/system/archfile** for the Guardian file $SYSTEM.SYSTEM.ARCHFILE)

If a tape device is used, **pinstall** displays the appropriate tape mount messages to make sure that the correct tapes are mounted on the tape drives before proceeding.

-f does not preserve file access times.

-i Interactively renames archive members. For each archive member matching the specified *pattern* operand, a prompt is written to the terminal. The prompt contains the name of the archive member that is to be renamed. The user's response is read from the terminal.

If the user's response to the prompt is blank, the archive member is skipped. If the response is a single period (dot), the archive member is processed with no modification to its name. Otherwise, the name is replaced with the contents of the response.

-k Prevents existing files from being overwritten.

-n Selects the first archive member that matches the specified *pattern* operand. No more than one archive member is matched for each *pattern* operand.

-p *string* Specifies one or more file characteristic options (privileges). The *string* argument must be a string specifying file characteristics to be retained or discarded on extraction. The *string* argument can consist of any combination of the following options:

a Do not preserve file access times.

e Preserve the user ID, group ID, file mode bits, file access times, and file modification times.

m Do not preserve file modification times.

o Preserve the user ID and group ID.

p Preserve the file mode bits.

Multiple **-p** flags are allowed in one command. If an option in *string* duplicates or conflicts with another option in *string*, the option given last takes precedence.

If neither the **e** nor the **o** option is specified in *string*, or if the user ID and group ID are not retained, the **S_ISUID** and **S_ISGID** bits of the access permission are not set for the extracted archive members. If the retention of any of these items fails, **pinstall** writes a diagnostic message to the standard error file. Failure to retain any of the items affects the exit value but does not cause the extracted file to be deleted.

File mode bits cannot be preserved if the target is a Guardian file. The user's default Guardian file permissions or the existing Guardian file permissions are used instead.

-r Reads an archive file. **-r** is required if the archive members are to be restored to the OSS file system.

-s *replstr* Modifies archive-member names specified by the *pattern* operand according to the substitution expression *replstr* and using the syntax of the **ed** command. The substitution expression has the following format:

-s /*old*/*new*/ [**gp**]

where (as in the **ed** command) *old* is a basic regular expression and *new* is the replacement string to be inserted in place of strings that match the regular expression. The string *new* can contain an **&** (ampersand), *n* (where *n* is a digit), back references, or subexpression matching. A **g** following *new/* replaces all matches. A **p** following *new/* causes successful substitutions to be written to the standard error file.

Multiple **-s** flags can be specified in one command. Multiple expressions are applied in the specified order, terminating with the first successful substitution. Archive-member names that are replaced by an empty string are ignored.

-u Ignores files that are older (that have a less recent file modification time) than archive members with the same names. An archive member with the same name as a file in the file system is extracted if the archive member is newer than the file.

-v In list mode, produces a table of contents (see **Standard Output** under **DESCRIPTION**). Otherwise, **-v** writes the archive member pathnames to the standard error file (see **Standard Error** under **DESCRIPTION**).

-W clobber
Allows the matching files from an archive to be restored to a Guardian target and to overwrite any existing Guardian target file with the same name. The files are restored as unstructured files, and Guardian file attributes are not preserved. Be sure you understand the problems this flag can create (refer to the subsection **Guardian Filename Transformation** in the **pax(1)** reference page) before using this flag.

-W norewind
Specifies that the tape be left on the tape drive without being rewound and unloaded after **pinstall** has finished reading from it. The tape drive is closed. The default action is for the tape to be rewound. The **-W nounload** flag is assumed when the **-W norewind** flag is specified. The **-W unload** and **-W norewind** flags cannot be specified together.

-W [**unload** | **nounload**]
Specifies whether the tape is to be unloaded or left on the tape drive after it has been rewound. The default action is **-W unload**.

-W [**wait** | **nowait**]
Specifies which of the following the **pinstall** command is to do if a tape is not mounted on the tape drive or the tape drive goes off line:

- Issue a prompt and wait for a tape to be mounted
- Exit immediately, without waiting

The default action is **-W wait**.

DESCRIPTION

The **pinstall** command invokes a Guardian process that copies archived files such as OSS utilities to the OSS file system. The **pinstall** command reads **ustar** data interchange format files, such as archive files that are created by the **pax** utility. **pinstall** can only read from an archive; it cannot create an archive.

The name of the archive is specified by the *archive* argument of the **-f** flag. If the *archive* argument is a tape device, **pinstall** reads from the archive tapes that are mounted on the tape device. If the **-r** flag is not specified, the names of all members of the archive file are listed to the output file or to the process's controlling terminal; no files are extracted.

The root directory must exist and be mounted when **pinstall** is invoked.

For the H06.23 or J06.12 RVUs, or systems that have installed SPR T8626H03_ADE, **pinstall** assigns a permission of 0755 to the intermediate directories it creates. For the H06.24 or later H-series RVUs, or J06.13 or later J-series RVUs, or systems that have installed SPR T8626H03_ADF or later, **pinstall** creates intermediate directories with access permissions set as the bitwise inclusive OR of the values of **S_IRWXU**, **S_IRWXG**, and **S_IRWXO**. The access permissions are set using the Guardian **DEFINE =OSS^UMASK**. If **DEFINE =OSS^UMASK** is not set or invalid, **pinstall** assumes the default value of 022.

The **-s** flag defines where extracted files are to be stored in the OSS file system. Without the **-s** flag, the files could be stored to the current file system, which could be the Guardian file system.

pinstall also issues tape mount messages and checks the tape drive to make sure that the correct tapes are mounted before proceeding. (See the **pax(1)** reference page for mount messages and possible diagnostic messages.)

Operands

pattern Specifies the pathnames for one or more archive members to be copied or listed. If no *pattern* is specified, the default action is to select all archive members.

Standard Input

The standard input file is specified by the *archive* argument of the **-f** flag.

Input Files

The input file is specified by the *archive* argument of the **-f** flag. If the input file maps to a disk file, then it must be written in **ustar** archive format. If the input file maps to a tape device, then the tape mounted on the tape device must be written in **ustar** archive format.

Standard Output

The table of contents of the archive members is written to the standard output file as:

`%s,` *pathname*

If the **-v** flag is also specified, the output has the following format:

`%s,` *-ls_-l_listing*

where *ls_-l_listing* is the format used by the **ls** command with the **-l** flag specified.

Output Files
Extracted files are copied to the OSS file system.

Standard Error
Diagnostic messages are written to the standard error file. Tape messages are written to the process identified as HOMETERM in the Guardian environment.

EXAMPLES

To extract files from the archive tape that is currently mounted on the tape drive $TAPE and restore them to the same directory from which they were originally copied, enter:

```
pinstall -r -f /G/tape
```

To override the default access permissions used by **pinstall** with a value of 023, enter the following before running **pinstall**:

```
add DEFINE =OSS^UMASK, class MAP, file #023
```

NOTES

pinstall uses unlabeled-tape processing. There is no mechanism to check for tape sequence in the case of multiple-reel archives.

On systems where the Distributed Systems Management/Distributed Configuration Manager (DSM/SCM) product is used to install HP product files from the ZOSSUTL subvolume and maintain those files in the OSS file system, do not use **pinstall** to install HP product files from the ZOSSUTL subvolume.

On systems where DSM/SCM is not used to install HP product files from the ZOSSUTL subvolume and maintain those files in the OSS file system, do not use **pinstall** on files with Guardian file identifiers that begin with ZFB or ZPG. Such files are created and maintained by DSM/SCM. Using **pinstall** on such files can duplicate installation effort or overwrite current versions of product files with obsolete versions.

DIAGNOSTICS

A diagnostic message is written to the standard error file and a nonzero exit value is returned (but processing continues) when **pinstall** cannot create a file or preserve the user ID, group ID, or file mode bits.

If the extraction of a file from an archive is permanently terminated for any reason, either **pinstall** might have extracted only a partially restored file or the file attributes of the extracted files might not be set correctly.

If appropriate privileges are required to set one of the file access mode bits and if the user restoring the files from the archive does not have the appropriate privileges, the file access mode bits for which the user does not have privileges are ignored.

EXIT VALUES

The following exit values are returned by **pinstall**:

0 (zero)	All files were processed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **copyoss(8)**, **cp(1)**, **ed(1)**, **pax(1)**.

STANDARDS CONFORMANCE

The **pinstall** command is an extension to the XPG4 Version 2 specification.

NAME

pname - Displays the OSS pathname of a Guardian file

SYNOPSIS

pname [**-s**] *filename*

FLAGS

-s Suppresses formatting and displays only the OSS pathname.

DESCRIPTION

The **pname** command displays the OSS filename for the Guardian file identified by *filename*. This command displays either of the following:

- The full pathname of a file in an OSS fileset when given the filename of that file in the Guardian file system
- The full OSS pathname for a file in the Guardian file system

Operands

filename Specifies the Guardian filename whose OSS pathname is to be displayed. The following special characters must be preceded with a backslash: dollar sign ($) and backslash (\).

EXAMPLES

1. The command:

   pname \$XPG.ZYQ00000.Z00005LN

   results in the following output:

   ```
   pname: $XPG.ZYQ00000.Z00005LN ---> /bin/pname
   ```

2. The command:

   pname -s \\KT22.\$XPG.ZYQ00000.Z00005LS

   results in the following output:

   ```
   /bin/pname
   ```

3. The command:

   pname -s \\KT22.\$SYSTEM.SYSTEM.FUP

   results in the following output:

   ```
   /G/system/system/fup
   ```

DIAGNOSTICS

The following error message is returned if an invalid filename or a nonexistent file is specified:

```
Failed with Guardian error: 4022
```

EXIT VALUES

The **pname** command returns the following values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **gname(1)**.

Miscellaneous: **filename(5)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

pr - Writes a file to standard output

SYNOPSIS

pr [**-adfFmprt**] [**-e**][*character*][*number*] [**-h** *header***]** [**-i**][*character*][*gap*] [**-l** *lines*] [**-n**][*character*][*width*] | [**-x**][*character*][*number*] [**-o** *offset*] [**-s**][*character*] [**-w** *width*] [*-column*] [*+page*] [*file ...*]

The **pr** command writes *file* to standard output. If you do not specify *file* or if *file* is **-**, **pr** reads standard input.

FLAGS

-a Displays multicolumn output across the page. (This flag is useful only in combination with the *-column* flag. It modifies *-column* so that columns are filled across the page in a round robin order.)

-d Doublespaces the output.

-e[*character*][*number*]
Expands tabs to byte positions *number*+1, 2**number*+1, 3**number*+1, and so on. The default value of *number* is 8. Tab characters in the input expand to the appropriate number of spaces to line up with the next tab setting. If you specify *character* (any character other than a digit) that character becomes the input tab character. The default value of *character* is the ASCII tab character.

-f Uses a formfeed character to advance to a new page. (Otherwise, **pr** issues a sequence of newline characters.) Pauses before beginning the first page if the standard output is a tty.

-F Uses a formfeed character to advance to a new page. (Otherwise, **pr** issues a sequence of newline characters.) Does not pause before beginning the first page if the standard output if a tty.

-h *header*
Displays *header* instead of the filename in the page header.

-i[*character*][*gap*]
Replaces white space wherever possible by inserting tabs to positions *gap*+1, 2**gap*+1, 3**gap*+1, and so on. The default value of *gap* is 8. If you specify *character* (any character other than a digit), that character becomes the output tab character. (The default value of *character* is the ASCII tab character.)

-l *lines* Sets the length of a page to *lines* lines (the default is 66). If *lines* is less than the sum of the header and trailer, **pr** suppresses both header and trailer, as if **-t** were in effect.

-m Combines and writes all files at the same time, with each file in a separate column. (This overrides the *-column* and **-a** flags.)

-n[*character*][*width*]
Provides *width*-digit line numbering (the default value of *width* is 5). The number occupies the first *width* positions of each column of normal output (or each line of **-m** output). If you specify *character* (any character, other than a digit), that character is added to the line number to separate it from whatever follows. (The default value of *character* is an ASCII tab character.)

-o *offset* Indents each line of output by *offset* byte positions (the default is 0). This is in addition to output width (see **-w**).

-p Pauses before beginning each page if the output is directed to a tty. (**pr** sounds the bell at the tty and waits for you to press <**Return**>.)

-r Does not display diagnostic messages if the system cannot open files.

-s[*character*]
Separates columns by the single *character* instead of by the appropriate number of spaces (the default for *character* is an ASCII tab character).

-t Does not display the 5-line identifying header and the 5-line footer. Stops after the last line of each file without spacing to the end of the page.

-x[*character*][*number*]
Same as **-n**.

-w *width* Sets the width of a line to *width* byte positions. If neither **-w** or **-s** are specified, the default is 72. If only **-s** is specified, the default is 512. Single column output is not truncated.

-*column* Produces the specified number of columns (the default value is 1). The **-e** and **-i** flags are assumed for multicolumn output. A text column never exceeds the width of the page (see **-l**).

+*page* Begins the display at the specified page number (the default value is 1).

DESCRIPTION

A heading that contains the page number, date, time, and the name of the file separates the output into pages.

Unless specified, columns are of equal width and separated by at least one space. Lines that are too long for the page width are shortened. If the standard output is a tty, **pr** does not display any error messages until it has ended. By default, the input is separated into 66-line pages, including the 5-line header and 5-line footer.

EXAMPLES

1. To print a file with headings and page numbers on the printer, enter:

   pr prog.c | print

   This inserts a page break in **prog.c**, starts each page with a heading, and sends the output to the **print** command. The heading consists of the date the file was last modified, the filename, and the page number.

2. To specify a title, enter:

   pr -h "MAIN PROGRAM" prog.c | print

   This prints **prog.c** with the title **MAIN PROGRAM** in place of the filename. The modification date and page number are still printed.

3. To print a file in multiple columns, enter:

   pr -3 word.lst | print

   This prints the file **word.lst** in three vertical columns.

4. To print several files side-by-side on the paper, enter:

   pr -m -h "Members, Visitors" member.lst visitor.lst | print

   This prints the files **member.lst** and **visitor.lst** side-by-side with the title **Members, Visitors**.

5. To modify a file for later use, enter:

   pr -t -e prog.c > prog.notab.c

   This replaces tab characters in **prog.c** with spaces and puts the result in **prog.notab.c**. Tab positions are at byte positions 9, 17, 25, 33, and so on. The **-e** tells **pr** to replace the tab characters; the **-t** suppresses the page headings.

RELATED INFORMATION

Commands: **cat**(1).

NAME

print - The shell output mechanism

SYNOPSIS

print [-**Rnprsu**[*n*]] [*argument* ...]

FLAGS

-R Ignores escape conventions, which are the same as those followed by the **echo** command. Prints all subsequent arguments and options, other than **-n**.

-n Does not add newlines to the output.

-p Causes the arguments to be written onto the pipe of the process spawned with **|&** instead of onto the standard output.

-r Ignores escape conventions, which are the same as those followed by the **echo** command.

-u Specifies a 1-digit file descriptor unit number *n* on which the output will be placed. The default is 1.

- Causes the arguments to be written to the history file instead of to standard output.

DESCRIPTION

The **print** command prints the string of numbers and letters specified as *argument* to the standard output (default) or to a pipe (if the **-p** flag is used).

If no flags are specified or if a - or a -- is used, the argument is printed to standard output as descibed for the **echo** command.

EXAMPLES

The following command prints the sentence "This is a test." to the screen:

print This is a test.

This is a test.

NOTES

The **print** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **echo**(1), **sh**(1).

NAME

printf - Writes formatted output

SYNOPSIS

printf *format* [*argument* ...]

DESCRIPTION

The **printf** command converts, formats, and writes its arguments to the standard output. The values specified by the *argument* argument are formatted under the control of the *format* argument.

The **LC_NUMERIC** environment variable affects the format of numbers written using the **e**, **E**, **f**, **g**, and **G** conversion characters.

The *format* argument is a character string that contains three types of objects:

- Plain characters that are copied to the output stream.
- The following escape sequences, which are copied to the output stream, causing the associated action to occur on display devices that are capable of the action.

  \\ Backslash

  \a Alert

  \b Backspace

  \f Formfeed

  \n Newline

  \r Carriage return

  \t Tab

  \v Vertical tab

  ddd Where *ddd* is a one-, two-, or three-digit octal number. These escape sequences are displayed as a byte with the numeric value specified by the octal number.

- Conversion specifications, each of which causes zero or more items to be fetched from the value argument list.

The *argument* argument is a list of one or more strings to be written to the standard output under the control of the *format* argument. These are treated as strings if the corresponding conversion character is **b**, **c**, or **s**; otherwise, the argument is evaluated as a C constant, as described by ISO/IEC 9899:1990, with the following extensions:

- A leading + (plus sign) or - (minus sign) is allowed.
- If the leading character is a **'** (single quotation mark) or **"** (double quotation), the value is the numeric value in the underlying code set of the character following the single quotation mark or double quotation mark.

The *format* argument is reused as often as necessary to satisfy the *argument* arguments. Any extra **c** or **s** conversion specifications are evaluated as if a null string *argument* were supplied; other extra conversion specifications are evaluated as if a zero *argument* were supplied.

Each conversion specification in the *format* argument has the following syntax:

1. A **%** (percent sign).
2. Zero or more options, which modify the meaning of the conversion specification. The option characters and their meaning are as follows:

   - The result of the conversion is left aligned within the field.

   + The result of a signed conversion always begins with a **+** (plus) or **-** (minus).

   blank If the first character of a signed conversion is not a sign, a blank is prefixed to the result. If both the **blank** and **+** options appear, then the **blank** option is ignored.

   # The value is converted to an alternative form. For **c**, **d**, **i**, **u**, and **s** conversions, the option has no effect. For **o** conversion, it increases the precision to force the first digit of the result to be a 0 (zero). For **x** and **X** conversions, a nonzero result has **0x** or **0X** prefixed to it, respectively. For **e**, **E**, **f**, **g**, and **G** conversions, the result always contains a radix character, even if no digits follow the radix character. For **g** and **G** conversions, trailing zeros are not removed from the result as they usually are.

   0 For **d**, **i**, **o**, **u**, **x**, **X**, **e**, **E**, **f**, **g**, and **G** conversions, leading zeros (following any indication of sign or base) are used to pad to the field width; no space padding is performed. If the **0** (zero) and **-** options appear, the **0** (zero) option is ignored. For **d**, **i**, **o**, **u**, **x**, and **X** conversions, if a precision is specified, the **0** (zero) option is ignored.

3. An optional decimal digit string that specifies the minimum field width. If the converted value has fewer characters than the field width, the field is padded on the left to the length specified by the field width. If the left-adjustment option is specified, the field is padded on the right.
4. An optional precision. The precision is a **.** (dot) followed by a decimal digit string. If no precision is given, it is treated as **0** (zero). The precision specifies:
   - The minimum number of digits to appear for the **d**, **i**, **o**, **u**, **x**, or **X** conversions.
   - The number of digits to appear after the radix character for the **e** and **f** conversions.
   - The maximum number of significant digits for the **g** conversion.
   - The maximum number of bytes to be printed from a string in the **s** conversion.
5. A character that indicates the type of conversion to be applied, as follows:

   % Performs no conversion. Prints a **%** (percent sign).

   b Accepts a value as a string that may contain backslash-escape sequences. Bytes from the converted string are printed until the end of the string or the number of bytes indicated by the precision specification is reached. If the precision is omitted, all bytes until the first null character are printed.

The following backslash-escape sequences are supported:

- **\O***ddd*, where *ddd* is a zero-, one-, two-, or three-digit octal number that is converted to a byte with the numeric value specified by the octal number.
- The escape sequences previously listed under the description of the *format* argument. These are converted to the individual characters they represent.
- The **\c** sequence, which is not displayed and causes the **printf** command to ignore any remaining characters in the string parameter containing it, any remaining string parameters, and any additional characters in the *format* argument.

c Accepts an integer value and converts it to an unsigned character. The resulting byte is printed.

d, **i** Accepts an integer value and converts it to signed decimal notation in the style [-]*dddd*. The precision specifies the minimum number of digits to appear. If the value being converted can be represented in fewer digits, it is expanded with leading zeros. The default precision is 1. The result of converting a 0 (zero) value with a precision of 0 (zero) is a null string. Specifying a field width with a 0 (zero) as a leading character causes the field width value to be padded with leading zeros.

e, **E** Accepts a float or double value and converts it to the exponential form [-] *d***.***dd***e +|-** *dd*. There is one digit before the radix character (shown here as the decimal point), and the number of digits after the radix character is equal to the precision specification. The **LC_NUMERIC** locale category determines the radix character to use in this format. If no precision is specified, then six digits are output. If the precision is **0** (zero), then no radix character appears. The **E** conversion character produces a number with **E** instead of **e** before the exponent. The exponent always contains at least two digits. However, if the value to be printed requires an exponent greater than two digits, additional exponent digits are printed as necessary.

f Accepts a float or double value and converts it to decimal notation in the format [-] *ddd***.***ddd*. The number of digits after the radix character (shown here as the decimal point) is equal to the precision specification. The **LC_NUMERIC** locale category determines the radix character to use in this format. If no precision is specified, then six digits are output. If the precision is **0** (zero), then no radix character appears.

g, **G** Accepts a float or double value and converts it in the style of the **f** or **e** conversion characters (or **E** in the case of the **G** conversion), with the precision specifying the number of significant digits. Trailing zeros are removed from the result. A radix character appears only if it is followed by a digit. The style used depends on the value converted. Style **g** results only if the exponent resulting from the conversion is less than -4 or it is greater than or equal to the precision.

o Accepts an integer value and converts it to unsigned octal notation. The precision specifies the minimum number of digits to appear. If the value being converted can be represented in fewer digits, it is expanded with leading zeros. The default precision is 1. The result of converting a 0 (zero) value with a precision of **0** (zero) is a null string. Specifying a field width with a 0 (zero) as a

leading character causes the field width value to be padded with leading zeros. An octal value for field width is not implied.

s Accepts a value as a string, and bytes from the string are printed until the end of the string is encountered or the number of bytes indicated by the precision is reached. If no precision is specified, all characters up to the first null character are printed.

u Accepts an integer value and converts it to unsigned decimal notation. The precision specifies the minimum number of digits to appear. If the value being converted can be represented in fewer digits, it is expanded with leading zeros. The default precision is 1. The result of converting a 0 (zero) value with a precision of **0** (zero) is a null string. Specifying a field width with a 0 (zero) as a leading character causes the field width value to be padded with leading zeros.

x, **X** Accepts an integer value and converts it to unsigned hexadecimal notation. The letters **abcdef** are used for the **x** conversion, and the letters **ABCDEF** are used for the **X** conversion. The precision specifies the minimum number of digits to appear. If the value being converted can be represented in fewer digits, it is expanded with leading zeros. The default precision is 1. The result of converting a 0 (zero) value with a precision of **0** (zero) is a null string. Specifying a field width with a 0 (zero) as a leading character causes the field width value to be padded with leading zeros.

If the result of a conversion is wider than the field width, the field is expanded to contain the converted result. No truncation occurs. However, a small precision may cause truncation on the right.

EXAMPLES

The following command

```
printf "%5d%4d\n" 1 21 321 4321 54321
```

produces the following output:

```
    1  21
  3214321
54321   0
```

The *format* argument is used three times to print all of the given strings. The 0 (zero) is supplied by the **printf** command to satisfy the last **%4d** conversion specification.

RELATED INFORMATION

Commands: **read(1).**

Functions: **printf(3).**

NAME

ps - Displays process status

SYNOPSIS

ps [**-aA**] [**-defl**]
[**-G** *grouplist*]
[**-o** *format*] ...
[**-p** *proclist*]
[**-t** *termlist*]
[**-U** *userlist*]
[**-g** *grouplist*]
[**-n** *namelist*]
[**-u** *userlist*]

or

ps -W
all
[**,node=***system_name*]
[**,gpri=***Gpriority*]
[**,prog=**[/**E**/*systemname*]/**G**/*volume*/*subvolume*/*fileid*]
[**,term=[/E**/*system_name*]/**G**/*terminal_process_name*]
[**,guser={** *groupname***.***username* | *groupid***:***userid***}**]
[**,detail**]
. . .

or

ps -W
cpu=*processor_number*
[**,pin=***process_identifier_number*]
[**,node=***system_name*]
[**,detail**]
. . .

or

ps -W
name=[/**E**/*system_name*]/**G**/*process_name* . . .
[**,node=***system_name*]
[**,detail**]
. . .

or

ps -W files *pid*

or

ps -W loaded *filename*

FLAGS

The following flags can be used with the **ps** command:

-a Writes information about all processes, except the process group leaders and processes not associated with a terminal, to the standard output file.

-A Writes information about all processes.

-d Writes information about all processes, except the process group leaders, to the standard output file.

-e Writes information about all processes to the standard output file.

-f Generates a full listing that includes the fields **C**, **CMD**, **PID**, **PPID**, **STIME**, **TIME**, **TTY**, and **UID**.

-G *grouplist*
Writes information for processes whose real group ID numbers are given in the *grouplist* option. The *grouplist* is a list of process-group identifiers separated from one another by a comma or one or more spaces.

-l Generates a long listing, which includes the fields **ADDR**, **C**, **CMD**, **F**, **NI**, **PID**, **PPID**, **PRI**, **S**, **SZ**, **TIME**, **TTY**, **UID**, and **WCHAN**.

-n *namelist*
Specifies the name of an alternative system *namelist* file in place of the default file.

Note that the concept of an alternative system is meaningless in the Guardian or OSS environment. As a result, any argument is accepted for *namelist*. The **-n** flag is thus the equivalent of a **noop**, and **-n** is ignored by the **ps** utility. Nevertheless, the requirement to specify an argument to **-n** is enforced.

-o *format*
Specifies a list of format specifiers to describe the output format. Multiple **-o** flags can be specified; the format specification is interpreted as the concatenation of all the *format* options.

-p *proclist*
Displays only information about processes whose process numbers are specified in the *proclist* argument. The *proclist* argument is either a list of process ID numbers or a list of process ID numbers enclosed in " " (double quotes) and separated from one another by a comma or a space.

-t *termlist*
Displays only information about processes associated with the terminals listed in the *termlist* argument. The *termlist* argument is a list of terminal identifiers or a list of terminal identifiers separated from one another by a comma or one or more spaces.

Terminal identifiers must be in one of two forms:

- The device's filename
- The device's digit identifier, if the device's filename begins with **tty**

-u *userlist*
Displays only information about processes whose user ID numbers or login names are specified in the *userlist* argument. The *userlist* argument is either a list of user IDs or a list of user IDs enclosed in " " (double quotes) and separated from one another by a comma or one or more spaces or both. Because of the way the shell treats spaces and tabs, you need to quote space-separated lists.

In the listing, the numerical user ID is written unless the **-f** flag is used, in which case the login name is written.

-U *userlist*

Writes information for processes whose real user ID numbers or login names are given in the *userlist* argument. The *userlist* argument is either a list of user IDs or a list of user IDs enclosed in **" "** (double quotes) and separated from one another by a comma or one or more spaces or both. Because of the way the shell treats spaces and tabs, you need to quote space-separated lists.

HP Extensions

The **-W** flag and its options are HP extensions to the **ps** command. They write information about Guardian and OSS processes.

The **-W** flag can only be specified with the **-W** options. It cannot be specified with the other **ps** command flags. When the **-W** flag is is used with other flags, the usage message is displayed and **ps** exits in error.

Information displayed by the **-W** flag is formatted using Guardian environment display conventions, rather than UNIX conventions.

When none of the **-W** flags are used (or when no other options are used with them), only OSS processes are candidates for display.

When the **-W** flag and its options are used, all processes (both OSS and Guardian) are candidates for display.

All flag options used with the **-W** flag are used to select processes. Specifying any option causes the **ps** command to ignore the default list of options and select the processes represented by the inclusive "or" of all the selection criteria options. When specifying more than one option, do not use spaces after the commas.

If the **-W** flag is specified without one of its valid options (described below), the **ps** command displays the usage message and exits in error.

-W all Writes information about all Guardian and OSS processes. If the **node=** option is omitted, the information displayed is for the local node. If any of the options **gpri**, **prog**, **term**, or **guser** are also specified, only information about the matching subset of Guardian and OSS processes is displayed.

-W cpu=*processor_number*

If **pin=***process_identifier_number* is omitted, writes information about all Guardian and OSS processes that reside on the processor specified by the *processor_number* value. If **pin=***process_identifier_number* is also specified, specifying the **-W cpu** flag restricts the output to the process with the specified operating system CPU,PIN number on the specified processor. The *processor_number* value must be within the range 0 through 15.

-W detail

Displays detailed information about processes.

-W files *pid*

Displays the absolute pathnames of all loadfiles used by the process with the OSS process identifier (PID) specified as *pid*. Each pathname is marked with its type of loadfile (program, dynamic-link library (DLL), or other library).

-W gpri=*Gpriority*

Displays information about processes whose execution priority is equal to the *Gpriority* value. The specified priority is the Guardian priority, where 0 is the minimum and 255 is the maximum. If the specified priority is less than 0 or greater than 255, the usage message is displayed and the **ps** command exits in error.

-W guser=*groupname***.***username* | *groupid***:***userid*
Displays information on processes created by the user specified by *groupname***.***username* or *groupid***:***userid*.

-W loaded *pathname*
Displays the OSS process identifiers (PIDs) for all processes that have copies of the loadfile specified by *pathname*.

-W name=[**/E/***system_name*]**/G/***process_name*
Writes information about the process identified by the path **/E/***system_name***/G/***process_name*. The value specified for *process_name* must begin with the character immediately following the **$** in the Guardian process name.

-W node=*system_name*
Writes information about processes based on other selection criteria residing in the remote system identified by *system_name*. Do not specify the backslash (\) when entering the *system_name* value.

The equivalent to the TACL command STATUS *system_name* is:

ps -W node=*system_name* **-W all**

or

ps -W node=*system_name,* **all**

The **-W node=** option cannot be specified alone. It must be accompanied by one of the other options. For example:

ps -W node=foxii -W cpu=3

ps -W node=tsii,-W name=/G/cmon

-W prog=[/E/*system_name*] **/G/***volume/subvolume/fileid*
Displays information about processes that are executing the specified Guardian file. Remember to specify the Guardian filename using OSS pathname conventions.

-W term=[/E/*system_name*]**/G/***terminal_process_name*
Displays information on processes that are associated with the specified Guardian terminal.

DESCRIPTION

The **ps** command displays the current process status.

While the **ps** command provides a fairly accurate snapshot of the system, **ps** cannot begin and finish a snapshot as fast as some processes change state. At times there may be minor discrepancies between the **ps** command's output and the actual state of a process.

The state is given by a sequence of letters, for example, **RWN**. The first letter indicates the status of the process:

R Runnable process

U Uninterruptible sleeping process

S Process sleeping for less than about 20 seconds

I Idle (sleeping longer than about 20 seconds) process

T Stopped process

H Halted process

The second character, if any, indicates additional state information:

W Process is swapped out (shows a blank space if the process is loaded, or in core).

> Process has specified a soft limit on memory requirements and is exceeding that limit; such a process is (necessarily) not swapped.

The third character, if any, indicates whether a process is running with altered processor scheduling priority:

N Process priority is reduced.

< Process priority has been artificially raised.

\+ Process is a process group leader with a controlling tty.

Environment Variables

The following environment variables affect the execution of the **-ps** command:

COLUMNS
Overrides the default horizontal screen size. **COLUMNS** determines the number of text columns to display. The output wraps if **COLUMNS** exceeds the system screen size.

LANG Provides a default value for internationalization variables that are null. If **LANG** is unset or null, the corresponding value from the implementation-specific default locale is used. If internationalization variables contain invalid settings, the **ps** command behaves as though none of the variables are defined.

LC_ALL
Overrides the values of all other internationalization variables.

LC_CTYPE
Determines the locale for interpreting bytes of text data as characters (for example, single-byte versus multiple-byte characters in arguments).

LC_MESSAGES
Determines the locale for defining the format and contents of diagnostic messages that are written to the standard error file and of information messages that are written to the standard output file.

LC_TIME
Determines the format and contents of the date and time strings that are to be displayed.

NLSPATH
Determines the location of message catalogs for processing the **LC_MESSAGES** variable.

Standard Output

When the **-o** and **-W** flags are not specified, the standard output format is as follows:

- The column headings and descriptions of the columns in a **ps** listing are described
- The letters **f** and **l** indicate the flag (full or long) that causes the corresponding heading to appear
- The suboption **all** specifies that the heading always appears

The **-o** option allows the output format to be specified by the user. The format specification must be a list of names presented as a single argument, with the names separated by a space or a comma. Each variable has a default header that is displayed. The default header can be overwritten by appending an equal sign and the new text of the header to the default header. The characters in the argument following the equal sign are used as the header text. The fields are written in the order specified on the command line and are arranged in columns in the output. The field widths are selected automatically and are at least as wide as the header text. If all the header text fields are null, no header line is written.

The following names are recognized in the OSS environment:

args The command being executed with all of its arguments. The display is a string. The string is not truncated to match the field width length. However, the string is truncated if it wraps around the screen. The string is the argument list that was passed to the command when it was started. Any modifications that a process might do to its argument list are not reflected in the output of the **ps** command.

comm The name of the command being executed, displayed as a string. The name is truncated in the display if its value is larger than 80 bytes.

etime The elapsed time since the process was started. The value is displayed in the form [[*dd*-]*hh*:]*mm*:*ss*, where *dd* represents the number of days, *hh* represents the number of hours, *mm* represents the number of minutes, and *ss* represents the number of seconds. The *dd* field is a decimal integer. The *hh*, *mm*, and *ss* fields are two-digit decimal integers padded on the left with zeros.

group The effective group ID of the process. This is the textual group ID.

nice The decimal value of the system scheduling priority of the process.

pcpu The ratio of the processor time used recently to the available processor time during the same period. The value is expressed as a percentage. The meanings of "recently" and "processor time" are implementation-defined. This field has no meaning in the Guardian or OSS environments, and a hyphen is displayed instead of the field value.

pgid The decimal value of the process group ID.

pid The decimal value of the process ID.

ppid The decimal value of the parent process ID.

rgroup The real group ID of the process. This is the textual group ID.

ruser The real user ID of the process. This is the textual user ID.

time The cumulative CPU time of the process in the form [[*dd*-]*hh*:]*mm*:*ss*. The *hh*, *mm*, and *ss* fields have the same definitions that they do for **etime**.

tty The name of the controling terminal of the process (if any). If no controlling terminal is present, a question mark (**?**) is displayed

user The effective user ID of the process. This is the textual user ID.

vsz The size of the process in virtual memory in kilobytes as a decimal integer.

If you use the **-l** and **-A** flags together, a list of zombie processes also appears at the end of the display (similar to the following):

```
F   S    UID  PID  PPID  C PRI NI ADDR  SZ WCHAN TTY  TIME   CMD
 .
 .
 .
-   0000 xx    xx   xx    -  -  -   -    -   -     -    -  <defunct>
```

Format Specifiers

The following list contains all format specifiers that can be used with the **ps** command. The default header for each specifier appears in parentheses. Most of the headers can also be used as format specifier synonyms.

args (COMMAND)
The command with all its arguments as a string. The **ps** command truncates this value to the field width.

comm (COMMAND)
The name of the command being executed (*argv***[0]** value) as a string.

etime (ELAPSED)
Elapsed time since the process was started.

group (GROUP)
Effective group ID of the process.

nice (NI) Process scheduling increment for the process.

pcpu The ratio of processor time used recently to processor time available in the same time period. The meaning of the term "recently" as well as the meaning of "processor time available" is implementation defined. This field has no meaning in the Guardian or OSS environment.

pgid (PGID)
Process group ID of the process.

pid (PID)
Process ID of the process.

ppid (PPID)
Parent process ID of the process.

rgroup (RGROUP)
Real group ID of the process.

ruser or **runame** (RUSER)
Real user ID of the process.

user or **uname** (USER)
Real user name of the process.

time The cumulative processor time of the process in the form `[[dd-] hh: ] mm:ss`.

tty The name of the controlling terminal of the process. If no controlling terminal is present a question mark (**?**) is displayed.

vsz The size of the process in virtual memory in Kbytes, as a decimal integer.

EXAMPLES

1. To list all your processes, enter:

   ps

2. To list all processes except kernel processes, enter:

   ps -e

3. To list processes owned by specific users, enter:

   ps -f -l -uos.jim,software.jane,super.super

4. To display only the **pid**, **user**, and **comm** information for all processes, enter:

   ps -o pid,user,comm -e

5. The command:

   ps -l

   results in the following display:

```
F   S    UID    PID       PPID C PRI NI ADDR SZ WCHAN TTY TIME  CMD
011 8102 123456 318570500 1    - 106 -  1,27 -  -     tty 05:17 /bin/ps
```

6. The command:

   ps -W name=/G/cmon -W node=kt22

   results in the following display:

```
SYSTEM_NAME
\KT22
PROCESS BK CPU,PIN GPRI PFR %WT USERID  PROGRAMFILE          HOMETERM
/G/cmon    0,51    170  P   001 255,255 /G/system/cmon/cmon /G/osp
/G/cmon B  4,139   170  P   001 255,255 /G/system/cmon/cmon /G/osp
```

7. Each of the commands:

   ps -W cpu=3 -W pin=76 -W detail

   ps -W cpu=3, pin=76 -W detail

   ps -W cpu=3, pin=76, detail

results in the following display:

```
SYSTEM_NAME
foxii
PROCESS BK CPU,PIN GPRI PFR %WT USERID  PROGRAMFILE          HOMETERM
          3,76    150  P   001 255,255 /G/system/sys02/zexp /G/tl1/111
PROCESS_TIME GMOMJOBID SWAP_FILENAME      EXTENDED_SWAP
0:0:0.041    0         /G/SYSTEM/#0121577 /G/system/#0121578
PROCESS_STATE                      CREATION_TIME
no messages, forced low, runnable  March 26, 1993 16:16:34.408
```

8. The command:

   ps -Wloaded /G/system/sys01/zcresrl

   results in a display similar to the following:

```
       PID
 703594543
 938475558
 183500819
 821035021
1055916036
 988807201
```

9. The command:

   ps -Wfiles 1055916036

   results in a display similar to the following:

```
FILES
/bin/util/myprog (PROGRAM)
/usr/lib/mylib (DLL)
/G/SYSTEM/SYS01/zcresrl (SRL)
```

NOTES

The status of a system can change between the time **ps** polls a system for status and the time the information is displayed on your terminal. Thus the **ps** command gives a snapshot of a system's status that may be out of date by the time it is displayed.

The **COLUMNS** variable overrides the system-selected horizontal screen size.

EXIT VALUES

The **ps** command returns the following values:

0 Completion was successful.

>0 An error occurred.

RELATED INFORMATION

Commands: **kill(1)**, **nice(1)**.

STANDARDS CONFORMANCE

The **-W** flags are HP extensions to the XPG4 Version 2 specification.

NAME

pwd - Displays current directory pathname

SYNOPSIS

pwd

DESCRIPTION

The **pwd** command writes to standard output the full pathname of your current directory from the root directory. All directories are separated by a / (slash). The root directory is represented by the first /, and the last directory named is your current directory.

The OSS shell contains a built-in command named **pwd**, that functions in the same way as the regular OSS command named **pwd**, except that a new shell process is started for each execution of the regular form of **pwd**. The shell built-in version is the default. A new subshell is not started for each invocation of the shell built-in form of **pwd**. Both the regular form and the shell built-in form of **pwd** are described in this reference page.

NOTES

The OSS **pwd** command has both a shell built-in version and a regular version. The two versions have the same features and functionality. The only difference between the two versions is that the shell built-in version does not start a new shell process when it is invoked. Both versions are described in the reference page for **pwd**. The shell built-in version is the default. To specify the regular version use the full pathname: **/bin/pwd** For more information about shell built-in commands refer to the reference page for **sh(1).**

RELATED INFORMATION

Commands: **cd**(1), **sh**(1).

Functions: **stat**(2).

NAME

read - Reads one line from the standard input file

SYNOPSIS

read [**-r**] *variablename* ...

FLAGS

-r Specifies that the **read** command treat a \ (backslash) character as just part of the input line, not as a control character.

DESCRIPTION

The **read** command reads one line from the standard input file and assigns the values of each field in the input line to a shell variable.

The OSS shell contains a built-in command named **read** that functions in the same way as the regular OSS command named **read**, except that a new shell process is started for each execution of the regular OSS command **read**. The shell built-in version of **read** is the default. Both the regular version and the shell built-in version of the **read** command are described in this reference page.

Operands

variablename Specifies the name of a shell variable.

The first variable specified by *variablename* is reached if the line of standard corresponding shell variables specified is given the value of all the remaining fields. If there are fewer fields than shell variables, the remaining shell variables are set to empty strings. The setting of shell variables by the **read** command affects the current shell execution environment. The environment variable **IFS** determines the internal field separators used to delimit the fields.

The **read** command prompts for a continuation line when the shell readsan input line ending with a \ (backslash), unless the **-r** flag is specified.

Environment Variables

IFS Determines the internal field separators used to delimit the fields.

EXAMPLES

The following is a command sequence to print a file with the first field of each line moved to the end of the line:

```
while read -r xx yy
do
        printf "%s %s\n" "$yy" "$xx"
done < inputfile
```

NOTES

The OSS **read** command has both a shell built-in version and a regular version. The two versions have the same features and function. The only difference between the two versions is that the shell built-in version does not start a new shell process when it is invoked. Both versions are described in this reference page. The shell built-in version is the default version. To specify the regular version, use the full pathname: **/bin/read**. For more information about shell built-in commands, refer to the **sh(1)** reference page.

EXIT VALUES

The **read** command returns the following exit values:

0

>0 An End-of-File was detected or an error occurred during execution.

RELATED INFORMATION

Commands: **printf(1)**.

NAME

readonly - Sets environment variables as read only

SYNOPSIS

readonly [*name*[=*value* ...]]

readonly -p

FLAGS

-p Writes to standard output the names and values of all read-only variables.

DESCRIPTION

The names and assigned values of environmental variables specified as *name* and *value* are marked as read only and cannot be changed by subsequent assignment.

If **-p** is specified, **readonly** displays a list of the names and values of all read-only variables. The shell formats the output, including the proper use of quoting, so that is it suitable for reinput to the shell as commands that achieve the same attribute-setting results. The **-p** flag allows portable access to the values that can be saved and then later restored by using, for example, a **.** (dot) script.

EXAMPLES

1. In the following series of commands the variable **x** is made read only and the read only status is tested:

   ```
   x = 3
   readonly x
   x = 5
   ```

   The last command in the above series results in the following error message:

   ```
   x: is read only
   ```

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.
- Words in the format of a parameter assignment are expanded with the same rules as a parameter assignment. This means that ~ (tilde) substitution is performed after the = (equal sign). Word splitting and filename generation are not performed.

The **readonly** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh(1)**.

NAME

reset_define - Restores a DEFINE's attributes to their initial settings

SYNOPSIS

reset_define {*attribute-name*}...

DESCRIPTION

The **reset_define** command is specific to OSS and a built-in command to the OSS shell. It is similiar to the TACL RESET DEFINE command. The **reset_define** command restores the attributes of one or more DEFINEs to their initial settings. If you reset a default attribute, the default value is restored. (Optional attributes cannot be reset, because they have no initial value.) Refer to the RESET DEFINE command in the *TACL Reference Manual* for related information.

attribute-name

Specifies the attribute whose setting is to be restored. Valid attribute names are described in the *TACL Reference Manual*.

Environment Variables

LANG Determines the locale to use for the locale categories when neither the **LC_ALL** variable nor the corresponding environment variable (beginning with **LC_**) specify a locale.

LC_ALL

Determines the locale to be used to override any values for locale categories specified by the **LANG** variable or any environment variable whose name begins with **LC_**.

LC_CTYPE

Determines the locale for interpretation of bytes of text data as characters (for example, single-byte as opposed to multibyte characters in arguments).

LC_MESSAGES

Determines the locale that should be used to affect the format and contents of diagnostic messages written to the standard error file and of information messages written to the standard output file.

EXAMPLES

1. To reset and display the entire working attribute set that specifies a CLASS TAPE DEFINE, enter:

   reset_define class=tape

   This command might result in the following display:

   ```
   CLASS       =TAPE
   VOLUME      =(25436, 75444, 23121)
   LABELS      =ANSI
   REELS       =
   OWNER       =
   FILESECT    =
   FILESEQ     =
   FILEID      =
   RETENTION   =
   EXPIRATION  =
   GEN         =
   VERSION     =
   RECFORM     =
   BLOCKLEN    =
   RECLEN      =
   ```

```
DENSITY     =
USE         =IN
DEVICE      =\KT22.$TAPE
EBCDIC      =
MOUNTMSG    =
SYSTEM      =
TAPEMODE    =
```

EXIT VALUES

The following exit values are returned:

0 DEFINE attribute values were reset successfully.

>0 An error occurred.

NOTES

The **reset_define** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **add_define(1)**, **del_define(1)**, **info_define(1)**, **set_define(1)**, **show_define(1)**.

STANDARDS CONFORMANCE

The **reset_define** command is an HP extension to the XPG4 Version 2 specification.

NAME

return - Returns a shell function to its invoking script

SYNOPSIS

return [*n*]

DESCRIPTION

The **return** command causes a shell function to return to the script that started it, with the return status specified by the argument *n*.

If the argument *n* is not specified, the return status is that of the last command executed. If the **return** command is executed while not in a function or in a . (dot) script, it has the same effect as an **exit** command.

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **return** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

NAME

rm - Removes (unlinks) files or directories

SYNOPSIS

rm [-**firR**] [**-W NOG**] [**-W NOE**] *file* ...

FLAGS

-f Does not prompt before removing a file that does not have write permission set, and does not display an error message if a specified file does not exist. If you specify both the **-f** and **-i** flags when invoking the **rm** command, the flag that is specified last on the command line takes effect.

-i Prompts you before deleting each file (interactive). When you use both the **-i** and **-r** flags, the **rm** command also asks if you want to examine directories. If you specify both the **-f** and **-i** flags when invoking the **rm** command, the flag that is specified last on the command line takes effect.

-r Permits recursive removal of directories and their contents.

-R Permits recursive removal of directories and their contents (same as **-r**).

HP Extensions

-W NOG Specifies that the **/G** directory should be omitted when the initial directory is root and a recursive flag (**-R** or **-r**) is used. This flag is ignored when the initial directory is not **/**, **/E**, or **/E**/*system* or when recursion does not occur.

-W NOE Specifies that the **/E** directory should be omitted when the initial directory is root and a recursive flag (**-R** or **-r**) is used. This flag is ignored when the initial directory is not root or when recursion does not occur.

Specify both the **-W NOG** and **-W NOE** flags to omit both the **/G** and **/E** directories.

DESCRIPTION

The **rm** command removes the entries for the specified files from a directory.

If the *file* argument is of the directory type, the following steps are taken:

1. If neither the **-R** or **-r** flag is specified, **rm** writes a diagnostic message to standard error, does nothing further with the file *file*, and goes on to any remaining files.
2. If the **-f** flag is not specified and either of the following is true, **rm** writes a prompt to standard error and reads a line from standard input:
   - The permissions of the file *file* do not permit writing, and standard input is a terminal.
   - The **-i** flag is specified.

   If the response is affirmative, **rm** does nothing further with the current file and goes on to any remaining files. (The same actions are taken if the **-f** flag is specified and *file* is *not* of the directory type.)
3. For each entry contained in the file *file*, other than **.** (dot) or **..** (dot dot) entries. Steps 1 through 3 are taken with the entry as if it were a *file* argument.

If an entry is the last link to a file, it is destroyed. To remove a file, you must have write permis-

sion for its parent directory, but you need neither read nor write permission for the file itself.

If a file has no write permission and standard input is a tty:

- If the system does not support OSS ACLs, the **rm** command displays the file permission code and reads a line from standard input. If that line begins with **y**, or the locale's equivalent of a **y**, **rm** deletes the file. If the response is anything else, the **rm** command does nothing to that file and continues with the next specified file.
- If the system supports OSS ACLs, the **rm** command displays the file name and the file permissions and reads a line from standard input. If that line begins with **y**, or the locale's equivalent of a **y**, **rm** deletes the file. If the response is anything else, the **rm** command does nothing to that file and continues with the next specified file.

  If the **-f** option is used or the standard input is not a tty, **rm** does not display any prompts.

  If the file has optional ACL entries, the **rm** command displays a plus sign (+) after the file permissions. The permissions shown by the **rm** command summarize the *st_mode* values returned
  by the **stat( )** function (see the **stat(2)** reference page). If you execute the **rm** command remotely from a system that does not support OSS ACLs, **rm** does not display a plus sign (+) for files that have optional ACLs.

  For more information about ACLs, see the **acl(5)** reference page.

The **-i** flag causes **rm** to prompt and read the standard input even if the standard input is not a terminal. In the absence of **-i**, however, **rm** does not prompt when the standard input is not a terminal.

Environment Variables

The following environment variables affect the execution of the **rm** command:

LC_MESSAGES
Determines the locale's equivalent of **y** or **n** (for yes/no queries).

UTILSGE Specifies that HP extensions to the root directory should be omitted when the initial directory is root and a recursive operation occurs in an OSS shell command. Application programs that test this variable might also honor its settings.

The **UTILSGE** value can be any of the following:

NOE	Omit the **/E** directory.
NOG	Omit the **/G** directory.
NOG:NOE	Omit both the **/G** and **/E** directories.

The effect of assigning a value to the **UTILSGE** environment variable is the same as specifying the **-W NOG** or **-W NOE** flag in the command.

EXAMPLES

1. To delete a file, enter:

   rm myfile

   If there is another link to this file, then the file remains under that name, but the file **myfile** is removed. If **myfile** is the only link, the file itself is deleted.

2. To delete a file silently, enter:

   rm -f core

   This command removes file **core** without asking any questions or displaying any error messages. This is normally used in shell procedures. It prevents confusing messages from being displayed when deleting files that may or may not exist.

3. To delete files interactively, enter:

   rm -i mydir/*

   After each filename is displayed, enter the affirmative response; press **<Return>** (or anything other than the affirmative response) to retain the file.

4. To delete a directory tree interactively, enter:

   rm -ir manual

   This recursively removes the contents of all subdirectories of the file **manual**, then removes **manual** itself, asking if you want to remove each file and directory.

5. To delete all OSS files on the local node, enter:

   export UTILSGE=NOG:NOE
   rm -r /

6. To delete all OSS files on the remote node **node1**, enter:

   rm -r -W NOG /E/node1

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no distinction between Guardian and OSS files or between local and remote files. You can use the **-W NOG** and **-W NOE** flags or the **UTILSGE** environment variable to exclude objects in the Guardian file system or objects accessible through the Expand product.

For G-series RVUs, H06.19 and earlier H-series RVUs, or J06.08 and earlier J-series RVUs, the OSS Network File System (NFS) you cannot use the **rm** command to remove OSS objects that have OSS ACLs that contain optional ACL entries.

For H06.20 and later H-series RVUs and J06.09 and later J-series RVUs, access by the OSS Network File System (NFS) to OSS objects that have OSS ACLs that contain optional ACL entries can be allowed, depending upon the NFSPERMMAP attribute value for the fileset that contains the object. For more information about NFS and ACLs, see the **acl(5)** reference page.

For H06.23 and later H-series RVUs, J06.12 and later J-series RVUs, the H06.22 RVU with the T9194H01^AFA SPR installed, or the J06.11 RVU with the T9194J01^AEZ SPR installed, a remote user can delete an SQL object that is in the OSS file system.

For H06.22 (without the T9194H01^AFA SPR installed) and earlier H-series RVUs, and J06.11 (without the T9194J01^AEZ SPR installed) and earlier J-series RVUs, if a remote user attempts

to delete an SQL object that is in the OSS file system, the **rm** command fails with error "Guardian or User Defined Error 197". To remove this type of file on these RVUs, a local user must execute the **rm** command.

File-Label and SQL Catalog Table Inconsistencies

When you issue the **rm** command to purge a remote OSS SQL program and the command fails due to certain failure scenarios such as a CPU failure, disk failure, or application outage, this failure can cause an inconsistency between the program file label and the corresponding SQL catalog. The program can lose its SQL properties, but have SQL catalog entries present. In this case, the following conditions occur:

- The program does not run after losing its SQL properties.
- The SQL catalog tables have stray entries for this program file. The program exists, but its just like a normal Enscribe file.
- If any DDL command is issued on a table on which the program depends, the program obtains partial SQL properties and remains invalid.

RELATED INFORMATION

Commands: **ln(1)**, **mv(1)**, **rmdir(1)**.

Functions: **rmdir(2)**, **unlink(2)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The **-W NOG** and **-W NOE** flags and the **UTILSGE** environment variable are HP extensions to the XPG4 Version 2 specification.

NAME

rmdir - Removes a directory

SYNOPSIS

rmdir [**-p**] *directory* ...

FLAGS

-p Removes all directories in a pathname. For each *directory* argument, the following operations are performed:

1. The directory entry specified by *directory* is removed.
2. If the *directory* argument includes more than one pathname component, effects equivalent to the following command occur:

   rmdir -p $(dirname *directory*)

DESCRIPTION

The **rmdir** command removes a directory from the system. The directory must be empty before you can remove it, and you must have write permission in its parent directory. Use the **ls -al** command to see whether a directory is empty.

If a directory and a subdirectory of that directory are specified in a single invocation of **rmdir**, the subdirectory must be specified before the parent directory so that the parent directory is empty when **rmdir** tries to remove it.

EXAMPLES

1. To empty and remove a directory, enter:

   ```
   rm mydir/* mydir/.*
   rmdir mydir
   ```

   This command removes the contents of directory **mydir**, then removes the empty directory. The **rm** command displays an error message about trying to remove the directories . (dot) and .. (dot dot), and then **rmdir** removes them.

   Note that **rm mydir/* mydir/.*** first removes files with names that do not begin with a **.** (dot), then those with names that do begin with a **.** (dot). You may not realize that the directory contains filenames that begin with a **.** (dot), because the **ls** command does not normally list them unless you use the **-a** flag.

2. To remove all of the directories in the pathname **a/b/c**, enter:

   ```
   rmdir -p a/b/c
   ```

   Use a command like this one if directory **a** in the current directory is empty except that it contains a directory **b** and **a/b** is empty except that it contains a directory **c**.

NOTES

Because **/G** and **/E** both appear in your local root directory, you should be very careful when using OSS shell commands on or from the root directory. OSS shell commands that perform recursive actions make no destinction between Guardian and OSS files or between local and remote files.

RELATED INFORMATION

Commands: **ls**(1), **rm**(1).

Functions: **rmdir**(2), **unlink**(2).

NAME

rsh - Executes the specified command remotely

SYNOPSIS

rsh [**-d**] [**-l** *user*] *hostname command* [*argument* ...]

rsh [**-d**] *hostname* [**-l** *user*] *command* [*argument* ...]

hostname [**-l** *user*] [*command*] [*argument* ...]

FLAGS

-d Turns on socket debugging (using the **setsockopt()** function) on the Transmission Control Protocol (TCP) sockets used for communication with the remote host.

-l *user* Specifies that **rsh** is to log in to the remote host using *user* instead of the user's local user name. If this flag is not specified, the local and remote user names are the same.

DESCRIPTION

The remote shell command (**rsh**) executes *command* at *hostname*. The **rsh** command sends the standard input file from the local host to the remote command and receives the standard output and standard error files from the remote command.

If the name of the file from which **rsh** is executed is anything other than **rsh**, **rsh** takes this name as its *hostname* operand. This feature allows you to create a symbolic link to **rsh** in the name of the host that, when executed, invokes a remote shell on that host. If you create a directory and populate it with symbolic links in the names of commonly used hosts, then, by including the directory in your shell's search path, you can run **rsh** by typing *hostname* to your shell.

If you do not specify the **-l** flag, the local user name is used at the remote host. If **-l** *user* is entered, the specified user name is used at the remote host. In either case, the remote host allows access only if at least one of the following conditions is satisfied:

- The local user ID is not the super ID, and the name of the local host is listed as an equivalent host in the remote **/etc/hosts.equiv** file.
- The remote user's home directory contains a **$HOME/.rhosts** file that lists the local host and user name.

For security reasons, any **$HOME/.rhosts** file must be owned by either the remote user or the super ID, and only the owner should have write access.

In addition to the preceding conditions, **rsh** also allows access to the remote host if the remote user account does not have a password defined. However, for security reasons, use of a password on all user accounts is recommended.

While the remote command is executing, pressing the Interrupt, Terminate, or Quit key sequence sends the corresponding signal to the remote process. However, pressing the Stop key sequence stops only the local process. Normally, when the remote command terminates, the local **rsh** process terminates.

To have shell metacharacters interpreted on the remote host, place the metacharacters inside ´ ´ (single quotes). Otherwise, the metacharacters are interpreted by the local shell.

EXAMPLES

In the following examples, the local host **host1** is listed in the **/etc/hosts.equiv** file at the remote host **host2**.

1. To check the amount of free disk space on the remote host **host2**, enter:

   $ rsh host2 df

2. To do the same job, create symbolic link **host2** to **rsh** and enter:

   $ host2 df

3. To append a remote file to another file on the remote host, place the >> metacharacters in "" (double quotes):

   $ rsh host2 cat test1 ">>" test2

4. To append a remote file at the remote host to a local file, omit the double quotes:

   $ rsh host2 cat test2 >> test3

5. To append a remote file to a local file and use a remote user's permissions at the remote host, use the **-l** flag:

   $ rsh host2 -l jane cat test4 >> test5

FILES

/etc/hosts.equiv	Specifies remote hosts from which users can execute commands on the local host (if these users have an account on the local host). This file can also specify a trusted user for each host.
$HOME/.rhosts	Specifies remote hosts from which users can execute commands on the local host (if these users have an account on the local host). This file can also specify a trusted user for each host.

NOTES

To remotely access an HP NonStop server from a UNIX system or another NonStop HP server through that system's **rsh** command, you must specify the **-l** flag. If you omit the **-l** flag, or if the target server does not have a login name configured with an INITIAL-DIRECTORY attribute, the command fails with the message "Connection refused."

RELATED INFORMATION

Commands: **rshd(8)**, **telnet(1)**.

Files: **hosts.equiv(4)**, **.rhosts(4)**.

STANDARDS CONFORMANCE

This command is an HP extension to the XPG4 Version 2 specification.

NAME

run - Runs a process with specific attributes

SYNOPSIS

run [**-cpu**=*cpu_number*
| **-debug**
| **-defmode**={**off** | **on**}
| **-inspect**={**off** | **on** | **saveabend**}
| **-jobid**=*jobid_number*
| **-mem**=*num_pages*
| **-maxheap**=*maxheapsize*
| **-name**=/**G**/*process_name*
| **-gpri**=*priority*
| **-swap**=/**G**/*volume*[/*subvolume*[/*swapfile*]]
| **-extswap**=/**G**/*volume*[/*subvolume*[/*swapfile*]]
| **-term**=/**G**/*terminal_name*
| **-lib**={/**G**/*volume/subvolume/*<*library_filename*> | **unset**}] ...
program_file_path [*arguments*]

FLAGS

-cpu=*cpu_number*
Specifies the number of the processor on which the process is to run. *cpu_number* is an integer in the range 0 through 15. If the **-cpu** flag is not specified, the process is created on the CPU from which the **run** command was executed.

-debug Causes the process to start in the symbolic debugger. If the **-debug** flag is not specified, the process is not started in the symbolic debugger.

-defmode={**off** | **on**}
Specifies the initial **-defmode** setting for the process being started. The setting controls both enablement and propagation of DEFINEs for the process and determines which DEFINEs are propagated to the process. If you specify **-defmode=off**, all DEFINEs are disabled for the new process and no DEFINE is propagated from the OSS shell to the new process. If you specify **-defmode=on**, all DEFINEs are enabled for the new process and all DEFINEs are propagated to the new process. The **-defmode** flag has no effect on the **-defmode** setting for the current OSS shell. If you do not specify the **-defmode** flag, the initial setting for **-defmode** is the same as the **-defmode** setting for the current OSS shell. The **-defmode** flag has no effect on the **-defmode** setting for the current OSS shell.

-gpri=*priority*
Specifies the Guardian execution priority of the new process. *priority* is specified as an integer in the range 1 through 199, with 199 being the highest priority. If an integer greater than 199 is specified, the process runs at priority 199 (a warning message might be displayed). In this case, the exit status is set to indicate that the operation was successful. If the **-gpri** flag is not specified, the default Guardian execution priority of a child process will be the same as the execution priority of the parent.

-inspect={**off** | **on** | **saveabend**}
Sets the debugging environment for the process being started. The **-inspect off** flag selects the default debugging utility, which is the default action. The **-inspect on** and **-inspect saveabend** flags select the current symbolic debugger. **-inspect on** and **-inspect saveabend** are the same except that **-inspect saveabend** automatically creates a saveabend file (process snapshot file or core file) if the program ends abnormally.

-jobid=*jobid_number*
Specifies the new job ID for the new process. If this flag is not specified, the job ID for the new process is 0.

-lib={**/G***/volume/subvolume/swapfile* | **unset**}
Specifies a user library file in the **/G** directory that is to be searched for external references required by the program being run. The user library file is searched before the OSS library file. The name of the library file in **/G** is linked to the program file being run and remains in use until the program is run with the **-lib=unset** option.

-mem=*num_pages*
Specifies the maximum number of virtual data pages to be allocated for the new process. *num_pages* is an integer in the range 1 through 64. If this flag is not specified, the default value for the maximum number of virtual data pages allocated for the new process is 64. If the value specified for *num_pages* is less than the compilation-time value, the compilation-time value is used. If the actual compilation-time value is less than the value specified for *num_pages* (or its default value), the compilation-time value is used.

The **-mem** flag does not apply to native processes, which use the kernel-managed swap facility.

-maxheap=*maxheapsize*
Overrides the default maximum heap size for a process. The *maxheapsize* value is a positive integer, optionally suffixed with either MB or GB. The default unit of the *maxheapsize* value is MB (MegaBytes).

This flag is supported on systems running J06.13 or later J-series RVUs or H06.24 or later H-series RVUs only.

-name=**/G***/process_name*
Specifies the Guardian process name to be assigned to the new process. *process_name* is an alphanumeric string whose first character must be alphabetic. If this flag is not specified, the new process is not named.

-swap=**/G***/volume*[*/subvolume*[*/swapfile*]]
Specifies either the name of a file in directory **/G** that holds the virtual data of a TNS process or the name of a volume to contain temporary files.

When a TNS process is running, by default the operating system allocates swap space from the kernel-managed swap facility for the purpose of swapping the data stack. However, if the **-swap** flag specifies a valid unstructured permanent file, that file is used as the swap file and is not purged when the process terminates. The volume that is specified with the **-swap** flag must exist in the **/G** directory on the local node. If not, the **run** command displays the usage message and exits in error.

For a TNS process, if the file specified with the **-swap** flag does not exist, it will be created. If you specify a swap volume (omitting the subvolume and swapfile names) for a TNS process, the operating system creates a temporary swap file for the TNS data segment and deletes it when the process completes.

For a native process, any volume specified in a **-swap** flag can be used for temporary file creation by appropriately coded processes. The process can inquire (through PROCESS_GETINFOLIST_ or PROCESS_GETINFO_) about its swap file and get back the name of a nonexistent temporary file, for example, $VOL.#0. (The volume name is as specified in the command; the #0 is syntactically valid but does not correspond to a real file [a temporary file would have seven digits after the #].) Some utilities use this feature to determine the volume on which to place temporary files of

their own creation. By default, the swap volume of a native process is reported as the program location.

-extswap=**/G***/volume*[*/subvolume*[*/swapfile*]]
Same as **-swap**, but any swap file specified is used to hold the extended data segment for the TNS process. This flag has no impact on native processes.

-term=/G*/terminal_name*
Specifies the home terminal for the new process. If this flag is omitted, the new process uses the OSS shell's home terminal. *terminal_name* must be a valid name for a terminal or process. If *terminal_name* is of the Guardian form $ZTNT.#PTY4, it must be specified as **/G/ZTNT/#PTY4**. If *terminal_name* is of the Guardian form $*name*, it must be specified as **/G/***name*. For example, $TTY translates to **/G/TTY**.

DESCRIPTION

The **run** utility starts OSS programs with specific attributes. The **run** utility is implemented as a built-in shell feature conforming to the POSIX.2 syntax standards. The child process of **run** belongs to the same process group as that of the shell from which the **run** command was issued. **run** allows OSS programs to run under the control of the symbolic debugger. **run** issues a **tdm_execve( )** function call with the specified executable image file path and the option operands with which it was invoked.

Operands

program_file_path
The OSS filename of an executable image. An executable image is any type of OSS file that has the execute permission bit set for the user, so that the OSS **stat( )** function identifies the file as an executable image.

If the full OSS pathname is not specified, the directories listed in the environment variable **PATH** are searched. If the **run** command is invoked without a *program_file_path*, the usage string is displayed and **run** exits in error.

arguments
An optional list of arguments that are to be passed to the program file. The arguments should be specified at the time the **run** command is invoked.

Environment Variables

LANG Determines the locale to be used for the locale categories when both the **LC_ALL** variable and the corresponding environment variable (whose name begins with **LC_**) do not specify a locale.

LC_ALL
Determines the locale to be used to override any values for locale categories specified by the settings of the **LANG** variable or any environment variable whose name begins with **LC_**.

LC_CTYPE
Determines the locale for the interpretation of bytes of text data as characters (for example, single-byte versus multibyte characters in arguments).

LC_MESSAGES
Determines the locale to be used to affect the format and content of diagnostic messages written to the standard error file and informational messages written to the standard output file.

Standard Output

The **run** command itself does not print messages to the standard output file. However, the program file to which **run** is applying the **tdm_execve( )** function can print messages to the standard output file if both the program file and the shell are sharing the same home terminal.

EXAMPLES

1. The following command line

   run -cpu=3 -gpri=150 -name=/G/myls ls -xaF

   starts the **ls** utility on processor 3 with a priority of 150. This instance of **ls** is passed the arguments **-xaF** and is named **/G/myls**.

2. The following command line

   run -debug -inspect=on myprogram

   starts the program **myprogram** in the symbolic debugger.

NOTES

When resources are not available, the shell can return the following message if the **run** command is used:

```
/bin/-sh:  sh: tdm_fork() failed with errno EAGAIN:
               cannot fork too many processes
```

EXIT VALUES

0 Command completed sucessfully.

>0 An error occurred.

RELATED INFORMATION

Functions: **tdm_execve(2)**.

STANDARDS CONFORMANCE

The **run** utility is an HP extension to the XPG4 Version 2 specification.

NAME

runcat - Invokes the **mkcatdefs** utility and pipes the resulting message-catalog source data to the **gencat** utility

SYNOPSIS

runcat *catalog_name source_file catalogfile*

FLAGS

Operands

catalog_name is the name of the message catalog to be used by the **mkcatdefs** utility to generate the name of the symbolic definition file.

source_file identifies the message text containing symbolic identifiers.

catalogfile is the name of the catalog file to be created by the **gencat** utility.

DESCRIPTION

The **runcat** utility invokes the **mkcatdefs** utility and pipes the resulting message-catalog source data to the **gencat** utility.

source_file contains the message text containing symbolic identifiers. The **mkcatdefs** utility uses the *catalog_name* argument to generate the name of the symbolic definition file by adding **_msg.h** to the end of the *catalog_name* value. **mkcatdefs** also uses the *catalog_name* value to generate the symbolic name for the catalog file by adding **MF_** to the beginning of the *catalog_name* value.

The symbolic definition file must be included in your application. The symbolic name for the catalog file can be used in library functions such as the **catopen** function. The *catalogfile* argument is the name of the catalog file created by the **gencat** utility. If you do not specify this argument, the **gencat** utility appends **.cat** to the end of the *catalog_name* value. This filename can also be used in the *catopen* library function.

EXAMPLES

To generate a catalog named test.cat from a message source file named `test.msg`, enter:

```
runcat test test.msg
```

DIAGNOSTICS

The **runcat** utility generates these error messages:

```
Usage: runcat catname srcfile [catfile]\n
Can´t open \%s\n
```

RELATED INFORMATION

Commands: **gencat(1)**, **mkcatdefs(1)**.

STANDARDS CONFORMANCE

The **runcat( )** utility is an extension to the XPG4 specification.

NAME

runv - Runs a process in the Visual Inspect debugger

SYNOPSIS

runv [**-wsaddr=**{*workstation_IP_address* }]
[*run_command_options*]
program_file_path [*arguments*]

FLAGS

-wsaddr={*workstation_IP_address*}

Specifies the workstation name or numeric IP address of the client workstation running Visual Inspect.

DESCRIPTION

The **runv** utility starts OSS programs in the Visual Inspect symbolic debugger. The **runv** utility is implemented as a stand-alone shell script. The **runv** utility accepts all **run** utility flags described in the **run(1)** reference page, except it silently ignores the **-inspect=off**, **-inspect=on**, and **-debug** flags. The **-debug** and **-inspect=on** flags are not necessary, because these features are implied by the use of **runv**. If **-inspect=off** is specified, it is ignored.

The **-wsaddr** flag specifies the workstation name or numeric IP address of the client workstation running the Visual Inspect debugger. This Visual Inspect session must have a connection to the HP NonStop host that invoked the **runv** utility. See the Visual Inspect online help for details on connecting to a host.

The **-wsaddr** flag overrides the setting of the **_TANDEM_VISUALINSPECT_WSADDR** environment variable if both are set. If neither the flag nor the variable is set, **runv** automatically detects the IP address and uses it.

Operands

program_file_path

The OSS filename of an executable image. An executable image is any type of OSS file that has the execute permission bit set for the user, so that the OSS **stat( )** function identifies the file as an executable image.

If the full OSS pathname is not specified, the directories listed in the environment variable **PATH** are searched. If the **runv** utility is invoked without a *program_file_path*, the usage string is displayed and **runv** exits in error.

arguments

An optional list of arguments that are to be passed to the program file. The arguments should be specified at the time the **runv** utility is invoked.

Environment Variables

_TANDEM_VISUALINSPECT_WSADDR

Specifies the default workstation name or IP address of the client system running Visual Inspect if the **-wsaddr** flag is not specified. This variable must be exported to be available to **runv**.

Standard Output

The **runv** utility prints error text to the standard output file if it encounters an error. The program file to which **runv** is applying the **tdm_execve( )** function can also print messages to the standard output file if both the program file and the shell are sharing the same home terminal.

NOTES

On H-series systems, the **runv** and **run -debug** commands start the same debugger and are functionally equivalent. However, the **run -debug** command is faster than the **runv** command in the OSS environment on H-series systems.

Unless you want to initiate debugging from one workstation but have the debugging session go to another workstation, you should use the **run -debug** command instead of **runv**. The **run -debug** command does not support the **-wsaddr** flag, so the **runv** command must be used to route debugging to a different workstation.

EXAMPLES

1. The command line

   runv myprog

   starts the program **myprog** under the control of the Visual Inspect session on the current workstation.

2. The command line

   runv -cpu=2 -wsaddr=mypc.mycompany.com myprogram -abc myfile

   starts the **myprogram** executable image under the control of Visual Inspect on the workstation **mypc.mycompany.com** and on processor 2. This instance of **myprogram** is passed the arguments **-abc** and **myfile**.

EXIT VALUES

The following exit values are returned by **runv**:

0 The file specified in *program_file_path* was run successfully.

>0 An error occurred.

RELATED INFORMATION

Commands: **run(1)**.

Functions: **tdm_execve(2)**.

STANDARDS CONFORMANCE

The **runv** utility is an HP extension to the XPG4 Version 2 specification.

Section 8. User Commands (s)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letter **s**.

NAME

sed - Provides a stream line editor

SYNOPSIS

sed [**-n**] [**-e** *escript*] ... [**-f** *script_file*] ... [*file* ...]

sed [**-n**] *script* [*file* ...]

The **sed** command modifies lines from the specified *file* or from standard input according to edit commands and writes them to standard output.

FLAGS

-e *escript*

Uses the string *escript* as an edit script. If you are using just one **-e** flag and no **-f** flag, you can omit the **-e** flag and include the single *escript* on the command line as an argument to **sed**.

-f *script_file*

Uses *script_file* as the source of the edit script. The *script_file* is a set of editing commands to be applied to *file*.

-n Suppresses the default action of writing each line to standard output after editing. Only lines explicitly selected for output are written.

DESCRIPTION

The **sed** command includes many features for selecting lines to be modified and making changes only to the selected lines.

The **sed** command uses two workspaces for holding the line being modified: the *pattern space*, where the selected line is held, and the *hold space*, where lines can be stored temporarily.

An edit script consists of individual subcommands, each one on a separate line. The general form of **sed** subcommands is as follows:

[*address*[**,***address*]] *command* [*argument* ...]

The **sed** command processes each input file by reading an input line into the pattern space, sequentially applying all **sed** subcommands in sequence whose addresses select that pattern space, and writing the pattern space to standard output. It then clears the pattern space and repeats this process for each line in the input file. Some of the subcommands use a hold space to save all or part of the pattern space for subsequent retrieval.

When a command includes an address, either a line number or a search pattern, only the addressed line or lines are affected by the command. Otherwise, the command is applied to all lines.

The sed Addresses

An address is either a decimal line number, a **$**, which addresses the last line of input, or a *context address*. A context address is a regular expression as described for **grep**, except that you can select the character delimiter for patterns. The general form of the expression is as follows:

*?pattern**?

The *?* represents a character delimiter you select. This delimiter cannot be a multibyte character.

The default form for the pattern is as follows:

/pattern/

- In a context address, the construction *\cexpressionc*, where *c* is any character other than a \ (backslash) or the newline character, is identical to */expression/*. If the character designated by *c* appears following a \ (backslash), then it is considered to be that literal

character, which does not terminate the regular expression (RE). For example, in the context address **\xabc\xdefx**, the second **x** stands for itself, so that the RE is **abcxdef**.

- The sequence **\n** matches a newline character in the pattern space, except the terminating new line. A literal newline character must not be used in the regular expression of a context address or in the **s** (substitute) subcommand.
- A **.** (dot) matches any character except a terminating newline character. That is, unlike **grep**, which cannot match a newline character in the middle of a line, **sed** can match a newline character in the pattern space.

Certain commands allow you to specify one line or a range of lines to which the command applies. These commands are called *addressed commands*. The following rules apply to addressed commands:

- A command line with no address selects every line.
- A command line with one address, expressed in context form, selects each line that matches the address.
- A command line with two addresses separated by a **,** (comma) or **;** (semicolon) selects the entire range from the first line that matches the first address through the next line that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter, the process is repeated, looking again for the first address.

SUBCOMMANDS

Backslashes in text are treated like backslashes in the replacement string of an **s** command and can be used to protect initial spaces and tabs against the stripping that is done on every script line.

The *text* argument accompanying the **a**, **c**, and **i** commands can continue onto more than one line, provided all lines but the last end with a \ (backslash) to quote the newline character.

The *read_file* and *write_file* arguments must end the command line and must be preceded by exactly one space. Each *write_file* is created before processing begins, up to a maximum of 10 files.

The **sed** command can process up to 99 commands in a file.

In the following list of subcommands, the maximum number of permissible addresses for each subcommand is indicated in parentheses. The **sed** script subcommands are as follows:

(2)**{***subcommand ...*

} Groups subcommands enclosed in **{ }** (braces). The **{** (left brace) can be preceded by spaces and can be followed by spaces or tabs. The list of subcommands must be separated by newline characters. The subcommands can also be preceded by spaces or tabs. The terminating **}** (right brace) must be preceded by a newline character and then zero or more spaces.

(1) **a**

text Places *text* in the output file before reading the next input line, whether by executing **N** or by beginning a new cycle.

(2)**b** [*label*]

Branches to the : command bearing the *label*. If *label* is empty, it branches to the end of the script.

(2)**c**\

text Deletes the pattern space. With a 0 or 1 address or at the end of a 2-address range, places *text* on the output. Then it starts the next cycle.

(2)**d** Deletes the pattern space. Then it starts the next cycle.

(2)**D** Deletes the initial segment of the pattern space through the first newline character. Then it starts the next cycle.

(2)**g** Replaces the contents of the pattern space with the contents of the hold space.

(2)**G** Appends the contents of the hold space to the pattern space, after first appending a newline character.

(2)**h** Replaces the contents of the hold space with the contents of the pattern space.

(2)**H** Appends the contents of the pattern space to the hold space, after first appending a newline character.

(1)**i**\

text Writes *text* to standard output before reading the next line into the pattern space.

(2)**l** Writes the pattern space to standard output, showing nonprinting characters as 3-digit octal values. Long lines are folded, with the point of folding indicated by **<Backslash><Return>**. The end of each line is marked with a **$**.

Certain characters are shown as escape sequences as follows:

\\	Backslash
\a	Alert
\b	Backspace
\f	Formfeed
\n	Newline
\r	Carriage return
\t	Tab
\v	Vertical tab

(2)**n** Writes the pattern space to standard output. It replaces the pattern space with the next line of input.

(2)**N** Appends the next line of input to the pattern space with an embedded newline character. (The current line number changes.) You can use this subcommand to search for patterns that are split onto two lines.

(2)**p** Writes the pattern space to standard output.

(2)**P** Writes the initial segment of the pattern space through the first newline character to standard output.

(1)**q** Branches to the end of the script. It does not start a new cycle.

(1)**r** *read_file*
Reads the contents of *read_file*. It places contents on the output before reading the

next input line. If *read_file* does not exist, it is treated as an empty file, causing no error condition.

(2)**s**/*pattern*/*replacement*/*flags*

Substitutes the *replacement* string for the first occurrence of the *pattern* in the pattern space. Any character that is entered after the **s** command can substitute for the **/** (slash) separator, except \ (backslash) and the newline character. Within the regular expression and replacement string, the delimiter can appear as a literal if it is preceded by a \ (backslash).

An **&** (ampersand) appearing in the replacement string is replaced by the string matching the RE. The special meaning of **&** in this context can be suppressed by preceding it with a \ (backslash). The characters *n*, where *n* is a digit, are replaced by the text matched by the corresponding backreference expression.

A line can be split by substituting a newline character into it. You must escape the newline character in the replacement string by preceding it with a \ backslash. A substitution is considered to have been performed even if the replacement string is identical to the string that it replaces.

You can add zero or more of the following *flags*:

n Substitutes *replacement* for the *n*th occurrence of *pattern* on each addressed line, rather than for the first occurrence.

g Substitutes *replacement* for all nonoverlapping instances of *pattern* on each addressed line, rather than for just the first one (or for the one specified by *n*).

p Writes the pattern space to standard output if a replacement was made.

w *write_file*

Writes the pattern space to *write_file* if a replacement was made. Appends the pattern space to *write_file*. If *write_file* was not already created by a previous write by this **sed** script, **sed** creates it. Each *write_file* is created before processing begins.

(2)**t** [*label*]

Branches to **:***label* in the script file if any substitutions were made since the most recent reading of an input line or execution of a **t** subcommand. If you do not specify *label*, control transfers to the end of the script.

(2)**w** *write_file*

Appends the pattern space to *write_file*.

(2)**x** Exchanges the contents of the pattern space and the hold space.

(2)**y**/*pattern1*/*pattern2*/

Replaces all occurrences of characters in *pattern1* with the corresponding characters from *pattern2*. *pattern1* and *pattern2* must contain the same number of characters. The delimiter itself can be used as a literal if it is preceded by a \ (backslash) character.

(2)**!***sed_command*

Applies the specified **sed** subcommand only to lines not selected by this address or addresses.

(2)**!{***subcommand ...*

} Applies the specified subcommand list only to lines not selected by this address or addresses. The **{** (left brace) can be preceded by spaces and can be followed by spaces

or tabs. The list of subcommands must be separated by newline characters. The subcommands can also be preceded by spaces or tabs. The terminating **}** (right brace) must be preceded by a newline character and then zero or more spaces.

(0)**:***label* This script entry simply marks a branch point to be referenced by the **b** and **t** commands. This label can be any sequence of eight or fewer bytes.

(1)**=** Writes the current line number to standard output as a line.

(0) Ignores an empty command.

(0)# If a **#** (number sign) appears as the first character on a line, that entire line is treated as a comment, with one exception. If the first two characters of a script file are **#n**, the default output is suppressed (same as **-n**).

EXAMPLES

1. To perform a global change, enter:

   sed "s/happy/enchanted/g" chap1 >chap1.new

   This replaces each occurrence of **happy** found in the file **chap1** with **enchanted**, and puts the edited version in a separate file named **chap1.new**. The **g** at the end of the **s** subcommand tells **sed** to make as many substitutions as possible on each line. Without the **g**, **sed** replaces only the first **happy** on a line.

   The **sed** stream editor operates as a filter. It reads text from standard input or from the files named on the command line (**chap1** in this example), modifies this text, and writes it to standard output. Unlike most editors, it does not replace the original file. This makes **sed** a powerful command when used in pipelines.

2. To use **sed** as a filter in a pipeline (**sh** only), enter:

   pr chap2 | sed "s/Page *[0-9]*$/(&)/" | print

   This encloses the page numbers in parentheses before printing **chap2**. The **pr** command puts a heading and page number at the top of each page, then **sed** puts the page numbers in parentheses, and the **print** command prints the edited listing.

   The **sed** pattern **/Page *[0-9]*$/** matches page numbers that appear at the end of a line. The **s** subcommand changes this to **(&)**, where the **&** stands for the pattern that was matched (for example, **Page 5**).

3. To display selected lines of a file, enter:

   sed -n "/food/p" chap3

   This displays each line in **chap3** that contains the word **food**. Normally, **sed** copies every line to standard output after it is edited. The **-n** flag stops **sed** from doing this. You then use subcommands like **p** to write specific parts of the text. Without the **-n**, this example displays all the lines in **chap3**, and it shows each line containing **food** twice.

4. To perform complex editing, enter:

   sed -f script.sed chap4 >chap4.new

   It is always a good idea to create a **sed** script file when you want to do anything complex. You can then test and modify your script before using it. You can also reuse your script to edit other files. Create the script file with an interactive text editor.

5. A sample **sed** script follows:

```
:join
/\\$/{N
s/\\\n//
b join
}
```

This **sed** script joins each line that ends with a \ (backslash) to the line that follows it. First, the pattern **/\\$/** selects a line that ends with a \ for the group of commands enclosed in **{ }**. The **N** subcommand then appends the next line, embedding a newline character. The **s/\\\n//** deletes the \ (backslash) and embedded newline character. Finally, **b join** branches back to the label **:join** to check for a \ (backslash) at the end of the newly joined line. Without the branch, **sed** writes the joined line and reads the next one before checking for a second \ character.

The **N** subcommand causes **sed** to stop immediately if there are no more lines of input (that is, if **N** reads the End-of-File character). It does not copy the pattern space to standard output before stopping. This means that if the last line of the input ends with a \ (backslash) character, then it is not copied to the output.

RELATED INFORMATION

Commands: **awk**(1), **grep**(1), **vi**(1).

NAME

set - Sets shell options and positional parameters

SYNOPSIS

set [**+** | **-aCefmnostuvx**] [**+** | **-o** *option* ...] [*argument* ...]

FLAGS

a All subsequent parameters defined are automatically exported.

b Causes the shell to notify the user asynchronously of background job completions (same as **-o notify**). When the shell notifies the user a job has been completed, it can remove the job's process ID from the list of those known in the current shell execution environment.

C Prevents existing files from being overwritten by the shell's > redirection operator (same as **-o noclobber**). The **>|** redirection operator overrides this noclobber option for an individual file.

e If a command has a nonzero exit status, is not part of the compound list following a **while**, **until**, or **if** keyword, is not part of an AND or OR list, and is not a pipeline preceded by the **!** reserved word, it executes the **ERR** trap, if set, and exits. This mode is disabled while reading profiles.

f Disables pathname expansion.

m Background jobs run in a separate process group and a line prints upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells.

n Reads commands and checks them for syntax errors, but does not execute them. Ignored for interactive shells.

o The argument can be one of the following option names:

allexport
Same as **a**.

errexit Same as **e**.

bgnice Runs all background jobs at a lower priority. This is the default mode.

emacs Invokes an **emacs** style inline editor for command entry.

gmacs Invokes a **gmacs**-style inline editor for command entry.

ignoreeof
The interactive shell does not exit on End-of-File. This setting prevents accidental exit when **<Ctrl-D>** is entered. The **exit** command must be used.

markdirs
All directory names resulting from filename generation have a trailing **/** (slash) appended.

monitor Same as **m**.

noclobber
Prevents redirection > from truncating existing files. Requires **>|** to truncate a file when turned on. (Same as **C**.)

noexec Same as **n**.

noglob Same as **f**.

nolog Prevents the entry of function definitions into the command history.

notify Same as **b**.

nounset Same as **u**.

trackall Each time the shell finds a new command when searching a path, it creates an alias for the command. The value of the alias is the full pathname for the command.

verbose Same as **v**.

vi Invokes, in insert mode, a **vi**-style inline editor until you press Escape (ASCII **033**). This changes to move mode. A return sends the line.

viraw Each character is processed as it is entered in **vi** mode.

xtrace Same as **x**.

If no option name is supplied, then the current option settings are printed.

s Sorts the positional parameters.

t Exits after reading and executing one command.

u Treats unset parameters as an error when substituting. An interactive shell does not exit.

v Prints shell input lines to standard error as they are read.

x Prints each command and its arguments to standard error after expanding the command and before executing it.

- Unsets **x** and **v** flags without changing the positional parameters. If other arguments are present, these arguments are assigned to the positional parameters in order. (Obsolescent)

-- Does not change any of the flags; useful in setting **$1** to a value beginning with **-**. If no arguments follow this flag, the positional parameters are unset.

These flags can also be used upon invocation of the shell. The current set of flags can be found in **$-**. The remaining arguments are positional parameters and are assigned, in order, to **$1 $2** .... If no arguments are given, the names and values of all named parameters are printed on the standard output. If the only argument is **+**, the names of all named parameters are printed.

DESCRIPTION

The **set** command is used to set various shell commands (listed under the **o** option below) and to assign positional parameters.

The flags can also be used upon invocation of the shell. The current set of flags can be found in **$-**.

Used without any of its options, the **set** command returns a list of all the variables that exist in your environment, local or exported.

If the only argument is **+**, the names of all named parameters are printed.

The argument *argument* ... is composed of positional parameters, which are assigned, in order, to **$1 $2** ....

The **set** command is also used to assign positional parameters.

EXAMPLES

1. Following is an example of the set command and its results.

   set a b c

   The above command assigns **a** to **$1**, **b** to **$2**, and **c** to **$3**. The following command verifies the values of a, b, and c.

   echo $1:$2:$3

   a:b:c

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **set** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1), **unset**(1).

NAME

setacl - Modifies access control lists (ACLs) for files

SYNOPSIS

setacl [-n] -f *acl_file file* **...**
setacl [-n] { -m | -d } *acl_entries* **[{ -m | -d }** *acl_entries***]...** *file* **...**
setacl [-n] -s *acl_entries file* **...**

FLAGS

-n Normally, the **setacl** command recalculates the group class entry to ensure that permissions specified in the additional ACL entries are actually granted, and the value specified in the **class** entry is ignored. If you specify the **-n** flag, the recalculation is not performed, and the value specified in the **class** entry is used.

-s Sets the ACL for a file. All old ACL entries are removed and replaced with the newly specified ACL. You must specify exactly one **user** entry for the owner of the file, exactly one **group** entry for the owning group of the file, and exactly one **other** entry. If you specify the **-n** flag, you must specify exactly one **class** entry also. You can specify additional **user** ACL entries and additional **group** ACL entries, but these entries must not duplicate **user** entries with the same user ID or **group** entries with the same group ID.

-m Adds or modifies the specified ACL entry.

-d Deletes the specified entry from the ACL of the specified file.

-f Sets the ACL entries for *file* to the entries specified in *acl_file*.

You must specifiy one of the flags **-s**, **-m**, **-d**, or **-f**. If you specify **-s** or **-f**, other flags are invalid. You can combine the **-m** and **-d** flags, and you can specify multiple **-m** and **-d** flags.

For a detailed description of the use of these flags, see DESCRIPTION.

DESCRIPTION

For each *file* specified, the **setacl** command either:

- Replaces the entire ACL.
- Adds, modifies, or deletes the specified ACL entries, including default entries for directories.

Only a user with a user ID equal to the super ID or file owner, or with a user ID or group affiliation qualifying for membership in the Safeguard SECURITY-OSS-ADMINISTRATOR group can use the **setacl** command to change the file access permissions for a file.

For the **-m** and **-s** flags, *acl_entries* are one or more comma-separated ACL entries selected from the following list. For the **-f** flag, *acl_file* must contain ACL entries, one to a line, selected from the same list. You can specify default ACL entries for directories only. Brackets denote optional characters. Items formatted as a variable denote fields for you to enter. For example, **u:***uid***:***perm* becomes **u:james:rwx**. Choices, of which exactly one must be selected, are separated by vertical bars.

u[ser]::*operm*|*perm*
u[ser]:*uid*:*operm*|*perm*
g[roup]::*operm*|*perm*
g[roup]:*gid*:*operm*|*perm*
c[lass]:*operm*|*perm*
o[ther]:*operm*|*perm*
d[efault]:u[ser]::*operm*|*perm*

d[efault]:u[ser]:*uid*:*operm*|*perm*
d[efault]:g[roup]::*operm*|*perm*
d[efault]:g[roup]:*gid*:*operm*|*perm*
d[efault]:c[lass]:*operm*|*perm*
d[efault]:o[ther]:*operm*|*perm*

For the **-d** flag, *acl_entries* are one or more comma-separated ACL entries, without permissions, selected from the following list. You cannot delete the entries for file owner (**user**), owning group (**group), class**, or **other**.

u[ser]:*uid*
g[roup]:*gid*
d[efault]:u[ser]:
d[efault]:u[ser]:*uid*
d[efault]:g[roup]:
d[efault]:g[roup]:*gid*
d[efault]:c[lass]:
d[efault]:o[ther]:

In the preceding lists:

perm A permissions string composed of the characters **r** (read), **w** (write), and **x** (execute), each of which can appear at most one time, in any order. You can specify the character **-** (dash) as a placeholder.

operm The octal representation of the preceding permissions, with 7 representing all permissions, or **rwx**, and 0 representing no permissions, or **---**.

uid A login name or user ID.

gid A group name or group ID.

The flags have the following meanings:

-n Specifies not to recalculate the group class entry. Normally, **setacl** recalculates the group class entry to ensure that permissions granted in the additional ACL entries are actually granted, and the value specified in the **class** entry is ignored. If you specify the **-n** flag, the recalculation is not performed, and the value specified in the **class** entry is used. The **setacl** command never recalculates the **default:class** entry.

-s Replaces the ACL for the specified file with the ACL specified in this command. All old ACLs are removed. You must specify exactly one **user** entry for the owner of the file, one **group** entry for the owning group of the file, and one **other** entry. If you specify the **-n** flag, you must specify exactly one **class** entry in addition to the entries for the owner, owning group, and **other**. You can specify additional **user** and **group** entries, but these entries cannot contain duplicate **user** entries with the same user ID or duplicate **group** entries with the same group ID.

If the file is a directory, you can specify default ACL entries. You can specify at most one **default:user** entry for the owner of the file, at most one **default:group** entry for the owning group of the file, at most one **default:class** entry for the file group class, and at most one **default:other** entry for other users. You can specify additional **default:user** entries and additional **default:group** entries, but these entries cannot include duplicate **default:user** entries with the same user ID or **default:group** entries with the same group ID.

If you specify no permissions for an entry (**---**), the user ID or group ID specified in the entry is denied access to the file.

The entries need not be in order. The **setacl** command sorts them before applying them to the file.

-m Adds one or more new ACL entries to the file, or changes one or more existing ACL entries on the file. If an entry already exists for a specified user ID or group ID, the specified permissions replace the current permissions. If an entry does not exist for the specified user ID or group ID, an entry is created.

-d Deletes one or more existing ACL entries from the file. You cannot delete entries for the file owner, the owning group, **class**, or **other**. Deleting an entry does not necessarily have the same effect as removing all permissions from the entry. Specifically, deleting an entry for a specific user causes the permissions for that user to be determined by the **other** entry (or other **group** entries, if the user is in those groups).

-f Sets the ACL for the specified file using the ACL entries contained in the file named *acl_file*. The constraints for entries in the *acl_file* are the same as the constraints for entries you specify using the **-s** flag. The character **#** in *acl_file* indicates a comment. All characters, starting with the **#**, until the end of the line, are ignored. If the *acl_file* has been created as the output of the **getacl** command, any effective permissions, which are written with a preceding **#**, are also ignored.

Using the **setacl** command can result in changes to the file permission bits. When you change the user ACL entry for the file owner, the file owner permission bits are modified. When you change the **other** ACL entry, the file **other** permission bits are modified. When you set or modify additional user ACL entries, any group ACL entries, or both, the **class** permission bits are modified to reflect the maximum permissions allowed by the additional user entries and all the group entries.

If an ACL contains no additional user or additional group entries, the permissions in the group entry for the object-owning group and the **class** entry must be the same. Therefore, if specifying the **-d** flag results in no additional **user** entries and no additional **group** entries, the **class** entry permissions are set to the permissions of the owning-group entry, whether or not the **-n** flag is specified.

A directory can contain default ACL entries. If a file is created in a directory that contains default ACL entries, the file inherits those default ACL entries as described in the **acl(5)** reference page.

If an ACL contains no additional **default:user** or additional **default:group** entries, and you specify a **default:group** entry for the owning group, you must also specify a **default:class** entry that has the same permissions as the **default:group** entry.

EXAMPLES

To add one ACL entry to file **filea**, giving user **archer** read permission only, use this command:

setacl -m user:archer:r-- filea

If an entry for user **archer** already exists, this command sets the permissions in that entry to **r--**.

To replace the entire ACL for file **filea** and add entries:

- Allowing read/write access for users **archer** and **fletcher**
- For the file owner allowing all access

- For the file group allowing read access only
- For others disallowing all access

use this command:

setacl -s user::rwx,user:archer:rw-,user:fletcher:rw-,group::r--,other:--- filea

After this command is executed, the file permission bits are set to **-rwxrw----**. Although the file-owning group has read permission only, the maximum permissions available to all additional user ACL entries and all group ACL entries are read and write, because the two additional user entries both specify these permissions.

To set an ACL for **filea** using a file, **sample.acl**, that contains ACL entries, enter this command:

setacl -f sample.acl filea

Edit the file **sample.acl** to contain:

user::rwx
user:archer:rw-
user:fletcher:rw-
class:rw-
group::r--
other:---

RELATED INFORMATION

Commands: **getacl(1)**.

Functions: **acl(2)**, **aclsort(3)**, **creat(2)**, **creat64(2)**, **getgrid(3)**, **getpwuid(3)**, **mkdir(2)**, **open(2)**, **open64(2)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

This command is an HP extension to the XPG4 Version 2 specification.

NAME

setfilepriv - Sets file privileges for one or more executable files

SYNOPSIS

setfilepriv {**-a**|**-d**} *privilege_value* [{**-a**|**-d**} *privilege_value*]... *file* ...
setfilepriv **-f** *privilege_file file* ...
setfilepriv **-s** *privilege_value*[**,***privilege_value*]... *file* ...

FLAGS

-a Adds the specified *privilege_value* to the file privileges of *file*.

-d Deletes the specified *privilege_value* from the file privileges of *file*.

-f Sets the privileges for the specified file using the privileges entries contained in the file *privilege_file*.

-s Sets the file privileges of *file to privilege_value*, replacing all existing file privileges. Multiple privilege values can be separated by commas. Spaces between values or after commas are not permitted.

Operands

file The pathname of a file for which you want to set privileges.

See DESCRIPTION for information about the requirements for these flags.

DESCRIPTION

The **setfilepriv** command sets file privileges for the specified file or files. A file specified by *file* can be either a Guardian disk file or an OSS regular file, but file privileges are ignored for files that are not executable files, ordinary DLLs, or user libraries.

The values for **privilege_value** are:

PRIVNONE If a file has the PRIVNONE file privilege only, the file has no special privileges. When used with the **-s** flag:

- If PRIVNONE is used alone, the file privileges are reset and the file has no special privileges.
- If PRIVNONE is used with another file privilege, such as PRIVSETID, the PRIVNONE privilege value has no effect and the file privileges are set to the other file privilge value or values you used.

When used with a flag other than the **-s** flag, the PRIVNONE privilege value has no effect.

PRIVSETID If the super ID (255,255 in the Guardian environment, 65535 in the OSS environment) runs an executable file that has this file privilege, the resultant process is permitted to perform a privileged switch (such as by using the **setuid( )** function) to another user ID, group ID, or both to access files in a restricted-access fileset.

PRIVSOARFOPEN
If a locally-authenticated member of the Safeguard SECURITY_OSS_ADMINISTRATOR (SOA) group runs an executable file that has this file privilege, the resultant process is permitted to perform additional system calls needed to back up and restore files in a restricted-access fileset. These system calls include **open( )**, **open64( )**, **creat( )**, **creat64( )**, **link( )**, **remove_oss( )**, **unlink( )**, **rmdir( )**, and **utime( )**,

The **-a** flag adds the privilege specified by *privilege_value* to the file privileges of the file *file*. Using PRIVNONE with this flag has no effect.

The **-d** flag deletes the privilege option specified by *privilege_value* from the file privileges of the file *file*. Using PRIVNONE with this flag has no effect.

The **-s** flag sets the file privileges for the file to the specified value or values, replacing all existing file privileges. Multiple privilege values can be separated by commas. Spaces between values or after commas are not permitted. See also the description of the PRIVNONE file privilege.

The file *privilege_file*, used with the **-f** flag, must contain one or more of the values listed for *privilege_value*, one on each line. The character **#** in a *privilege_file* indicates a comment. All characters, starting with the **#** through the end of the line, are ignored. You can use the output of the **getfilepriv** command to create this file.

Use on Guardian Objects

Specify Guardian files with the **/G** pathname convention.

EXAMPLES

1. To set PRIVSOARFOPEN privileges for the file **/G/SYSTEM/SYSTEM/PRIVOBJ**, enter:

   setfilepriv -s PRIVSOARFOPEN /G/SYSTEM/SYSTEM/PRIVOBJ

2. To add the PRIVSETID privilege to the file **/G/SYSTEM/SYSTEM/PRIVOBJ**, enter:

   setfilepriv -a PRIVSETID /G/SYSTEM/SYSTEM/PRIVOBJ

3. To remove the file privileges for the file **/G/SYSTEM/SYSTEM/PRIVOBJ**, enter:

   setfilepriv -s PRIVNONE /G/SYSTEM/SYSTEM/PRIVOBJ

4. To set both the PRIVSOARFOPEN and the PRIVSETID privileges file privileges for the file **/G/SYSTEM/SYSTEM/MYPRVOB**, enter:

   setfilepriv -s PRIVSOARFOPEN,PRIVSETID /G/SYSTEM/SYSTEM/MYPRVOBJ

5. To add the PRIVSETID privilege to all files in the XYZ directory, enter:

   setfilepriv -a PRIVSETID /G/system/xyz/*

6. To delete the PRIVSOARFOPEN privilege from the file **/G/SYSTEM/SYSTEM/MYPRVOB**, enter:

   setfilepriv -d PRIVSOARFOPEN /G/SYSTEM/SYSTEM/MYPRVOBJ

7. To add the PRIVSETID file privilege and delete PRIVSOARFOPEN file file privilege from the file /G/SYSTEM/SYSTEM/PRIVOBJ, enter:

   setfilepriv -a PRIVSETID -d PRIVSOARFOPEN /G/SYSTEM/SYSTEM/PRIVOBJ

8. To set the privileges for file **exe2** to the same privileges that are set for the file **exe1**, create a file from the output of the **getfilepriv** command for **exe1** and use it with the **-f** flag of the **setfilepriv** command:

   getfilepriv exe1>my_privilege_file
   setfilepriv -f my_privilege_file exe2

NOTES

This command is supported on systems running J06.11 or later J-series RVUs or H06.22 or later H-series RVUs only

Only Members of Safeguard SECURITY-PRV-ADMINISTRATOR (SEC-PRIV-ADMIN or SPA) group are permitted to explicitly set or reset file privileges. For example, only members of the Safeguard SECURITY_OSS_ADMINISTRATOR (SOA) group can use the **initfilepriv** command to set the file privileges needed by the Backup and Restore 2 product to access files in a restricted-access fileset.

File privileges are removed from a file if the file is modified. Any change to the file privileges of a file is audited. File privileges are inherited by child processes created using the **fork( )** function.

If the main executable of a process has a file privilege, then all user libraries and ordinary DLLs loaded into the process must also have that file privilege. Public DLLs and implicit DLLs do not need file privileges to be loaded into a process.

NFS client processes are not allowed to write to a file that has file privileges.

RELATED INFORMATION

Commands: **getfilepriv(1)**, **initfilepriv(1)**.

Functions: **setfilepriv(2)**, **stat(2)**.

STANDARDS CONFORMANCE

This command is an HP extension.

NAME

set_define - Sets values for DEFINE attributes in the working attribute set

SYNOPSIS

set_define -like=*define-name* {*attribute-specs*}...

FLAGS

-like=*define-name*

Specifies a DEFINE name. The name can be from 2 to 24 characters long. The first character must be an equal sign (=), and the second character must be a letter.

DESCRIPTION

The **set_define** command is specific to HP and an OSS shell built-in command. It sets values for the specified DEFINE attributes. The **set_define** command is similiar to the TACL SET DEFINE command. It accepts Guardian attributes. As a result, input must follow Guardian conventions.

attribute-specs

Specifies the names of one or more valid DEFINE attributes and the values they are to have. If the **-like** flag is specified, a DEFINE is created with the attributes and values of the specified *define_name* and modified by the clauses specified by *attribute-specs*. If the **-like** flag is not specified, a DEFINE is created with the attributes and values of the working attribute set and modified by the specified *attribute-specs*. *attribute-specs* is defined as:

class={catalog | defaults | map | search
| sort | spool | subsort | tape}
{*class_attributes*}...

Class Attributes

Certain characters are special in the OSS environment and must be preceded by the escape character or they will not be accepted by the **set_define** command. For a detailed description of the valid class attributes, refer to the SET DEFINE command in the *TACL Reference Manual*.

For **class=catalog** (a CATALOG DEFINE), you must use the escape character in *class-attributes* as follows:

subvol=\$a123

For **class=defaults** (a DEFAULTS DEFINE), you must use the escape character in *class-attributes* as follows:

volume=\$oss.joe
swap=\$null
catalog=\$system.catalog

For **class=map** (a MAP DEFINE), you must use the escape character in *class-attributes* as follows:

file=\$volume.subvolume.file

For **class=search** (a SEARCH DEFINE), you must use the escape character in *class-attributes* as follows:

subvol0=\(a,b,c,d\)
relsubvolO=\\foxii.\$coral.i
subvol2=\(\$data.y2,y22\)

For **class=sort** (a SORT DEFINE), you must use the escape character in *class-attributes* as follows:

```
scratch=\\foxii.\$osf.joe.scratch
swap=\\foxii.\$osf.joe.swap
program=\\foxii.\$osf.joe.suprsort
cpus=\(1,2\)
notcpus=\(0,3\)
subsorts=\(=subsort1,=subsort2\)
```

For **class=spool** (a SPOOL DEFINE), you must use the escape character in *class-attributes* as follows:

```
loc=\\kt22.\$s.#a
```

For **class=subsort** (a SUBSORT DEFINE), you must use the escape character in *class-attributes* as follows:

```
scratch=\\foxii.\$osf.joe.scratch
swap=\\foxii.\$osf.joe.swap
program=\\foxii.\$osf.joe.suprsort
```

For **class=tape** (a TAPE DEFINE), you must use the escape character in *class-attributes* as follows:

```
device=\$device
swap=\(v1,v2\)
program=\\foxii
```

EXAMPLES

1. The following command establishes a working attribute set that describes a tape file residing on three ANSI-standard tape volumes (1, 2, and 3). This file is to be read (USE IN), and the system is to do standard label processing.

   ```
   set_define class=tape labels=
   ansi volume=\(1,2,3\) reels=3 use=in
   ```

2. The following command establishes a working attribute set that contains the attributes common to two DEFINEs that are to be created. Each is a CLASS TAPE DEFINE that describes a tape file residing on volume 30 of an IBM standard labeled tape mounted on tape drive $TAPE2. The two **add_define** commands create the DEFINEs and set the attributes that are unique to each DEFINE, which in this case are the file names MAYRCDS and JUNRCDS.

   ```
   set_define class=tape labelsˆH=ibm fileid=
   \$TAPE device=\$TAPE2 volume=30
   add_define=ONE fileid=MAYRCDS
   add_define=TWO fileid=JUNRCDS
   ```

EXIT VALUES

The following exit values are returned:

0	DEFINEs were set successfully.
>0	An error occurred.

NOTES

The **set_define** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **add_define(1)**, **del_define(1)**, **info_define(1)**, **reset_define(1)**, **show_define(1)**.

STANDARDS CONFORMANCE

The **set_define** command is an HP extension to the XPG4 Version 2 specification.

NAME

sh - Describes the OSS shell

SYNOPSIS

sh [**-i**] [**-c** *command_string* | **-s**] [**+** | **-abCefmnosuvx**] [**+** | **-o**][*option* ...] | [*argument* ...] | [*file*] [*argument* ...]

The OSS shell is an interactive command interpreter and a command programming language. The OSS shell is based on the UNIX Korn shell.

FLAGS

-c *command_string*
Causes **sh** to read commands from *command_string*.

-i Causes **sh** to run as an interactive shell. The **SIGTERM** signal is thus ignored, and the **SIGINT** signal is caught, causing the current command to be terminated and a new prompt to be output.

-r Causes **sh** to run as a restricted shell.

-s Causes **sh** to read commands from standard input. If you do not specify the **-c** flag or do not specify any arguments to **sh** other than flags, **sh** automatically invokes the **-s** flag. The **-c** flag overrides the **-s** flag, however.

The rest of the flags that can be used with **sh** are described under the **set** subcommand, in the subsection **Special sh Commands**.

DESCRIPTION

ksh is an alias for **sh**.

The OSS shell carries out commands either interactively from a terminal keyboard or from a file.

Some important features of the OSS shell are as follows:

- Command aliasing
- Filename substitution
- Tilde substitution
- Command substitution
- Parameter substitution
- Job control
- Inline editing

A file from which the shell carries out commands is usually called a *shell script*, a *shell procedure*, or a *command file*.

A *simple command* is a sequence of words separated by spaces or tabs. A *word* is a sequence of characters that contains no unquoted spaces or tabs. The first word in the sequence (numbered as 0) usually specifies the name of a command. Any remaining words, with a few exceptions, are passed to that command. A *space* refers to both spaces and tabs.

The *value* of a simple command is its exit value if it ends normally or (octal) 200 added to the signal number if it terminates due to a signal. For a list of *status* values, see the **signal()** system call.

A *pipeline* is a sequence of one or more commands separated by a | (vertical bar). In a pipeline, the standard output of each command becomes the standard input of the next command. Each

command runs as a separate process, and the shell waits for the last command to end. A *filter* is a command that reads its standard input, transforms it in some way, then writes it to its standard output. A pipeline normally consists of a series of filters. Although the processes in a pipeline (except the first process) can execute in parallel, they are synchronized to the extent that each program needs to read the output of its predecessor.

The exit value of a pipeline is the exit value of the last command.

A *list* is a sequence of one or more pipelines separated by ; (semicolon), **&** (ampersand), **&&** (two ampersands), or || (two vertical bars) and optionally ended by a ; (semicolon), an **&** (ampersand), a **|&** (coprocess), or a newline. These separators and terminators have the following effects:

; Causes *sequential execution* of the preceding pipeline; the shell waits for the pipeline to finish.

& Causes *asynchronous execution* of the preceding pipeline; the shell does *not* wait for the pipeline to finish.

&& Causes the list following it to be executed *only* if the preceding pipeline returns a 0 (zero) exit value.

|| Causes the list following it to be executed *only* if the preceding pipeline returns a nonzero exit value.

The **cd** command is an exception; if it returns a nonzero exit value, no subsequent commands in a list are executed, regardless of the separator characters.

The **;** and **&** separators have equal precedence, as do **&&** and ||. The single-character separators have lower precedence than the double-character separators. An unquoted newline character following a pipeline functions the same as a ; (semicolon).

Access Control Lists (ACLs)

If the shell creates a file, for example when you redirect **stdout** or **stderr** to a file, and the parent directory for that file has an ACL that contains default ACL entries, the file inherits ACL entries from the parent directory as described in the **acl(5)** reference page.

Comments

The shell treats as a comment any word that begins with a **#** character and ignores that word and all characters following up to the next newline character.

Shell Flow Control Statements

Unless otherwise stated, the value returned by a command is that of the last simple command executed in the command.

for *identifier* [**in** *word*...] **;do** *list* **;done**

Each time a **for** command is executed, *identifier* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, the **for** command executes the **do** *list* once for each positional parameter that is set. (See **Parameter Substitution**.) Execution ends when there are no more words in the list.

select *identifier* [**in** *word*...] **;do** *list* **;done**

Prints on standard error (file descriptor 2) the set of words, each word preceded by a number. If **in** *word*... is omitted, then the positional parameters are used instead. (See **Parameter Substitution**.) The **PS3** prompt is printed and a line is read from the standard input. If this line consists of the number of one of the listed words, then the value of the parameter *identifier* is set to the word corresponding to this number. If this line is empty, the selection list is printed again. If neither of these cases is true, the value of the parameter *identifier* is set to null. The contents of the line read from standard input

is saved in the **REPLY** parameter. The *list* is executed for each selection until a **break** or end-of-file character is encountered.

case *word* **in** [[**(**]
pattern [| *pattern*] ...**)** *list* **;;**] ... **esac**" Executes the list associated with the first pattern that matches *word*. The form of the patterns is the same as that used for filename generation. (See **Filename Generation**.)

if *list* **;then** *list* [**elif** *list* **;then** *list*] ... [**;else** *list*] **;fi**
Executes the list following **if** and, if it returns a 0 (zero) exit status, executes the list following the first **then**. Otherwise, the list following **elif** is executed and, if its value is 0 (zero), the list following the next **then** is executed. Failing that, the **else** list is executed. If no **else** list or **then** list is executed, then the **if** command returns a 0 (zero) exit status.

while *list* **;do** *list* **;done**

until *list* **;do** *list* **;done**
Executes the **while** list repeatedly, and if the exit status of the last command in the list is 0 (zero), executes the **do** list; otherwise the loop terminates. If no commands in the **do** list are executed, then the **while** command returns a 0 (zero) exit status; **until** can be used in place of **while** to negate the loop termination test.

(*list***)** Executes *list* in a separate environment. Note that if two adjacent open parentheses are needed for nesting, a space must be inserted to avoid arithmetic evaluation.

{*list***;**} Executes *list*. Note that unlike the metacharacters **(** and **)**, **{** and **}** are reserved words and must be at the beginning of a line or after a **;** (semicolon) in order to be recognized.

[[*expression*]]
Evaluates *expression* and returns a 0 (zero) exit status when *expression* is **TRUE**. See **Conditional Expressions** for a description of *expression*.

function *identifier* **{***list***;}**

identifier **() {***list***;}**
Defines a function that is referenced by *identifier*. The body of the function is the list of commands between **{** and **}**. (See **Functions**.)

time *pipeline*
Executes *pipeline* and prints the elapsed time as well as the user and system time on standard error.

The following reserved words are recognized only when they appear, without single or double quotation marks, as the first word of a command:

case
do
done
elif
else
esac
fi
for
function
if
select
then

time
until
while
{ }
[[ ]]

Command Aliasing

The first word of each command is replaced by the text of an alias (if an alias for this word was defined). The first character of an alias name can be any nonspecial printable character, but the rest of the characters must be the same as for a valid *identifier*. The replacement string can contain any valid shell script, including the metacharacters previously listed. The first word of each command in the replaced text, other than any that are in the process of being replaced, will be tested for aliases. If the last character of the alias value is a space, the word following the alias will also be checked for alias substitution.

Aliases can be used to redefine special built-in commands but cannot be used to redefine the reserved words previously listed. Aliases can be created, listed, and exported with the **alias** command and can be removed with the **unalias** command. Exported aliases remain in effect for scripts invoked by name, but must be reinitialized for separate invocations of the shell. (See **Invocation**.)

Aliasing is performed when scripts are read, not while they are executed. Therefore, for an alias to take effect, the **alias** definition command has to be executed before the command that references the alias is read.

Aliases are frequently used as shorthand for full pathnames. An option to the aliasing facility allows the value of the alias to be automatically set to the full pathname of the corresponding command. These aliases are called *tracked* aliases.

The value of a tracked alias is defined the first time the corresponding command is looked up and becomes undefined each time the **PATH** environment variable is reset. These aliases remain tracked so that the next subsequent reference will redefine the value. Several tracked aliases are compiled into the shell. The **-h** flag of the **set** command makes each referenced command name into a tracked alias.

The following exported aliases are compiled into the shell, but can be unset or redefined:

autoload='typeset -fu'
command='command '
functions='typeset -f'
hash='alias -t -'
history='fc -l'
integer='typeset -i'
local=typeset
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
type='whence -v'

Tilde Substitution

After alias substitution is performed, each word is checked to see if it begins with an unquoted ~ (tilde). If it does, then the word up to a / (slash) is checked to see if it matches a username in the **/etc/passwd** file. If a match is found, the tilde and the matched name are replaced by the login directory of the matched user. This is called a *tilde substitution*. If no match is found, the

original text is left unchanged. A tilde by itself, or in front of a /, is replaced by the value of the **HOME** parameter. A tilde followed by a **+** (plus sign) or **-** (dash) is replaced by **$PWD** and **$OLDPWD**, respectively.

In addition, tilde substitution is attempted when the value of a variable assignment parameter begins with a tilde.

Command Substitution

The standard output from a command enclosed in parentheses preceded by a dollar sign **$( )** or a pair of **‘‘** (grave accents) can be used as part or all of a word; trailing newlines are removed. In the second (archaic) form, the string between the grave accents is processed for special quoting characters before the command is executed. (See **Quoting**.) The command substitution **$(cat** *file*) can be replaced by the equivalent but faster **$(<***file*). Command substitution of most special commands that do not perform input/output redirection are carried out without creating a separate process. An arithmetic expression enclosed in double parentheses preceded by a dollar sign (**$(( ))**) is replaced by the value of the arithmetic expression within the double parentheses.

Parameter Substitution

A *parameter* is an identifier, one or more digits, or any of the characters *****, **@@@@**, **#**, **?**, **-**, **$**, and **!**. A *named parameter* (a parameter denoted by an identifier) has a *value* and 0 (zero) or more *attributes*. Named parameters can be assigned values and attributes by using the **typeset** special command. The attributes supported by the shell are described later with the **typeset** special command. Exported parameters pass values and attributes to the environment.

The shell supports a 1-dimensional array facility. An element of an array parameter is referenced by a *subscript*. A subscript is denoted by an *arithmetic expression* enclosed with [ ] (brackets). To assign values to an array, use values of subscripts in the range of 0 to 1023. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array will be created if necessary. Referencing an array without a subscript is equivalent to referencing the element 0 (zero).

The value of a named parameter can be assigned by the following:

name=*value* [*name*=*value*]

If the integer attribute, **-i**, is set for *name*, the value is subject to arithmetic evaluation. *Positional parameters*, which are denoted by a number, can be assigned values with the **set** special command. Parameter **$0** is set from argument 0 (zero) when the shell is invoked. The **$** (dollar sign) character is used to introduce substitutable parameters.

${*parameter***}**

Reads all the characters from the **${** (dollar sign left brace) to the matching **}** (right brace) as part of the same word even if it contains braces or metacharacters. The value, if any, of the parameter is substituted. The braces are required when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name or when a named parameter is subscripted. If *parameter* is one or more digits, it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces. If *parameter* is ***** (asterisk) or **@@@@** (at sign), all the positional parameters, starting with **$1**, are substituted (separated by a field separator character). If an array identifier with subscript ***** or **@@@@** is used, the value for each of the elements is substituted (separated by a field separator character).

${#*parameter***}**

Substitutes the number of positional parameters if *parameter* is ***** or **@@@@**; otherwise, the length of the value of the *parameter* is substituted.

${#*identifier*[*****]**}**
Substitutes the number of elements in the array *identifier*.

${*parameter***:-***word***}**
Substitutes the value of *parameter* if it is set and non-null; otherwise, substitutes *word*.

${*parameter***:=***word***}**
Sets *parameter* to *word* if it is not set or is null; the value of the parameter is then substituted. Positional parameters cannot be assigned values in this way.

${*parameter***:?***word***}**
Substitutes the value of *parameter* if it is set and is non-null; otherwise, print *word* and exit from the shell. If *word* is omitted, a standard message is printed.

${*parameter***:+***word***}**
Substitutes *word* if *parameter* is set and is non-null; otherwise, substitutes nothing.

${***parameter***#***pattern***}** | **${*parameter***##***pattern***}**
Causes the value of this substitution to be the value of *parameter* with the matched portion deleted if the shell *pattern* matches the beginning of the value of *parameter*; otherwise the value of *parameter* is substituted. In the first form, the smallest matching pattern is deleted and in the second form, the largest matching pattern is deleted.

${***parameter***%***pattern***}** | **${*parameter***%%***pattern***}**
Causes the value of this substitution to be the value of *parameter* with the matched part deleted if the shell *pattern* matches the end of the value of *parameter*; otherwise, substitute the value of *parameter*. In the first form, the smallest matching pattern is deleted and in the second form, the largest matching pattern is deleted.

If the **:** (colon) is omitted from the previous expressions, then the shell checks only whether *parameter* is set or not.

In the previous expressions, *word* is not evaluated unless it is to be used as the substituted string, so that, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo ${d:-$(pwd)}
```

The following parameters are automatically set by the shell:

(hash mark)
The number of positional parameters in decimal.

- (dash)
Flags supplied to the shell on invocation or by the **set** command.

? (question mark)
The decimal value returned by the last executed command.

$ (dollar sign)
The process number of this shell.

_ (underscore)
Initially, an absolute pathname of the shell or script being executed as passed in the environment. Subsequently, the value is assigned the last argument of the previous command. This parameter is not set for commands that are asynchronous.

! (exclamation point)
The process number of the last background command invoked.

ERRNO The value of *errno* as set by the most recently failed system call. This value is system dependent and is intended for debugging purposes.

LINENO
The line number of the current line within the script or function being executed.

OLDPWD
The previous working directory set by the **cd** command.

OPTARG
The value of the last option argument processed by the **getopts** special command.

OPTIND
The index of the last option argument processed by the **getopts** special command.

PPID The process number of the parent of the shell.

PWD The present working directory set by the **cd** command.

RANDOM
Each time this parameter is referenced, a random integer, uniformly distributed between 0 and 32767, is generated. The sequence of random numbers can be initialized by assigning a numeric value to **RANDOM**.

REPLY This parameter is set by the **select** statement and by the **read** special command when no arguments are supplied.

SECONDS
Each time this parameter is referenced, the number of seconds since shell invocation is returned. If this parameter is assigned a value, then the value returned upon reference will be the value that was assigned plus the number of seconds since the assignment.

The following parameters are used by the shell:

CDPATH
The search path for the **cd** command.

COLUMNS
If this variable is set, the value is used to define the width of the edit window for the shell edit modes and for printing **select** lists.

EDITOR
If the value of this variable ends in **vi** and the **VISUAL** variable is not set, then the corresponding option (see **set** under **Special sh Commands**) will be turned on.

ENV If this parameter is set, then parameter substitution is performed on the value to generate the pathname of the script that will be executed when the shell is invoked. (See **Invocation**.) This file is typically used for **alias** and **function** definitions.

FCEDIT
The default editor name for the **fc** command.

FPATH The search path for function definitions. This path is searched when a function with the **-u** attribute is referenced and when a command is not found. If an executable file is found, then it is read and executed in the current environment.

IFS Internal field separators, normally spaces, tabs, and newlines that are used to separate command words which result from command or parameter substitution and for separating words with the **read** special command. The first character of the **IFS** parameter is used to separate arguments for the **$*** substitution. (See **Quoting**.)

HISTFILE
If this parameter is set when the shell is invoked, then the value is the pathname of the file that will be used to store the command history. (See **Command Reentry**.)

HISTSIZE
If this parameter is set when the shell is invoked, the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128.

HOME The default argument (home directory) for the **cd** command.

LANG Specifies the locale of your system, which is comprised of three parts: language, territory, and code set. The default locale is the **C** locale, which specifies the value **English** for language, **U.S.** for territory, and **ASCII** for code set.

LC_ALL
Specifies the behavior for all aspects of the locale.

LC_COLLATE
Specifies the collating sequence to use when sorting names and when character ranges occur in patterns. The default value is the collating sequence for American English. If absent, the collating sequence can be taken from the **LANG** parameter. If both **LC_COLLATE** and **LANG** are absent, the ANSI C collating sequence is used.

LC_CTYPE
Specifies the character classification information to use on your system. The default value is American English.

LC_MESSAGES
Specifies the language in which system messages appear and the language that the system accepts for user input of yes and no strings. The default is American English.

LC_MONETARY
Specifies the monetary format for your system. The default value is the monetary format for American English.

LC_NUMERIC
Specifies the numeric format for your system. The default value is the numeric format for American English.

LC_TIME
Specifies the date and time format for your system. The default value is the date and time format for American English.

LINES If this variable is set, the value is used to determine the column length for printing **select** lists. Select lists will print vertically until about two-thirds of **LINES** lines are filled.

LOCPATH
Specifies a series of colon-separated search rules that describe where to look for locales. These rules override the default search path of **/usr/lib/nls/loc**.

NLSPATH
Specifies a list of directories to search to find message catalogs.

PATH The search path for commands. (See **Execution**.) You cannot change **PATH** if executing under **rsh**, except in **.profile**.

PS1 The value of this parameter is expanded for parameter substitution to define the primary prompt string which by default is the **$** (dollar sign). The **!** (exclamation point) in the primary prompt string is replaced by the command number. (See **Command Reentry**.)

PS2 Secondary prompt string, by default > (right angle bracket).

PS3 Selection prompt string used within a **select** loop, by default **#?** (number sign, question mark).

PS4 The value of this parameter is expanded for parameter substitution and precedes each line of an execution trace. If omitted, the execution trace prompt is **+** (plus sign).

SHELL The pathname of the shell is kept in the environment.

TMOUT
If set to a value greater than 0 (zero), the shell terminates if a command is not entered within the prescribed number of seconds after issuing the **PS1** prompt. (Note that the shell can be compiled with a maximum bound for this value that cannot be exceeded.)

TZ Current value for the time zone, if any.

VISUAL
If the value of this variable ends in **vi**, the corresponding option (see the **set** command in **Special sh Commands**) will be turned on.

The shell gives default values to **PATH**, **PS1**, **PS2**, **TMOUT**, and **IFS**, while **HOME**, **SHELL**, and **ENV** are not set by the shell.

Interpretation of Spaces
After parameter and command substitution, the results of substitutions are scanned for the field separator characters (those found in **IFS**), and split into distinct arguments where such characters are found. Explicit null arguments (**‘‘** or **’’**) are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

Filename Generation
Following substitution, each command *word* is scanned for the characters ***** (asterisk), **?** (question mark), and **[ ]** (brackets), unless the **-f** option was set. If one of these characters appears, the word is regarded as a pattern. The word is replaced with lexicographically sorted filenames that match the pattern. If no filename is found that matches the pattern, the word is left unchanged. When a pattern is used for filename generation, the **.** (dot) character at the start of a filename or immediately following a / (slash), as well as the / character itself, must be matched explicitly. In other instances of pattern matching, the / and **.** are not treated specially.

***** Matches any string, including the null string.

? Matches any single character.

[...] Matches any one of the enclosed characters. In an expression such as [**a-z**], the **-** (dash) means "through" according to the current collating sequence. The collating sequence is determined by the value of the **LC_COLLATE** environment variable. If the first character following the **[** (left bracket) is a **!** (exclamation point), then any character not enclosed is matched. A **-** can be included in the character set by putting it as the first or last character.

A *pattern_list* is a list of one or more patterns separated from each other with a **|** (vertical bar). Composite patterns can be formed with one or more of the following:

?(*pattern_list***)**
Optionally matches any one of the given patterns.

(pattern_list***)**
Matches zero or more occurrences of the given patterns.

+(*pattern_list***)**
Matches one or more occurrences of the given patterns.

@@@@(*pattern_list***)**
Matches exactly one of the given patterns.

!(*pattern_list***)**
Matches anything, except one of the given patterns.

Character Classes

You can use the following notation to match filenames within a range indication:

[:*charclass***:]**

This format instructs the system to match any single character belonging to *charclass*; the defined classes correspond to **ctype( )** subroutines as follows:

alnum
alpha
blank
cntrl
digit
graph
lower
print
punct
space
upper
xdigit

Your locale might define additional character properties, such as the following:

[:vowel:]

The preceding character class could be TRUE for **a**, **e**, **i**, **o**, **u**, or **y**. You could then use **[:vowel]** inside a **set** construction to match any vowel. Refer to **The LC_CTYPE Category** section of the **locale** file format reference page for more information.

Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > <newline> <space> <tab>

Each of the *metacharacters* previously listed has a special meaning to the shell and causes termination of a word unless quoted. A character can be *quoted* (that is, made to stand for itself) by preceding it with a \ (backslash). The pair **newline** is ignored. All characters enclosed between a pair of **‘’** (single quotes) are quoted. A single quote cannot appear within single quotes.

Inside **""** (double quotes) parameter and command substitution occurs and \ quotes the characters \, **‘**, **’**, and **$**. The meaning of **$*** and **$@@@@** is identical when not quoted or when used as a parameter assignment value or as a filename. However, when used as a command argument, **’$*’** is equivalent to **’$1***d***$2***d*. . .**’**, where *d* is the first character of the **IFS** parameter, whereas **’$@@@@’** is equivalent to **’$1’ ’$2’** . . . Inside **‘‘** (grave accents) \ (backslash) quotes the characters \, **‘**, and **$**. If the grave accents occur within double quotes, then \ also quotes the **’** (single quote) character.

The special meaning of reserved words or aliases can be removed by quoting any character of the reserved word. The recognition of function names or special command names listed later cannot be altered by quoting them.

Arithmetic Evaluation

An ability to perform integer arithmetic is provided with the **let** special command. Evaluations are performed using *long* arithmetic. Constants are of the form [*base*#]*n*, where *base* is a decimal number between 2 and 36 representing the arithmetic base and *n* is a number in that base. If *base* is omitted, then base 10 is used.

An arithmetic expression uses the same syntax, precedence, and associativity of expression of the C language. All the integral operators, other than **++**, **--**, **?:**, and **,** are supported. Named parameters can be referenced by name within an arithmetic expression without using the parameter substitution syntax. When a named parameter is referenced, its value is evaluated as an arithmetic expression.

An internal integer representation of a named parameter can be specified with the **-i** option of the **typeset** special command. Arithmetic evaluation is performed on the value of each assignment to a named parameter with the **-i** attribute. If you do not specify an arithmetic base, the first assignment to the parameter determines the arithmetic base. This base is used when parameter substitution occurs.

Because many of the arithmetic operators require quoting, an alternative form of the **let** command is provided. For any command that begins with a **((**, all the characters until a matching **))** are treated as a quoted expression. More precisely, **((**...**))** is equivalent to **let** "...".

Note that **((**...**))** is a command with a return value, whereas **$((**...**))** is the way to put the string representation of the value of an arithmetic expression into the command line (that is, it is like a **$** variable).

Prompting

When used interactively, the shell prompts with the value of **PS1** before reading a command. If at any time a newline character is typed and further input is needed to complete a command, then the secondary prompt (that is, the value of **PS2**) is issued.

Conditional Expressions

A *conditional expression* is used with the **[[** compound command to test attributes of files and to compare strings. Word splitting and filename generation are not performed on the words between **[[** and **]]**. Each expression can be constructed from one or more of the following unary or binary expressions:

-a *file* **TRUE**, if *file* exists.

-b *file* **TRUE**, if *file* exists and is a block-special file.

-c *file* **TRUE**, if *file* exists and is a character-special file.

-d *file* **TRUE**, if *file* exists and is a directory.

-f *file* **TRUE**, if *file* exists and is an ordinary file.

-g *file* **TRUE**, if *file* exists and has its **setgid** bit set.

-G *file* **TRUE**, if *file* exists and its group ID matches the effective group ID of this process.

-h *file* **TRUE**, if *file* exists and is a symbolic link.

-k *file* **TRUE**, if *file* exists and has its sticky bit set.

-L *file* **TRUE**, if *file* exists and is a symbolic link.

-n *string* **TRUE**, if length of *string* is nonzero.

-o *option*
TRUE, if option named *option* is on.

-O *file* **TRUE**, if *file* exists and is owned by the effective user ID of this process.

-p *file* **TRUE**, if *file* exists and is a FIFO special file or a pipe.

-r *file* **TRUE**, if *file* exists and is readable by current process.

-s *file* **TRUE,** if *file* exists and has size greater than 0 (zero).

-S *file* **TRUE**, if *file* exists and is a socket.

-t *file_des*
TRUE, if file descriptor number *file_des* is open and associated with a terminal device.

-u *file* **TRUE**, if *file* exists and has its **setuid** bit set.

-w *file* **TRUE**, if *file *O exists and is writable by current process.*

-x *file* **TRUE**, if *file* exists and is executable by current process. If *file* exists and is a directory, then the current process has permission to search in the directory.

-z *string* **TRUE**, if length of *string* is 0 (zero).

file1 **-nt** *file2*
TRUE, if *file1* exists and is newer than *file2*.

file1 **-ot** *file2*
TRUE, if *file1* exists and is older than *file2*.

file1 **-ef** *file2*
TRUE, if *file1* and *file2* exist and refer to the same file.

string **=** *pattern*
TRUE, if *string* matches *pattern*.

string **!=** *pattern*
TRUE, if *string* does not match *pattern*.

string1 < *string2*
TRUE, if *string1* collates before *string2*.

string1 > *string2*
TRUE, if *string1* collates after *string2*.

expression1 **-eq** *expression2*
TRUE, if *expression1* is equal to *expression2*.

expression1 **-ne** *expression2*
TRUE, if *expression1* is not equal to *expression2*.

expression1 **-lt** *expression2*
TRUE, if *expression1* is less than *expression2*.

expression1 **-gt** *expression2*
TRUE, if *expression1* is greater than *expression2*.

expression1 **-le** *expression2*
TRUE, if *expression1* is less than or equal to *expression2*.

expression1 **-ge** *expression2*
TRUE, if *expression1* is greater than or equal to *expression2*.

A compound expression can be constructed from these primitives by using any of the following, listed in decreasing order of precedence.

(*expression***)**
TRUE, if *expression* is **TRUE**. Used to group expressions.

! *expression*
TRUE if *expression* is **FALSE**.

expression1 **&&** *expression2*
TRUE, if *expression1* and *expression2* are both **TRUE**.

expression1 || *expression2*
TRUE, if either *expression1* or *expression2* is **TRUE**.

Input/Output

Before a command is executed, you can redirect its input and output by using a special notation interpreted by the shell. The following can appear anywhere in a simple command or can precede or follow a command and are *not* passed on to the invoked command. Command and parameter substitution occurs before *word* or *digit* is used, except as noted in the following text. Filename generation occurs only if the pattern matches a single file and interpretation of spaces is not performed.

<*word* Use file *word* as standard input (file descriptor 0).

>*word* Use file *word* as standard output (file descriptor 1). If the file does not exist, it is created. If the file exists, and the **noclobber** option is on, this causes an error; otherwise, it is truncated to 0 (zero) length.

>|*word* Same as >, except that it overrides the **noclobber** option.

>>*word* Use file *word* as standard output. If the file exists, output is appended to it (by first seeking to the End-of-File); otherwise, the file is created.

<>*word* Open file *word* for reading and writing as standard input.

<<[-]*word*
The shell input is read up to a line that is the same as *word*, or to an End-of-File. No parameter substitution, command substitution, or filename generation is performed on *word*. The resulting document, called a *here document*, becomes the standard input. If any character of *word* is quoted, then no interpretation is placed upon the characters of the document; otherwise, parameter and command substitution occurs, **\newline** is ignored, and \ must be used to quote the characters \, **$**, **‘**, and the first character of *word*. If **-** is appended to <<, then all leading tabs are stripped from *word* and from the document.

<**&***digit* The standard input is duplicated from file descriptor *digit* (see **dup(2)**). The standard output is duplicated using **>&** *digit*.

<**&-** The standard input is closed. The standard output is closed using **>&-**.

<**&p** The input from the coprocess (or background process) is moved to standard input.

>**&p** The output to the coprocess is moved to standard output.

If one of the preceding redirections is preceded by a digit, then the file descriptor number referred to is that specified by the digit (instead of the default 0 or 1). For example:

```
... 2>&1
```

means file descriptor 2 is to be opened for writing as a duplicate of file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates each redirection in terms of the (*file descriptor*, *file*) association at the time of evaluation. For example:

```
... 1>fname >&1
```

first associates file descriptor 1 with file **fname**. It then associates file descriptor 2 with the file associated with file descriptor 1 (that is, **fname**). If the order of redirections is reversed, file descriptor 2 is associated with the terminal (assuming file descriptor 1 is) and then file descriptor 1 is associated with file **fname**.

If a command is followed by **&** and job control is not active, the default standard input for the command is the empty **/dev/null** file. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Environment

The *environment* is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The names must be identifiers and the values are character strings. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value and marking it **export**. Executed commands inherit the environment. If you modify the values of these parameters or create new ones, using the **export** or **typeset -x** commands, they become part of the environment. The environment used by any executed command is thus composed of any name-value pairs originally inherited by the shell, whose values can be modified by the current shell, plus any additions that must be noted in the **export** or **typeset -x** commands.

You can augment the environment for any simple command or function by prefixing it with one or more parameter assignments. A parameter assignment argument is a word of the form *identifier=value*.

Thus, the following two expressions are equivalent (as far as the execution of *command* is concerned):

TERM=450 *command argument ...*

(export TERM; TERM=450; *command argument ...*)

Functions

The **function** reserved word is used to define shell functions. Shell functions are read in and stored internally. Alias names are resolved when the function is read. Functions are executed like commands with the arguments passed as positional parameters. (See **Execution**.)

Functions execute in the same process as the caller and share all files and the present working directory with the caller. Traps caught by the caller are reset to their default action inside the function. A trap condition that is not caught or ignored by the function causes the function to terminate and the condition to be passed on to the caller. A trap on **EXIT** set inside a function is executed after the function completes in the environment of the caller. Ordinarily, variables are shared between the calling program and the function. However, the special command **typeset** used within a function defines local variables whose scope includes the current function and all functions it calls.

The special command **return** is used to return from function calls. Errors within functions return control to the caller.

Function identifiers can be listed with the **-f** or **+f** option of the **typeset** special command. The text of functions is also listed with **-f**. Functions can be undefined with the **-f** option of the **unset** special command.

Ordinarily, functions are unset when the shell executes a shell script. The **-xf** option of the **typeset** command allows a function to be exported to scripts that are executed without a separate invocation of the shell. Functions that need to be defined across separate invocations of the shell should be specified in the **ENV** file with the **-xf** option of **typeset**.

Jobs

If the **monitor** option of the **set** command is turned on, an interactive shell associates a job with each pipeline. It keeps a table of current jobs, printed by the **jobs** command, and assigns them small integer numbers. When a job is started asynchronously with **&**, the shell prints a line that looks like:

```
[1] 1234
```

This line indicates that the job, which was started asynchronously, was job number 1 and had one (top-level) process, whose process ID was 1234.

If you are running a job and want to do something else, you can enter the Suspend key sequence (normally **<Ctrl-z>**), which sends a **SIGTSTP** signal to the current job. The shell then normally indicates that the job has been stopped, and it prints another prompt. You can then manipulate the state of this job, putting it in the background with the **bg** command, or run some other commands and then eventually bring the job back into the foreground with the foreground command **fg**. The job suspension takes effect immediately, and corresponds to the Interrupt key sequence in that pending output and unread input are discarded. A special key sequence, **<Ctrl-y>**, does not generate a **SIGTSTP** signal until a program attempts to read it. (See the **read(2)** reference page for more information.) This key sequence can usefully be typed ahead when you have prepared some commands for a job that you wish to stop after it has read them.

A job being run in the background will stop if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by issuing the **stty tostop** command. If you set this **tty** option, then background jobs will stop when they try to produce output like they do when they try to read input.

There are several ways to refer to jobs in the shell. A job can be referred to by the process ID of any process of the job, or by one of the following:

%*job_number*
The job with the given number.

%*string* Any job whose command line begins with *string*.

%?*string*
Any job whose command line contains *string*.

%% Current job.

%+ Equivalent to **%%**.

%- Previous job.

This shell knows immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work.

When the monitor mode is on, each background job that is completed triggers any trap set for **CHLD**.

When you try to leave the shell while jobs are stopped or running, you are warned that `You have stopped(running) jobs.` You can use the **jobs** command to see what they are. If you do this or immediately try to exit again, the shell does not warn you a second time, and the stopped jobs are terminated.

Signals

The **SIGINT** and **SIGQUIT** signals for an invoked command are ignored if the command is followed by **&** and job **monitor** option is not active. Otherwise, signals have the values inherited by the shell from its parent (but see also the **trap** command).

Execution

Each time a command is executed, the previous substitutions are carried out. If the command name matches one of the shell built-in commands it is executed within the current shell process. Next, the command name is checked to see if it matches one of the user-defined functions. If it does, the positional parameters are saved and then reset to the arguments of the function call. When the function is completed or issues a **return**, the positional parameter list is restored and any trap set on **EXIT** within the function is executed. The value of a function is the value of the last command executed. A function is also executed in the current shell process. If a command name is not a special command or a user-defined function, a process is created and an attempt is made to execute the command via **exec**.

The **PATH** shell parameter defines the search path for the directory containing the command. Alternative directory names are separated by a **:** (colon). The default path is **/usr/bin:** (specifying **/usr/bin**, and the current directory in that order). The current directory can be specified by two or more adjacent colons, or by a colon at the beginning or end of the path list. If the command name contains a / (slash), then the search path is not used. Otherwise, each directory in the path is searched for an executable file.

If the file has execute permission but is not a directory or an **a.out** file, it is assumed to be a file containing shell commands. A subshell is spawned to read it. All nonexported aliases, functions, and named parameters are removed in this case. If the shell command file does not have read

permission, or if the **setuid** and/or **setgid** bits are set on the file, the shell executes an agent whose job it is to set up the permissions and execute the shell with the shell command file passed down as an open file. A command in parentheses is executed in a subshell without the removal of nonexported quantities.

Command Reentry

The text of the last **HISTSIZE** (default 128) commands entered from a terminal device is saved in a history file. The **$HOME/.sh_history** file is used if the **HISTFILE** variable is not set or is not writable. A shell can access the commands of all *interactive* shells that use the same named **HISTFILE**. The **fc** special command is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by giving the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to **fc**, then the value of the **FCEDIT** parameter is used. If **FCEDIT** is not defined, then **/usr/bin/ed** is used. The edited commands are printed and reexecuted upon leaving the editor. The editor name **-** (dash) is used to skip the editing phase and to reexecute the command. In this case, a substitution parameter of the form *old=new* can be used to modify the command before execution. For example, if **r** is aliased to **'fc -e -'**, then typing '**r bad=good c**' reexecutes the most recent command, which starts with the letter **c**, replacing the first occurrence of the string **bad** with the string **good**.

Inline Editing Options

Normally, each command line entered from a terminal device is simply typed followed by a newline character (**<Return>** or linefeed). If the **vi** option is active, you can edit the command line. To be in this edit mode, **set** the corresponding option. An editing option is automatically selected each time the **VISUAL** or **EDITOR** variable is assigned a value ending in the option name.

The editing features require that the terminal accept **<Return>** as carriage-return without linefeed and that a space must overwrite the current character on the screen. ADM terminal users should set the space-advance switch to **Space**. Hewlett-Packard series 2621 terminal users should set the straps to **bcGHxZ etX**.

The editing modes create the impression that the user is looking through a window at the current line. The window width is the value of **COLUMNS** if it is defined, otherwise it is 80 bytes. If the line is longer than the window width minus 2, a mark is displayed at the end of the window to notify the user. As the cursor moves and reaches the window boundaries, the window will be centered about the cursor. The mark is a **>** (right angle bracket) if the line extends on the right side of the window, a **<** (left angle bracket) if the line extends on the left side of the window, and an ***** (asterisk) if the line extends on both sides of the window.

The search commands in each edit mode provide access to the history file. Only strings are matched, not patterns, although if the leading character in the string is a **^** (circumflex), the match is restricted to begin at the first character in the line.

The vi Editing Mode

There are two typing modes. Initially, when you enter a command you are in the **input** mode. To edit, the user enters **control** mode by typing **<Esc>** (ASCII **033**) and moves the cursor to the place needing correction and then inserts or deletes characters or words as needed. Most control commands accept an optional repeat count prior to the command. When in **vi** mode on most systems, canonical processing is initially enabled and the commands are echoed again if the speed is 1200 baud or greater, if it contains any control characters, or if less than 1 second has elapsed since the prompt was printed. The Escape character terminates canonical processing for the remainder of the command and the user can then modify the command line.

This scheme of using two typing nodes has the advantages of canonical processing with the type-ahead echoing of raw mode. If the option **viraw** is also set, the terminal always has canonical processing disabled. This mode is implicit for systems that do not support two alternative End-of-Line delimiters, and can be helpful for certain terminals.

Input Edit Commands

By default the editor is in input mode.

Erase (User-defined Erase character as defined by the **stty** command, often **<Ctrl-h>** or **#**.) Deletes the previous character.

<Ctrl-w>
Deletes the previous space-separated word.

<Ctrl-d>
Terminates the shell (at the beginning of a line only).

<Ctrl-v>
Escapes the next character. Editing characters and the user's Erase or Kill characters can be entered in a command line or in a search string if preceded by a **<Ctrl-v>**. **<Ctrl-v>** removes the next character's editing features (if any).

\ Escapes the next Erase or Kill character.

Motion Edit Commands

These commands move the cursor:

[*count*]**l** Cursor forward (right) one character.

[*count*]**w** Cursor forward one word. A word is a string of characters delimited by spaces or tabs.

[*count*]**W**
Cursor to the beginning of the next word that follows a space.

[*count*]**e** Cursor to the end of the word.

[*count*]**E** Cursor to end of the current space-delimited word.

[*count*]**h** Cursor backward (left) one character.

[*count*]**b** Cursor backward one word.

[*count*]**B** Cursor to the preceding space-delimited word.

[*count*]| Cursor to the column *count*.

[*count*]**f***c* Finds the next character *c* in the current line.

[*count*]**F***c*
Finds the previous character *c* in the current line.

[*count*]**t***c* Equivalent to **f** followed by **h**.

[*count*]**T***c*
Equivalent to **F** followed by **l**.

[*count*]**;** Repeats *count* times, the last single character find command: **f**, **F**, **t**, or **T**.

[*count*]**,** Reverses the last single character find command *count* times.

0 Cursor to the start of the line.

^ Cursor to the first nonspace character in the line.

$ Cursor to the end of the line.

Search Edit Commands

These commands access your command history.

[*count*]**k** Fetches the previous command. Each time **k** is entered, the previous command back in time is accessed.

[*count*]**-** Equivalent to **k**.

[*count*]**j** Fetches the next command. Each time **j** is entered, the next command forward in time is accessed.

[*count*]**+** Equivalent to **j**.

[*count*]**G** Fetches the command number *count*. The default is the least recent **history** command.

/string Searches backward through history for a previous command containing the specified *string*. *string* is terminated by **<Return>** or a newline character. If the specified string is preceded by a ^ (circumflex), the matched line must begin with *string*. If *string* is null, the previous string is used.

?*string* Same as / (slash) except that the search is in the forward direction.

n Searches for next match of the last pattern to the **/** or **?** commands.

N Searches for next match of the last pattern to the **/** or **?** commands, but in reverse direction. Searches the command history for the *string* entered by the previous / command.

Text Modification Edit Commands

These commands modify the line.

a Enters input mode and enters text after the current character.

A Appends text to the end of the line. Equivalent to **$a**.

[*count*]**c***motion*

c[*count*]*motion*
Deletes the current character through the character to which *motion* would move the cursor, and enters input mode. If *motion* is **c**, the entire line is deleted and input mode is entered.

C Deletes the current character through the end of line, and enters input mode. Equivalent to **c$**.

S Equivalent to **cc**.

D Deletes the current character through the end of line. Equivalent to **d$**.

[*count*]**d***motion*

d[*count*]*motion*
Deletes the current character through the character to which *motion* would move. If *motion* is **d**, the entire line is deleted.

i Enters input mode and inserts text before the current character.

I Inserts text before the beginning of the line. Equivalent to **0i**.

[*count*]**P** Places the previous text modification before the cursor.

[*count*]**p** Places the previous text modification after the cursor.

R Enters input mode and replaces characters on the screen with the characters you type, overlay fashion.

[*count*]**r***c*
Replaces the *count* characters, starting at the current cursor position with *c* and advancing the cursor.

[*count*]**x** Deletes the current character.

[*count*]**X** Deletes the preceding character.

[*count*]**.** Repeats the previous text modification command.

[*count*]**~** Inverts the case of the *count* characters, starting at the current cursor position and advancing the cursor.

[*count*]_ Causes the *count* word of the previous command to be appended and input mode entered. The last word is used if *count* is omitted.

***** Causes an ***** (asterisk) to be appended to the current word and filename generation to be attempted. If no match is found, it sounds the bell. Otherwise, the word is replaced by the matching pattern and input mode is entered.

\ Filename completion. Replaces the current word with the longest common prefix of all filenames matching the current word with an * (asterisk) appended. If the match is unique, a / (slash) is appended if the file is a directory; a space is appended if the file is not a directory.

Miscellaneous vi Commands

[*count*]**y***motion*

y[*count*]*motion*
Yanks the current character through the character to which *motion* would move the cursor and puts the characters into the delete buffer. The text and cursor are unchanged.

Y Yanks from current position to the end of line. Equivalent to **y$**.

u Undoes the last text-modifying command.

U Undoes all the text-modifying commands performed on the line.

[*count*]**v** Returns the command **fc -e vi** *count* in the input buffer. If *count* is omitted, the current line is used.

<Ctrl-l> Performs a linefeed and prints the current line. Effective only in control mode.

<Ctrl-j> Executes the current line, regardless of mode (newline).

<Ctrl-m>
Executes the current line, regardless of mode (enter).

Sends the line after inserting a **#** (number sign) in front of the line. Useful for causing the current line to be inserted in the history without being executed.

= Lists the filenames that match the current word if an * (asterisk) is appended to it.

@@@@*letter*
Searches the alias list for an alias by the name *_letter* . If an alias of this name is defined, its value is inserted in the input queue for processing.

Special sh Commands

Shell built-in commands are executed by the OSS shell and run entirely within the shell process. A subshell process is not created for shell built-in commands as it is for a command that is not a shell built-in command.

The following shell built-in commands also have counterparts that are regular OSS commands having the same names:

echo.1
kill.1
pwd.1
read.1

The shell built-in command is the default. To run the regular version of a command (instead of the shell built-in version) specify the command as follows:

/bin/*command_name*

To make the regular version the default, create an alias to the regular version.

The shell built-in version and the regular version of a command may not behave the same way or have the same flags.

The shell commands described below are executed in the shell process. Input/output redirection is permitted.

DESCRIPTION

:[*argument* ...]
The command only expands arguments. It is used when a command is needed, as in the **then** condition of an **if** command, but nothing is to be done by the command.

Parameter assignment lists that precede the command remain in effect when the command completes.

I/O redirections are processed after parameter assignments.

Errors cause a script that contains the commands so marked to abort.

. *file* [*argument* ...]
Reads the complete file and executes the commands. The commands are executed in the current shell environment. The search path specified by **PATH** is used to find the directory containing *file*. Unlike normal command search, however, the file searched for by the **.** command need not be executable. If any arguments are specified, they become the positional parameters. Otherwise, the positional parameters are unchanged. If no readable file is found, a noninteractive shell aborts; an interactive shell writes a diagnostic message to standard error, but this condition is not considered a syntax error. The exit status is the exit status of the last command executed, or a 0 (zero) if no command is executed.

Parameter assignment lists that precede the command remain in effect when the command completes.

I/O redirections are processed after parameter assignments.

Errors cause a script that contains the commands so marked to abort.

add_define
Creates DEFINEs for the Guardian environment. An HP extension.

alias Creates or lists aliases.

bg Puts each specified *job* into the background.

break Exits from the enclosing **for**, **while**, **until**, or **select** loop.

cd Changes the current directory.

continue
Resumes the next iteration of the enclosing **for**, **while**, **until**, or **select** loop.

del_define
Deletes DEFINEs from the current shell process. An HP extension.

echo Sends the string given as an argument to the standard output.

eval Reads arguments as input to the shell and executes arguments as commands.

exec Executes commands specified as arguments

exit Causes the shell to exit.

export Marks names automatic export to the shell environment.

fc Lists or edits and reexecutes commands previously entered to an interactive shell.

fg Moves processes into the foreground.

getopts Checks argument for legal options.

hash Affects the way the shell remembers the locations of utilities.

history Lists the contents of the **history** file, which contains a list of previously executed commands.

info_define
Displays information about DEFINEs. An HP extension.

jobs Lists information about jobs.

kill Sends either the **TERM** signal or the specified signal to the specified jobs or processes.

let Evaluates arguments as arithmetic expressions.

print The shell output mechanism; prints arguments to standard output as described for **echo**.

pwd Prints the current working directory to standard output.

read The shell input mechanism.

readonly
The variable names given as arguments are marked read only. These names cannot be changed by subsequent assignment.

reset_define
Restores the attributes of a DEFINE to their original settings. An HP extension.

return Causes a shell function to return to the invoking script.

set Sets parameters.

set_define
Sets the values for DEFINE attributes. An HP extension.

shift Renames positional parameters.

show_define
Displays values of DEFINE attributes. An HP extension.

times Prints the accumulated user and system times for the shell and for processes run from the shell.

trap Defines variables to be read and executed when the shell receives the specified signals.

type Returns the location of commands.

typeset Sets attributes and values for shell parameters.

umask Sets the user file-creation mask to *mask*.

unalias Removes alias definions.

unset Erases values assigned to variables.

wait Waits for the specified process and reports its termination status.

whence Indicates how names would be interpreted if used as commands.

Invocation

If the shell is invoked by **exec**, and the first character of argument zero (**$0**) is **-** (dash), the shell is assumed to be a login shell and commands are read from **/etc/profile** and then from either **.profile** in the current directory or **$HOME/.profile**, if either file exists. Next, commands are read from the file named by performing parameter substitution on the value of the **ENV** environment variable, if the file exists. If the **-s** flag is not present and *argument* is present, a path search is performed on the first *argument* to determine the name of the script to execute. The script *argument* must have read permission and any **setuid** and **getgid** settings are ignored. Commands are then read, as described in the following text.

See the **FLAGS** section for a complete description of flags that can be interpreted by the shell when it is invoked.

FILES

/etc/profile	System profile.
$HOME/.profile	User profile.

NOTES

1. If a command is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will execute the original command. Use the **hash** command to correct this situation.

2. When the shell encounters the >> characters, it does not open the file in append mode; instead, the shell opens the file for writing and seeks to the end.
3. Failure (nonzero exit status) of a special command preceding a || symbol prevents the list following || from executing.
4. If a command that is a *tracked alias* is executed, and then a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell continues to **exec** the original command. Use the **-t** flag of the **alias** command to correct this situation.
5. Using the **fc** built-in command within a compound command causes the whole command to disappear from the history file.
6. The built-in **.file** command reads the whole file before any commands are executed. Therefore, the **alias** and **unalias** commands in the file do not apply to any functions defined in the file.
7. Traps are not processed while a job is waiting for a foreground process. Thus, a trap on **CHLD** is not executed until the foreground job terminates.
8. The shell displays the following progress message if it needs to retry the fork operation during an attempt at process creation:

   ```
   sh: Resource temporarily unavailable....
       will retry fork() for MAX 62 secs...
   ```

   If the indicated time passes before the fork operation is successful, the shell returns the following message:

   ```
   /bin/-sh:  sh: tdm_fork() failed with errno EAGAIN:
                  cannot fork too many processes
   ```

EXIT VALUES

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. Otherwise, the shell returns the exit status of the last command executed. (See also the **exit** command, described previously.) If the shell is being used noninteractively, execution of the shell file is abandoned. Run-time errors detected by the shell are reported by printing the command or function name and the error condition. If the line number that the error occurred on is greater than 1, the line number is also printed in **[ ]** (brackets) after the command or function name.

RELATED INFORMATION

Commands: **cd(1)**, **chmod(1)**, **echo(1)**, **env(1)**, **ksh(1)**, **setacl(1)**, **stty(1)**, **test(1)**, **umask(1)**, **vi(1)**.

Functions: **exec(2)**, **fcntl(2)**, **fork(2)**, **ioctl(2)**, **lseek(2)**, **pipe(2)**, **rand(3)**, **umask(2)**, **ulimit(3)**, **wait(2)**.

Files: **locale(4)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

The following commands are HP extensions to the shell built-in commands of the XPG4 Vewrsion 2 specification. They are described in detail in their own reference pages:

add_define

Creates DEFINEs for the Guardian environment.

del_define
Deletes DEFINEs from the current shell process.

info_define
Displays information about DEFINEs.

reset_define
Restores the attributes of a DEFINE to their original settings.

set_define
Sets the values for DEFINE attributes.

show_define
Displays values of DEFINE attributes.

NAME

shift - Shifts positional parameters

SYNOPSIS

shift [*n*]

DESCRIPTION

The **shift** command moves the specified positional parameter so that it takes the place of the specified parameter to its left. Thus, the positional parameters from **$***n***+1** ... are renamed **$***n* ... . The parameters represented by the numbers **$#** down to **$#-***n*+1 are unset, and the parameter **#** is updated to reflect the new number of positional parameters. The argument *n* can be any arithmetic expression that evaluates to a nonnegative number less than or equal to **$#**.

EXIT VALUES

The exit status is greater than 0 (zero) if *n* is greater than **$#**; otherwise, it is 0 (zero).

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **shift** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

NAME

show_define - Displays the values of DEFINE attributes

SYNOPSIS

show_define [*attribute-name*]

DESCRIPTION

The **show_define** command is specific to HP and a shell built-in command. It displays the values associated with the specified DEFINE attributes. It can display all attribute values that are currently set or defaulted, and it can show all attributes in the current working set and the value of each attribute. The **show_define** command is similiar to the TACL INFO DEFINE command.

The **show_define** command has the same output as the TACL INFO DEFINE command. Refer to the *TACL Reference Manual* for more information on the output of **show_define**.

attribute-name

Specifies the attribute whose value is to be displayed. Valid *attribute-names* are:

catalog | defaults | map | search |
sort | spool | subsort | tape

If no argument is specified with the **show_define** command, all attribute names and their current values for the working attribute set are displayed. (Optional attributes that have no current value are listed with a blank value.)

Environment Variables

LANG Determines the locale to use for the locale categories when neither the **LC_ALL** variable nor the corresponding environment variable (beginning with **LC_**) specify a locale.

LC_ALL

Determines the locale to be used to override any values for locale categories specified by the **LANG** variable or any environment variable whose name begins with **LC_**.

LC_CTYPE

Determines the locale for interpretation of bytes of text data as characters (for example, single-byte as opposed to multibyte characters in arguments).

LC_MESSAGES

Determines the locale that should be used to affect the format and contents of diagnostic messages written to the standard error file and of informational messages written to the standard output file.

EXAMPLES

1. To display the value currently set for the tape attribute, enter:

   show_define tape

   This command might result in the following display:

   DEVICE =\KT22.$TAPE

EXIT VALUES

The following exit values are returned:

0 DEFINE attribute values were displayed successfully.

>0 An error occurred.

NOTES

The **show_define** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **add_define(1)**, **del_define(1)**, **info_define(1)**, **reset_define(1)**, **set_define(1)**.

STANDARDS CONFORMANCE

The **show_define** command is an HP extension to the XPG4 Version 2 specification.

NAME

sleep - Suspends execution for a specified time

SYNOPSIS

sleep *seconds*

DESCRIPTION

The **sleep** command suspends execution of a process for the interval specified by *seconds*, which can range from 0 to 2,147,483,647 seconds.

seconds can be entered as a non-negative decimal, octal, or hexadecimal value.

EXAMPLES

1. To display a message at 4-minute intervals for 20 minutes, create a shell script called **remind** containing the following:

   ```
   for i
   do
   sleep 240; echo $i
   sleep 240; echo $i
   sleep 240; echo $i
   sleep 240; echo $i
   sleep 240; echo $i
   done
   ```

   To display the message `Try calling NHK` at 4-minute intervals, enter:

   ```
   remind 'Try calling NHK'
   ```

2. To run a command at regular intervals, create a shell script containing the following:

   ```
   while true
   do
           date
           sleep 60
   done
   ```

   This displays the date and time once a minute.

NOTES

If **sleep** receives a **SIGALARM** signal before process execution has resumed, **sleep** terminates normally with a 0 (zero) exit status. (See the **sleep(3)** reference page for more information.)

RELATED INFORMATION

Functions: **alarm(3)**, **pause(3)**, **sigaction(2)**, **sleep(3)**.

NAME

sort - Sorts or merges files

SYNOPSIS

Current syntax

sort

[**-m**]
[**-o** *output_file*]
[**-Abdfinru**]
[**-k** *keydef*] ...
[**-t** *character*]
[**-T** *directory*]
[**-y**][*kilobytes*]
[**-z** *record_size*] ...
file ...

sort

-c
[**-u**]
[**-Abdfinru**]
[**-k** *keydef*] ...
[**-t** *character*]
[**-T** *directory*]
[**-y**][*kilobytes*]
[**-z** *record_size*] ...
file ...

Obsolescent syntax

sort

[**-Abcdfimnru**]
[**-o** *output_file*]
[**-t** *character*]
[**-T** *directory*]
[**-y**][*kilobytes*]
[**-z** *record_size*]
[**+***fskip*][**.***cskip*]
[**-***fskip*][**.***cskip*]
[**-bdfinr**] ...
file ...

FLAGS

The **sort** command sorts lines in its input files and writes the result to standard output.

The **-d**, **-f**, **-i**, **-n**, and **-r** flags override the default ordering rules. When ordering flags appear independent of any key field specifications, the requested field ordering rules are applied globally to all sort keys. When attached to a specific key (see **-k**), the specified ordering flags override all global ordering flags for that key. In the obsolescent forms, if one or more of these flags follows a **+***fskip* flag, it affects only the key field specified by that preceding flag.

-A Sorts on a byte-by-byte basis using each character's encoded value. On some systems, extended characters will be considered negative values, and so sort before ASCII characters. If you are sorting ASCII characters in a non-C/POSIX locale, this flag performs much faster.

-b Ignores leading spaces and tabs when determining the starting and ending positions of a restricted sort key. If the **-b** flag is specified before the first **-k** flag, the **-b** flag is

applied to all **-k** flags on the command line; otherwise, the **-b** flag can be independently attached to each **-k** *field_start* or *field_end* argument.

-c Checks that the input is sorted according to the ordering rules specified in the flags and the collating sequence of the current locale. No output is produced; only the exit code is affected.

-d Specifies that only spaces and alphanumeric characters (according to the current setting of **LC_TYPE**) are significant in comparisons.

-f Treats all lowercase characters as their uppercase equivalents (according to the current setting of **LC_TYPE**) for the purposes of comparison.

-i Sorts only by printable characters (according to the current setting of **LC_TYPE**).

-k *keydef* Specifies one or more (up to 10) restricted sort key field definitions. This flag replaces the obsolescent *+fskip.cskip* and *-fskip.cskip* flags. A field comprises a maximal sequence of non-separating characters and, in the absence of the **-t** flag, any preceding field separator.

The format of a key field definition is as follows:

field_start[*type*][**,***field_end*[*type*]]

where the *field_start* and *field_end* arguments define a key field that is restricted to a portion of the line, and *type* is a modifier specified by **b**, **d**, **f**, **i**, **n**, or **r**. The **b** modifier behaves like the **-b** flag, but applies only to the *field_start* or *field_end* argument to which it is attached. The other modifiers behave like their corresponding flags, but apply only to the key field to which they are attached; these modifiers have this effect if specified with *field_start*, *field_end* or both. Modifiers attached to a *field_start* or *field_end* argument override any specifications made by the flags. A missing *field_end* argument means the last character of the line.

The *field_start* portion of the *keydef* argument takes the following form:

field_number[**.***first_character*]

Fields and characters within fields are numbered starting with 1. The *field_number* and *first_character* pieces, interpreted as positive decimal integers, specify the character to be used as part of a sort key. If *first_character* is not specified, the default is the first character of the field.

The *field_end* portion of the *keydef* argument takes the following form:

field_number[**.***last_character*]

The *field_number* is the same as that described for *field_start*. The *last_character* argument, interpreted as a nonnegative decimal integer, specifies the last character to be used as part of the sort key. If *last_character* evaluates to 0 (zero) or is not specified, the default is the last character of the field specified by *field_number*.

If **-b** is in effect, characters within a field are counted from the first nonspace character in the field. (This applies separately to *first_character* and *last_character*.)

If **-k** is not specified, the default sort key is the entire line.

When there are multiple key fields, later keys are compared only after all earlier keys compare as equal. Except when the **-u** flag is specified, lines that otherwise compare as equal are ordered as though none of the flags **-d**, **-f**, **-i**, **-n**, or **-k** were present (but with **-r** still in effect, if it was specified) and with all bytes in the lines significant to the comparison.

The algorithm for the **-k** flag can be summarized as follows:

```
/*
 * -ka.b,c.d = if d==0 then +(a-1).(b-1) -c.d
 *              else +(a-1).(b-1) -(c-1).d
 */
```

-m Merges only (assumes sorted input).

-n Sorts any initial numeric strings (consisting of optional spaces, optional dashes, and zero or more digits with optional radix character and thousands separator, as defined by the current locale) by arithmetic value. An empty digit string is treated as zero; leading zeros and signs on zeros do not affect ordering.

-o *output_file*
Directs output to *output_file* instead of standard output. *output_file* can be the same as one of the input files.

-r Reverses the order of the specified sort.

-t *character*
Sets the field separator character to *character*. The *character* argument is not considered to be part of a field (although it can be included in a sort key). Each occurrence of *character* is significant (for example, two consecutive occurrences of *character* delimit an empty field). To specify the tab character as the field separator, you must enclose it in **' '** (single quotes).

The default field separator is one or more spaces.

-T *directory*
Places all the temporary files that are created in *directory*.

-u Suppresses all but one in each set of equal lines. Ignored characters (such as leading tabs and spaces) and characters outside of sort keys are not considered in this type of comparison.

If used with the **-c** flag, **-u** checks that there are no lines with duplicate keys, in addition to checking that the input file is sorted.

-y [*kilobytes*]
Starts the **sort** command using *kilobytes* of main storage and adds storage as needed. (If *kilobytes* is less than the minimum storage size or greater than the maximum, the minimum or maximum is used instead.) If the **-y** flag is omitted, the **sort** command starts with the default storage size; **-y 0** starts with minimum storage, and **-y** (with no value) starts with the maximum storage. The amount of storage used by the **sort** command has a significant impact on performance. Sorting a small file in a large amount of storage is wasteful.

-z *record_size*
Prevents abnormal termination if lines being sorted are longer than the default buffer size can handle. When the **-c** or **-m** flags are specified, the sorting phase is omitted and a system default size buffer is used. If sorted lines are longer than this size, **sort** terminates abnormally. The **-z** option specifies that the longest line be recorded in the sort phase so that adequate buffers can be allocated in the merge phase. *record_size* must be a value in bytes equal to or greater than the number of bytes in the longest line to be merged.

+fskip.cskip
Specifies the start position of a key field. See the **-k** flag for a description of the current way to perform this operation. (Obsolescent)

The *fskip* variable specifies the number of fields to skip from the beginning of the input line, and the *cskip* variable specifies the number of additional characters to skip to the right beyond that point. For both the starting point (+*fskip.cskip*) and the ending point (-*fskip.cskip*) of a sort key, *fskip* is measured from the beginning of the input line, and *cskip* is measured from the last field skipped. If you omit *.cskip*, .0 (zero) is assumed. If you omit *fskip*, 0 (zero) is assumed. If you omit the ending field specifier (-*fskip.cskip*), the end of the line is the end of the sort key.

You can supply more than one sort key by repeating +*fskip.cskip* and -*fskip.cskip*. In cases where you specify more than one sort key, keys specified further to the right on the command line are compared only after all earlier keys are sorted. For example, if the first key is to be sorted in numerical order and the second according to the collating sequence, all strings that start with the number 1 are sorted according to the collating order before the strings that start with the number 2. Lines that are identical in all keys are sorted with all characters significant. You can also specify different flags for different sort keys in multiple sort keys.

-*fskip***.***cskip*

Specifies the end position of a key field. See the **-k** flag for a description of the current way to perform this operation. (Obsolescent)

DESCRIPTION

The **sort** command performs one of the following functions:

1. Sorts lines of all the named files together and writes the result to the specified output.
2. Merges lines of all the named (presorted) files together and writes the result to the specified output.
3. Checks that a single input file is correctly presorted.

Comparisons are based on one or more sort keys extracted from each line of input (or the entire line if no sort keys are specified), and are performed using the collating sequence of the current locale.

The **sort** command treats all of its input files as one file when it performs the sort. A **-** (dash) in place of a filename specifies standard input. If you do not specify a filename, it sorts standard input.

The **sort** command can handle a variety of collation rules typically used in Western European languages, including primary/secondary sorting, one-to-two character mapping, N-to-one character mapping, and ignore-character mapping. To summarize briefly:

Primary/Secondary Sorting

In this system, a group of characters all sort to the same primary location. If there is a tie, a secondary sort is applied. For example, in French, the plain and accented **a**'s all sort to the same primary location. If two strings collate to the same primary location, the secondary sort goes into effect. These words are in correct French order:

à
abord
âpre
après
âpreté
azur

NOTE: If you are viewing this reference page online using the **man** command, the special characters are not displayed. See this reference page in the *Open System Services Shell and*

Utilities Reference Manual.

One-to-Two Character Mappings
This system requires that certain single characters be treated as if they were two characters. For example, in German, the **ß** (scharfes-S) is collated as if it were **ss**.

NOTE: If you are viewing this reference page online using the **man** command, the special characters are not displayed. See this reference page in the *Open System Services Shell and Utilities Reference Manual*.

N-to-One Character Mappings
Some languages treat a string of characters as if it were one single collating element. For example, in Spanish, the **ch** and **ll** sequences are treated as their own elements within the alphabet. (**ch** comes between **c** and **d** in the alphabet, and **ll** comes between **l** and **m**.)

Ignore-Character Mappings
In some cases, certain characters may be ignored in collation. For example, if **-** were defined as an ignore-character, the strings **re-locate** and **relocate** would sort to the same place.

The results that you get from **sort** depend on the collating sequence as defined by the current setting of the **LC_COLLATE** environment variable. The configuration files for collation and character classification information are **/usr/lib/nls/loc/src/***locale***.src**.

A field is one or more characters bounded by the beginning of a line and the current field separator, or one or more characters bounded by a field separator on either side. The space character is the default field separator.

Lines longer than 1024 bytes are truncated by **sort**. The maximum number of fields on a line is 10.

EXAMPLES
The following examples apply to the C locale, unless it is specifically stated otherwise.

1. To perform a simple sort, enter:

   sort fruits

   This displays the contents of **fruits** sorted in ascending lexicographic order. This means that the characters in each column are compared one by one, including spaces, digits, and special characters.

   For instance, if **fruits** contains the text:

   banana
   orange
   Persimmon
   apple
   %%banana
   apple
   ORANGE

   then **sort fruits** displays:

   ```
   %%banana
   ORANGE
   Persimmon
   apple
   apple
   banana
   orange
   ```

This order follows from the fact that in the ASCII collating sequence, symbols (such as **%**) precede uppercase letters, and all uppercase letters precede the lowercase letters. If you are using a different collating order, your results may be different.

2. To group lines that contain uppercase and special characters with similar lowercase lines, and remove duplicate lines, enter:

   sort -d -f -u fruits

   The **-u** flag tells **sort** to remove duplicate lines, making each line of the file unique. This displays:

   ```
   apple
   %%banana
   orange
   Persimmon
   ```

   Note that not only was the duplicate **apple** removed, but **banana** and **ORANGE** were removed as well. The **-d** flag told **sort** to ignore symbols, so **%%banana** and **banana** were considered to be duplicate lines and **banana** was removed. The **-f** flag told **sort** not to differentiate between uppercase and lowercase, so **ORANGE** and **orange** were considered to be duplicate lines and **ORANGE** was removed.

   When the **-u** flag is used with input that contains nonidentical lines that are considered by **sort** (due to other flags) to be duplicates, there is no way to predict which lines **sort** will keep and which it will remove.

3. To sort as in Example 2, but remove duplicates unless capitalized or punctuated differently, enter:

   sort -u -k 1df -k 1 fruits

   Flags appearing between sort key specifiers apply only to the specifier preceding them. There are two sorts specified in this command line. **-k 1df** specifies the first sort, of the same type done with **-d -f** in Example 3. Then **-k 1** performs another comparison to distinguish lines that are not actually identical. This prevents **-u**, which applies to both sorts because it precedes the first sort key specifier, from removing lines that are not exactly identical to other lines.

   Given the **fruits** file shown in Example 1, the added **-k 1** distinguishes **%%banana** from **banana** and **ORANGE** from **orange**. However, the two instances of **apple** are exactly identical, so one of them is deleted.

   ```
   apple
   %%banana
   banana
   ORANGE
   orange
   Persimmon
   ```

4. To specify a new field separator, enter:

   sort -t : -k 2 vegetables

   This sorts **vegetables**, comparing the text that follows the first colon on each line. **-t :** tells **sort** that colons separate fields. **-k 2** tells **sort** to ignore the first field and to compare from the start of the second field to the end of the line. If **vegetables** contains:

   yams:104
   turnips:8

potatoes:15
carrots:104
green beans:32
radishes:5
lettuce:15

then **sort -t : -k 2 vegetables** displays:

```
carrots:104
yams:104
lettuce:15
potatoes:15
green beans:32
radishes:5
turnips:8
```

Note that the numbers are not in ascending order. This is because a lexicographic sort compares each character from left to right. In other words, **3** comes before **5** so **32** comes before **5**.

5. To sort on more than one field, enter:

   sort -t : -k 2n -k 1r vegetables

   This performs a numeric sort on the second field (**-k 2n**) and then, within that ordering, sorts the first field in reverse collating order (**-k 1r**). The output looks like this:

   ```
   radishes:5
   turnips:8
   potatoes:15
   lettuce:15
   green beans:32
   yams:104
   carrots:104
   ```

   The lines are sorted in numeric order; when two lines have the same number, they appear in reverse collating order.

6. To replace the original file with the sorted text, enter:

   sort -o vegetables vegetables

   The **-o vegetables** flag stores the sorted output into the file **vegetables**.

7. To collate using Spanish rules, set the **LC_COLLATE** (or **LANG**) environment variable to a Spanish locale, and then use **sort** in the regular way, enter:

   sort sp.words

   If an input file named **sp.words** contains the following Spanish words:

   dama
   loro
   chapa
   canto
   mover
   chocolate
   curioso
   llanura

The sorted file looks like this:

```
canto
curioso
chapa
chocolate
dama
loro
llanura
mover
```

If you sort the file in the default C locale, the output looks like this:

```
canto
chapa
chocolate
curioso
dama
llanura
loro
mover
```

FILES

/usr/lib/nls/loc/src/*locale***.src**
Configuration files.

EXIT VALUES

The **sort** command returns the following exit values:

0	All input files were output successfully, or **-c** was specified and the input file was correctly sorted.
1	Under the **-c** flag, the file was not ordered as specified, or if the **-c** and **-u** flags were both specified, two input lines were found with equal keys.
>1	An error occurred.

RELATED INFORMATION

Commands: **comm**(1), **join**(1), **uniq**(1).

Files: **locale**(4).

NAME

split - Splits a file into pieces of a specified size

SYNOPSIS

Current syntax

split

[**-l** *line_count*]
[**-a** *suffix*]
[*file* [*prefix*] | **-**]

split

-b *n*[**k** | **m**]
[**-a** *suffix*]
[*file* [*prefix*] | **-**]

Obsolescent syntax

split

[*-number*]
[**-a** *suffix*]
[*file* [*prefix*] | **-**]

FLAGS

-a *suffix* Uses *suffix* letters to form the suffix portion of the filenames of the split files. If **-a** is not specified, the default suffix length is two letters.

If the sum of the length of the prefix and the suffix would create a filename exceeding **NAME_MAX** - 2 bytes, an error occurs. In this case, **split** exits with a diagnostic message and no files are created.

-b *n*[**k** | **m]**

Splits a file into pieces of the specified size. The *n* argument without **k** or **m** specifies the size of each piece in bytes. The *n***k** argument specifies the size of each piece in *n**1024 bytes. The *n***m** argument specifies the size of each piece in *n**1048576 bytes.

-l *line_count*

Specifies the number of lines in each output file. The *line_count* argument is an unsigned decimal integer. The default value is 1000. If the input does not end with a newline character, the partial line is included in the last output file.

-number (Obsolescent) Specifies the number of lines in each output file. The default value is 1000 lines per output file.

DESCRIPTION

The **split** command reads *file* and writes it in *number*-line pieces (the default *number* is 1000 lines) to a set of output files. The size of the output files can be modified by using the **-b** or **-l** flag.

Each output file is created with a unique suffix consisting of exactly *suffix* lowercase letters from the POSIX locale. The letters of the suffix are used as if they were a base-26 digit system, with the first suffix to be created consisting of all **a** characters, the second with **b** replacing the last **a**, and so forth, until a suffix of all **z**s is created.

By default, the names of the output files are **x**, followed by a two-character suffix from the character set as described in the preceding paragraph, starting with **aa**, **ab**, **ac**, and so on, and continuing until the suffix **zz**, for a maximum of 676 files.

The filename specified by *prefix* and *suffix* cannot be longer than the value of **NAME_MAX** minus two (from the **limits.h** header file).

If the number of files required is greater than the maximum allowed by the effective suffix length (such that the last allowable file would be larger than the requested size), **split** fails after creating the last possible file with a valid suffix. The **split** command does not delete the files it creates with valid suffixes. If the file limit is not exceeded, the last file created contains the remainder of the input file and thus might be smaller than the requested size.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

Operands

file Specifies the pathname of the file to be read. If you do not specify an input file, or if you specify **-** (dash) in place of *file*, **split** reads the standard input file.

prefix Specifies the prefix to be used for the resulting output filenames.

EXAMPLES

1. To split a file into 1000-line segments, enter:

   split book

   This splits **book** into 1000-line segments named **xaa**, **xab**, **xac**, and so forth.

2. To split a file into 50-line segments and specify the filename prefix, enter:

   split -l 50 book sect

   This splits **book** into 50-line segments named **sectaa**, **sectab**, **sectac**, and so forth.

EXIT VALUES

The **split** command returns the following values:

0 (zero) The command completed successfully.

>0 An error occurred.

RELATED INFORMATION

Commands: **join(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions, with the following exception:

- The length of an output filename cannot be greater than **NAME_MAX** - 2 bytes.

NAME

strings - Finds printable strings in binary files

SYNOPSIS

strings [**-a**] [**-t** *format*] [**-n** *number*] [*file* ...]

FLAGS

-a Searches an entire object file, linkfile, or loadfile rather than just the:

- Code, data, and extended data areas of a TNS or accelerated object file
- **.data**, **.rodata**, and **.sdata** areas of a TNS/R native non-position-independent (non-PIC) or PIC linkfile or loadfile
- **.data**, **.rdata**, **.sdata**, **.rconst** areas of a TNS/E native PIC linkfile or loadfile

-n *number* Sets the minimum string length to *number* rather than the default length of 4.

-t *format* Writes each string preceded by its byte offset from the start of the file. The format depends on the single character used as the *format* argument, as follows:

d The offset is written in decimal.

o The offset is written in octal.

x The offset is written in hexadecimal.

DESCRIPTION

The **strings** command looks for strings in an ASCII or binary file.

The **strings** command looks for printable strings in regular files and writes the strings to the standard output file. A printable string is any sequence of four (the default value) or more printable characters terminated by a newline or NULL character.

By default, **strings** scans for strings in the:

- code, data, and extended data areas of a TNS or accelerated object file
- **.data**, **.rodata**, and **.sdata** areas of a TNS/R native non-position-independent (non-PIC) or PIC linkfile or loadfile
- **.data**, **.rdata**, **.sdata**, **.rconst** areas of a TNS/E native PIC linkfile or loadfile

Operand

file Specifies the pathname of a regular file to be used as input. The input files can be regular files of any format. If you do not specify a *file* operand, **strings** reads from the standard input file.

Environment Variables

This utility supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

Standard Output

Strings that are found are written to the standard output file, one per line.

- When the **-t** flag is not specified, the format of the output is:

  `"%s",` *string*

- When the **-t** or **-t o** flag is specified, the format of the output is:
  `"%o %s",` *byte-offset, string*
- When the **-t x** flag is specified, the format of the output is:
  `"%x %s",` *byte-offset, string*
- When the **-t d** flag is specified, the format of the output is:
  `"%d %s",` *byte-offset, string*

EXIT VALUES

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

RELATED INFORMATION

Commands: **od(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions, except that the HP implementation does not support the obsolescent form of the command.

The scanning of code and extended code areas of files is an HP extension to the XPG4 Version 2 specification.

NAME

strip - Removes unnecessary information from loadfiles or executable files

SYNOPSIS

strip [[**-Wa**] | [**-Ws**]] *file* ...

FLAGS

-Wa Removes the TNS/R Accelerator region or TNS/E Object Code Accelerator region of an accelerated object file.

-Ws Removes the Inspect symbols region of a TNS or accelerated object file.

DESCRIPTION

The **strip** utility removes information that is considered unnecessary for proper execution of a file:

TNS object files
: This information includes the Binder and Inspect symbols regions.

accelerated object files
: This information includes the Binder, debugger symbols, and TNS/R Accelerator or TNS/E Object Code Accelerator regions.

TNS/R native non-position-independent code (non-PIC) or PIC loadfiles
: This information includes the debugger and **nld** or **ld** symbols regions.

TNS/E native PIC loadfiles
: This information includes the **eld** symbols region.

Using **strip** is the same as using the **-s** flag with the **c89** utility.

If neither **-Wa** nor **-Ws** is specified, **strip** removes:

- Both the Binder and debugger symbols regions from TNS and accelerated object files
- The symbols region from native loadfiles

Removing the Binder and debugger symbols regions of an executable object file or loadfile does not affect the behavior or outcome of the program. However:

- Files stripped of the debugger symbols region cannot be debugged symbolically.
- TNS and accelerated object files stripped of the Binder region cannot be relinked by the Binder.

Environment Variables

This utility supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXIT VALUES

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

RELATED INFORMATION

Commands: **ar(1)**, **c89(1)**, **eld(1)**, **ld(1)**, **nld(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

The **-W** flags are HP extensions to the XPG4 Version 2 specification.

NAME

stty - Sets terminal characteristics

SYNOPSIS

stty [**-a** | **-g**] [**-f** *special_device*]

stty [**-f** *special_device*] [*argument* ...]

The **stty** command sets or reports on terminal I/O characteristics for the device that is its standard input.

FLAGS

-a Writes to standard output all the current settings for the terminal.

-f *special_device*
Allows you to specify an alternative terminal or tty device. Normally, the **stty** command works on your standard input.

-g Writes to standard output the current settings in an unspecified form that can be used as input arguments to another **stty** command on the same system. (The form used will not contain any characters that would require quoting; therefore, word expansion by the shell is avoided.)

DESCRIPTION

Without flags or arguments specified, **stty** reports the settings of certain characteristics, usually those that differ from implementation-defined defaults. Otherwise, **stty** modifies the terminal state according to the specified arguments. Some combinations of arguments are mutually exclusive on some terminal types.

Control Modes

The following arguments are available to set the terminal characteristics:

parenb (**-parenb**)
Enables (disables) parity generation and detection.

parodd (**-parodd**)
Selects odd (even) parity.

cs7 cs8 Selects character size, if possible.

hupcl (**-hupcl**)
Stops asserting modem control (does not stop asserting modem control) on last close.

hup (**-hup**)
Same as **hupcl** (**-hupcl**).

cstopb (**-cstopb**)
Uses two (one) stop bits per character.

cread (**-cread**)
Enables (disables) the receiver.

clocal (**-clocal**)
Assumes a line without (with) modem control.

Input Modes

ignbrk (**-ignbrk**)
Ignores (does not ignore) break on input.

brkint (**brkint**)
Signals (does not signal) **INTR** on break.

parmrk (**-parmrk**)
Marks (does not mark) parity errors.

istrip (**-istrip**)
Strips (does not strip) input characters to seven bits.

inlcr (**-inlcr**)
Maps (does not map) newline to carriage-return on input.

igncr (**-igncr**)
Ignores (does not ignore) carriage-return on input.

icrnl (**-icrnl**)
Maps (does not map) carriage-return to newline on input.

ixon (**-ixon**)
Enables (disables) Start/Stop output control. Output from the system is stopped when the system receives Stop and started when the system receives Start.

ixoff (**-ixoff**)
Requests that the system send (not send) Stop characters when the input queue is nearly full and Start characters to resume data transmission.

Local Modes

isig (**-isig**)
Enables (disables) the checking of characters against the special control characters **INTR**, **QUIT**, and **SUSP**.

icanon (**-icanon**)
Enables (disables) canonical input (Erase and Kill processing).

tostop (**-tostop**)
Sends (does not send) **SIGTTOU** for background output. This stops or allows output from background jobs to the terminal.

echo (**-echo**)
Echoes back (does not echo back) every character typed.

echoe (**-echoe**)
Causes the Erase character to (to not) visually erase the last character in the current line from the display, if possible.

echok (**-echok**)
Echoes (does not echo) newline after the Kill character.

echonl (**-echonl**)
Echoes (does not echo) newline, even if **echo** is disabled.

noflsh (**-noflsh**)
Disables (enables) flush after **INTR**, **QUIT**, **SUSP**.

Control Assignments

special_character string

Sets *special_character* to *string*. The special character is set to the first character in *string* and subsequent characters are ignored, with the following exceptions:

- The *string*s **undef** and **^-** set the special character to **{_POSIX_VDISABLE}** if it is in effect for the device.
- The *string* **^?** sets the special character to <**Delete**>.
- Any other *string* beginning with the character ^ sets the special character to the control character corresponding to the second character of *string* (subsequent characters are ignored). For example, the *string* **^c** sets the special character to **^C**; the string **^zq** sets the special character to **^Z**.

  Note that you can set a special character to a control character in two ways: by entering the control character itself or by entering ^ and another character. This allows you to enter a control character that is already assigned to a special character without entering that special character; for example, you can enter **^C** even if it is already assigned to the **intr** special character, by entering ^ and then **c**.

  Recognized *special_character*s include **dsusp**, **eof**, **eol**, **eol2**, **erase**, **discard**, **status**, **intr**, **kill**, **lnext**, **quit**, **reprint**, **start**, **stop**, **susp**, and **werase**.

min *number*

time *number*

Sets the value of **min** or **time** to *number*. **MIN** and **TIME** are used in Noncanonical mode input processing (**-icanon**).

Combination Modes

saved settings

Sets the current terminal characteristics to the saved settings produced by **-g**.

evenp or **parity**

Enables **parenb** and **cs7**; disables **parodd**.

oddp Enables **parenb**, **cs7**, and **parodd**.

-parity, **-evenp**, **-oddp**

-parity enables **parodd**, **-evenp** sets odd parity (that is, disables **parenb**), and **-oddp** sets even parity (that is, sets **cs8**).

nl (**-nl**) Enables (disables) **icrnl** and **onlcr**. **-nl** also unsets **inlcr**, **igncr ocrnl**, and **onlret**.

sane Resets all modes to some reasonable values.

Compatibility Mode

ek Resets Erase and Kill characters back to system defaults.

lfkc (**-lfkc**)

Same as **echok**.

flow (**-flow**)

Same as **ixon** (**-ixon**).

tandem (**-tandem**)

Same as **ixoff** (**-ixoff**).

crterase (**-crterase**)

crtbs (**-crtbs**)
Same as **echoe** (**-echoe**).

ctlecho (**-ctlecho**)
Same as **echoctl** (**-echoctl**).

litout (**-litout**)

all

everything
Same as **-a**.

nohang (**-nohang**)
Does not (does) send **HANGUP** signal if carrier drops.

nul-fill Does character fill and uses Null character.

del-fill Does character fill and uses Delete character.

If no options are specified, an unspecified subset of the information displayed for the **-a** flag is displayed.

Control-characters are displayed as follows:

control_character = *value*

where *value* is either the character, or some visual representation of the character if it is nonprinting, or the string **undef** if the character is disabled.

EXIT VALUES

The **stty** utility exits with one of the following values:

0 The terminal options were read or set successfully.

>0 An error occurred.

RELATED INFORMATION

Commands: **tty**(1).

Functions: **tcsetattr**(3), **tcgetattr**(3), **ttyname**(3)

Files: **termios**(4)

NAME

su - Substitutes user ID temporarily and changes password

SYNOPSIS

su [[**-f**] | [**-**]
[*username* | *username*,*password*[,]]
[**-c** *string*]]

FLAGS

- Starts a login shell using **/bin/sh**; **/etc/profile** and **.profile** are processed if they exist, and the current working directory is set to the home directory (the initial working directory) of the new user ID. The **ENV** and **HOME** environment variables are set accordingly. If the **-** option is not specified, the **ENV** and **HOME** environment variables are not changed.

-f The **-f** option allows the command to skip certain steps to load faster. A new shell (/bin/sh) is started.

-c *string* Specifies a string to be passed to the shell as a command to execute. This option must follow a specified *username* value.

The string value is subject to all of the rules of character substitution and is usually enclosed in quotation marks. Refer to the **osh(1)** reference page either online or in the *Open System Services Shell and Utilities Reference Manual* for a discussion of using **-c** options.

DESCRIPTION

The **su** command can change:

- The login name and therefore the user ID of the current shell
- The password for the user ID of the current shell
- The login name and therefore the user ID used for a new shell
- The password used for the user ID of a new shell

Security is enforced by requiring the user to complete a normal login dialog for the new login name.

The new user ID stays in force until the shell exits. The new password stays in force until changed again.

Changing the Password

The password can be changed either on the command line or during the login dialog by specifying a comma immediately after the current password. See the **EXAMPLES** section of this reference page for more information. For information about valid passwords, including information about special characters in passwords, see the USER_AUTHENTICATE_ procedure in the *Guardian Procedure Calls Reference Manual*.

The ability to change the password of the new login name can be disabled by setting the BLINDLOGON attribute of the new login name.

Operands

username Specifies the login name to which the command applies. If no value is specified for *username*, the **SUPER.SUPER** username is assumed; this usually corresponds to the super ID of 65535. Only users who belong to group number **255** can issue **su** to become the super ID, even with the appropriate password for the super ID.

,*password*[**,**] Specifies the password for the new login name. The initial comma cannot have any spaces before it. When this option is used, login dialog is bypassed unless the value entered is incorrect for the login name specified. Passwords that contain characters that have special meaning to the shell must be enclosed in quotes if the password is specified on the command line. For information about shell metacharacters, see "Quoting" in the **sh(1)** reference page.

If the comma is specified after the value, the dialog for changing the password of the new login name must be completed.

To remind super ID users of their responsibilities, the shell substitutes a **#** (number sign) for its usual prompt.

Environment Variables

This command supports the following environment variables: **ENV**, **HOME**, and **LOGNAME**.

EXAMPLES

1. The following command starts a new login shell for the login name **myalias**, runs the **/etc/profile** and **.profile** files, and changes the user to the initial working directory of **myalias**:

   su - myalias

2. The following command does not start a new shell for the user named **SUPER.SUPER** but does begin the editing of the system-wide profile file:

   su SUPER.SUPER -c "vi /etc/profile"

3. The following command changes the password for the login name **myalias** in the current shell:

   su myalias
   ```
   Password:  oldpw,
   Enter new password:  newpw
   Reenter new password:  newpw
   ```
   The password values **oldpw** and **newpw** are not echoed during this dialog.

4. The following command also changes the password for the login name **myalias** in the current shell:

   su myalias,oldpw,
   ```
   Enter new password:  newpw
   Reenter new password:  newpw
   ```
   The new password value **newpw is not echoed during this dialog.**

RELATED INFORMATION

Commands: **sh(1)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

sum - Displays the checksum and block count of a file

SYNOPSIS

sum [**-o** | **-r**] [*file* ...]

FLAGS

-o Computes the checksum using a word-by-word computation algorithm.

-r Computes the checksum using the more rigorous byte-by-byte computation algorithm. This is the default action.

DESCRIPTION

The **sum** command reads *file* and calculates a 16-bit checksum and the number of 512-byte blocks in the file. If the *file* operand is omitted, **sum** reads the standard input file.

The checksum and number of blocks are written to the standard output file.

Environment Variables

The following environment variables affect the execution of the **sum** command:

LANG Provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the corresponding value from the default locale is used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables have been defined.

LC_ALL If set to a nonempty string value, overrides the values of all the other internationalization variables.

LC_CTYPE Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multibyte characters in arguments).

LC_MESSAGES
Determines the locale to be used to affect the format and contents of diagnostic messages written to the standard error file.

NLSPATH Determines the location of message catalogs for the processing of **LC_MESSAGES**.

EXAMPLES

To display the checksum of **datafile** and the number of blocks in this file, enter:

```
sum datafile
```

If the checksum of **datafile** is 1605 and if the file contains 3 blocks, **sum** displays:

```
1605 3 datafile
```

NOTES

The **sum** command is typically used to determine whether a file that was copied or communicated over transmission lines is an exact copy of the original.

Portable applications should use the **cksum** command instead of the **sum** command.

EXIT VALUES

The following exit values are returned:

0 Successful completion.

>0	An error occurred.

RELATED INFORMATION

Commands: **cksum(1)**, **wc(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification, with the following exception:

- The **-o** flag is an HP extension to the specification.

Section 9. User Commands (t - u)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letters **t** and **u**.

NAME

tail - Writes a file from a specified point

SYNOPSIS

Current Syntax

tail

[**-f** | **-r**]
[**-c** [+|-]*number* | **-n** [+|-]*number*]
[*file*]

Obsolescent Syntax

tail

[+[*number*][*unit*] | -[*number*][*unit*][**f**]]
[**-f** | **-r**]
[*file*]

FLAGS

-c [+|-]*number*

Writes *number* bytes of the file. The sign specified before *number* affects the location in the file from which to begin writing:

+ (plus) Writing begins relative to the beginning of the file.

- (minus) Writing begins relative to the end of the file.

no sign Writing begins relative to the end of the file.

The value *number* begins at 1; that is, **-c +1** writes the first byte of the file, and **-c -1** writes the last byte of the file.

-f Prevents the **tail** command from terminating after it writes the last line of the input file if the input file is not read from a pipe (that is, if the input file is a regular file or the *file* operand specifies a FIFO). Instead, **tail** enters an endless loop in which it sleeps for a second and then attempts to read and write further records from the input file. Thus, the **tail** command with the **-f** flag set can be used to monitor the growth of a file being written by another process.

The **-f** flag has no effect if specified with the **-r** flag.

-n [+|-]*number*

Writes *number* lines of the file. The sign specified before *number* affects the location in the file from which to begin writing:

+ (plus) Writing begins relative to the beginning of the file.

- (minus) Writing begins relative to the end of the file.

no sign Writing begins relative to the end of the file.

The value *number* begins at 1; that is, **-n +1** writes the first line of the file, and **-n -1** writes the last line of the file.

-r Causes **tail** to write lines from the end of the file in reverse order. This flag overrides the **-f** flag.

If the size of the file is not larger than **BUFFSIZE**, the default action is to write the entire file; otherwise, **-r** writes the last **BUFFSIZE** bytes of the file. (Note that **BUFFSIZE** is 10 * **LINE_MAX**, or 20,480 bytes.)

+[*number*][*unit*]
(Obsolescent) Begins writing from a location that is the specified number of units after the beginning of the input file.

The default value for *number* is 10.

The possible values for *unit* are:

b	Specifies 512-byte blocks
c	Specifies characters, counted byte by byte
k	Specifies 1-kilobyte blocks
l	Specifies lines
m	Specifies characters, counting a multibyte character as a single character

The default value for *unit* is **l**.

-[*number*][*unit*][**f**]
(Obsolescent) Begins writing from a location that is the specified number of units before the end of the input file.

The default value for *number* is 10.

The possible values for *unit* are:

b	Specifies 512-byte blocks
c	Specifies characters, counted byte by byte
k	Specifies 1-kilobyte blocks
l	Specifies lines
m	Specifies characters, counting a multibyte character as a single character

The default value for *unit* is **l**. Specifying **f** has the same effect as specifying the **-f** flag.

DESCRIPTION

The **tail** command writes from the named file (or, if no *file* is specified, from the standard input file) to the standard output file, beginning at a point you specify. If you do not specify the flags **-f**, **-r**, *-number*, or *+number*, **tail** begins reading 10 lines before the end of the file. **-** (end of input file) is the default starting point, **l** (lines) is the default *unit*, and 10 is the default *number*.

By specifying **+**, you can direct **tail** to write from the beginning of the input file. By specifying a *number* or a *unit* or both, you can change the point at which **tail** begins writing.

The *unit* argument can specify lines, blocks, or characters. The block size is either 512 bytes or 1 kilobyte.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

1. To write the last 10 lines of a file named **notes**, enter:

   tail notes

2. To write the last 20 lines of **notes**, enter:

   tail -n -20 notes

3. To write **notes** a page at a time, starting with the 200th byte from the beginning, enter:

   tail +200c notes | more

4. To follow the growth of a file named **accounts**, enter:

   tail -n -1 -f accounts

   This command writes the last line of **accounts** and then continues to write any lines that have been added to the end of the file until the command is stopped by the Interrupt key sequence.

EXIT VALUES

The **tail** command returns the following exit values:

0 (zero)	The command completed successfully.
>0	An error occurred.

RELATED INFORMATION

Commands: **cat(1)**, **more(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

The **-r** flag is an HP extension to the XPG4 Version 2 specification.

NAME

tar - Manipulates tape-archive-format files

SYNOPSIS

tar [-]*required_flag*[**Abflmovw**]
[*operand* ...]

FLAGS

The function performed by **tar** is specified by one of the following values for *required_flag*:

c Creates a new archive file. If the archive file is on tape, writing begins at the beginning of the tape instead of after the last file. The use of this flag implies the function of the *required_flag* value of **r**.

r Writes the files specified by *operand* values at the end of the tape or appends them to the end of an existing archive file. The use of the *required_flag* value of **c** implies the function of this flag.

t Lists the names of the files specified by *operand* values each time they occur in the archive file. If no value is given for *operand*, all the names in the archive file are listed to the standard output file.

u Adds the files specified by *operand* values to the archive file, if the files are not already there or if they were modified since last archived.

x Extracts the files specified by *operand* values from the archive file. If a specified file matches a directory whose contents were written into the archive file, this directory is (recursively) extracted. The owner, modification time, and mode are restored (if possible).

If no value is given for *operand*, the entire content of the archive is extracted. If multiple entries specifying the same file are in the archive, the last one overwrites all earlier ones.

You can use the following flags with *required_flag*:

A Tells the **tar** command to suppress warning messages about optional access control list (ACL) entries. Because the **tar** utility does not archive optional ACL entries, a warning message is printed for each file that has optional ACL entries. However, if **tar** is executed remotely from a system that does not support OSS ACLs, no warnings are printed.

b Tells the **tar** command to use the next operand as the blocking factor for tape records. The default value is less than or equal to 20, and the maximum value is greater than or equal to 20 (larger values can be specified at the risk of creating a tape archive that some systems' tape drives might not be able to restore).

Use this flag only when creating or writing raw magnetic tape archives. The block size is determined automatically when reading tapes (*required_flag* values of **x** or **t**).

f Tells the **tar** command to use the first *operand* value (or the second, if the **b** flag has been specified) as the name of the archive file.

When **f** is used with the *required_flag* value of **t** or **x** and the pathname specified for *operand* is **-** (dash), the standard input file is an archive file formatted as with **pax -x ustar**. When **f** is used with the *required_flag* value of **r**, **u**, or **c** and the pathname specified for *operand* is **-** (dash), the standard output file is an archive file formatted as with **pax -x ustar**. For example, the following command line is valid:

(cd fromdir; tar cf -.)|(cd todir; tar xf -)

l Tells **tar** to generate an error message if it cannot resolve all the links to the files archived. If this flag is not specified, no error messages are generated.

m Tells **tar** not to restore the modification times. The modification time assigned will be the time of extraction, which is always the case with symbolic links.

o Is provided for backward compatibility. Specify this flag if the archive will be restored on a system with an earlier version of **tar**.

On output, **tar** normally places information specifying owner and modes of directories in the archive file. Earlier versions of **tar**, when encountering this information, give an error message of the form:

name/: cannot create

The **o** flag suppresses the directory information. It also prevents archiving of special files and FIFOs that earlier versions of **tar** would not be able to extract properly. (Although anyone can archive special files, only a user who has appropriate privileges can extract them from the archive files.)

When **o** is used for reading, it causes the extracted file to take on the user ID and group ID (UID and GID) of the user running the program, rather than those of the archive file. This is the default action for the ordinary user.

v Makes **tar** display progress messages containing the name of each file it processes preceded by the flag letter. When the **v** flag is omitted, **tar** does not produce progress messages.

When used with the *required_flag* value of **t**, the **v** (verbose) flag gives more information about the archive entries than just their names.

w Causes **tar** to display the action to be taken followed by the name of the file, and then to wait for the user's confirmation.

If the user responds with a word beginning with **y**, or the locale's equivalent of a **y**, the action is performed. If any other response is given, the action is not performed.

All flags must be specified together (with no separating spaces). For all flags that require *operand* values, the operands must follow the string of flags and be in the same order as the corresponding flags. For example, **tar -cbf** *n file* **.** and **tar -cfb** *file n* |**.** both use correct ordering, while **tar -cbf** *file n* **.** specifies a filename where a blocking factor is expected.

DESCRIPTION

The **tar** utility is used to save and restore data from traditional format **tar** archives.

The actions of the **tar** command are controlled by a string containing one required flag and one or more optional flags. Most operands to **tar** are filenames or directory names specifying which files to dump or restore. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

Environment Variables

The **LC_MESSAGES** variable determines the locale's equivalent of **y** or **n** (for yes/no user responses).

This command supports the use of the **LANG**, **LC_ALL**, **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, **LC_TIME**, **NLSPATH**, and **TZ** environment variables.

FILES

/tmp/tar* Temporary file used with the *required_flag* value of **u**.

NOTES

- There is no way to ask for the *n*th occurrence of a file.
- Tape errors are handled ungracefully.
- The function of the *required_flag* value of **u** can be slow.
- The limit on filename length is 256 bytes. The limit on file links (hard or soft) is 100 bytes.
- There is no way to selectively follow symbolic links.
- When extracting archive files created with the *required_flag* value of **r** or **u**, directory modification times might not be set correctly.
- The **tar** command can fail with the error message `Name too long` when an attempt is made to archive a file with a filename longer than 100 characters. This message is displayed because the USTAR format is used to create an archive. The command fails because the USTAR format does not support filenames longer than 100 characters, in conformance with the 1990 edition of the POSIX Standard IEEE 1003.1. A practical workaround is to use the **pax** command with the **-x cpio** flag, because the **cpio** archive format supports filenames longer than 100 characters.

EXIT VALUES

The **tar** command returns the following values:

0 (zero) The command completed successfully.

>0 An error occurred.

RELATED INFORMATION

Commands: **cpio(1)**, **pax(1)**.

Functions: **chdir(2)**, **umask(2)**.

Files: **tar(4)**.

Miscellaneous topics: **acl(5)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

NAME

tee - Displays program output and copies to a file

SYNOPSIS

tee [**-ai**] [*file* ...]

The **tee** command reads standard input and writes both to standard output, and to each specified *file*.

FLAGS

-a Adds the output to the end of *file* instead of writing over it.

-i Ignores the **SIGINT** signal.

DESCRIPTION

The **tee** command is useful when you wish to view program output as it is displayed, and also want to save it in a file. The **tee** command can accept up to 20 file arguments.

The **tee** command does not buffer output.

EXAMPLES

1. To view and save the output from a command at the same time, enter:

   ls | tee mylist

   This displays the standard output of the command **ls** at the terminal, and at the same time saves a copy of it in the file **mylist**. If **mylist** already exists, it is deleted and replaced.

2. To display the output of a command or program and append it to a file, enter:

   ls | tee -a listrecord

   This displays the standard output of **ls** at the terminal and at the same time appends a copy of it to the end of **listrecord**. If the file **listrecord** does not exist, it is created.

RELATED INFORMATION

Commands: **echo**(1).

NAME

telnet - Allows login to a remote host (not supported in OSS)

SYNOPSIS

telnet

DESCRIPTION

The **telnet** command implements the TELNET protocol, which allows remote login to other hosts. The **telnet** client is not supported in the OSS environment.

As a substitute, the Guardian environment Telserv interface for the TELNET protocol can be used through the **gtacl** command. The corresponding syntax is:

gtacl -p telnet [*hostname* | *ipaddress* [*port_number*]]

EXAMPLES

The following example creates a connection for login to the remote host with the specified IP address; the port number defaults to a well-known value (usually 23):

```
gtacl -p telnet 130.255.198.4
```

RELATED INFORMATION

Commands: **gtacl(1)**.

NAME

test - Evaluates conditional expressions

SYNOPSIS

test *expression*

[*expression* **]**

DESCRIPTION

The **test** command evaluates an *expression* constructed of functions and operators. If the value of *expression* is **TRUE**, **test** returns an exit value of 0 (zero); otherwise, it returns a value of 1 (**FALSE**). The **test** command also returns an exit value of 1 if there are no arguments.

The alternative form of the command surrounds *expression* with **[ ]** (brackets). When you use this form, you must surround the brackets with spaces.

The test Expressions

All of the listed functions and operators are separate arguments to **test**.

The following functions are used to construct *expression*:

-a *file* **TRUE** if *file* exists.

-b *file* **TRUE** if *file* exists and is a block special file.

-c *file* **TRUE** if *file* exists and is a character special file.

-d *file* **TRUE** if *file* exists and is a directory.

-e *file* **TRUE** if *file* exists.

-f *file* **TRUE** if *file* exists and is a regular file.

-g *file* **TRUE** if *file* exists and its set-group ID bit is set.

-h *file* **TRUE** if *file* exists and is a symbolic link. This function was used with previous versions of this program. Use -L instead of -h.

-k *file* **TRUE** if *file* exists and its sticky bit is set.

-L *file* **TRUE** if *file* exists and is a symbolic link.

-n *string1* **TRUE** if the length of *string1* is nonzero.

-p *file* **TRUE** if *file* exists and is a named pipe (FIFO).

-r *file* **TRUE** if *file* exists and has read permission.

-s *file* **TRUE** if *file* exists and has a size greater than 0 (zero).

-t [*file_descriptor*]
TRUE if the open file with file descriptor number *file_descriptor* (1 by default) is associated with a terminal device.

-u *file* **TRUE** if *file* exists and its set-user ID bit is set.

-w *file* **TRUE** if *file* exists and has write permission.

-x *file* **TRUE** if *file* exists and has execute permission. If *file* is a directory, **TRUE** indicates that it can be searched.

-z *string1* **TRUE** if the length of *string1* is 0 (zero).

string1 **=** *string2*
TRUE if *string1* and *string2* are identical.

string1 **!=** *string2*
TRUE if *string1* and *string2* are not identical.

string1 **TRUE** if *string1* is not the null string.

number1 **-eq** *number2*
TRUE if the integers *number1* and *number2* are algebraically equal. Any of the comparisons **-ne** (not equal to), **-gt** (greater than), **-ge** (greater than or equal to), **-lt** (less than), and **-le** (less than or equal to) can be used in place of **-eq**.

The listed functions can be combined with the following operators:

! Unary negation operator.

-a Binary AND operator.

-o Binary OR operator (**-a** has higher precedence than **-o**).

\(*expression* **\)** Parentheses for grouping. There must be a space after **\(** and before **\)**.

EXAMPLES

1. To test whether a file exists and is not empty, enter:

```
if test -s "$1"
then
        echo $1 does not exist or is empty.
fi
```

If the file specified by the first positional parameter to the shell procedure does not exist, this example displays an error message. If **$1** exists, this example displays nothing. Note that there must be a space between **-s** and the filename.

The double quotes around **$1** ensure that the test will work properly even if the value of **$1** is the empty string. If the double quotes are omitted and **$1** is the empty string, **test** displays the error message `test: parameter expected`.

2. To do a complex comparison, enter:

```
if [ $# -lt 2  -o  ! -s "$1" ]
then
        exit
fi
```

If the shell procedure was given fewer than two positional parameters or if the file specified by **$1** does not exist or is empty, then this example exits the shell procedure. The special shell variable **$#** represents the number of positional parameters entered on the command line that started this shell procedure.

Note that there must be a space before and after the **[** character and before the **]** character. There must also be a space before the **-lt** flag and before the **-s** flag.

EXIT VALUES

The **test** command evaluates *expression* and, if its value is **TRUE**, returns an exit value of 0 (zero); otherwise, it returns a value of 1 (**FALSE**); the **test** command also returns an exit value of 1 if there are no arguments.

RELATED INFORMATION

Commands: **find(1)**, **sh(1)**.

STANDARDS CONFORMANCE

The **-a** *file* function (an *expression* primary operand) is an extension to the XPG4 Version 2 specification.

NAME

time - Times the execution of a command

SYNOPSIS

time [**-p**] *command* [*argument* ...]

The **time** command prints the elapsed time during the execution of a command, the time spent in the system, and the time spent in execution of the command on the diagnostic output system.

FLAGS

-p Writes the timing output to standard error. This is the default.

DESCRIPTION

Time is reported in seconds.

The **time** command (with a different format) is also built into **sh**.

EXAMPLES

To measure the time required to run a program, enter:

time a.out

This runs the program **a.out** and writes to the standard error output the amount of real, system, and user time that it uses:

```
real    10.5
user     0.3
sys      3.6
```

EXIT VALUES

The **time** command returns the following exit values:

1-125 An error occurred.

126 The utility was found but could not be invoked.

127 The utility could not be found.

RELATED INFORMATION

Commands: **sh(1)**.

NAME

times - Prints accumulated running times

SYNOPSIS

times

DESCRIPTION

The **times** command returns the total time that has been used by the shell and the accumulated run times of processes run by the shell.

Two sets of times are returned for both the shell time and process time. The first two numbers represent the accumulated user time and accumulated system time used by the shell. The second two numbers represent the accumulated user time and accumulated system time used for processes.

EXAMPLES

1. The following example shows the time command and an example of its output:

   times

   0m0.23s 0m0.33s
   0m0.13s 0m0.09s

   The first line represents times used by the shell. The first number is the time used by the shell for the user functions. The second number represents time used by the system for shell functions.

   The second line represents times used by the shell for subshell processes. The first number represents user time for subshells, the second number represents system time used for subshells.

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **times** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

NAME

touch - Updates file access and modification times

SYNOPSIS

Current Syntax

touch [**-acfm**] [**-r** *reference_file* | **-t** *time*] *file ...*

Obsolescent Syntax

touch [**-acfm**] [*time*] *file ...*

The **touch** command updates the access and modification times of each file or directory named to the time specified on the command line.

Do not use the **touch** command on files in the Guardian environment.

FLAGS

-a Changes only the access time.

-c Suppresses the creation of the file without any diagnostic messages.

-f Attempts to force the **touch** in spite of read and write permissions on a file. **-f** is actually a dummy flag; it is not used by the **touch** code, but is recognized by **getopt()**.

-m Changes only the modification time.

-r *reference_file*
Uses the time of the file named by the pathname *reference_file* instead of the current time.

-t *time* Uses the specified time instead of the current time. The *time* argument is a decimal number in the following form:

[[*CC*]*YY*]*MMDDhhmm*[**.***SS*]

The paired decimal numbers in the preceding syntax line represent the following:

CC The first two digits of the year (the century).

YY The second two digits of the year (00-99).

MM The month of the year (01-12).

DD The day of the month (01-31).

hh The hour of the day (00-23).

mm The minute of the hour (00-59).

SS The second of the minute (00-61).

Both *CC* and *YY* are optional. If neither is specified, the current year is assumed. If *YY* is specified, but *CC* is not, *CC* is derived as follows:

- If *YY* is 69-99, *CC* is 19.
- If *YY* is 00-68, *CC* is 20.

The resulting time is affected by the value of the **TZ** environment variable. If the resulting time value precedes the Epoch, **touch** exits immediately with an error status. The range of valid times past the Epoch extends to at least midnight 1 January 2000 UT.

The range for *SS* is 00-61 rather than 00-59 because of leap seconds. If *SS* is 60 or 61, and the resulting time, as affected by the **TZ** environment variable, does not refer to a leap second, the resulting time is one or two seconds after a time where *SS* is 59. If *SS* is not given a value, it is assumed to be 0 (zero).

DESCRIPTION

The time used can be specified by **-t**, **-r**, or by the *time* argument. If you do not specify a time, **touch** uses the current time. If you specify a file that does not exist, **touch** creates a file with that name unless you request otherwise with the **-c** flag.

If neither the **-a** or **-m** flags are specified, **touch** behaves as though both of these flags were specified.

The **LC_TIME** environment variable, if defined, specifies the order of month and day in the date specification and of hour and minute in the time specification. Otherwise, these orders default to *MMdd* and *hhmm*.

The obsolescent format for the *time* argument is *MMddhhmm*[*yy*].

EXAMPLES

1. To update the access and modification times of a file, enter:

   touch program.c

   This sets the last access and last modification times of **program.c** to the current date and time. If **program.c** does not exist, **touch** creates an empty file with that name.

2. To avoid creating a new file, enter:

   touch -c program.c

3. To update only the modification time, enter:

   touch -m *.o

   This updates only the last modification times of the files in the current directory that end with **.o**. The **touch** command is often used in this way to alter the results of the **make** command.

4. To explicitly set the access and modification times, enter:

   touch -c -t 02171425 program.c

   This sets the access and modification dates to 14:25 (2:25 p.m.) February 17 of the current year. (This assumes that you are using the default format.)

5. To touch a file with a numeric filename, include its full pathname or precede it with **./**, so that the filename is not mistaken for the *time* argument. For example, to touch the file **123.abc**, enter:

   touch -c ./123.abc

CAUTION

Do not use the **touch** command on files in the Guardian Environment.

EXIT VALUES

The return code from **touch** is the number of files for which the times could not be successfully modified (including files that did not exist and were not created). If no errors occur, the exit status is 0 (zero).

RELATED INFORMATION

Commands: **date**(1).

Functions: **utime**(2).

Files: **locale**(4).

NAME

tr - Translates characters

SYNOPSIS

tr [**-Acs**] *string1 string2*

tr -s [**-Ac**] *string1*

tr -d [**-Ac**] *string1*

tr -ds [**-Ac**] *string1 string2*

The **tr** command copies characters from the standard input to the standard output with substitution or deletion of selected characters.

FLAGS

-A Translates on a byte-by-byte basis. When you specify this flag, **tr** does not support extended characters.

-c Complements (inverts) the set of characters in *string1*, which is the set of all characters in the current character set, as defined by the current setting of **LC_CTYPE**, except for those actually specified in the *string1* argument. These characters are placed in the array in ascending collation sequence, as defined by the current setting of **LC_COLLATE**.

-d Deletes all occurrences of input characters or collating elements found in the array specified in *string1*.

If **-c** and **-d** are both specified, all characters except those specified by *string1* are deleted. The contents of *string2* are ignored, unless **-s** is also specified. Note, however, that the same string cannot be used for both the **-d** and the **-s** flags; when both flags are specified, both *string1* (used for deletion) and *string2* (used for squeezing) are required.

If **-d** is not specified, each input character or collating element found in the array specified by *string1* is replaced by the character or collating element in the same relative position in the array specified by *string2*.

-s Replaces any character specified in *string1* that occurs as a string of two or more repeating characters as a single instance of the character in *string2*.

If the *string2* contains a character class, the argument's array contains all of the characters in that character class. For example:

tr -s '[:space:]'

In a case conversion, however, the *string2* array contains only those characters defined as the second characters in each of the **toupper** or **tolower** character pairs, as appropriate. For example:

tr -s '[:upper:]' '[:lower:]'

DESCRIPTION

Input characters from *string1* are replaced with the corresponding characters in *string2*. If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the shell.

The following abbreviations can be used to introduce ranges of ASCII characters or repeated characters:

a-z Stands for a string of characters whose ASCII codes run from character **a** to character **z**, inclusive. No multicharacter collating elements will be included in this range.

[a***number***]
Stands for *number* repetitions of **a**. *number* is considered to be in decimal unless the first digit of *number* is **0**; then it is considered to be in octal. Because this expression is used to map multiple characters to one character, it is only valid when it occurs in *string2*. If *number* is omitted or is 0 (zero), it is interpreted as large enough to extend the *string2*-based sequence to the length of the *string1*-based sequence.

[=***equiv*****=]**
Represents all characters or collating elements belonging to the equivalence class specified by *equiv*, as defined by the **LC_COLLATE** locale category. An equivalence class expression can be used for *string1* or *string2* only when used in combination with the **-d** and **-s** flags. (For more information, see the reference page for the **locale** file.)

[:***class*****:]** Represents all characters belonging to the defined character class, as defined by the current setting of the **LC_CTYPE** locale category. The following character class names are accepted when specified in *string1*:

alnum	**cntrl**	**lower**	**space**
alpha	**digit**	**print**	**upper**
blank	**graph**	**punct**	**xdigit**

When the **-d** and **-s** flags are specified together, any of the character class names are accepted in *string2*; otherwise, only character class names **lower** or **upper** are accepted in *string2* and then only if the corresponding character class (**upper** and **lower**, respectively) is specified in the same relative position in *string1*. Such a specification is interpreted as a request for case conversion.

When **[:lower:]** appears in *string1* and **[:upper:]** appears in *string2*, the arrays contain the characters from the **toupper** mapping in the **LC_CTYPE** category of the current locale. When **[:upper:]** appears in *string1* and **[:lower:]** appears in *string2*, the arrays contain the characters from the **tolower** mapping in the **LC_CTYPE** category of the current locale.

The first character from each mapping pair is in the array for *string1* and the second character from each mapping pair is in the array for *string2* in the same relative position. (For more information about possible character class settings, see the reference page for **sh**.)

Use the escape character \ (backslash) to remove the special meaning from any character in a string. Use the \ (backslash) followed by 1, 2, or 3 octal digits for the code of a character.

If an ordinary digit (representing itself) is to follow an octal sequence, the octal sequence must use the full three digits to avoid ambiguity.

When *string2* is shorter than *string1*, a difference results between historical System V and BSD systems. A BSD system pads *string2* with the last character found in *string2*. Thus, it is possible to do the following:

tr 0123456789 d

The preceding command translates all digits to the letter **d**. A portable application cannot rely on the BSD behavior; it would have to code the example in the following way:

tr 0123456789 '[d*]'

Despite their similarity to regular expressions, string arguments used by **tr** are not regular expressions.

The **tr** command correctly processes NULL characters in its input stream. NULL characters can be stripped using the following command:

tr -d '\000'

EXAMPLES

1. To translate braces into parentheses, enter:

   tr '{}' '()'

   This translates each **{** (left brace) to a **(** (left parenthesis) and each **}** (right brace) to **)** (right parenthesis). All other characters remain unchanged and are sent to standard output.

2. To translate lowercase ASCII characters to uppercase, enter:

   tr '[:lower:]' '[:upper:]' < infile > outfile

3. To translate each digit to a **#** (number sign), enter:

   tr '0-9' '[#*]' < infile > outfile

   The ***** (asterisk) tells **tr** to repeat the **#** (number sign) enough times to make the second string as long as the first one.

4. To translate each string of digits to a single **#** (number sign), enter:

   tr -s '0-9' '[#*]' < infile > outfile

5. To translate all ASCII characters that are *not* specified, enter:

   tr -c '[-~\177]' '[A-_?]' < infile > outfile

   This translates each nonprinting ASCII character to the corresponding control key letter (**\001** translates to **A**, **\002** to **B**, and so on). ASCII DEL (**\177**), the character that follows **~**(tilde), translates to a **?** (question mark).

6. To create a list of all words in **file1** one per line in **file2**, where a word is taken to be a maximal string of letters, enter:

   tr -cs '[:alpha:]' '[\n*]' < file1 > file2

7. To use an equivalence class to identify accented variants of the base character **e** in **file1**, which are stripped of diacritical marks and written to **file2**, enter:

   tr '[=e=]' '[e*]' < file1 > file2

Specifying the **-A** flag improves ASCII performance.

RELATED INFORMATION

Commands: **sh**(1).

NAME

trap - Provides instructions to a process

SYNOPSIS

trap [*argument*] [*signal* ...]

DESCRIPTION

The **trap** command provides instructions to a program when signals are received.

The *argument* variable specifies a command to be read and executed when the shell receives the specified signals. (Note that *argument* is scanned once when the trap is set and once when the trap is taken.) Each *signal* can be given as a number or as the name of the signal. Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective.

If *argument* is omitted or is **-**, all traps are reset to their original values.

If *argument* is a null string, this signal is ignored by the shell and by the commands it invokes.

If *signal* is **ERR**, *argument* is executed whenever a command has a nonzero exit status.

If *signal* is **DEBUG**, *argument* is executed after each command.

If *signal* is **0** or **EXIT** and the **trap** statement is executed inside the body of a function, the command *argument* is executed after the function completes.

If *signal* is **0** (zero) or **EXIT** for a **trap** set outside any function, the command *argument* is executed on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **trap** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1).

NAME

true - Returns a standard exit value

SYNOPSIS

true

DESCRIPTION

The **true** command returns a 0 (zero) exit value. These commands are usually used in input to the **sh** command.

EXAMPLES

To construct a loop in a shell procedure, enter:

```
while true
do
        date
        sleep 60
done
```

This procedure displays the date and time once a minute. To stop it, press the Interrupt key sequence.

RELATED INFORMATION

Commands: **false**(1), **sh**(1).

NAME

tty - Returns pathname of terminal device

SYNOPSIS

tty [**-s**]

The **tty** command writes the full pathname of your terminal device to standard output.

FLAGS

-s Suppresses reporting the pathname so that only the exit status is affected. (Obsolescent.) You can perform the same operation with the **test -t** *file_descriptor* command.

DESCRIPTION

The **tty -s** command evaluates as TRUE if standard output is a display and FALSE if it is not.

The **/dev/tty** special file always refers to your controlling terminal, although it also may have another name like **/dev/console** or **/dev/tty2**. To avoid writing undesirable output to an output file (for example, to write a prompt in a shell script to the screen, while writing the response to the prompt to an output file), redirect standard output to **/dev/tty**.

EXAMPLES

1. To display the full pathname of your terminal device, enter:

   ```
   tty
   ```

2. To test whether or not the standard input is a terminal device, enter:

   ```
   if test -t 0
   then
           echo "Output is a display"
   else
           echo "Output is not a display"
   ```

FILES

/dev/tty Pseudodevice representing the user's controlling terminal.

DIAGNOSTICS

`not a tty` Your standard input is not an interactive terminal.

EXIT VALUES

The exit value has the following possible meanings:

0 Standard input is a tty.

1 Standard input is not a tty.

2 Invalid flags specified or other error.

RELATED INFORMATION

Commands: **stty(1)**.

Subroutines: **isatty(3)**, **ttyname(3)**.

Files: **tty**(7).

NAME

type - Returns type and location of commands

SYNOPSIS

type *argument*

DESCRIPTION

The **type** command returns the location of the command name given as *argument*. The **type** command is an alias to the **whence -v** command, another shell built-in command.

EXAMPLES

1. The following example returns the command type and location of the **grep** command.

   type grep

   grep is a tracked alias for /bin/grep

   In this example, the name **grep** is a tracked alias for the **grep** command located at **/bin/grep**.

NOTES

The **type** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh**(1), **whence**(1).

NAME

typeset - Sets attributes and values for shell parameters

SYNOPSIS

typeset [**+** | **-HLRZfilrtux**[*n*]] [*name*[=*value* ...]]

FLAGS

-f The names refer to function names rather than parameter names. No assignments can be made and the only other valid flags are **-t**, **-u**, and **-x**. The **-t** flag turns on execution tracing for this function. The **-u** flag causes this function to be marked undefined. The **FPATH** variable is searched to find the function definition when the function is referenced. The **-x** flags allows the function definition to remain in effect across shell procedures invoked by name.

-H Provides system-to-hostname file mapping on machines that restrict the set of characters in filenames.

-i Parameter is an integer. This makes arithmetic faster. If *n* is nonzero, it defines the output arithmetic base; otherwise, the first assignment determines the output base.

-l All uppercase characters are converted to lowercase. The uppercase **-u** flag is turned off.

-L Left justifies and removes leading spaces from *value*. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. When the parameter is assigned, it is filled on the right with spaces or truncated, if necessary, to fit into the field. Leading zeros are removed if the **-Z** flag is also set. The **-R** flag is turned off.

-r The given *names* are marked read-only, and these names cannot be changed by subsequent assignment.

-R Right justifies and fills with leading spaces. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. The field is left-filled with spaces or truncated from the end if the parameter is reassigned. The **L** flag is turned off.

-t Tags the named parameters. Tags are user definable and have no special meaning to the shell.

-u All lowercase characters are converted to uppercase characters. The lowercase **-l** flag is turned off.

-x The given names are marked for export.

-Z Right justifies and fills with leading zeros if the first nonspace character is a digit and the **-L** flag was not set. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment.

DESCRIPTION

The **typeset** command assigns values to named parameters. The flags assign attributes to the parameters.

When the **typeset** command is invoked inside a function, a new instance of the parameter *name* is created. The parameter value and type are restored when the function completes.

Using **+** (plus sign) rather than **-** (dash) causes the flags to be turned off.

If no *name* arguments are given but flags are specified, a list of *names* (and optionally the *values*) of the parameters that have these flags set is printed. (Using **+** rather than **-** keeps the values from being printed.)

If no names and flags are given, the names and attributes of all parameters are printed.

NOTES

The **typeset** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **set(1)**, **sh(1)**.

NAME

umask - Sets the user file-creation mask.

SYNOPSIS

umask [-S] [*mask*]

-S Produces symbolic output.

DESCRIPTION

The **umask** command sets the user file-creation mask to the value specified in the argument *mask*. The default user file-creation mask for an OSS process is 0022.

The default permissions is 666 (rw-rw-rw) for files and 777 (rwxrwxrwx) for directories. The permissions value you specify with the **umask** command will modify the default permissions.

The *mask* argument can be either octal number or a symbolic value. If an octal value is given, the new **umask** value is the complement of the result of applying *mask* to the complement of the current **umask** value.

If *mask* is omitted, the current value of the mask is printed.

Symbolic Mode

Symbolic mode has the form:

[*who*] *operation permission*,[*operation permission* ...]

The *who* argument specifies whether you are defining permissionsfor a user, group, or all others, or any combination of these. The *operation* argument specifies whether the permission is being added, removed, or assigned absolutely. The *permission* argument identifies the operation that the specified users can perform.

Valid options for the *who* argument are as follows:

a User, group, and all others (same effect as the combination **ugo**)

g Group

o All others

u User (owner)

If the *who* argument is omitted, the default value is **a**, but the setting of the file creation mask, **umask** (see the reference page for **sh(1)**), is applied.

Valid options for the *operation* argument are as follows:

- Removes specified permissions.

+ Adds specified permissions.

Valid options for the *permission* argument are as follows:

r Read permission.

w Write permission.

x Execute permission for files, search permission for directories.

s Set-user-ID or set-group-ID permission.

This permission bit sets the effective user ID or group ID to that of the owner or group owner of *file* whenever the *file* is run. Use this permission setting with the **u** or **g** option to allow temporary or restricted access to files not normally accessible to other users.

An **s** appears in the user or group execute position of a long listing (see the reference page for the **ls** command) to show that the file runs with set-user-ID or set-group-ID permission.

Note that the command **umask o+s** has no effect (the set-user-ID-on-execution and set-group-ID-on-execution bits are not modified).

Absolute Mode

Absolute mode lets you use octal notation to set each bit in the permission code.

0400 Permits read by owner.

0200 Permits write by owner.

0100 Permits execute or search by owner.

0040 Permits read by group.

0020 Permits write by group.

0010 Permits execute or search by group.

0004 Permits read by others.

0002 Permits write by others.

0001 Permits execute or search by others.

EXAMPLES

1. To change the default permissions from 666 (wr-wr-wr) to 644 (rw-r--r--) using octals (absolute mode) enter the following command.

   umask 022

   Notice that the octal number 022 is used. The complement of 666 is 111; when 022 is applied to 111 the result is 133. The complement of 133 is 644, the value of the new permissions.

2. To change the default permissions from 666 to 644 using symbolic mode, enter the following command.

   umask go-w

   This command removes the write permissions for both the group and others.

NOTES

The **umask** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **chmod(1), sh(1).**

NAME

unalias - Removes aliases

SYNOPSIS

unalias *name ...*
unalias -a

FLAGS

-a Removes all alias definitions from the current shell environment.

DESCRIPTION

The **unalias** command removes alias definitions.

Without the **-a** flag, the **alias** command removes the names specified as *name ...* from the shell's alias list.

With the **-a** flag, the **alias** command removes all alias definitions from the current shell execution environment, but not from the shell's alias list.

EXIT VALUES

If one of the *name* arguments does not represent a valid alias definition or an error occurs, the exit value is greater than 0 (zero).

NOTES

The **unalias** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **alias**(1), **sh**(1).

NAME

uname - Displays information about the operating system

SYNOPSIS

uname [**-amnrsv**]

FLAGS

-a Displays all information specified with the **-m**, **-n**, **-r**, **-s**, and **-v** flags.

-m Displays the type of hardware running the system.

-n Displays the name of the node (this might be a name that the system is known by to a communications network).

-r Displays the release version of the running system software.

-s Displays the operating system name. (This flag is on by default.)

-v Displays the update number of the release version currently running on the server.

DESCRIPTION

The **uname** command displays the name of the operating system that you are using and provides other system information.

The **uname** command writes system information to standard output. This is mainly useful to determine which system you are using. The flags cause selected information returned by **uname** to be displayed.

EXAMPLES

To display the complete system name and version banner, enter:

```
uname -a
```

RELATED INFORMATION

Functions: **uname(2).**

NAME

uncompress - Expands compressed data

SYNOPSIS

uncompress [**-cdfnqv**] [*file*[**.Z**] ...]

FLAGS

-c Makes the **uncompress** command write to the standard output file; no files are changed. The nondestructive behavior of the **zcat** command is identical to that of **uncompress -c**.

-d Specifies that decompression should occur.

-f Except when **uncompress** is run in the background under the **/usr/bin/sh** file, suppresses the prompt about whether an existing file given by the *file* operand should be overwritten.

-n Specifies that no header is added or expected. This flag might be useful for decompressing old files.

-q Specifies a quiet operation. This flag is the default action; diagnostic messages are displayed only if the **-v** flag is specified.

-v Prints the percentage expansion of each file.

DESCRIPTION

The **uncompress** command replaces the compressed **.Z** file with a decompressed version of the file, identical to the file that was originally compressed with the **compress** command. The new file has the same filename as the compressed file with the **.Z** suffix removed.

You can specify the compressed target file with or without the **.Z** suffix; if you do not specify the suffix, **uncompress** assumes it.

If the file has an access control list (ACL), the ACL is preserved when the file is decompressed. For more information about ACLs, see the **acl(5)** reference page.

Operands

file[**.Z**] Specifies the pathname of a compressed file to be decompressed. If the **.Z** suffix is omitted, it is assumed.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXIT VALUES

The **uncompress** command returns the following values:

0 (zero) The command completed successfully.

>0 An error occurred or an attempt was made to expand a file that is not compressed.

RELATED INFORMATION

Commands: **compress(1)**, **zcat(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

The following features are extensions to the XPG4 Version 2 specification:

- The **-d**, **-n**, and **-q** flags are supported.

NAME

unexpand - Replace tab or space characters

SYNOPSIS

unexpand [**-a**] [**-t** *tablist*] [*file* ...]

The **unexpand** command puts tab characters back into the data from the standard input file, or the named files and writes the result to the standard output file.

FLAGS

-a Replaces spaces at the beginning of each line with a tab, and inserts tab characters wherever their presence compresses the resultant file, by replacing two or more characters. When the **-t** flag is specified with the **unexpand** command, the **-a** flag has no effect.

-t *tablist* Specifies the tab stops. The *tablist* argument consists of a single positive decimal integer or multiple positive decimal integers, separated by spaces or commas, in ascending order. If a single number is specified, tabs are set *tablist* column positions apart instead of the default width (8). If multiple numbers are specified, tabs are set at those specific column positions. Tabbing to tab stop position *n* thus causes the next character output to be in the (*n*+1)th column position on that line.

DESCRIPTION

Backspace characters are preserved in the output and decrement the column count for tab calculations. The column position count cannot be decremented below zero.

By default, the **unexpand** command converts only spaces within sequences of spaces and tab characters at the beginnings of lines. Use the **-a** flag to convert other sequences of spaces.

EXAMPLES

1. To replace the spaces in **file** with tab characters, enter:

   unexpand -a file

RELATED INFORMATION

Commands: **expand**(1).

NAME

uniq - Removes or lists repeated lines in a file

SYNOPSIS

Current syntax

uniq [**-c** | **-d** | **-u**] [**-f** *fields*] [**-s** *characters*] [*input_file*] [*output_file*]

Obsolescent syntax

uniq [**-c** | **-d** | **-u**] [*-number*] [*+number*] [*input_file*] [*output_file*]

The **uniq** command reads standard input by default or, the specified *input_file* compares adjacent lines, removes the second and succeeding occurrences of a line, and writes to standard output or the specified file *output_file*.

FLAGS

-c Precedes each output line with a count of the number of times each line appears in the file. This flag supersedes **-d** and **-u**.

-d Displays repeated lines only.

-f *fields* Ignores the first *fields* fields on each input line when doing comparisons, where *fields* is a positive decimal integer. A field is the maximal string matched by the basic regular expression:

[[:blank:]]*[^[:blank:]]*

If the *fields* argument specifies more fields than appear on an input line, a null string is used for comparisons.

-s *characters*

Ignores the specified number of characters when doing comparisons. The *characters* argument is a positive decimal integer.

If specified with the **-f** flag, the first *characters* characters after the first *fields* fields are ignored. If the *characters* argument specifies more characters than remain on an input line, **uniq** uses a null string for comparison.

-u Displays unique lines only.

-number Skips over the first *number* fields. A field is a string of nonspace, nontab characters separated by tabs or spaces, or both, from adjacent data on the same line. Equivalent to **-f** *fields*. (Obsolescent)

+number Skips over the first *number* characters. Fields specified by *number* are skipped before characters. Equivalent to **-s** *characters*. (Obsolescent)

DESCRIPTION

The *input_file* and *output_file* arguments must specify different files.

Repeated lines must be on consecutive lines to be found. You can arrange them with the **sort** command before processing.

EXAMPLES

To delete repeated lines in the following file called **fruit** and save it to a file named **newfruit**, enter:

uniq fruit newfruit

The file **fruit** contains the following lines:

```
apples
apples
bananas
cherries
cherries
peaches
pears
```

The file **newfruit** contains the following lines:

```
apples
bananas
cherries
peaches
pears
```

RELATED INFORMATION

Commands: **comm**(1), **sort**(1).

NAME

unpack - Expands files compressed by the **pack** command

SYNOPSIS

unpack *file*[**.z**] ...

DESCRIPTION

The **unpack** command expands files created by **pack**. For each file specified, **unpack** searches for a file named *file***.z**. If this file is a packed file, **unpack** replaces it with its expanded version. The **unpack** command names the new file by removing the **.z** suffix from *file***.z**. The **unpack** command tries to preserve the access modes, access and modification dates, and owner from the compressed file, but it can do so only if you have the appropriate privileges (see the **chmod(1)** reference page); otherwise, **unpack** expands the compressed file and assigns your owner and group ID to the new file.

The exit value is the number of files the **unpack** command was unable to expand (unpack). A file cannot be unpacked if any one of the following occurs:

- The file cannot be opened.
- The file is not a packed file.
- A file with the unpacked filename already exists.
- The unpacked file cannot be created.

If the file has an access control list (ACL), the ACL is preserved when the file is unpacked. For more information about ACLs, see the **acl(5)** reference page.

Operands

file[**.z**] Specifies the filename of the file to be unpacked. If the **.z** suffix is omitted, it is assumed.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXAMPLES

To unpack packed files, enter:

```
unpack chap1.z chap2
```

This command expands the packed files **chap1.z** and **chap2.z**, replacing them with files named **chap1** and **chap2**. Note that you can give **unpack** filenames either with or without the **.z** suffix.

NOTES

1. **unpack** operates only on files whose names end in **.z**. As a result, when you specify a filename *file* that does not end in **.z**, **unpack** adds that suffix and searches the directory for the filename *file***.z**.
2. The **unpack** command writes a warning to the standard output file if the file it is unpacking has links. Any other files linked to the packed file's original inode still exist and are still packed.
3. If the file being unpacked has a symbolic link, the new unpacked file has a different inode than the packed file from which it was created.

EXIT VALUES

The **unpack** command returns the following values:

0 (zero)	The command completed successfully; all files were unpacked.
>0	An error occurred because some of the files could not be unpacked. The number returned is the number of files the **unpack** command was unable to unpack.

RELATED INFORMATION

Commands: **cat(1)**, **compress(1)**, **pack(1)**, **uncompress(1)**, **zcat(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions, except for the following features:

- The size of the *file* value is restricted to **NAME_MAX**-2 bytes.

NAME

unset - Removes environment variable or function definitions

SYNOPSIS

unset [**-fv**] *name*

FLAGS

-f The *name* argument refers to function names.

-v The *name* argument refers to an environment variable name.

DESCRIPTION

The values of the variables or functions given as the *name* argument are unassigned; that is, the values and attributes of the names are erased. Read-only variables cannot be unset.

If the **-f** flag is set, the name given as the *name* argument refers to a function name.

If the **-v** flag is specified, the name given as the *name* parameter refers to a variable name, and the shell unsets it and removes it from the environment.

If neither the **-f** nor **-v** flag is specified, *name* refers to a variable.

Unsetting **ERRNO**, **LINENO**, **OPTARG**, **OPTIND**, **RANDOM**, **SECONDS**, **TMOUT**, and **_** removes their special meaning, even if they are subsequently assigned.

EXAMPLES

1. The following commands set, check, reset, and recheck the value of a variable **x**. After **x** has been unset, the **echo x** command returns null, confirming that the value of **x** has been unset.

```
x=100
echo $x
100
unset x
echo $x
```

NOTES

- Parameter assignment lists that precede the command remain in effect when the command completes.
- I/O redirections are processed after parameter assignments.
- Errors cause a script that contains the commands so marked to abort.

The **unset** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **set**(1), **sh**(1).

NAME

uudecode - Decodes a binary file

SYNOPSIS

uudecode [*file* ...]

DESCRIPTION

The **uudecode** command reads an encoded file, strips off any leading and trailing lines added by mailers, and recreates the original file with its original file access permissions and pathname.

If the pathname of the file to be produced exists and the user does not have write permission on that file, **uudecode** terminates with an error. If the pathname of the file to be produced exists and the user has write permission on that file, the existing file is overwritten.

Operands

file Specifies the pathname of a binary file to be decoded. If this operand is omitted, the standard input file is decoded.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXIT VALUES

The **uudecode** command returns the following values:

0 (zero) The command completed successfully.

>0 An error occurred.

RELATED INFORMATION

Commands: **uuencode(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

The following features are extensions to the XPG4 Version 2 specification:

- More than one file can be specified.

NAME

uuencode - Encodes a binary file

SYNOPSIS

uuencode [*infile*] *remotefile*

DESCRIPTION

The **uuencode** command reads the named *infile* (the default value for *infile* is the standard input file) and produces an encoded version of that file on the standard output file. The encoding uses only printing ASCII characters and includes the file access permission bits of the file and the pathname to be used when the file is decoded by **uudecode**. This pathname is specified by *remotefile*.

The output file is approximately 35 percent larger than the original file.

When the command is reading from the standard input file, the **umask** setting determines the file permissions.

Operands

infile Specifies the pathname of the file to be encoded.

remotefile Specifies the pathname that the file should be given when it is decoded by the **uudecode** command.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXIT VALUES

The **uuencode** command returns the following values:

0 (zero) The command completed successfully.

>0 An error occurred.

RELATED INFORMATION

Commands: **umask(1)**, **uudecode(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

Section 10. User Commands (v - z)

This section contains reference pages for the Open System Services (OSS) user commands with names starting with the letters **v** through **z**.

NAME

vi - Edits files

SYNOPSIS

vi [**-ls**] [**-R**] | [**-r**] [**-c** *subcommand*] [**-t** *tag*] [**-w***number*] [**+***subcommand*] [**-**] [*file* ...]

The **vi** command is a display editor that is based on an underlying line editor (**ex**).

FLAGS

-c *subcommand*
Executes the specified **ex** subcommand (*command*) before displaying the file for which the editor was invoked.

-l Indents appropriately for LISP code, and accepts the **(**, **)**, **{**, **}**, **[**, and **]** characters (parentheses, braces, and brackets) as text rather than interpreting them as **vi** subcommands. The LISP modifier is active in **open** or **visual** modes.

-r[*file*] Recovers *file* after an editor or system crash. If you do not specify a *file*, **vi** displays a list of all saved files.

-R Sets the **readonly** option to protect the file against overwriting.

-s Invokes **vi** in **open** mode. (Silent mode.)

-t *tag* Edits the file containing the *tag* and positions the editor at its definition. To use this flag, you must first create a database of function names and their locations using the **ctags** command. OSS does not support the **ctags** command, however OSS **vi** does support **ctags** databases imported from other systems.

-w*number*
Sets the default window size to *number*. This is useful when you use the editor over a low-speed line.

- Suppresses all interactive user feedback. If you use this flag, file input/output errors do not generate an error message.

+[*subcommand*]
Performs the **ex** subcommand before editing begins. If you do not specify *subcommand*, the cursor is placed on the first line of the file. (Obsolescent)

DESCRIPTION

The **ex** editor subcommands can be used within the **vi** editor, because **vi** is based on **ex**. (For a complete description of **ex** subcommands, see the **ex** reference page.) The *file* argument specifies the files to be edited. If you supply more than one *file* on the command line, **vi** edits each file in the order specified.

When you use **vi**, changes you make to a file are reflected on your display. The position of the cursor on the display indicates its position within the file. The subcommands affect the file at the cursor position.

Limitations of the vi Editor

The maximum limits of the **vi** editor are as follows:

- 2048 bytes per line.
- 256 bytes per global command list.
- 128 bytes in the previous inserted and deleted text.

- 128 bytes in a shell escape command.
- 128 bytes in a string-valued option.
- 30 bytes in a tag name.
- 128 map macros with 2048 bytes total.

Editing Modes

The **vi** editor has the following operational modes:

Command mode

When you start the **vi** editor, it is in Command mode. Any subcommand can be entered from this mode, except commands that can only be used in the Text Input mode (those subcommands that make corrections during text insertion). When subcommands and the other modes end, **vi** returns to Command mode. Pressing **<Esc>** cancels a partial subcommand.

Text Input mode

Entered by the **a**, **A**, **i**, **I**, **o**, **O**, **c***x* (where *x* represents the scope of the subcommand), **C**, **s**, **S,** and **R** subcommands. After entering one of these commands, you can enter text into the editing buffer at the current cursor position. To return to Command mode, press **<Esc>** for normal exit or press the Interrupt key sequence to end abruptly.

Setting Options

The **vi** editor allows you to customize options so that you can use the editor for a specific task. Use the **set** command to set or change an option. To view the current setting of options, enter **:set all** while in **vi** Command mode.

Some options are set to a string or a number value; other options are simply turned on or off. To change an option that is set to a value, enter a command in the form **:set** *option=value*. To toggle an option that can be set to on or off, enter a line of the form **:set** *option* to set it to on or **:set no***option* to set it to off.

Options can be abbreviated in a **set** command. The following list describes some of **vi**'s options, along with their abbreviations and descriptions:

[**no**]**autoindent** (**ai**)

Indents automatically in Text mode to the indentation on the previous line by using the spacing between tab stops specified by the **shiftwidth** option. The default is **noai**. To back the cursor up to the previous tab stop, type **<Ctrl-d>**. This option is not in effect for global commands.

[**no**]**autoprint** (**ap**)

Prints the current line after any command that changes the editing buffer. The default is **ap**. This option applies only to the last command in a sequence of commands on a single line, and is not in effect for global commands.

[**no**]**autowrite** (**aw**)

Writes the editing buffer to the file automatically before the **:e**, **:n**, **:ta**, **:rew**, **:st**, **:su**, **<Ctrl-a>**, and **!** subcommands if the editing buffer was changed since the last **write** command. The default is **noaw**.

[**no**]**beautify** (**bf**)

Prevents user from entering control characters (except for tab, newline, and formfeed) in the editing buffer during text entry. The default is **nobf**. This option does apply to command input. If **beautify** is set, all nonprintable characters other than the Tab, newline, and formfeed characters are discarded from text read in from files.

directory (**dir=**)
Displays the directory that contains the editing buffer. The default is **dir=/tmp**.

[**no**]**edcompatible** (**ed**)
Causes the presence of global (**g**) and confirm (**c**) suffixes on substitute commands to be remembered and toggled by repeating the suffixes during substitutions and causes the read (r) suffix to work like the **r** subcommand. The default is **noed**.

[**no**]**errorbells** (**eb**)
Precedes error messages with an **<Alert>** character. Setting this option off (**noeb**) does not suppress the alerting in visual mode. The default is **noeb**.

[**no**]**exrc** If not set, ignores any **.exrc** file in the current directory during initialization, unless the current directory is that named by the **HOME** variable. The default is **noexrc**.

[**no**]**flash** (**fl**)
Uses visual flash rather than audible bell. The default is **fl**.

hardtabs (**ht=**)
Tells **vi** the distance between the hardware tab stops on your display. The default is **ht=8**.

[**no**]**ignorecase** (**ic**)
Ignores the distinction between uppercase and lowercase while searching for regular expressions. The default is **noic**.

[**no**]**lisp** Enters **vi** in LISP mode. In this mode, **vi** appropriately indents for LISP code and the **(**, **)**, **{**, **}**, **[[**, and **]]**. The default is **nolisp**.

[**no**]**list** Displays text with tabs and the end of lines marked. Tabs are displayed as **^I** and the end of lines as **$**. The default is **nolist**.

[**no**]**magic**
Treats the characters **.**, **[**, and ***** as special characters in scans. In Off mode, only the **(**, **)**, and **$** characters retain special meanings; however, special meaning of other characters can still be invoked by preceding the characters with a \ (backslash). The default is **magic**.

[**no**]**mesg**
Turns on write permission to the terminal while in visual mode. The default is on.

[**no**]**modeline**
Runs an editor command line if found in the first five and the last five lines of the file. An editor command line may be anywhere in a line. To be recognized as a command line, it must contain a space or a tab followed by the string **ex:** or **vi:**. The command line is ended by a second **:** (colon). The editor tries to interpret any data between the first and second **:** as editor commands. The default is **nomodeline**.

[**no**]**number** (**nu**)
Displays lines prefixed with their line numbers. The default is **nonu**.

[**no**]**optimize** (**opt**)
Speeds up the operation of terminals that do not have cursor addressing. The default is **noopt**.

paragraphs (**para=**)
Defines macro names that start paragraphs. The default is **para=IPLPPPQPP LIp-plpipbp**. Single-letter **nroff** macros, such as **.P**, must include the space as a quoted character if respecifying a paragraph. (See **RELATED INFORMATION**.)

[**no**]**prompt**
Prompts for command mode input with a **:** (colon). When not set, no prompt is displayed. The default is on.

readonly
Allows writing to a different file. In addition, the write can be forced by using the **!** (exclamation point) character (see the editor command **write**). The default is off, unless the file lacks write permission or the **-R** flag is specified.

[**no**]**redraw** (**re**)
Simulates a smart display on a dumb display. The default is **re**.

[**no**]**remap**
Enables multi-step translation of map keys. For example, if **r** maps to **u**, and **u** maps to **k**, **remap** causes **r** to map to **k**. If **noremap** is set, **r** maps to **u**. The default is **remap**.

report Sets the number of repetitions of a command before a message is displayed. For subcommands that can produce a number of messages, such as global subcommands, the messages are displayed when the command is completed. The default is **report=5**.

scroll (**scr=**)
Sets the number of lines to be scrolled when you scroll the screen up or down. The default scroll is one-half the size of the screen.

sections (**sect=**)
Defines macro names that start sections. The default is **sect=NHSHH HUuhsh+c**. Single-letter **nroff** macros, such as **.P**, must include the space as a quoted character if respecifying a paragraph. (See **RELATED INFORMATION**.)

shell (**sh=**)
Defines the shell for **!** or **:!** commands. The default is the value of the **SHELL** environment variable.

shiftwidth (**sw=**)
Sets the distance for the software tab stops used by **autoindent**, the shift commands (> and <), and the text input commands (<**Ctrl-d**> and <**Ctrl-t**>) to allow the editor to indent text and move back to a previous indentation. The default is **sw=8**.

[**no**]**showmatch** (**sm**)
Shows the matching open parenthesis **(** or open brace **{** as you type the close parenthesis) or close brace }. The default is **nosm**.

[**no**]**showmode** (**smd**)
Displays mode indicator at the bottom of the screen when in the insert or replace mode. The default is **nosmd**.

[**no**]**slowopen** (**slow**)
Postpones updating the display during inserts. Setting this option alters the display algorithm to accommodate slow or unintelligent terminals. The default is **noslow**.

[**no**]**sourceany**
Allows the use of the **source** command on a file that a user does not own. The default is **nosourceany**.

tabstop (**ts=**)
Sets distance between tab stops when a file is displayed. The default is **ts=8**.

taglength (**tl=**)
Determines length of tag.

[**no**]**tags** (**tag**)
Specifies a list of possible filenames of tag files. The default is **tags /usr/lib/tags**.

term Sets the kind of display you are using. The default is **term=$TERM**, where **$TERM** is the value of the **TERM** shell variable.

[**no**]**terse**
Allows **vi** to display the short form of messages. The default is **noterse**.

[**no**]**timeout** (**to**)
Sets a time limit of 2 seconds on entry of characters. This limit allows the characters in a macro to be entered and processed as separate characters when **timeout** is set. To resume use of the macro, set **notimeout**. The default is **to**.

ttytype (**tty=**)
Same as **term**.

[**no**]**warn**
Displays a warning message before the **!** subcommand executes a shell command if this is the first time you issued a shell command after a given set of changes were made in the editing buffer, but not written to a file. The default is **warn**.

window (**wi=**)
Sets the number of lines displayed in one window of text. The default is dependent on the baud rate at which you are operating: 600 baud or less / 8 lines, 1200 baud / 16 lines, higher speeds / full screen minus 1.

wrapmargin (**wm=**)
Sets the margin for automatic wordwrapping from one line to the next. A value of 0 indicates no wordwrapping. The default is **wm=0**.

[**no**]**wrapscan** (**ws**)
Allows string searches to wrap from the end of the editing buffer to the beginning. The default is **ws**.

wraptype (**wt=**)
Causes words to be wrapped in Japanese style. **wraptype=general** causes general-purpose wrap on word breaks, where word break is defined as whitespace or space between two nonASCII characters. **general** is a combination of **word** and **flexible**.

wraptype=word causes wrap on words. **wraptype=rigid** causes wrap on column and before closing punctuation. **wraptype=flexible** causes wrap on column, but closing punctuation may extend past the margin.

[**no**]**writeany** (**wa**)
Turns off the checks usually made before a **write** command. The default is **nowa**.

Defining Macros

If you use a subcommand or sequence of subcommands frequently, you can create a macro that issues the subcommand or sequence when you call a macro. To create a macro, enter the sequence of subcommands into an editing buffer named with a letter of the alphabet. When used as buffer names, lowercase ASCII letters **a** through **z** overlay the contents of the buffer, while uppercase ASCII letters **A** through **Z** append text to the previous contents of the buffer, allowing the building of a macro piece by piece.

To invoke the macro, enter **@***x*, where *x* is the letter name of the buffer. Enter **@@** to repeat the last macro you invoked.

Mapping Keys

You can use the **map** command to set a keystroke to a subcommand or a sequence of subcommands for use during visual mode. To set a key mapping, enter **:map** *key subcommand* where *key* is the key to which you want to assign a subcommand or sequence of subcommands and *subcommand* is the subcommand or sequence of subcommands. For example, to set **X** to delete lines, enter:

:map X dd

In this example, **X** is the key to which the subcommand is assigned and **dd** is the subcommand.

In the next example, a subcommand sequence is mapped to a key:

:map * {>}

The ***** (asterisk) is the key to which the subcommand sequence is assigned and **{>}** is the subcommand sequence. The **{** (open brace) moves the cursor to the beginning of the paragraph and the **>** (right angle bracket) indents the paragraph to the next **shiftwidth**.

To display the list of the current key mappings while you are in Command mode, enter the **:map** *key* command. To remove a key mapping, enter **:unmap** *key* or **:unmap!** *key* where *key* is the string used after the **:map** command to set the key and subcommand sequence. For example, to remove key mapping for the previous example, enter:

:unmap *

If function keys are defined for your terminal, they can be put in a **map** or **unmap** command by typing **<Ctrl-v>** and then pressing the desired key. In this way, function keys that are unused during editing can be mapped to useful editing subcommand sequences.

If the **!** (exclamation point) character is appended to the command name **map** (**map!**), the mapping is effective during input mode rather than during visual mode.

Abbreviations

You can define abbreviations for long phrases that you use often. **vi** then automatically expands these abbreviations whenever you enter them in insert mode.

To define an abbreviation, enter:

:abbr *abbreviation phrase*

where *abbreviation* is the abbreviation you specify for the longer text specified by *phrase*. For example, to specify the abbreviation **imho** for the phrase **In my humble opinion**, enter:

:abbr imho In my humble opinion

Keeping a Customized Change

The editing environment defaults to certain configuration options. When an editing session is initiated, **vi** attempts to read the **EXINIT** environment variable. If it exists, the editor uses the values defined in **EXINIT**, otherwise the values set in **$HOME/.exrc** are used. If **$HOME/.exrc** does not exist, the default values are used.

The vi Character Sets

The collation sequence, as defined by the value of the **LC_COLLATE** environment variable, defines the alphanumeric set used by your system. This table affects the performance of **vi** macros and subcommands.

The **vi** editor uses the collation sequence to distinguish between a small word and a big word. A small word is bounded by letters or numbers as defined in the collation table. For example, **isn't** is two small words. The ' (apostrophe) is not a number or an alphabetic character, and it bounds both the small word **t** and the small word **isn**. A big word is bounded by spaces, tabs, or newline indicators. For example, **accommodate** is a big word. For more information, see the section **Moving to Words**.

SUBCOMMANDS

Subcommand Syntax

[*named_buffer*] [*operator*] [*number*] *argument*

Surrounding brackets indicate optional items.

[*named_buffer***]**
A temporary text storage area.

[*operator***]**
Specifies the subcommand or action; instructs the **vi** editor.

[*number***]** A whole decimal value that specifies either the extent of the action or a line address. The **vi** editor interprets this number in one of the following ways:

1. Go to line *number*:

   5G
   10z<Return>

2. Go forward *number* columns.

   25<Space>

3. Scroll *number* of lines:

   10<Ctrl-d>
   10<Ctrl-u>

4. Delete *number* lines:

   6dd

5. **%** means *all*. To yank *all* lines:

   %y

   (The preceding command replaces **1,$y**.)

argument
Specifies what to act on. This can be a text object (a character, word, sentence, paragraph, section, or character string) or a text position (a line, position in the current line, or screen position).

Moving Within a File

Enter the following subcommands in Command mode. You can cancel an incomplete subcommand by pressing **<Esc>**.

<Left Arrow>, **h**, **<Ctrl-h>**
Moves the cursor one character to the left.

<Down Arrow>, **j**, **<Ctrl-j>**, **<Ctrl-n>**
Moves the cursor down one line, remaining in the same column.

<Up Arrow>, **k**, **<Ctrl-p>**
Moves the cursor up one line, remaining in the same column.

<Right Arrow>, **l** **<Space>**
Moves the cursor one character to the right.

Long lines: Lines over one screen width are wrapped but not broken. When using the Up Arrow or Page Up key, **@** lines are added at the bottom of the screen when too few physical lines are available to display the complete line. The Down Arrow key moves the entire line off the screen at once.

Character Positioning Within a Line

Enter the following subcommands in Command mode.

^ Moves the cursor to the first nonspace character.

0 Moves the cursor to the beginning of the line.

$ Moves the cursor to the end of the line.

f*x* Moves the cursor to the next *x* character.

F*x* Moves the cursor to the previous *x* character.

t*x* Moves the cursor to one column before the next *x* character.

T*x* Moves the cursor to one column after the previous *x* character.

; Repeats the last **f**, **F**, **t**, or **T** subcommand.

, Repeats the last **f**, **F**, **t**, or **T** subcommand in the opposite direction.

*number***<Space>**
Moves the cursor to the specified column.

Moving to Words

Enter the following subcommands in Command mode.

w Moves the cursor forward to the beginning of a word.

b Moves the cursor backward to the beginning of a word.

e Moves the cursor forward to the end of a word.

W Moves the cursor forward to the beginning of a big word.

B Moves the cursor backward to the beginning of a big word.

E Moves the cursor forward to the end of a big word.

Moving by Line Positioning

Enter the following subcommands in Command mode.

G Moves to the line number given as preceding argument, or the end of the file if no preceding count is given.

H Moves the cursor to the top line on the screen.

L Moves the cursor to the last line on the screen.

M Moves the cursor to the middle line on the screen.

+ Moves the cursor to the next line, at its first nonspace character.

- Moves the cursor to the previous line, at its first nonspace character.

<Return>
Moves the cursor to the next line, at its first nonspace character.

Moving to Sentences, Paragraphs, or Sections

Enter the following subcommands in Command mode. You can cancel an incomplete subcommand by pressing **<Esc>**.

(Places the cursor at the beginning of the previous sentence (or the previous S-expression if you are in LISP mode).

) Places the cursor at the beginning of the next sentence (or the next S-expression if you are in LISP mode).

{ Places the cursor at the beginning of the previous paragraph (or at the next list if you are in LISP mode).

} Places the cursor at the beginning of the next paragraph, at the next section if you are in C mode, or at the next list if you are in LISP mode.

| Requires a count; the cursor is placed in that column (if possible).

]] Places the cursor at the next section, or at the next function if you are in LISP mode.

[[Places the cursor at the previous section, or at the next function if you are in LISP mode.

Paging and Scrolling

<Ctrl-u>
Scrolls up (defaults to value of **scroll** option).

<Ctrl-d>
Scrolls down (defaults to value of **scroll** option).

<Ctrl-f> Pages forward one screen (defaults to value of **window** option).

<Ctrl-b>
Pages backward one screen (defaults to value of **window** option).

<Ctrl-y>
Scrolls the window up one line.

<Ctrl-e> Scrolls the window down one line.

<Ctrl-m>
Moves to the first nonwhite character in the next line. A count specifies the number of lines to go forward.

<Ctrl-t> Inserts *shiftwidth* white space in input mode, if at the beginning of the line or preceded only by white space. This inserted space can only be backed over using **<Ctrl-d>**.

<Ctrl-[> Cancels a partially formed command; sounds the bell if there is none.

In input mode, terminates input mode.

When entering a command on the bottom line of the screen (**ex** command line or search pattern with \ or **?**), terminates input and executes command.

z+ Pages up (defaults to value of **window** option).

z^ Pages down (defaults to value of **window** option).

Searching for Patterns

The following commands allow you to search for patterns within a file. Patterns can be regular expressions as described for **grep**.

/pattern Places the cursor at the next line containing *pattern*.

?*pattern* Places the cursor at the next previous line containing *pattern*.

n Repeats the last search for *pattern* in the same direction.

N Repeats the last search for *pattern* in the opposite direction.

/pattern/+*number*
Places the cursor at the *number*th line after the line matching *pattern*.

?*pattern*?*-number*
Places the cursor at the *number*th line before the line matching *pattern*.

% Finds the parenthesis or brace that matches the one at the current cursor position.

<Ctrl-t> Finds the word at the cursor in the tags file then edits the proper file, placing the cursor at the tag. If the tag is in the current file, moves cursor to it.

Marking and Returning

Enter the following subcommands in Command mode. You can cancel an incomplete subcommand by pressing **<Esc>**.

‘‘ Moves the cursor to the same cursor position of the previous current line.

’’ Moves the cursor to the beginning of the previous current line.

m*x* Marks the current position with the letter specified by *x*.

‘*x* Moves the cursor to the same cursor position of line marked *x*.

'*x* Moves the cursor to the beginning of the line marked *x*.

Adjusting the Screen

Enter the following subcommands in Command mode. An incomplete subcommand can be cancelled by pressing **<Esc>**.

<Ctrl-l> Clears and redraws the screen.

<Ctrl-r> Redraws the screen and eliminates blank lines marked with a **@**.

z<Return>
Redraws the screen with the current line at the top of the screen.

z- Redraws the screen with the current line at the bottom of the screen.

z. Redraws the screen with the current line at the center of the screen.

/*pattern*/**z-**
Redraws the screen with the line containing *pattern* at the bottom.

z*number***<Return>**
Makes the window *number* lines long.

Adding Text to a File--Text Input Mode

The following subcommands are entered in Command mode and bring the **vi** editor into Text Input mode to allow you to add text to your file. End Text Intput mode by pressing **<Esc>**.

a*text* Inserts *text* after the cursor.

A*text* Adds *text* to the end of the line.

i*text* Inserts *text* before the cursor.

I*text* Inserts *text* before the first nonspace character in the line.

o Adds an empty line below the current line.

O Adds an empty line above the current line.

Changing Text While in Input Mode

Use the following commands only while in Text Entry mode. They have different meanings in Command mode.

<Ctrl-h>
Erases the last character.

<Ctrl-w>
Erases the last small word. (For more information about small words, see the section **vi Character Sets**.)

\ Quotes the Erase and Kill characters.

<Esc> Ends insertion, sends the program back to Command mode.

Quit key sequence
Interrupts and terminates insert or **<Ctrl-d>**.

<Ctrl-d>
Goes back to the previous **autoindent** stop.

^<Ctrl-d>
Ends **autoindent** for this line only.

0<Ctrl-d>
Moves the cursor back to the left margin.

<Ctrl-v>
Quotes a nonprinting character.

Changing Text from Command Mode

Use the following subcommands in Command mode. An incomplete subcommand can be cancelled by pressing **<Esc>**.

C Changes the rest of the line (**c$**).

c Must be followed by a movement command. Deletes the specified region of text and enters input mode to replace it with the entered text. If more than part of a single line is affected, the deleted text is saved in the numeric buffers. If only part of the current line is affected, the last character to be deleted is marked with a **$**. A count is passed through to the **move** command. If the command is **cc**, the whole of the current line is changed.

cc Changes a line.

cw Changes a word.

D Deletes the rest of the line (**d$**) and puts it into the **undo** buffer.

d Must be followed by a movement command. Deletes the specified region of text. If more than part of a line is affected, the text is saved in the numeric buffers. A count is passed through to the **move** command. If the command is **dd**, the whole of the current line is deleted.

dd Deletes a line and puts it into the undo buffer.

dw Deletes a word and puts it into the undo buffer.

J Joins lines.

r*x* Replaces the current character with the character specified by *x*.

R Overwrites characters.

s Substitutes characters (**cl**).

S Substitutes lines (**cc**).

u Undoes the previous change.

x Deletes a character.

X Deletes characters before cursor (**dh**).

<< Shifts one line to the left.

<L Shifts all lines from the cursor to the end of the screen to the left. (The < character describes a range upon which the **L** subcommand acts.)

>> Shifts one line to the right.

>L Shifts all lines from the cursor to the end of the screen to the right. (The > character describes a range upon which the **L** subcommand acts.)

~ Changes the letter at the cursor to the opposite case.

Copying and Moving Text

Use the following subcommands in Command mode. An incomplete subcommand can be cancelled by pressing **<Esc>**.

p (**P**) Puts back text in the **undo** buffer after (before) the cursor.

"*x*p (*x*P) Puts back text from the buffer *x* after (before) the cursor. You must precede the character *x* with a double quote.

"*x*d*object*
Deletes *object* into the buffer *x*. You must precede the character *x* with a double quote.

y*object* Yanks *object* into the **undo** buffer (for example, **yw** to yank a word).

"*x***y***object*
Yanks *object* into buffer *x*. You must precede the character *x* with a double quote.

Y Places the line in the **undo** buffer.

Restoring and Repeating Changes

Use the following subcommands in Command mode. An incomplete subcommand can be cancelled by pressing **<Esc>**.

u Undoes the last command.

U Restores the current line if the cursor has not left the line since the last change.

. Repeats the last change or increments the *n***p** command.

Note that this command is not meant for use with a macro. Enter **@@** to repeat a macro.

"*n*p Retrieves the *n*th last delete of a complete line or block of lines. You must precede the character *n* with a double quote.

Saving Changes to a File

Use the following subcommands in Command mode. An incomplete subcommand can be cancelled by pressing **<Esc>**. If you are using these subcommands within the **ex** editor, you do not need to type the : (colon).

:w Writes the editing buffer contents to the original file.

:w *file* Writes the editing buffer contents to the named *file*.

:w! *file* Overwrites *file* with the editing buffer contents.

Interrupting, Cancelling, and Exiting vi

Q Enters the **ex** editor in Command mode.

q Enters the **ex** editor in Command mode if no filename was specified on the **vi** command line when the program was invoked.

ZZ Exits **vi**, saving changes, if any were made.

:q Quits **vi**. If you have changed the contents of the editing buffer, **vi** displays a warning message and does not quit.

:q! Quits **vi**, discarding the editing buffer with no warning.

:sh Runs a shell. You can return to **vi** by pressing **<Ctrl-d>**.

:w !*command*
Runs the file through the specified shell command (causes no change to the file).

:!*command*
Runs *command*, then returns.

:!! Repeats the last **:!***command* command.

n!!*command*
Executes the shell command identified by *command* and replaces the number of lines specified by *n* with the output of *command*. If *n* is not specified, the default is 1. If *command* expects standard input, the lines specified are used as input. (**10!!sort** sorts the next 10 lines.)

!*linescommand*
Works like **n!!***command*, except that *lines* is a line address (for example, **!Gsort** sorts the rest of the file).

Quit key sequence
Interrupts a subcommand.

Editing a Second File

Enter the following subcommands in Command mode. An incomplete subcommand can be cancelled by pressing **<Esc>**.

:e *file* Edits *file*. If you are using this subcommand from the **ex** editor, you do not need to type the : (colon).

:e! Reedits the current file and discards all changes.

:e + *file* Edits *file*, starting at the end.

:e +*number*
Edits *file*, starting at the line *number*.

:e # and **<Ctrl-a>**
Edits the alternate file. The alternate file is usually the previous current filename. However, if changes are pending on the current file when a new file is called, the new file becomes the alternate file.

:r *file* Reads the file into the editing buffer by adding new lines below the current line. If you are using this subcommand from the **ex** editor, you do not need to type the : (colon).

:r !*command*
Runs the shell command identified by *command* and places its output in the file by adding new lines below the current cursor position.

:ta *tag* Edits a file containing *tag* at the location of *tag*. If the tag is in another file and the current file has been changed (and **noaw** is set), a warning is posted. To use this command, you must first create a database of function names and their locations using the **ctags** command. OSS does not support the **ctags** utility. However, **vi** does support **ctags** databases imported from other systems. If you are using this subcommand from the **ex** editor, you do not need to type the : (colon).

<Ctrl-t> Finds the word at the cursor in the tags file and edits the indicated file, placing the cursor at the tag. Same as **:ta**, but the tag is the word to the right of the cursor.

Editing a List of Files

Enter the following subcommands in Command mode.

:n Edits the next file in the list entered on the command line.

:n *file ...* Specifies a new list of files to edit.

Displaying File Information

Enter the following subcommand in Command mode to show the current filename, the current line number, the number of lines in the file, and the percentage of lines of the file that are before the cursor:

<Ctrl-g>

RELATED INFORMATION

Commands: **ed**(1), **grep**(1).

NAME

vproc - Displays version information for program and object files

SYNOPSIS

There are two ways to obtain version-procedure information:

Through the Guardian VPROC utility
Through the OSS **vproc** command

To use the Guardian VPROC utility from a TACL prompt for an object in the Guardian file system:

vproc
[/ *RUN_option*, ... /]
[*filename*]

To use the Guardian VPROC utility from a TACL prompt for an object in the OSS file system:

vproc
/ { *RUN_option*, ... } /
[*pathname*]

To use the Guardian VPROC utility from an OSS shell prompt for an object in the Guardian file system:

gtacl -p vproc
[**'***filename***'**]

To use the Guardian VPROC utility from an OSS shell prompt for an object in the OSS file system:

gtacl -p vproc
[*pathname*]

To use the OSS command from an OSS shell prompt:

vproc
[*path*] ...

To use the OSS command from a TACL prompt:

osh -p vproc
[*path*] ...

FLAGS

The Guardian VPROC utility has the following flags and operands:

RUN_option Specifies one or more TACL RUN command options as flags with arguments. Use these options only when running the Guardian VPROC utility from a TACL prompt. Refer to the *TACL Reference Manual* for a complete list and description of these options.

When using the Guardian VPROC utility from a TACL prompt, you must enter at least one valid value for *RUN_option* if you enter a *pathname* operand.

Do not use RUN options when using the Guardian VPROC utility through the **gtacl** command or when using the OSS **vproc** command. Refer to the **gtacl(1)** reference page for possible ways to redirect Guardian input or output. Use **gtacl** options instead of RUN options.

filename Specifies the Guardian filename of the program file or object file whose version information is to be displayed.

When the Guardian VPROC utility is used from a TACL prompt, the value of **filename** need not be fully qualified. When the Guardian VPROC utility is used through the **gtacl** command, the value of *filename* should be fully qualified and enclosed in single quotes.

TACL wildcard-matching can be used.

pathname Specifies the absolute OSS pathname of the program file or object file whose version information is to be displayed. Information can be returned for any OSS regular file.

When the Guardian VPROC utility is used from a TACL prompt, the pathname must be preceded by at least one specification for *RUN_option*. If no run option is needed, use the NAME option.

Wildcard-matching cannot be used.

The OSS **vproc** command is a script that calls the Guardian VPROC utility. The OSS **vproc** command has the following flags and operands:

path Specifies the relative or absolute pathname of the program file or object file whose version information is to be displayed. Information can be returned for any OSS regular file.

If more than one pathname is specified, all pathnames must be relative to the current working directory.

If **filename**, **pathname**, and *path* are omitted, the Guardian VPROC utility and the OSS **vproc** command run in interactive mode. Pathnames or Guardian filenames can be entered interactively.

DESCRIPTION

The Guardian VPROC utility and the OSS **vproc** command display information that identifies the version of the file pointed to by the *filename*, *pathname*, or *path* operand.

Information can be displayed for any Guardian file with a file code (type) of the following:

0 Indicates a binary file

100 Indicates a TNS format or TNS/R COFF format executable object file or a file in the OSS file system

180 Indicates a file containing text in the Guardian file system or a Guardian C data file (format is consistent with an OSS regular file containing ASCII text)

510 Indicates a standard microcode file

700 Indicates a TNS/R native (COFF or ELF format) relinkable or executable object file

800 Indicates a TNS/E native (ELF format) linkfile or loadfile

860 Indicates a TNS/R millicode file.

870, 871, or 872
: Indicates a TNS/R millicode file.

880 or 881
: Indicates a TNS/R millicode file.

All program and object files have a version. Use the version information returned by **vproc** to identify the code you are using when you report a problem to HP.

An OSS archive file (or other file in **ar** format) might contain member files that do not contain version information. Member files that contain version information are listed. Member files that do not contain version information are not listed. If no member of an archive file contains version information, the timestamp for the last modification to the file is returned.

The **vproc** display has the following fields:

Archive member
: Displays the archive member OSS or UNIX filename. This information appears only for files in **ar** format.

Binder timestamp
: Displays the date and time the program was created. This information appears only for appropriate file types.

GMT Binder Timestamp
: Displays the date and time the ELF file was created in GMT (UTC), rather than displaying it as local time. This information appears only for appropriate file types.

Version procedure
: Displays a string of the form:

cttttrvv_ddmmmyy_nnnnnn_xxxxxx

or

cttttrvv_ddmmmyyyy_nnnnnn_xxxxx

ctttt is the Tandem number (T number) of the corresponding product.

rvv is the version of the product.

ddmmmyy or *ddmmyyyy*
: is the release date of the product version.

nnnnnn is the abbreviated product name of the code.

xxxxxx or *xxxxx*
: is optional information for use by HP.

Target CPU Indicates which type of processor must be used to execute the code.

AXCEL timestamp
: Displays the date and time the code was accelerated. This information appears only for files that have been accelerated for TNS/R.

OCA timestamp
: Displays the date and time the code was accelerated. This information appears only for files that have been accelerated for TNS/E.

Privileged code
This information appears only for files with file code 100 that contain privileged code. The value is **YES**.

Native Mode Indicates whether the file can be executed. This information appears only for files with file code 700. The possible values are **runnable file** and **Not runnable file**.

TNS/E Native Mode
Indicates whether the file can be executed. This information appears only for files with file code 800. The possible values are **runnable file** and **Not runnable file**.

EXAMPLES

1. To display the process version information for the **ed** editor from the OSS shell, enter:

   gtacl -p vproc '/bin/ed'

   This displays information similiar to the following:

   ```
   GMT Binder timestamp: 04JAN2003 11:43:43
   Version procedure:    T8432G07_01AUG99_CRTLMAIN
   Version procedure:    T8626G06_30JAN2003_ABU_OSSUTIL
         Native Mode:    runnable file
   ```

2. To display the process version information for the **ed** editor from a TACL prompt, enter:

   vproc /NAME/ /bin/ed

   This displays information similiar to the following:

   ```
   GMT Binder timestamp: 04JAN2003 11:43:43
   Version procedure:    T8432G07_01AUG99_CRTLMAIN
   Version procedure:    T8626G06_30JAN2003_ABU_OSSUTIL
         Native Mode:    runnable file
   ```

3. To use **vproc** interactively, enter:

   gtacl -p vproc

   and enter filenames or absolute pathnames in response to the prompt:

   ```
   > Enter filename:
   ```

   To leave **vproc**, enter CTRL/Y at any prompt.

4. To display version information about Binder-format object files in an OSS **ar** format archive file named **/nonnative/usr/lib/libc.a**, enter the following from an OSS shell prompt:

   vproc /nonnative/usr/lib/libc.a

   This displays information similiar to the following:

   ```
   Archive member:         versiono
   Binder timestamp:       14FEB2000 00:02:32
   Version procedure:      T8305D40_08MAR2000_STDLIBS_AAO
   Target CPU:             UNSPECIFIED
   .
   .
   .
   ```

5. To display version information about an OSS **ar** format archive file named **/usr/lib/liby.a** that contains no VPROC information, enter the following from an OSS shell prompt:

   vproc /usr/lib/liby.a

   This displays information similiar to the following:

```
/usr/lib/liby.a
        Last modified timestamp: 04JAN2003 04:46:26
        No VPROC found in this ar-format file
```

6. To display information about the OSS **ls** command object file using the OSS **vproc** command from a TACL prompt, enter the following:

   osh -p vproc /bin/ls

   This displays information similar to the following:

```
GMT Binder timestamp:   04JAN2003 12:00:41
Version procedure:      T6523G05_31MAY2000_RTLABB
Version procedure:      S7035D40^17JUL97^03OCT97^AAB
Version procedure:      T8432G07_01AUG99_CRTLMAIN
Version procedure:      T8626G06_30JAN2003_ABU_OSSUTIL
Version procedure:      T8626G05_01Jun00_LS
  Native Mode:  runnable file
```

FILES

/bin/vproc Contains the OSS **vproc** command script.

/tmp/*username*/**.vproc.tempfile.***OSS_process_ID*
A temporary file that is used during processing by the OSS **vproc** command script.

NOTES

To use the Guardian VPROC utility or the OSS **vproc** command, you must first know the location of the file for which you want information. To locate a Guardian file, follow the procedure in the *Guardian User's Guide*. To locate an OSS file, use the **find** command. For example, to determine the location of **sh** and then display the process version for **sh**, enter:

find / -name sh

This returns:

```
find: cannot chdir to </G/oss/zyq00000> : Operation not permitted
 .
 .
 .
/bin/sh
```

Next enter:

gtacl -p vproc '/bin/sh'

This displays information similiar to the following:

```
/bin/sh
        GMT Binder timestamp:   04JAN2003 12:34:12
        Version procedure:      T8432G07_01AUG99_CRTLMAIN
        Version procedure:      T8626G06_30JAN2003_ABU_OSSUTIL
```

```
Native Mode:     runnable file
```

You can also use the **whence** or **type** command to find a file if you are interested only in those files accessible through your **PATH** environment variable values.

DIAGNOSTICS

`ERROR:` [*filename* | *pathname*] `does not exist.`
Either the specified file does not exist or you made a typographical error when entering the filename or pathname value.

`ERROR - NO SUCH VOLUME:` [*filename* | *pathname*]
Either the Guardian volume that you specified does not exist or you made a typographical error when entering the volume-name portion of the filename or pathname value.

`>> NO T9xxx PROC <<`
The version procedure information is not stored in the specified program or object file.

`No VPROC found in this ar-format file`
No VPROC information is stored in the indicated archive file.

`Version procedure:` [*filename* | *pathname*]`: Not object file`
Either the indicated file does not have a file code of a type that VPROC can read, or the file contains only text data.

RELATED INFORMATION

Commands: **find(1)**, **gtacl(1)**, **osh(1)**, **type(1)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

wait - Reports termination status of processes

SYNOPSIS

wait [*job*]

DESCRIPTION

The **wait** command waits for the specified *job* and reports its termination status. If *job* is not given, all currently active child processes are waited for. The exit status from this command is that of the process waited for. (See **Jobs** for a description of the format of *job*.)

EXIT VALUES

If a specified *job* is not known, **wait** returns an exit status of 127. If **wait** is invoked with no arguments and all process IDs known by the invoking shell have terminated, **wait** returns an exit status of 0 (zero). If **wait** detects an error, it returns an exit status in the range 1-126.

NOTES

The **wait** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **sh(1).**

NAME

wall - Sends a message to all users

SYNOPSIS

wall [*file*]

DESCRIPTION

The **wall** command takes its input from the contents of *file* if you specify it; if you do not specify *file*, **wall** reads from the standard input file until either you press the End-of-File key sequence or an end-of-file is encountered. **wall** then sends that input as a message to all users who are logged in. The message is preceded by the heading:

`Broadcast Message from` *user*`@`*node* `(`*tty*`) at` *hh*`:`*mm*

where *user* is the user invoking **wall**, *node* and *tty* are the node and terminal of that user, and *hh*`:`*mm* is the time of the message.

To override any protections other users have set up, you must be operating with appropriate privileges. Typically, the system administrator uses **wall** to warn all users of an impending system shutdown.

The **wall** command sends messages only to the local node.

Operands

file Specifies the pathname of a file to be used as the source of the broadcast message.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification.

The following features are extensions to the XPG4 Version 2 specification:

- The alternate message source *file* can be specified.

NAME

wc - Counts lines, words, characters, and bytes

SYNOPSIS

wc [**-c** | **-k** | **-m**] [**-lw**] [*file* ...]

FLAGS

-c Counts bytes.

-k Counts characters.

-m Counts characters.

-l Counts lines.

-w Counts words.

DESCRIPTION

The **wc** command counts the lines, words, characters, and bytes in a file, or in the standard input file if you do not specify any files, and writes the results to the standard output file. It also keeps a total count for all named files.

A word is defined as a nonzero-length string of characters delimited by spaces, tabs, or newline characters. A line is defined as zero or more characters followed by a newline character.

The **wc** command counts lines, words, and bytes by default; use the appropriate flags to limit **wc** output. Specifying **wc** without flags is the equivalent of specifying **wc -lwc**.

The order in which the counts appear in the output line matches the order in which the flags are entered on the command line. (If you do not specify any flags, the order is lines, words, bytes.)

When you specify more than one file, **wc** displays the name of the file along with the counts.

EXAMPLES

1. To display the number of lines, words, and bytes in the file **text**, enter:

   wc text

   This command results in the following output:

   ```
   27 185 722 text
   ```

   The numbers 27, 185, and 722 are the number of lines, words, and bytes, respectively, in the file **text**.

2. To display only one or two of the three counts, or to display the counts you want in a particular order, include the appropriate flags in the order you want. For example, the following command displays only byte and line counts:

   wc -cl text

   ```
   722 27 text
   ```

3. To count lines, words, and bytes in more than one file, use **wc** with more than one input file or with a filename pattern. For example, the following command can be issued in a directory containing the files **text**, **text1**, and **text2**:

   wc -l text*

   ```
   27      text
   112     text1
   5       text2
   144     total
   ```

The numbers 27, 112, and 5 are the numbers of lines in the files **text**, **text1**, and **text2**, respectively, and 144 is the total number of lines in the three files.

RELATED INFORMATION

Commands: **ls(1)**.

STANDARDS CONFORMANCE

The **-k** flag is an extension to the XPG4 Version 2 specification.

NAME

whatis - Describes a command's function

SYNOPSIS

whatis [**-M** *pathname*] [*keyword* ...]

The **whatis** command looks up a keyword, which may be a command, system call, library function, special filename, or POSIX regular expression and displays the **NAME** line from the reference page. You can then issue the **man** command to display additional information.

FLAGS

-M *pathname*

Specifies an alternative search path. The search path is a list of directories, separated by **:** (colons), in which **whatis** expects to find the standard manual subdirectories.

DESCRIPTION

The **whatis** command is equivalent to the **man** command with the **-f** flag.

The *keyword* argument can be a POSIX regular expression. For more information, see the **grep** reference page.

EXAMPLES

To find out what function the **ls** command performs, enter:

```
whatis ls
```

FILES

/usr/share/man/whatis Keyword database.

RELATED INFORMATION

Commands: **man**(1).

NAME

whence - Interprets command names

SYNOPSIS

whence [**-pv**] *name* ...

FLAGS

-p The **-p** flag does a path search for *name* even if *name* is an alias, a function, or a reserved word.

-v The **-v** flag produces a more verbose report.

DESCRIPTION

The **whence** command indicates how each name given in the argument *name* would be interpreted if it were used as a command name. The flags provide more specific information about the name or its use as a command name.

EXAMPLES

1. The following example shows the **whence** command used with the OSS **grep** command as an argument. The output of the command shows that the **grep** command is located in **/bin**.

   whence -rv grep

   grep is /bin/grep

2. The following example shows the results when a name that is not a command name is used as an agument:

   whence -rv shell

   shell not found

NOTES

The **whence** command is a shell built-in command. It differs from the regular commands in that it does not open a new shell process when it executes.

A general discussion of shell built-in commands and a list of the OSS shell built-in commands are in the reference page for **sh(1)**.

RELATED INFORMATION

Commands: **type(1)**, **sh(1)**.

NAME

who - Identifies users currently logged in

SYNOPSIS

who [**-mTu**]

The **who** command displays information about users on the local system.

FLAGS

-m Displays information about the current terminal.

-T Displays the status of the terminal line and indicates who can write to that terminal as follows:

+ Writable by anyone.

- Writable only by the superuser or its owner.

? Bad line encountered.

-u Displays the username, terminal name, login time (time login shell was started), line activity, and process-ID of each current user. The **LC_TIME** environment variable controls the format of the login time.

DESCRIPTION

The **who** command displays the following information for the users and/or processes you specify:

- User name
- Terminal name
- Date
- Time of login

The general output format of the **who** command is as follows:

user [*state*] *line time* [*activity*] [*process_ID*]

where:

- The *user* argument specifies the user's login name.
- The *state* argument indicates whether or not the line is readable by everyone (see the **-T** flag).
- The *line* argument specifies the name of the TELNET line as found in the **/dev** directory.
- The *time* argument specifies the time that user logged in.
- The *activity* argument specifies the hours and minutes since activity last occurred on that user's line. A **.** (dot) here indicates line activity within the last minute. If the line has been quiet more than 24 hours or has not been used since the last system start-up, the entry is marked as old.
- The *process_ID* argument specifies the process-ID of the user's shell.

Note that **who** only identifies users on the local node.

EXAMPLES

1. To display information about who is using the local system, enter:

   who

   Information similar to the following is displayed:

   ```
   super.super      /G/ZTNT/#PTY6      JUN 08      09:10
   software.rdas    /G/ZTNT/#PTY5      Jun 06      08:20
   ```

2. To display information about who is using the local system and their associated host machine name, enter:

   who -m

   Information similar to the following is displayed:

   ```
   software.rdas    /G/ZTNT/#PTY5      Jun 06     08:20
   ```

RELATED INFORMATION

Commands: **date**(1).

Functions: **wait**(2).

NAME

whoami - Displays the user name for the effective user ID

SYNOPSIS

whoami

DESCRIPTION

The **whoami** command displays the user name associated with your effective user ID.

RELATED INFORMATION

Commands: **who(1)**.

STANDARDS CONFORMANCE

The **whoami** command is an extension to the XPG4 Version 2 specification.

NAME

xargs - Constructs argument lists and runs commands

SYNOPSIS

xargs [**-e** *eof_string*] [**-i**][*replace_string*] | [**-I** *replace_string*] | [**-r**] [**-l**][*number*] |[**-L** *number*] |[**-n** *number*] [**-ptx**] [**-s** *size*] [*command*] [*argument ...*]

The **xargs** command constructs a command line by combining a command string, containing a command and its flags or arguments with additional arguments read from standard input.

FLAGS

-e *eof_string*

Sets the logical End-of-File string to *eof_string*. By default, there is no logical End-of-File. The **xargs** command reads standard input until it encounters either an End-of-File character or the logical **EOF** string.

-i[*replace_string*]

This flag is the same as **-I**. The default *replace_string* for **-i** is **{}**.

The **-i**, **-I**, and **-r** flags are mutually exclusive; the last one of these flags specified takes effect.

-I *replace_string*

Takes an entire line as a single argument and inserts it in each instance of *replace_string* found in the command string. A maximum of five arguments in the command string can each contain one or more instances of *replace_string*. The **xargs** command discards spaces and tabs at the beginning of each line. The argument constructed cannot be larger than 255 bytes. This flag also turns on the **-x** flag.

The **-i**, **-I**, and **-r** flags are mutually exclusive; the last one of these flags specified takes effect.

-l[*number*]

This flag is the same as **-L**. The default *number* is 1. This flag turns on the **-x** flag.

The **-l**, **-L**, and **-n** flags are mutually exclusive; the last one of these flags specified takes effect.

-L *number*

Runs the command string with the specified *number* of nonempty argument lines read from standard input. The last invocation of the command string can have fewer argument lines if fewer than *number* remain. A line ends with the first newline character unless the last character of the line is a space or a tab. A trailing space or tab indicates a continuation through the next nonempty line.

The **-l**, **-L**, and **-n** flags are mutually exclusive; the last one of these flags specified takes effect.

-n *number*

Executes the command string using as many standard input arguments as possible, up to a maximum of *number* (a positive decimal integer). The **xargs** command uses fewer arguments if their total length is greater than the number of characters specified by the **-s** *size* flag (or **LINE_MAX** if there is no **-s** flag). It also uses fewer arguments for the last invocation if fewer than *number* arguments remain. When **-x** is present, each *number* argument must fit the *length* limitation specified by **-s**. When the replacement string **{}** is in effect for the **-i** flag (the deafult), the maximum number of input arguments that can be substituted is 1. Therefore, the use of the **-n** flag in this case has no effect.

The **-l**, **-L**, and **-n** flags are mutually exclusive; the last one of these flags specified takes effect.

-p Asks whether or not to run the command string. Trace mode (**-t**) is turned on to write the command instance to be executed, followed by a prompt to standard error (?...). An affirmative response read from **/dev/tty** executes the command. Any other response causes **xargs** to skip that particular invocation of the command string. You are asked about each invocation.

-r This flag is the same as **-I {}**.

The **-i**, **-I**, and **-r** flags are mutually exclusive; the last one of these flags specified takes effect.

-s *size* Invokes the specified command using as many standard input arguments as possible, yielding a command line length less than *size* (a positive decimal integer) bytes. Fewer arguments are used if any of the following conditions is true:

- The total number of arguments exceeds that specified by **-n**.
- The total number of lines exceeds that specified by **-L**.
- End-of-File is encountered on standard input before *size* bytes are accumulated.

Note that the character count for *size* includes one extra character for each argument and the number of characters in the command name. Values of *size* up to at least **LINE_MAX** bytes are supported.

-t Echoes the command string and each constructed argument list to file descriptor **2** (usually standard error).

-x Stops running **xargs** if any argument list is greater than the number of characters specified by the **-s** *size* flag. This flag is turned on if you specify either the **-I** or **-L** flags. If you do not specify **-I**, **-L**, or **-n**, the total length of all arguments must be within the *length* limit.

DESCRIPTION

The **xargs** command runs the command string as many times as necessary to process all input arguments. The default command string is **/usr/bin/echo**.

Arguments read from standard input are character strings delimited by one or more spaces, tabs, or newline characters. You can embed a space or a tab in arguments by preceding it with a \ (backslash) or by quoting it. The **xargs** command reads characters enclosed in single or double quotes as literals and removes the delimiting quotes. It always discards empty lines.

The **xargs** command ends if it cannot run the command string or if it receives an exit code of **255**. When the command string calls a shell procedure, the shell procedure should explicitly exit with an appropriate value to avoid accidentally returning **255**. (See the **sh** command.)

The **LC_MESSAGES** variables determines the locale's equivalent of **y** and **n** (for yes/no queries).

EXAMPLES

1. To use a command on files whose names are listed in a file, use a command line similar to the following:

   xargs ls -l < cfiles

   If **cfiles** contains the text, enter:

```
main.c  readit.c
gettoken.c
putobj.c
```

Then **xargs** constructs and runs the command:

```
ls -l  main.c  readit.c  gettoken.c  putobj.c
```

Each shell command line can be up to **LINE_MAX** bytes long. If **cfiles** contains more filenames than fit on a single line, then **xargs** runs the **ls** command with the filenames that fit. It then constructs and runs another **ls** command using the remaining filenames. Depending on the names listed in **cfiles**, the commands might look like the following:

```
ls  -l  main.c  readit.c  gettoken.c...
ls  -l  getisx.c  getprp.c  getpid.c...
ls  -l  fltadd.c  fltmult.c  fltdiv.c...
```

2. To construct commands that contain a certain number of filenames, use a command line similar to the following:

```
xargs  -t  -n 2  diff  <<end
starting  chap1  concepts  chap2  writing
chap3
end
```

This constructs and runs **diff** commands that contain two filenames each (**-n 2**):

```
diff  starting  chap1
diff  concepts  chap2
diff  writing  chap3
```

The **-t** flag tells **xargs** to display each command before running it so that you can see what is happening. The **<<end** and **end** arguments define a *Here Document*, which uses the text entered before the **end** line as standard input for the **xargs** command. (For more details, see the section **Inline Input (Here) Documents** in the **sh** reference page.)

3. To insert filenames into the middle of commands, use a command line similar to the following:

```
ls | xargs  -t  -r  mv  {}  {}.old
```

This renames all files in the current directory by adding **.old** to the end of each name. The **-r** tells **xargs** to insert each line of the **ls** directory listing where **{ }** (braces) appear. (You might need to precede the braces with shell escape characters, depending on what shell you are using.)

If the current directory contains the files **chap1**, **chap2**, and **chap3**, then this constructs the following commands:

```
mv  chap1  chap1.old
mv  chap2  chap2.old
mv  chap3  chap3.old
```

4. To run a command on files that you select individually, use a command line similar to the following:

```
ls | xargs  -p  -n 1  ar  r  lib.a
```

This allows you to select files to add to the library **lib.a**. The **-p** flag tells **xargs** to display each **ar** command it constructs and ask if you want to run it. Press **y**, or the

locale's equivalent of a **y**, and press **<Return>** to run the command. Press **<Return>** alone if you do not want to run it.

EXIT STATUS

The **xargs** command returns the following exit values:

0 All invocations of *command* returned exit status 0 (zero).

1-125 A command line meeting the specified requirements could be assembled, one or more of the invocations of *command* returned a nonzero exit status, or some other error occurred.

126 The specified comand was found but could not be invoked.

127 The specified command could not be found.

255 No further invocations using the current data stream will succeed.

RELATED INFORMATION

Commands: **sh(1)**.

NAME

yacc - Generates an LR(1) parsing program from input

SYNOPSIS

yacc [**-vltds**] [**-b** *prefix*] [**-N** *number*] [**-p** *symbol_prefix*] [**-P** *pathname*] *grammar*

The **yacc** command converts a context-free grammar specification into a set of tables for a simple automaton that executes an LR(1) parsing algorithm.

FLAGS

-b *prefix* Uses *prefix* instead of **y** as the prefix for all output filenames (*prefix***.tab.c**, *prefix***.tab.h**, and *prefix***.output**).

-d Produces the **y.tab.h** file, which contains the **#define** statements that associate the **yacc**-assigned token codes with your token names. This allows source files other than **y.tab.c** to access the token codes by including this header file.

-l Does not include any **#line** constructs in **y.tab.c**.

-N *number*
Provides **yacc** with extra storage for building its LALR tables, which may be necessary when compiling very large grammars. *number* should be larger than 40,000 when you use this flag.

-p *symbol_prefix*
Allows multiple **yacc** parsers to be linked together. Use *symbol_prefix* instead of **yy** to prefix global symbols.

-P *pathname*
Specifies an alternative parser (instead of **/usr/ccs/lib/yaccpar**). *pathname* specifies the filename of the skeleton to be used in place of **yaccpar**.

-s Breaks the **yyparse()** function into several smaller functions. Because its size is somewhat proportional to that of the grammar, it is possible for **yyparse()** to become too large to compile, optimize, or execute efficiently.

-t Compiles runtime debugging code. By default, this code is not included when **y.tab.c** is compiled. If **YYDEBUG** has a nonzero value, the C compiler (**cc**) includes the debugging code, whether or not the **-t** flag was used. Without compiling this code, **yyparse()** will run more quickly.

-v Produces the **y.output** file, which contains a readable description of the parsing tables and a report on conflicts generated by grammar ambiguities.

DESCRIPTION

The **yacc** grammar can be ambiguous; specified precedence rules are used to resolve ambiguities.

You must compile the **y.tab.c** output file with a C language compiler to produce the **yyparse()** function. This function must be loaded with a **yylex** lexical analyzer function, as well as the **main()** routine and **yyerror()**, an error-handling routine (you must provide these routines). The **lex** command is useful for creating lexical analyzers usable by **yacc**.

The **yacc** program reads its skeleton parser from the file **/usr/ccs/lib/yaccpar**. Use the **-P** flag or the environment variable **PARSER** to specify another location for **yacc** to read from.

Syntax for yacc Input

This section contains a formal description of the **yacc** input file (or grammar file), which is normally named with a **.y** suffix. The section provides a listing of the special values, macros, and functions recognized by **yacc**.

The general format of the **yacc** input file is:

[*definitions*]
%%
[*rules*]
[**%%**
[*user functions*]]

where

definitions Is the section where you define the variables to be used later in the grammar, such as in the rules section. It is also the file where files are included (**#include**) and processing conditions are defined. This section is optional.

rules Is the section that contains grammar rules for the parser. A **yacc** input file must have a rules section.

user functions Is the section that contains user-supplied functions that can be used by the actions in the rules section. This section is optional.

Each line in the *definitions* can be:

%{

%} When placed on lines by themselves, these enclose C code to be passed into the global definitions of the output file. Such lines commonly include preprocessor directives and declarations of external variables and functions.

%token *token* [*token* ...]
Lists tokens or terminal symbols to be used in the rest of the input file. This line is needed for tokens that do not appear in other **%** definitions.

%left *token* [*token* ...]
Indicates that each *token* is an operator, that all tokens in this definition have equal precedence, and that a succession of the operators listed in this definition are evaluated left to right.

%right *token* [*token* ...]
Indicates that each *token* is an operator, that all tokens in this definition have equal precedence, and that a succession of the operators listed in this definition are evaluated right to left.

%nonassoc *token* [*token* ...]
Indicates that each *token* is an operator, and that the operators listed in this definition cannot appear in succession.

%start *symbol* Indicates the highest-level production rule to be reduced; in other words, the rule where the parser can consider its work done and terminate. If this definition is not included, the parser uses the first production rule. *symbol* must be non-terminal (not a token).

%type < *type* > *symbol* [*symbol* ...]
Defines each *symbol* as data type *type*, to resolve ambiguities.

%union *union-def*
Defines the **yylval** global variable as a union, where *union-def* is a standard C definition in the format:

{ *type member* ; [*type member* ; ...] }

At least one member should be an **int**. Any valid C data type can be defined, including structures. When you run **yacc** with the **-d** option, the definition of **yylval** is placed in the **y.tab.h** file and can be referred to in a **lex** input file.

Every token (non-terminal symbol) must be listed in one of the preceding **%** definitions. Multiple tokens can be separated by white space or commas. All the tokens in **%left**, **%right**, and **%nonassoc** definitions are assigned a precedence, with tokens in later definitions having precedence over those in earlier definitions.

In addition to symbols, a token can be a literal character enclosed in single quotes. (Multibyte characters are recognized by the lexical analyzer and returned as tokens.) The following special characters can be used, just as in C programs:

\a	Alert
\n	Newline
\t	Tab
\v	Vertical tab
\r	Carriage return
\b	Backspace
\f	Formfeed
\\	Backslash
\'	Single quote
\?	Question mark
n	One or more octal digits specifying the integer value of the character

The *rules* section consists of a series of production rules that the parser tries to reduce. The format of each production rule is:

symbol : *symbol-sequence*
[*action*] [| *symbol-sequence*
[action] ...] ;

where *symbol-sequence* consists of zero or more symbols separated by white space. The first *symbol* must be the first character of the line, but new lines and other white space can appear anywhere else in the rule. All terminal symbols must be declared in **%token** definitions.

Each *symbol-sequence* represents an alternative way of reducing the rule. A symbol can appear recursively in its own rule. Always use left-recursion (where the recursive symbol appears before the terminating case in *symbol-sequence*).

The specific sequence:

%prec *token*

indicates that the current sequence of symbols is to be preferred over others, at the level of precedence assigned to *token* in the *definitions* section.

The specially defined token **error** matches any unrecognized sequence of input. This token causes the parser to invoke the **yyerror** function. By default, the parser tries to synchronize with the input and continue processing it by reading and discarding all input up to the symbol following **error**. (You can override this behavior through the **yyerrok** action.) If no **error** token appears in the **yacc** input file, the parser exits with an error message upon encountering

unrecognized input.

The parser always executes *action* after encountering the symbol that precedes it. Thus, an action can appear in the middle of a *symbol-sequence*, after each *symbol-sequence*, or after multiple instances of *symbol-sequence*. In the last case, *action* is executed when the parser matches any of the sequences.

The *action* consists of standard C code within braces and can also take the following values, variables, and keywords.

yylval If the token returned by the **yylex** function is associated with a significant value, **yylex** should place the value in this global variable. By default, **yylval** is of type **int**. The definitions section can include a **%union** definition to associate with other data types, including structures. If you run **yacc** with the **-d** option, the full **yylval** definition is passed into the **y.tab.h** file for access by **lex**.

yyerrok Causes the parser to start parsing tokens immediately after an erroneous sequence, instead of performing the default action of reading and discarding tokens up to a synchronization token. The **yyerrok** action should appear immediately after the **error** token.

$ [<*type*>] *n* Refers to symbol *n*, a token index in the production, counting from the beginning of the production rule, where the first symbol after the colon is **$1**. The *type* variable is the name of one of the **union** lines listed in the **%union** directive in the declaration section. The <*type*> syntax (non-standard) allows the value to be set to a specific data type. Note that you will rarely need to use the *type* syntax.

$ [<*type*>] $ Refers to the value returned by the matched *symbol-sequence* and used for the matched symbol when reducing other rules. The *symbol-sequence* generally assigns a value to **$$**. The *type* variable is the name of one of the **union** lines listed in the **%union** directive in the declaration section. The <*type*> syntax (non-standard) allows the value to be set to a specific data type. Note that you will rarely need to use the *type* syntax.

The *user functions* section contains user-supplied programs. If you supply a lexical analyzer (**yylex**) to the parser, it must be contained in the user functions section.

The following functions, which are contained in the user functions section, are invoked within the **yyparse** function generated by **yacc**.

yylex( ) The lexical analyzer called by **yyparse** to recognize each token of input. Usually this function is created by **lex**. **yylex** reads input, recognizes expressions within the input, and returns tokens. The function returns an **int** value. A return value of 0 (zero) means the end of input.

yyerror(*string*)
The function that the parser calls upon encountering an input error. The default function, defined in **liby.a**, simply prints *string* to the standard error. The user can redefine the function. The function's type is **void**.

The **liby.a** library contains default **main( )** and **yyerror( )** functions. These look like the following, respectively:

```
main()
{
     setlocale(LC_ALL, "");
     (void) yyparse();
     return(0);
```

```
}

int yyerror(s);
     char *s;
{
     fprintf(stderr,"%s\n",s);
     return (0);
}
```

Comments, in C syntax, can appear anywhere in the *user functions* or *definitions* sections. In the *rules* section, comments can appear wherever a symbol is allowed. Blank lines or lines consisting of white space can be inserted anywhere in the file, and are ignored.

EXAMPLES

This section describes the example programs for the **lex** and **yacc** commands, which together create a simple desk calculator program that performs addition, subtraction, multiplication, and division operations. The calculator program also allows you to assign values to variables (each designated by a single lowercase ASCII letter), and then use the variables in calculations. The files that contain the program are as follows:

calc.l The **lex** specification file that defines the lexical analysis rules.

calc.y The **yacc** grammar file that defines the parsing rules and calls the **yylex()** function created by **lex** to provide input.

The remaining text expects that the current directory is the directory that contains the **lex** and **yacc** example program files.

Compiling the Example Program

Perform the following steps to create the example program using **lex** and **yacc**:

1. Process the **yacc** grammar file using the **-d** flag. The **-d** flag tells **yacc** to create a file that defines the tokens it uses in addition to the C language source code.

   yacc -d calc.y

2. The following files are created:

   y.tab.c The C language source file that **yacc** created for the parser.

   y.tab.h A header file containing **#define** statements for the tokens used by the parser.

3. Process the **lex** specification file:

   lex calc.l

4. The following file is created:

   lex.yy.c The C language source file that **lex** created for the lexical analyzer.

5. Compile and link the two C language source files:

   cc -o calc y.tab.c lex.yy.c

6. The following files are created (the ***.o** files are created temporarily and then removed):

   y.tab.o The object file for **y.tab.c**.

   lex.yy.o The object file for **lex.yy.c**.

calc The executable program file.

You can then run the program directly by entering:

calc

Then enter numbers and operators in calculator fashion. After you press **<Return>**, the program displays the result of the operation. If you assign a value to a variable as follows, the cursor moves to the next line:

m=4 <Return>

_

You can then use the variable in calculations and it will have the value assigned to it:

m+5 <Return>

9

The Parser Source Code

The text that follows shows the contents of the file **calc.y**. This file has entries in all three of the sections of a **yacc** grammar file: declarations, rules, and programs.

```
%{
#include <stdio.h>

int regs[26];
int base;

%}

%start list

%token DIGIT LETTER

%left '|'
%left '&'
%left '+' '-'
%left '*' '/' '%'
%left UMINUS /*supplies precedence for unary minus */

%%      /*beginning of rules section */

list    :       /*empty */
        |       list stat '\n'
        |       list error '\n'
                {       yyerrok;        }
        ;

stat    :       expr
                {       printf("%d\n",$1);      }
        |       LETTER '=' expr
                {       regs[$1] = $3;  }
        ;

expr    :       '(' expr ')'
        {       $$ = $2;        }
```

```
  |     expr '*' expr
                { $$ = $1 * $3;   }
        |       expr '/' expr
        {       $$ = $1 / $3;   }
        |       expr '%' expr
                { $$ = $1 % $3;   }
        |       expr '+' expr
                { $$ = $1 + $3;   }
        |       expr '-' expr
                { $$ = $1 - $3;   }
        |       expr '&' expr
                { $$ = $1 & $3;   }
        |       expr '|' expr
                { $$ = $1 | $3;   }
        |       '-' expr %prec UMINUS
                { $$ = -$2;       }
        |       LETTER
                { $$ = regs[$1];  }
        |       number
        ;

number  :       DIGIT
                { $$ = $1; base = ($1==0) ? 8:10; }
        |       number  DIGIT
                { $$ = base * $1 + $2;    }
        ;

%%
main()
{
        return(yyparse());
}

yyerror(s)
char *s;
{
        fprintf(stderr,"%s\n",s);
}

yywrap()
{
        return(1);
}
```

Declarations Section

This section contains entries that perform the following functions:

- Includes standard I/O header file.
- Defines global variables.
- Defines the **list** rule as the place to start processing.
- Defines the tokens used by the parser.

- Defines the operators and their precedence.

Rules Section
The rules section defines the rules that parse the input stream.

Programs Section
The programs section contains the following routines. Because these routines are included in this file, you do not need to use the **yacc** library when processing this file.

main()	The required main program that calls **yyparse()** to start the program.
yyerror(s)	This error handling routine only prints a syntax error message.
yywrap()	The wrap-up routine that returns a value of 1 when the end of input occurs.

The Lexical Analyzer Source Code
This shows the contents of the file **calc.lex**. This file contains **include** statements for standard intput and output, as well as for the **y.tab.h** file. The **yacc** program generates that file from the **yacc** grammar file information, if you use the **-d** flag with the **yacc** command. The file **y.tab.h** contains definitions for the tokens that the parser program uses. In addition, **calc.lex** contains the rules used to generate the tokens from the input stream.

```
%{

#include <stdio.h>
#include "y.tab.h"
int c;
extern int yylval;
%}
%%
" "     ;
[a-z]   {
                c = yytext[0];
                yylval = c - 'a';
                return(LETTER);
        }
[0-9]   {
                c = yytext[0];
                yylval = c - '0';
                return(DIGIT);
        }
[^a-z 0-9]      {
                c = yytext[0];
                return(c);
                }
```

FILES

y.output	A readable description of parsing tables and a report on conflicts generated by grammar ambiguities.
y.tab.c	Output file.
y.tab.h	Definitions for token names.

yacc.tmp	Temporary file.
yacc.debug	Temporary file.
yacc.acts	Temporary file.
/usr/ccs/lib/yaccpar	Default skeleton parser for C programs.
/usr/ccs/lib/liby.a	**yacc** library.

RELATED INFORMATION

Commands: **lex**(1).

NAME

zcat - Expands compressed data

SYNOPSIS

zcat [**-n**] [*file*[**.Z**] ...]

FLAGS

-n Specifies that no header is added or expected. This flag might be useful for expanding old files.

DESCRIPTION

The **zcat** command writes the uncompressed version of a compressed file to the standard output file. The compressed (**.Z**) file remains intact.

zcat is identical to the command **uncompress -c**.

You can specify the compressed target file with or without the **.Z** suffix; if you do not specify the suffix, **zcat** assumes it. If you do not specify a file or if you specify **-** by itself, then the **zcat** command expands the standard input file.

Operands

file[**.Z**] Specifies the pathname of the compressed file to be expanded. If the **.Z** suffix is omitted, it is assumed.

Environment Variables

This command supports the use of the **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH** environment variables.

EXIT VALUES

The **zcat** command returns the following values:

0 (zero) The command completed successfully.

>0 An error occurred or an attempt was made to expand a file that is not compressed.

RELATED INFORMATION

Commands: **compress(1)**, **uncompress(1)**.

STANDARDS CONFORMANCE

This command conforms to the XPG4 Version 2 specification with extensions.

Section 11. File Format Reference Pages

This section contains reference pages for selected OSS file formats. See the *Open System Services System Calls Reference Manual* for other reference pages you might expect to find in this section.

NAME

charmap - Defines character symbols as character encodings

DESCRIPTION

The character set description (**charmap**) source file defines character symbols as character encodings.

The CHARMAP Section

The **CHARMAP** section must precede all other sections in the **charmap** file. The **CHARMAP** section of the **charmap** file maps symbolic character names to code points. All supported code sets have the portable character set as a proper subset. The portable character set consists of the following character symbols (listed by their standardized symbolic names) and hexadecimal encodings:

Table 11–1. The Portable Character Set

Symbol Name	Code	Symbol Name	Code
<NUL>	\x00	**<comma>**	\x2C
<SOH>	\x01	**<hyphen>**	\x2D
<STX>	\x02	**<period>**	\x2E
<ETX>	\x03	**<slash>**	\x2F
<EOT>	\x04	**<zero>**	\x30
<ENQ>	\x05	**<one>**	\x31D
<ACK>	\x06	**<two>**	\x32E
<alert>	\x07	**<three>**	\x33F
<backspace>	\x08	**<four>**	\x34
<tab>	\x09	**<five>**	\x35
<newline>	\x0A	**<six>**	\x36
<vertical-tab>	\x0B	**<seven>**	\x37
<form-feed>	\x0C	**<eight>**	\x38
<carriage-return>	\x0D	**<nine>**	\x39
<SO>	\x0E	**<colon>**	\x3A
<SI>	\x0F	**<semi-colon>**	\x3B
<DLE>	\x10	**<less-than>**	\x3C
<DC1>	\x11	**<equal-sign>**	\x3D
<DC2>	\x12	**<greater-than>**	\x3E
<DC3>	\x13	**<question-mark>**	\x3F
<DC4>	\x14	**<commercial-at>**	\x40
<NAK>	\x15	**<A>**	\x41
<SYN>	\x16	**<B>**	\x42
<ETB>	\x17	**<C>**	\x43
<CAN>	\x18	**<D>**	\x44
<EM>	\x19	**<E>**	\x45
<SUB>	\x1A	**<F>**	\x46
<ESC>	\x1B	**<G>**	\x47
<IS4>	\x1C	**<H>**	\x48
<IS3>	\x1D	**<I>**	\x49
<IS2>	\x1E	**<J>**	\x4A
<IS1>	\x1F	**<K>**	\x4B
<space>	\x20	**<L>**	\x4C
<exclamation-mark>	\x21	**<M>**	\x4D

<quotation-mark>	\x22	**<N>**	\x4E
<number-sign>	\x23	**<O>**	\x4F
<dollar-sign>	\x24	**<P>**	\x50
<percent>	\x25	**<Q>**	\x51
<ampersand>	\x26	**<R>**	\x52
<apostrophe>	\x27	**<S>**	\x53
<left-parenthesis>	\x28	**<T>**	\x54
<right-parenthesis>	\x29	**<U>**	\x55
<asterisk>	\x2A	**<V>**	\x56
<plus-sign>	\x2B	**<W>**	ex57
<X>	\x58	**<l>**	\x6C
<Y>	\x59	**<m>**	\x6D
<Z>	\x5A	**<n>**	\x6E
<left-bracket>	\x5B	**<o>**	\x6F
<backslash>	\x5C	**<p>**	\x70
<right-bracket>	\x5D	**<q>**	\x71
<circumflex>	\x5E	**<r>**	\x72
<underscore>	\x5F	**<s>**	\x73
<grave-accent>	\x60	**<t>**	\x74
<a>	\x61	**<u>**	\x75
<b>	\x62	**<v>**	\x76
<c>	\x63	**<w>**	\x77
<d>	\x64	**<x>**	\x78
<e>	\x65	**<y>**	\x79
<f>	\x66	**<z>**	\x7A
<g>	\x67	**<left-brace>**	\x7B
<h>	\x68	**<vertical-line>**	\x7C
<i>	\x69	**<right-brace>**	\x7D
<j>	\x6A	**<tilde>**	\x7E
<k>	\x6B	**<DEL>**	\x7F

The **CHARMAP** section contains the following four components:

- The **CHARMAP** section header.
- An optional special symbolic name declarations section. The symbolic name and value must be separated by one or more space characters. The following are the special symbolic names and their meanings:

  <code_set_name>
  Specifies the name of the coded character set for which the **charmap** file is defined. This value determines the value returned by the **nl_langinfo** (**CODESET**) subroutine.

  <mb_cur_max>
  Specifies the maximum number of bytes in a character for the coded character set. Valid values are 1 to 4. The default value is 1.

  <mb_cur_min>
  Specifies the minimum number of bytes in a character for the coded character set. Since all supported code sets have the portable character set as a proper subset, this value must be 1.

<escape_char>
Specifies the escape character that indicates encodings in hexadecimal or octal notation. The default value is a \ (backslash).

<comment_char>
Specifies the character used to indicate a comment within a **charmap** file. The default value is a # (number sign).

- Mapping statements for the defined coded character set.

  Each statement in this section lists a symbolic name for a character and its associated encodings. A symbolic name begins with the < (left angle bracket) character and ends with the > (right angle bracket) character. The characters between the < and > can be any characters from the portable character set, except for control and space characters. The > character may be used if it is escaped with the escape character (as specified by the **<escape_char>** special symbolic name). A symbolic name cannot exceed 32 bytes in length.

  The format of a symbolic name definition is:

  <char_symbol> *encoding*

  An encoding is specified as one or more character constants, with the maximum number of character constants specified by the **<mb_cur_max>** special symbolic name. The encodings may be listed as decimal, octal, or hexadecimal constants with the following formats:

Hexadecimal constant	**x***xxx*, where *x* is a hexadecimal digit.
Octal constant	*ooo*, where *o* is an octal digit.
Decimal constant	**d***ddd*, where *d* is a decimal digit.

  Some examples of character symbol definitions are the following:

```
<A>             \d65            #decimal constant
<B>             \x42            #hexadecimal constant
<j10101>        \x81\xA1        #multiple hexadecimal constants
```

  A range of symbolic names and corresponding encoded values may also be defined, where the non-numeric prefix for each symbolic name is common, and the numeric portion of the second symbolic name is equal to or greater than the numeric portion of the first symbolic name. In this format, a symbolic name value consists of zero or more non-numeric characters followed by an integer of one or more decimal digits. This format defines a series of symbolic names. For example, the string **<j0101>...<j0104>** is interpreted as the **<j0101>**, **<j0102>**, **<j0103>**, and **<j0104>** symbolic names, in that order.

  In statements defining ranges of symbolic names, the encoded value listed is the value for the first symbolic name in the range. Subsequent symbolic names have encoded values in increasing order. For example:

```
<j0101>...<j0104>         \d129\d254
```

  The preceding statement is interpreted as follows:

```
<j0101> \d129\d254
<j0102> \d129\d255
<j0103> \d130\d0
<j0104> \d130\d1
```

Although you cannot assign multiple encodings to one symbolic name, you can create multiple names for one encoded value. This is because some characters have several common names. For example, the "**.**" character is called a *period* in some parts of the world, and a *full stop* in others. Both names may appear in the **charmap**. For example:

```
<period>        \x2e
<full-stop>     \x2e
```

If used, comments must begin with the character specified by the <**comment_char**> special symbolic name.

- The **END CHARMAP** section trailer.

The following is an example of a portion of a possible **CHARMAP** section from a **charmap** file:

```
CHARMAP
<code_set_name>         "ISO8859-1"
<mb_cur_max>            1
<mb_cur_min>            1
<escape_char>           \
<comment_char>          #

<NUL>                   \x00
<SOH>                   \x01
<STX>                   \x02
<ETX>                   \x03
<EOT>                   \x04
<ENQ>                   \x05
<ACK>                   \x06
<alert>                 \x07
<backspace>             \x09
<tab>                   \x09
<newline>               \x0a
<vertical-tab>          \x0b
<form-feed>             \x0c
<carriage-return>       \x0d
END CHARMAP
```

RELATED INFORMATION

Commands: **locale(1)**, **localedef(1)**.

Files: **locale(4)**.

NAME

hosts - Contains information about the hosts in the network

DESCRIPTION

The **/etc/hosts** file contains information about the hosts in the network. A host entry consists of a host address in standard dot notation and the host name. The entry can optionally contain aliases for the host name. Each entry takes the following form:

address name aliases

The fields contain the following information.

address The host address in standard dot notation.

name The name of the host.

aliases Any alias names for the host.

Fields are separated by one or more spaces or tab characters. Comments begin with the number sign (#). Routines that search the **hosts** file do not interpret characters from the beginning of a comment to the end of the line. A host name can contain any printable character except a field delimiter, newline character, or comment character (#).

EXAMPLES

Example lines from a **hosts** file are shown below:

```
131.253.124.22   toms       #Tom Smith
155.187.223.67   granada    #name server
```

FILES

/etc/hosts

RELATED INFORMATION

Files: **networks(4)**, **protocols(4)**, **services(4)**.

Functions: **endhostent(3)**, **gethostbyaddr(3)**, **gethostbyname(3)**, **gethostent(3)**, **sethostent(3)**.

NAME

hosts.equiv - Describes node file for trusted remote hosts and users

SYNOPSIS

/etc/hosts.equiv

DESCRIPTION

The **/etc/hosts.equiv** and **.rhosts** files provide the "remote authentication" database for the **rsh** command. The files specify remote hosts and users that are considered trusted. Only trusted users from the remote host are allowed to access the local system. The **/etc/hosts.equiv** file applies to the entire system, while individual users can maintain their own **.rhosts** files in their home directories. To maintain system security, these files must be carefully created and maintained. The remote authentication procedure determines whether a user from a remote host should be allowed to access the local system with the identity of a local user. This procedure first checks the **/etc/hosts.equiv** file and then checks the **.rhosts** file in the home directory of the local user who is requesting access. The **rsh** command fails if the remote authentication procedure fails.

Each entry in the file has the form:

hostname [*username*]

If the entry is in the form:

hostname

then all users from the named host are trusted. That is, they are allowed to access the system with the same user name as they have on the remote system. This form of entry can be used in both the **/etc/hosts.equiv** and **.rhosts** files.

If the entry is in the form:

hostname username

then the named user from the named host is allowed to access the system. This form can be used in individual **.rhosts** files to allow a remote user to access the system as a different local user. If this form is used in the **/etc/hosts.equiv** file, the named remote user is allowed to access the system with the same capabilities as any local user.

FILES

/etc/hosts.equiv

~/.rhosts

NOTES

The *hostname* value used in the **/etc/hosts.equiv** and **.rhosts** files must be the official name of the host, not one of its nicknames.

Super ID access is handled as a special case. Only the **.rhosts** file is checked when access is attempted for the super ID. To help maintain system security, the **/etc/hosts.equiv** file is not checked.

RELATED INFORMATION

Commands: **rsh(1)**.

Files: **hosts(4)**, **.rhosts(4)**.

STANDARDS CONFORMANCE

This file is an extension to the XPG4 Version 2 specification.

NAME

ipnodes - Defines the hosts using IPv6 network addresses

DESCRIPTION

The Guardian IPNODES file is the IPv6-equivalent of the IPv4 **/etc/hosts** file. It contains IP addresses and host names, where the IP addresses can be either in IPv4 or IPv6 format.

Guardian DEFINEs determine whether a name resolution service is searched first, or whether the local databases (the IPNODES and **hosts** files) are searched first. If the DEFINEs =TCPIPˆNODEˆFILE, $SYSTEM.ZTCPIP.IPNODES and =TCPIPˆHOSTˆFILE, $SYSTEM.ZTCPIP.HOSTS are in effect, the Guardian copies of the local files are searched first.

No **/etc** directory link is used for IPNODES. NonStop TCP/IPv6 does not require an OSS **/etc/ipnodes** file.

Each entry in the IPNODES file has the format:

node_address stack_name

where

node_address Specifies a text string version of the address. This has one of the following forms:

IPv4 Dotted decimal format as *ddd.ddd.ddd.ddd*, for example:

172.17.201.43

IPv6 Hexadecimal string format as *x*:*x*:*x*:*x*:*x*:*x*:*x*:*x*, for example:

1080:0:0:0:8:800:200C:417A

Compressed hexadecimal string format that omits zero values, for example:

1080:::8:800:200C:417A

In mixed form as *x*:*x*:*x*:*x*:*x*:*x*:*d*.*d*.*d*.*d*, for example:

::FFFF:13.1.68.3

as a mapped value, or

::13.1;68.3

as a compatible value.

stack_name Is an arbitrary character string that identifies the TCP/IP stack or host for the corresponding host address.

NOTES

The maximum length of an IPv6 address as a text string is defined as **INET6_ADDRSTRLEN** in the header file **in6.h**.

EXAMPLES

Example lines from an **ipnodes** file are shown below:

```
3ffe:1111:200:1:a00:2bff:fe55    foo-ipv6
10.12.1.100              foo-ipv4
fe80:12::1234              test1
```

FILES

/G/SYSTEM/ZTCPIP/IPNODES
Contains the network host definitions for IPv6 addresses.

/etc/hosts Contains the network host definitions for IPv4 addresses.

RELATED INFORMATION

Files: **hosts(4)**, **networks(4)**, **protocols(4)**, **services(4)**.

Functions: **freeaddrinfo(3)**, **gai_strerror(3)**, **getaddrinfo(3)**, **getnameinfo(3)**, **if_freenameindex(3)**, **if_indextoname(3)**, **if_nameindex(3)**, **if_nametoindex(3)**, **inet_ntop(3)**, **inet_pton(3)**.

STANDARDS CONFORMANCE

This file is an extension to the XPG4 specification.

NAME

locale - Contains one or more categories that describe a locale

DESCRIPTION

For information on writing programs that use the internationalization features of Open System Services, refer to the *Software Internationalization Guide*.

A locale definition source file contains one or more categories that describe a locale. Files using this format can be converted into a locale by using the **localedef** command. Locales can be modified only by editing a locale definition source file and then using the **localedef** command again on the new source file.

Each locale source file section defines a category of locale data. A source file should not contain more than one section for the same category. The following standard categories are supported:

LC_COLLATE
Defines character or string collation information.

LC_CTYPE
Defines character classification, case conversion, and other character attributes.

LC_MESSAGES
Defines the format for affirmative and negative responses.

LC_MONETARY
Defines rules and symbols for formatting monetary numeric information.

LC_NUMERIC
Defines a list of rules and symbols for formatting nonmonetary numeric information.

LC_TIME
Defines a list of rules and symbols for formatting time and date information.

The category source definition consists of the following:

- The category header (category name).
- The associated keyword/value pairs that comprise the category body.
- The category trailer (**END** *category_name*).

For example:

LC_CTYPE
source for **LC_CTYPE** *category*
END LC_CTYPE

The source for all of the categories is specified using keywords, strings, character literals, and character symbols. Each keyword identifies either a definition or a rule. The remainder of the statement containing the keyword contains the operands to the keyword. Operands are separated from the keyword by one or more spaces. A statement may be continued on the next line by placing a \ (backslash) as the last character before the newline character that terminates the line. Lines containing the # (comment character) in the first column are treated as comment lines.

A symbolic name begins with the < (left angle bracket) character and ends with the > (right angle bracket) character. The characters between the < and the > can be any characters from the portable character set except for control and space characters. A symbolic name cannot exceed 32 bytes in length. For example, **<A-diaeresis>** is a valid symbolic name. Any symbolic name referenced in the source file should either be one of the portable character set symbols, or should

be defined in the provided character set description source file (**charmap**).

A character literal is the character itself, or else a decimal, hexadecimal, or octal constant. A decimal constant is of the following form:

\d*ddd*

where *d* is a decimal digit. A hexadecimal constant is of the following form:

\x*xxx*

where *x* is a hexadecimal digit. An octal constant is of the following form:

\ooo

where *o* is an octal digit.

The explicit definition of each category in a locale definition source file is not required. When a category is undefined in a locale definition source file, it defaults to the C locale definition.

The LC_COLLATE Category

The **LC_COLLATE** category defines the relative order between collating elements.

A collation element is the unit of comparison for collation. A collation element may be a character or a sequence of characters. Every collation element in the locale has a set of weights, which determine if the collation element collates before, equal to, or after the other collation elements in the locale. Each collation element is assigned collation weights by the **localedef** command when the locale definition source file is compiled. These collation weights are then used by applications programs that compare strings.

Comparison of strings is performed by comparing the collation weights of each character in the string until either a difference is found or the strings are determined to be equal. This comparison may be performed several times if the locale defines multiple collation orders. For example, in the French locale, the strings are compared using a primary set of collation weights. If they are equal on the basis of this comparison, they are compared again using a secondary set of collation weights. A collating element has a set of collation weights associated with it that is equal to the number of collation orders defined for the locale.

Every character defined in the **charmap** file (or every character in the portable character set if no **charmap** file is specified) is itself a collating element. Additional collating elements can be defined using the **collating-element** statement. The syntax is as follows:

collating-element <*character_symbol*> **from** <*string*>

The **LC_COLLATE** category begins with the keyword **LC_COLLATE** and ends with the **END LC_COLLATE** keyword.

The following keywords are recognized in the **LC_COLLATE** category:

collating-element

The **collating-element** statement is used to specify multicharacter collating elements.

The *character_symbol* argument defines a collating element that is a string of one or more characters as a single collating element. The *character_symbol* argument cannot duplicate any symbolic name in the current **charmap** file, or any other symbolic name defined in this collation definition. The *string* argument specifies a string of two or more characters that define the *character_symbol* argument. The following are examples of the syntax for the **collating-element** statement:

collating-element <ch> from <c><h>
collating-element <e-acute> from <acute><e>
collating-element <11> from <1><1>

A *character_symbol* argument defined by the **collating-element** statement is recognized only within the **LC_COLLATE** category.

collating-symbol

The **collating-symbol** statement is used to specify collation symbols for use in collation sequence statements.

The syntax for the **collating-symbol** statement is as follows:

collating-symbol <*collating_symbol*>

The *collating_symbol* argument cannot duplicate any symbolic name in the current **charmap** file, or any other symbolic name defined in this collation definition. The following are examples of the syntax for the **collating-symbol** statement:

collating-symbol <UPPER_CASE>
collating-symbol <HIGH>

A *collating_symbol* argument defined by the **collating-symbol** statement is recognized only within the **LC_COLLATE** category.

substitute

The **substitute** statement is used to define a substring substitution in a string to be collated.

The syntax for the **substitute** statement is as follows:

substitute "*regular_expr*" **with** "*replacement*"

The *regular_expr* argument defines a basic regular expression enclosed in **""** (double quotes). The *replacement* argument consists of zero or more characters and regular expression back references (for example, **\1** through **\9**) enclosed in **""** (double quotes).

When strings are collated based on a collation definition containing **substitute** statements, the collation acts as if occurrences of substrings matching the *regular_expr* string are replaced by the *replacement* string in the same manner as the **ed** command performs a global substitution. After all substitutions are complete, the strings are compared based on the specified collation order. Ranges in the regular expression are interpreted according to the character classification specified by the **LC_CTYPE** environment variable at collation time. If more than one **substitute** statement is defined in the collation definition, the collation process acts as if the **substitute** statements are applied to the strings in the order they occur in the source definition.

A *regular_expr* argument defined by the **substitute** statement is recognized only with the **LC_COLLATE** category.

A situation where substitution is necessary for dictionary order is the expansion of **Mc** to **Mac** in personal names. The following **substitute** statement performs this substitution:

substitute "^Mc\(\{0,1\}\)\([[:upper:]]\)" with " Mac\1\2"
substitute " Mc\(\{0,1\}\)\([[:upper:]]\)" with " Mac\1\2"

order_start

The **order_start** statement is followed by one or more collation order statements, assigning collation weights to collating elements. This statement is mandatory.

The syntax for the **order_start** statement is as follows:

order_start <*sort_rules*>**;**<*sort_rules*>**;...;**<*sort_rules*>
collation_order_statements
order_end

The <*sort_rules*> have the following syntax:

*keyword***,** *keyword***,...,***keyword*

where *keyword* is one of the keywords **forward**, **backward**, **no-substitute**, and **position**.

The *sort_rules* directives are optional. If present, they define the rules to apply during string comparison. The number of specified *sort_rules* directives defines the number of weights each collating element is assigned; that is, the number of collation orders in the locale. If no *sort_rules* directives are present, one **forward** directive is assumed and comparisons are made on a character basis rather than a string basis. If present, the first *sort_rules* directive applies when comparing strings using primary weight, the second when comparing strings using the secondary weight, and so on. Each set of *sort_rules* directives is separated by a **;** (semicolon). A *sort_rules* directive consists of one or more comma-separated keywords. The following keywords are supported:

copy Specifies the name of an existing locale to be used as the definition of this category. If you include a **copy** statement, no other keyword shall be specified.

forward Specifies that collation weight comparisons proceed from the beginning of a string toward the end of the string.

backward
Specifies that collation weight comparisons proceed from the end of a string toward the beginning of the string.

no-substitute
Specifies that substitution operations defined by the **substitute** statement are disabled for this collation order.

position Specifies that collation weight comparisons consider the relative position of nonignored elements in the string. That is, if strings compare as equal, the element with the shortest distance from the starting point of the string collates first.

The **forward** and **backward** keywords are mutually exclusive. The following is an example of the syntax for the *sort_rules* directives:

order_start forward;backward,no-substitute

The syntax of the collation order statements is as follows:

- Each collation element consists of a <*character_symbol*> specification, followed by whitespace and a set of collation orders.
- All characters in the character set must be placed in the collation order, either explicitly, or implicitly via the ellipsis symbol (**...**) or by using the **UNDEFINED** special symbol.
- The number of collation order statements must match the **order_start** specification.

The optional operands for each collation element are used to define the primary, secondary, or subsequent weights for the collating element. The special symbol **IGNORE** is used to indicate a collating element that is to be ignored when strings are compared.

An **ellipsis** keyword appearing in place of a *collating_element_list* indicates the weights are to be assigned, for the characters in the identified range, in numerically increasing order from the weight for the character symbol on the left-hand side of the preceding statement.

The use of the **ellipsis** keyword results in a locale that may collate differently when compiled with different character set description (**charmap**) source files. For this reason, the **localedef** command will issue a warning when the **ellipsis** keyword is encountered.

The **UNDEFINED** special symbol includes all coded character set values not specified explicitly or with an ellipsis symbol. These characters are inserted in the character collation order at the point indicated by the **UNDEFINED** special symbol in the order of their character code set values. If no **UNDEFINED** special symbol exists and the collation order does not specify all collation elements from the coded character set, a warning is issued and all undefined characters are placed at the end of the character collation order.

The following is an example of a collation order statement in the **LC_COLLATE** locale definition source file category:

```
order_start     forward;backward
UNDEFINED       IGNORE;IGNORE
<LOW>
<space>         <LOW>;<space>
..              <LOW>;...
<a>             <a>;<a>
<a-acute>       <a>;<a-acute>
<a-grave>       <a>;<a-grave>
<A>             <a>;<A>
<A-acute>       <a>;<A-acute>
<A-grave>       <a>;<A-grave>
<ch>            <ch>;<ch>
<Ch>            <ch>;<Ch>
<s>             <s>;<s>
<ss>            <s><s>;<s><s>
<eszet>         <s><s>;<eszet><eszet>
...             <HIGH>;...
<HIGH>
order_end
```

This example is interpreted as follows:

- The **UNDEFINED** special symbol indicates that all characters not specified in the definition (either explicitly or by the ellipsis symbol) are ignored for collation purposes.
- All collating elements between **<space>** and **<a>** have the same primary equivalence class and individual secondary weights based on their coded character set values.
- All versions of the letter **a** — uppercase and lowercase, and with or without diacriticals — belong to the same primary collation class.
- The **<c><h>** multicharacter collating element is represented by the **<ch>** collating symbol and belongs to the same primary equivalence class as the **<C><h>** multicharacter collating element.
- The **<eszet>** character is collated as an **<s><s>** string. That is, one **<eszet>** character is expanded to two characters before comparing.

The LC_CTYPE Category

The **LC_CTYPE** category of a locale definition source file defines character classification, case conversion, and other character attributes. This category begins with an **LC_CTYPE** category header and terminates with an **END LC_CTYPE** category trailer.

All operands for **LC_CTYPE** category statements are defined as lists of characters. Each list consists of one or more semicolon-separated characters or symbolic character names.

The following keywords are recognized in the **LC_CTYPE** category. In the descriptions, the term automatically included means that an error does not occur if the referenced characters are included or omitted. The characters will be provided if they are missing and will be accepted if they are present.

copy Specifies the name of an existing locale to be used as the definition of this category. If you include a **copy** statement, no other keyword shall be specified.

upper Defines uppercase letter characters. No character defined by the **cntrl**, **digit**, **punct**, or **space** keyword can be specified. If **upper** is not defined, **A** through **Z** default to **upper**.

lower Defines lowercase letter characters. No character defined by the **cntrl**, **digit**, **punct**, or **space** keyword can be specified. If **lower** is not defined, **a** through **z** default to **lower**.

alpha Defines all letter characters. No character defined by the **cntrl**, **digit**, **punct**, or **space** keyword can be specified. Characters defined by the **upper** and **lower** keywords are automatically included in this character class.

digit Defines numeric digit characters. Only the digits **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, and **9** can be specified. If **digit** is not defined, **0** through **9** default to **digit**.

alnum Defines alphanumeric characters. No character defined by the **cntrl**, **punct**, or **space** keyword can be specified. Characters defined by the **alpha** and **digit** keywords are automatically included in this character class.

space Defines white-space characters. No character defined by the **upper**, **lower**, **alpha**, **digit**, **graph**, **cntrl**, or **xdigit** keyword can be specified. If **space** is not defined, the space, formfeed, newline, carriage-return, tab, and vertical tab characters default to **space**.

cntrl Defines control characters. No character defined by the **upper**, **lower**, **alpha**, **digit**, **punct**, **graph**, **print**, **xdigit**, **blank**, or **space** keyword can be specified.

punct Defines punctuation characters. A character defined as the space character and characters defined by the **upper**, **lower**, **alpha**, **digit**, **cntrl**, or **xdigit** keyword, or as the **<space>** character cannot be specified.

graph Defines printable characters, excluding the space character. If this keyword is not specified, characters defined by the **upper**, **lower**, **alpha**, **digit**, **xdigit**, and **punct** keywords are automatically included in this character class. No character defined by the **cntrl** keyword can be specified.

print Defines printable characters, including the space character. If this keyword is not specified, the space character and characters defined by the **upper**, **lower**, **alpha**, **digit**, **xdigit**, and **punct** keywords and the **<space>** character are automatically included in this character class. No character defined by the **cntrl** keyword can be specified.

xdigit Defines hexadecimal digit characters. Only the digits **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, and **9** can be specified. Any character can be specified for the hexadecimal values for **10** to **15**, however. These alternate hexadecimal digits are not used by standard conversion routines when converting digit strings from hexadecimal to numeric quantities. If **xdigit** is not defined, the numbers **0** through **9** and the letters **A** through **F** and **a** through **f** default to **xdigit**.

blank Defines blank characters. If this keyword is not specified, the space and horizontal tab characters are included in this character class. Any characters defined by this statement are automatically included in the **space** class.

toupper Defines the mapping of lowercase characters to uppercase characters. Operands for this keyword consist of comma-separated character pairs. Each character pair is enclosed in **()** (parentheses) and separated from the next pair by a **;** (semicolon). The first character in each pair is considered a lowercase character; the second character is considered an uppercase character. Only characters defined by the **lower** and **upper** keywords can be specified. If **toupper** is not defined, **a** through **z** is mapped to **A** through **Z** by default.

tolower Defines the mapping of uppercase characters to lowercase characters. Operands for this keyword consist of comma-separated character pairs. Each character pair is enclosed in **()** (parentheses) and separated from the next pair by a **;** (semicolon). The first character in each pair is considered an uppercase character; the second character is considered a lowercase character. Only characters defined by the **lower** and **upper** keywords can be specified.

The **tolower** keyword is optional. If this keyword is not specified, the mapping defaults to the reverse mapping of the **toupper** keyword, if specified. If the **toupper** and **tolower** keywords are both unspecified, the mapping for each defaults to that of the C locale.

Additional keywords can be provided to define new character classifications. For example:

vowel **<a>;<e>;<i>;<o>;<u>;<y>**

The **LC_CTYPE** category does not support multicharacter elements. For example, the German Eszet character is traditionally classified as a lowercase letter. There is no corresponding uppercase letter; in proper capitalization of German text, the Eszet character is replaced by the two characters **SS**. This kind of conversion is outside of the scope of the **toupper** and **tolower** keywords.

The following is an example of a possible **LC_CTYPE** category listed in a locale definition source file:

```
LC_CTYPE
#"alpha" is by default "upper" and "lower"
#"alnum" is by default "alpha" and "digit"
#"print" is by default "alnum", "punct" and the space character
#"graph" is by default "alnum" and "punct"
#"tolower" is by default the reverse mapping of "toupper"
#
upper     <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
          <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>
#
lower     <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
          <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>
#
digit     <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
          <seven>;<eight>;<nine>
#
space     <tab>;<newline>;<vertical-tab>;<form-feed>;\
          <carriage-return>;<space>
#
cntrl     <alert>;<backspace>;<tab>;<newline>;<vertical-tab>;\
```

```
  <form-feed>;<carriage-return>;<NUL>;<SOH>;<STX>;\
          <ETX>;<EOT>;<ENQ>;<ACK>;<SO>;<SI>;<DLE>;<DC1>;<DC2>;\
          <DC3>;<DC4>;<NAK>;<SYN>;<ETB>;<CAN>;<EM>;<SUB>;\
          <ESC>;<IS4>;<IS3>;<IS2>;<IS1>;<DEL>
#
punct     <exclamation-mark>;<quotation-mark>;<number-sign>;\
          <dollar-sign>;<percent-sign>;<ampersand>;<asterisk>;\
          <apostrophe>;<left-parenthesis>;<right-parenthesis>;\
          <plus-sign>;<comma>;<hyphen>;<period>;<slash>;\
          <colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
          <greater-than-sign>;<question-mark>;<commercial-at>;\
          <left-square-bracket>;<backslash>;<circumflex>;\
          <right-square-bracket>;<underline>;<grave-accent>;\
          <left-curly-bracket>;<vertical-line>;<tilde>;\
          <right-curly-bracket>
#
xdigit    <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
          <seven>;<eight>;<nine>;<A>;<B>;<C>;<D>;<E>;<F>;\
          <a>;<b>;<c>;<d>;<e>;<f>
#
blank     <space>;<tab>
#
toupper   (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
          (<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
          (<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
          (<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
          (<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);\
          (<z>,<Z>)
#
END LC_CTYPE
```

The LC_MESSAGES Category

The **LC_MESSAGES** category of a locale definition source file defines the format for affirmative and negative system responses. This category begins with an **LC_MESSAGES** category header and terminates with an **END LC_MESSAGES** category trailer.

All operands for the **LC_MESSAGES** category are defined as strings or extended regular expressions bounded by " " (double quotes). These operands are separated from the keyword they define by one or more spaces. Two adjacent " " (double quotes) indicate an undefined value. The following keywords are recognized in the **LC_MESSAGES** category:

copy Specifies the name of an existing locale to be used as the definition of this category. If you include a **copy** statement, no other keyword shall be specified.

yesexpr Specifies an extended regular expression that describes the acceptable affirmative response to a question expecting an affirmative or negative response.

noexpr Specifies an extended regular expression that describes the acceptable negative response to a question expecting an affirmative or negative response.

yesstr Specifies the locale's equivalents of an acceptable affirmative response. This string is accessible to applications through the **nl_langinfo** subroutine as **nl_langinfo** (**YESSTR**).

nostr Specifies the locale's equivalents of an acceptable negative response. This string is accessible to applications through the **nl_langinfo** subroutine as **nl_langinfo** (**NOSTR**).

The following is an example of a possible **LC_MESSAGES** category listed in a locale definition source file:

```
LC_MESSAGES
#
yesexpr   "<circumflex><left-square-bracket><y><Y><right-square-bracket>"
noexpr    "<circumflex><left-square-bracket><n><N><right-square-bracket>"
yesstr    "<y>:<Y>:<y><e><s>"
nostr     "<n>:<N>:<n><o>"
#
END LC_MESSAGES
```

The LC_MONETARY Category

The **LC_MONETARY** category of a locale definition source file defines rules and symbols for formatting monetary numeric information. This category begins with an **LC_MONETARY** category header and terminates with an **END LC_MONETARY** category trailer.

All operands for the **LC_MONETARY** category keywords are defined as string or integer values. String values are bounded by " " (double quotes). All values are separated from the keyword they define by one or more spaces. Two adjacent " " (double quotes) indicate an undefined string value. A **-1** (negative one) indicates an undefined integer value. The following keywords are recognized in the **LC_MONETARY** category:

copy Specifies the name of an existing locale to be used as the definition of this category. If you include a **copy** statement, no other keyword shall be specified.

int_curr_symbol
Specifies the string used for the international currency symbol. The operand for the **int_curr_symbol** keyword is a 4-character string. The first three characters contain the alphabetic international currency symbol. The fourth character specifies a character separator between the international currency symbol and a monetary quantity.

currency_symbol
Specifies the string used for the local currency symbol.

mon_decimal_point
Specifies the string used for the decimal delimiter used to format monetary quantities.

mon_thousands_sep
Specifies the character separator used for grouping digits to the left of the decimal delimiter in formatted monetary quantities.

mon_grouping
Specifies a string that defines the size of each group of digits in formatted monetary quantities. The operand for the **mon_grouping** keyword consists of a sequence of semicolon-separated integers. Each integer specifies the number of digits in a group. The initial integer defines the size of the group immediately to the left of the decimal delimiter. The following integers define succeeding groups to the left of the previous group. If the last digit is a 0 (zero), subsequent grouping is performed using the previous digit. If the last digit is nonzero, grouping is only performed for the number of groups specified.

The following is an example of the interpretation of the **mon_grouping** statement. Assuming the value to be formatted is **123456789** and the operand for the

mon_thousands_sep keyword is **’**, the following results occur:

mon_grouping Value	**Formatted Value**
3	123456’789
3;0	123’456’789
3;2	1234’56’789
3;2;0	12’34’56’789

positive_sign
Specifies the string used to indicate a nonnegative-valued formatted monetary quantity.

negative_sign
Specifies the string used to indicate a negative-valued formatted monetary quantity.

int_frac_digits
Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the **int_curr_symbol** value.

frac_digits
Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the **currency_symbol** value.

p_cs_precedes
Specifies an integer value indicating whether the **int_curr_symbol** or **currency_symbol** string precedes or follows the value for a nonnegative-formatted monetary quantity. The following integer values are recognized:

0 Indicates that the currency symbol follows the monetary quantity.

1 Indicates that the currency symbol precedes the monetary quantity.

p_sep_by_space
Specifies an integer value indicating whether the **int_curr_symbol** or **currency_symbol** string is separated by a space from a nonnegative-formatted monetary quantity. The following integer values are recognized:

0 Indicates that no space separates the currency symbol from the monetary quantity.

1 Indicates that a space separates the currency symbol from the monetary quantity.

2 Indicates that a space separates the currency symbol and the **positive_sign** string, if adjacent.

n_cs_precedes
Specifies an integer value indicating whether the **int_curr_symbol** or **currency_symbol** string precedes or follows the value for a negative-formatted monetary quantity. The following integer values are recognized:

0 Indicates that the currency symbol follows the monetary quantity.

1 Indicates that the currency symbol precedes the monetary quantity.

n_sep_by_space
Specifies an integer value indicating whether the **int_curr_symbol** or **currency_symbol** string is separated by a space from a negative-formatted monetary quantity. The following integer values are recognized:

0 Indicates that no space separates the currency symbol from the monetary quantity.

1 Indicates that a space separates the currency symbol from the monetary quantity.

2 Indicates that a space separates the currency symbol and the **negative_sign** string, if adjacent.

p_sign_posn
Specifies an integer value indicating the positioning of the **positive_sign** string for a nonnegative-formatted monetary quantity. The following integer values are recognized:

0 Indicates that a **left_parenthesis** and **right_parenthesis** symbol enclose both the the monetary quantity and the **int_curr_symbol** or **currency_symbol** string.

1 Indicates that the **positive_sign** string precedes the quantity and the **int_curr_symbol** or **currency_symbol** string.

2 Indicates that the **positive_sign** string follows the quantity and the **int_curr_symbol** or **currency_symbol** string.

3 Indicates that the **positive_sign** string immediately precedes the **int_curr_symbol** or **currency_symbol** string.

4 Indicates that the **positive_sign** string immediately follows the **int_curr_symbol** or **currency_symbol** string.

n_sign_posn
Specifies an integer value indicating the positioning of the **negative_sign** string for a negative-formatted monetary quantity. The following integer values are recognized:

0 Indicates that a **left_parenthesis** and **right_parenthesis** symbol enclose both the the monetary quantity and the **int_curr_symbol** or **currency_symbol** string.

1 Indicates that the **negative_sign** string precedes the quantity and the **int_curr_symbol** or **currency_symbol** string.

2 Indicates that the **negative_sign** string follows the quantity and the **int_curr_symbol** or **currency_symbol** string.

3 Indicates that the **negative_sign** string immediately precedes the **int_curr_symbol** or **currency_symbol** string.

4 Indicates that the **negative_sign** string immediately follows the **int_curr_symbol** or **currency_symbol** string.

debit_sign
Specifies the string used for the debit symbol (**DB**) to indicate a negative-formatted monetary quantity.

credit_sign
Specifies the string used for the credit symbol (**CR**) to indicate a nonnegative-formatted monetary quantity.

left_parenthesis
Specifies the character, equivalent to a **(** (left parenthesis), used by the **p_sign_posn** and **n_sign_posn** statements to enclose a monetary quantity and currency symbol.

right_parenthesis
Specifies the character, equivalent to a **)** (right parenthesis), used by the **p_sign_posn** and **n_sign_posn** statements to enclose a monetary quantity and currency symbol.

A unique customized monetary format can be produced by changing the value of a single statement. For example, the following table shows the results of using all combinations of defined values for the **p_cs_precedes**, **p_sep_by_space**, and **p_sign_posn** statements:

p_cs_precedes	p_sign_posn	p_sep_by_space		
		0	1	2
1	0	($1.25)	($ 1.25)	($1.25)
	1	+ $1.25	+$ 1.25	+$1.25
	2	$1.25 +	$ 1.25+	$1.25+
	3	+ $1.25	+$ 1.25	+$1.25
	4	$ +1.25	$+ 1.25	$+1.25
0	0	(1.25 $)	(1.25 $)	(1.25$)
	1	+1.25 $	+1.25 $	+1.25$
	2	1.25$ +	1.25 $+	1.25$+
	3	1.25+ $	1.25 +$	1.25+$
	4	1.25$ +	1.25 $+	1.25$+

The following is an example of a possible **LC_MONETARY** category listed in a locale definition source file:

```
LC_MONETARY
#
int_curr_symbol         "<U><S><D>"
currency_symbol         "<dollar-sign>"
mon_decimal_point       "<period>"
mon_thousands_sep       "<comma>"
mon_grouping            <3>;<0>
positive_sign           "<plus-sign>"
negative_sign           "<hyphen>"
int_frac_digits         <2>
frac_digits             <2>
p_cs_precedes           <1>
p_sep_by_space          <2>
n_cs_precedes           <1>
n_sep_by_space          <2>
p_sign_posn             <3>
n_sign_posn             <3>
debit_sign              "<D><B>"
credit_sign             "<C><R>"
```

```
left_parenthesis        "<left-parenthesis>"
right_parenthesis       "<right-parenthesis>"
#
END LC_MONETARY
```

The LC_NUMERIC Category

The **LC_NUMERIC** category of a locale definition source file defines rules and symbols for formatting nonmonetary numeric information. This category begins with an **LC_NUMERIC** category header and terminates with an **END LC_NUMERIC** category trailer.

All operands for the **LC_NUMERIC** category keywords are defined as string or integer values. String values are bounded by **" "** (double quotes). All values are separated from the keyword they define by one or more spaces. Two adjacent **" "** (double quotes) indicate an undefined string value. A **-1** (negative one) indicates an undefined integer value. The following keywords are recognized in the **LC_NUMERIC** category:

copy Specifies the name of an existing locale to be used as the definition of this category. If you include a **copy** statement, no other keyword shall be specified.

decimal_point
Specifies the decimal delimiter string used to format nonmonetary numeric quantities.

thousands_sep
Specifies the string separator used for grouping digits to the left of the decimal delimiter in formatted nonmonetary numeric quantities.

grouping
Defines the size of each group of digits in formatted monetary quantities. The operand for the grouping keyword consists of a sequence of semicolon-separated integers. Each integer specifies the number of digits in a group. The initial integer defines the size of the group immediately to the left of the decimal delimiter. The following integers define succeeding groups to the left of the previous group. Grouping is performed for each integer specified for the grouping keyword, unless the last integer is 0 (zero), in which case the size of the last group is repeatedly used for remaining digits.

The following is an example of the interpretation of the **grouping** statement. Assuming the value to be formatted is **123456789** and the operand for the **thousands_sep** keyword is **'** (single quote), the following results occur:

grouping Value	**Formatted Value**
3	123456'789
3;0	123'456'789
3;2	1234'56'789
3;2;0	12'34'56'789

The following is an example of a possible **LC_NUMERIC** category listed in a locale definition source file:

```
LC_NUMERIC
#
decimal_point       "<period>"
thousands_sep       "<comma>"
grouping  <3>;<0>
#
END LC_NUMERIC
```

The LC_TIME Category

The **LC_TIME** category of a locale definition source file defines rules and symbols for formatting time and date information. This category begins with an **LC_TIME** category header and terminates with an **END LC_TIME** category trailer.

All operands for the **LC_TIME** category keywords are defined as string or integer values. String values are bounded by **" "** (double quotes). All values are separated from the keyword they define by one or more spaces. Two adjacent **" "** (double quotes) indicate an undefined string value. A **-1** (negative one) indicates an undefined integer value. Field descriptors are used by commands and subroutines that query the **LC_TIME** category to represent elements of time and date formats. The field descriptors used by commands and subroutines that query the **LC_TIME** category for time formatting are described in this section, immediately following the descriptions of valid keywords.

The following keywords are recognized in the **LC_TIME** category:

copy Specifies the name of an existing locale to be used as the definition of this category. If you include a **copy** statement, no other keyword shall be specified.

abday Defines the abbreviated weekday names corresponding to the **%a** field descriptor. Recognized values consist of 7 semicolon-separated strings. The first string corresponds to the abbreviated name for the first day of the week (**Sun**), the second to the abbreviated name for the second day of the week, and so on.

day Defines the full spelling of the weekday names corresponding to the **%A** field descriptor. Recognized values consist of 7 semicolon-separated strings. The first string corresponds to the full spelling of the name of the first day of the week (**Sunday**), the second to the name of the second day of the week, and so on.

abmon Defines the abbreviated month names corresponding to the **%b** field descriptor. Recognized values consist of 12 semicolon-separated strings. The first string corresponds to the abbreviated name for the first month of the year (**Jan**), the second to the abbreviated name for the second month of the year, and so on.

mon Defines the full spelling of the month names corresponding to the **%B** field descriptor. Recognized values consist of 12 semicolon-separated strings. The first string corresponds to the full spelling of the name for the first month of the year (**January**), the second to the full spelling of the name for the second month of the year, and so on.

d_t_fmt Defines the string used for the standard date and time format corresponding to the **%c** field descriptor. The string can contain any combination of characters and field descriptors.

d_fmt Defines the string used for the standard date format corresponding to the **%x** field descriptor. The string can contain any combination of characters and field descriptors.

t_fmt Defines the string used for the standard time format corresponding to the **%X** field descriptor. The string can contain any combination of characters and field descriptors.

am_pm Defines the strings used to represent a.m. (before noon) and p.m. (after noon) corresponding to the **%p** field descriptor. Recognized values consist of two semicolon-separated strings. The first string corresponds to the a.m. designation, the last string to the p.m. designation.

t_fmt_ampm
Defines the string used for the standard 12-hour time format that includes an **am_pm** value (**%p** field descriptor). This statement corresponds to the **%r** field descriptor. The string can contain any combination of characters and field descriptors.

era Defines how the years are counted and displayed for each era in a locale, corresponding to the **%E** field descriptor modifier. For each era, there must be one string in the following format:

direction:*offset*:*start_date*:*end_date*:*name*:*format*

The variables for the era string format are defined as follows:

direction Specifies a **-** (minus) or **+** (plus) character. The **+** character indicates that years count in the positive direction when moving from the start date to the end date. The **-** character indicates that years count in the negative direction when moving from the start date to the end date.

offset Specifies a number representing the first year of the era.

start_date
Specifies the starting date of the era in *yyyy/mm/dd* format, where *yyyy*, *mm*, and *dd* are the year, month, and day, respectively on the Gregorian calendar. Years prior to the year AD 1 are represented as negative numbers. For example, an era beginning March 5th in the year 100 BC would be represented as **-100/03/05**.

end_date Specifies the ending date of the era in the same form used for the *start_date* variable or one of the two special values **-*** or **+***. A **-*** value indicates that the ending date of the era extends backward to the beginning of time. A **+*** value indicates that the ending date of the era extends forward to the end of time. Therefore, the ending date can be chronologically before or after the starting date of the era. For example, the strings for the Christian eras AD and BC would be entered as follows:

+:0:0000/01/01:+*:AD:%o %N
+:1:-0001/12/31:-*:BC:%o %N

name Specifies a string representing the name of the era that is substituted for the **%N** field descriptor.

format Specifies a **strftime( ) format string to use when formatting the %EY** field descriptor. This string can contain any **strftime( ) format control characters (except %EY**) and locale-dependent multibyte characters.

An era value consists of one string (enclosed in quotes) for each era. If more than one era is specified, each era string is separated by a **;** (semicolon).

era_year
Defines the string used to represent the year in alternate-era format corresponding to the **%Ey** field descriptor. The string can contain any combination of characters and field descriptors.

era_d_fmt
Defines the string used to represent the date in alternate-era format corresponding to the **%Ex** field descriptor. The string can contain any combination of characters and field descriptors.

era_t_fmt
Defines the locale's alternative time format, as represented by the **%EX** field descriptor for **strftime()**.

era_d_t_fmt
Defines the locale's alternative date and time format, as represented by the **%Ec** field descriptor for **strftime()**.

alt_digits
Defines alternate strings for digits corresponding to the **%O** field descriptor. Recognized values consist of a group of semicolon-separated strings. The first string represents the alternate string for 0 (zero), the second string represents the alternate string for 1, and so on. A maximum of 100 alternate strings can be specified.

m_d_recent
Defines the string used to print out month/date/time format for some commands (**ls**, **find**, **who**, **ar**). This format corresponds to the **"%b %e %H:%M"** format for the POSIX locale. (Optional)

m_d_old Defines the string used to print out month/date/year format for some commands (**ls**, **find**, **who**, **ar**). This format corresponds to the **"%b %e %Y"** format for the POSIX locale. (Optional)

The **LC_TIME** locale definition source file uses field descriptors to represent elements of time and date formats. Combinations of these field descriptors create other field descriptors or create time and date format strings. When used in format strings containing field descriptors and other characters, field descriptors are replaced by their current values. All other characters are copied without change. The following field descriptors are used by commands and subroutines that query the **LC_TIME** category for time formatting:

%a Represents the abbreviated weekday name (for example, **Sun**) defined by the **abday** statement.

%A Represents the full weekday name (for example, **Sunday**) defined by the **day** statement.

%b Represents the abbreviated month name (for example, **Jan**) defined by the **abmon** statement.

%B Represents the full month name (for example, **January**) defined by the **month** statement.

%c Represents the date and time format defined by the **d_t_fmt** statement.

%C Represents the century as a decimal number (**00** to **99**).

%d Represents the day of the month as a decimal number (**01** to **31**).

%D Represents the date in **%m/%d/%y** format (for example, **01/31/91**).

%e Represents the day of the month as a decimal number (**1** to **31**). The **%e** field descriptor uses a 2-digit field. If the day of the month is not a 2-digit number, the leading digit is filled with a space character.

%Ec Specifies the locale's alternate appropriate date and time representation.

%EC Specifies the name of the base year (period) in the locale's alternate representation.

%Ex Specifies the locale's alternate date representation.

%Ey Specifies the offset from **%EC** (year only) in the locale's alternate representation.

%EY Specifies the full alternate year representation.

%h Represents the abbreviated month name (for example, **Jan**) defined by the **abmon** statement. This field descriptor is a synonym for the **%b** field descriptor.

%H Represents the 24-hour clock hour as a decimal number (**00** to **23**).

%I Represents the 12-hour clock hour as a decimal number (**01** to **12**).

%j Represents the day of the year as a decimal number (**001** to **366**).

%m Represents the month of the year as a decimal number (**01** to **12**).

%M Represents the minutes of the hour as a decimal number (**00** to **59**).

%n Specifies a newline character.

%N Represents the alternate era name.

%o Represents the alternate era year.

%Od Specifies the day of the month using the locale's alternate numeric symbols.

%Oe Specifies the day of the month using the locale's alternate numeric symbols.

%OH Specifies the hour (24-hour clock) using the locale's alternate numeric symbols.

%OI Specifies the hour (12-hour clock) using the locale's alternate numeric symbols.

%Om Specifies the month using the locale's alternate numeric symbols.

%OM Specifies the minutes using the locale's alternate numeric symbols.

%OS Specifies the seconds using the locale's alternate numeric symbols.

%OU Specifies the week number of the year (Sunday as the first day of the week) using the locale's alternate numeric symbols.

%Ow Specifies the weekday as a number in the locale's alternate representation (Sunday = 0).

%OW Specifies the week number of the year (Monday as the first day of the week) using the locale's alternate numeric symbols.

%Oy Specifies the year (offset from **%C**) in alternate representation.

%p Represents the a.m. or p.m. string defined by the **am_pm** statement.

%r Represents the 12-hour clock time with a.m./p.m. notation as defined by the **t_fmt_ampm** statement.

%S Represents the seconds of the minute as a decimal number (**00** to **59**).

%t Specifies a tab character.

%T Represents 24-hour clock time in the format **%H:%M:%S** (for example, **16:55:15**).

%U Represents the week of the year as a decimal number (**00** to **53**). Sunday, or its equivalent as defined by the **day** statement, is considered the first day of the week for calculating the value of this field descriptor.

%w Represents the day of the week as a decimal number (**0** to **6**). Sunday, or its equivalent as defined by the **day** statement, is considered as **0** (zero) for calculating the value of this field descriptor.

%W Represents the week of the year as a decimal number (**00** to **53**). Monday, or its equivalent as defined by the **day** statement, is considered the first day of the week for calculating the value of this field descriptor.

%x Represents the date format defined by the **d_fmt** statement.

%X Represents the time format defined by the **t_fmt** statement.

%y Represents the year of the century (**00** to **99**).

%Y Represents the year as a decimal number (for example, **1989**).

%Z Represents the time zone name, if one can be determined (for example, **EST**); no characters are displayed if a time zone cannot be determined.

%% Specifies a **%** (percent sign) character.

The following is an example of a possible **LC_TIME** category listed in a locale definition source file:

```
LC_TIME
#
#Abbreviated weekday names (%a)
abday   "<S><u><n>";"<M><o><n>";"<T><u><e>";"<W><e><d>";\
        "<T><h><u>";"<F><r><i>";"<S><a><t>"

#Full weekday names (%A)
day     "<S><u><n><d><a><y>";"<M><o><n><d><a><y>";\
        "<T><u><e><s><d><a><y>";"<W><e><d><n><e><s><d><a><y>";\
        <T><h><u><r><s><d><a><y>";"<F><r><i><d><a><y>";\
        <S><a><t><u><r><d><a><y>"

#Abbreviated month names (%b)
abmon   "<J><a><n>";"<F><e><b>";"<M><a><r>";"<A><p><r>";\
        "<M><a><y>";"<J><u><n>";"<J><u><l>";"<A><u><g>";\
        <S><e><p>";"<O><c><t>";"<N><o><v>";"<D><e><c>"
```

```
#Full month names (%B)
mon     "<J><a><n><u><a><r><y>";"<F><e><b><r><u><a><r><y>";\
        "<M><a><r><c><h>";"<A><p><r><i><l>";"<M><a><y>";\
        <J><u><n><e>";"<J><u><l><y>";"<A><u><g><u><s><t>";\
        "<S><e><p><t><e><m><b><e><r>";"<O><c><t><o><b><e><r>";\
        <N><o><v><e><m><b><e><r>";"<D><e><c><e><m><b><e><r>"

#Date and time format (%c)
#Note that for improved readability, this section uses actual
#characters, rather than symbolic names, and is inconsistent with
#the other sections in this example.  This is bad form.
#In practice, symbolic names should be used.
d_t_fmt         "%a %b %d %H:%M:%S %Y"
#
#Date format (%x)
d_fmt           "%m/%d/%y"
#
#Time format (%X)
t_fmt           "%H:%M:%S"
#
#Equivalent of AM/PM (%p)
am_pm           "<A><M>";"<P><M>"
#
#12-hour time format (%r)
#Note that for improved readability, this section uses actual
#characters, rather than symbolic names, and is inconsistent with
#the other sections in this example.  This is bad form.
#In practice, symbolic names should be used.
t_fmt_ampm      "%I:%M:%S %p"
#
era             "+:0:0000/01/01:+*:AD:%o %N";\
                "+:1:-0001/12/31:-*:BC:%o %N"
  era_year      ""
  era_d_fmt     ""
  alt_digits    "<0><t><h>";"<1><s><t>";"<2><n><d>";"<3><r><d>";\
                "<4><t><h>";"<5><t><h>";"<6><t><h>";"<7><t><h>";\
                "<8><t><h>";"<9><t><h>";"<1><0><t><h>"
#
END LC_TIME
```

FILES

/usr/lib/nls/loc/src/* Locale definition source files for supported locales.

/usr/lib/nls/loc/charmap/*
Character set description (**charmap**) source files for supported locales.

RELATED INFORMATION

Commands: **locale(1)**, **localedef(1)**.

Files: **charmap(4)**.

NAME

netrc - file for ftp remote login data

DESCRIPTION

The **.netrc** file contains data for logging in to a remote host over the network for file transfers by **ftp(1)**. This file resides in the user's home directory on the machine initiating the file transfer. Its permissions should be set to disallow read access by group and others (see the **chmod(1)** reference page).

The following tokens are recognized; they may be separated by SPACE, TAB, or NEWLINE characters:

machine*name*

Identify a remote machine name. The auto-login process searches the **.netrc** file for a **machine** token that matches the remote machine specified on the **ftp** command line or as an **open** command argument. Once a match is made, the subsequent **.netrc** tokens are processed, stopping when the EOF is reached or another **machine** token is encountered.

login *name*

Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified name.

password *string*

Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note: if this token is present in the **.netrc** file, **ftp** will abort the auto-login process if the **.netrc** is readable by anyone besides the user.

account *string*

Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an command if it does not.

macdef *name*

Define a macro. This token functions as the **ftp macdef** command functions. A macro is defined with the specified name; its contents begin with the next **.netrc** line and continue until a null line (consecutive NEWLINE characters) is encountered. If a macro named **init** is defined, it is automatically executed as the last step in the auto-login process.

EXAMPLE

The command:

machine ray login demo password mypassword

allows an autologin to the machine **ray** using the login name **demo** with password **mypassword**.

FILES

~/.netrc

RELATED INFORMATION

Commands: **chmod(1), ftp(1), ftpserver(7).**

NAME

networks - Contains network name information

DESCRIPTION

The **/etc/networks** file contains information about the known networks that constitute the DARPA (Defense Advanced Research Projects Agency) Internet. Each network is represented by a single line in the **networks** file. The format for the entries in the **networks** file is as follows:

name number aliases

The fields contain the following:

name The official network name.

number The network number.

aliases The unofficial names used for the network.

Items on a line are separated by one or more spaces or tab characters. Comments begin with a # (number sign). Routines that search the **networks** file do not interpret characters from the beginning of a comment to the end of that line. Network numbers are specified in dotted-decimal notation. A network name can contain any printable character except a field delimiter, newline character, or comment character (#).

The **networks** file is normally created from the official network database maintained at the Network Information Center (NIC). The file may need to be modified locally to include unofficial aliases or unknown networks.

EXAMPLES

Example lines from a **networks** file are shown below:

```
loopback        127
arpanet         10      arpa
```

FILES

/etc/networks Specifies the pathname of the file.

RELATED INFORMATION

Commands: **routed(8)**

Functions: **getnetbyaddr(3)**, **getnetbyname(3)**, **getnetent(3)**.

Files: **hosts(4)**, **protocols(4)**.

NAME

.proto - Defines the environment for a job to be processed by an **at** or **batch** command

SYNOPSIS

/var/adm/cron/.proto

DESCRIPTION

This file contains a set of shell commands that are added to the end of each **at** or **batch** job file to create an environment for the job. This set of commands is defined by the site; a prototype is provided by Compaq.

When a job is submitted to an **at** or **batch** queue, the job is constructed as a shell script. The **at** or **batch** command places the job file in **/var/spool/cron/atjobs** and the following steps occur:

1. The **at** or **batch** process adds a header describing the job:

   : at job Defines an **at** job

   : batch job Defines a **batch** job

   Jobs submitted to any queue other than queue **a** are always given a batch job header.

2. The process adds a set of shell commands to create an environment for the job identical to the current shell environment.

3. The process then copies text from the **/var/adm/cron/.proto** file into the job file. The following special variables can be used in the **/var/adm/cron/.proto** file to help define or modify the shell environment for the job:

   $d Replace with the pathname of the current working directory.

   $l Replace with the current file size limit.

   $m Replace with the current value of **umask**.

   $t Replace with the time at which the job should be run, expressed as seconds since the Epoch of Coordinated Universal Time (UTC) and prefixed by a colon (**:**).

   $< Replace with text read by the process from the standard input file (that is, with the commands to be run in the job).

When this job is dispatched by **cron** for execution, a new shell is started to execute this script.

EXAMPLES

When the following **at** command is given, a job file is created in **/var/spool/cron/atjobs** with the name **TCS.ALI.1008378001.a**:

```
Whit10:/home/ali: at -f /home/arindam/test.sh 5 pm Friday
job TCS.ALI.1008378001.a at Fri Dec 14 17:00:00 2001
Whit10:/home/ali:
```

The file **/var/adm/cron/.proto** contains attribution comments and the following three shell commands:

```
cd $d
ulimit $l
umask $m
```

The file **TCS.ALI.1008378001.a** contains:

```
: at job
```

```
export CDPATH;
  .
  .
  .
TERM='dumb';
export TERM;
  .
  .
  .
export PATH;
  .
  .
  .
export HOME;
#
# COPYRIGHT NOTICE
#
  .
  .
  .
#

cd /home/ali
ulimit 4194303
umask 0
/bin/sh << 'QAZWSXEDCRFVTGBYHNUJMIKOLP'
/home/arindam /test.sh
```

The prototype file (**.proto**) contents begin with the **#** characters and have been appended after the environment variables are set. The **/bin/sh** command and the **/home/arindam /test.sh** command have been appended to the end of the job file from the standard input file.

RELATED INFORMATION

Commands: **at(1)**, **batch(1)**.

STANDARDS CONFORMANCE

This file is an extension to the XPG4 Version 2 specification.

NAME

protocols - Defines the Internet protocols used on the local host

DESCRIPTION

The **/etc/protocols** file contains information about the known protocols used in the DARPA (Defense Advanced Research Projects Agency) Internet. Each protocol is represented by a single line in the **protocols** file. Each entry is of the following form:

name number aliases

The fields contain the following information:

name Official Internet protocol name.

number Protocol number.

aliases Unofficial names used for the protocol.

Items on a line are separated by one or more spaces or tab characters. Comments begin with the **#** (number sign), and routines that search the **protocols** file do not interpret characters from the beginning of a comment to the end of the line. A protocol name can contain any printable character except a field delimiter, newline character, or comment character (**#**).

EXAMPLES

Example lines from a **protocols** file are shown below:

```
#
# Internet (IP) protocols
#       @(#)protocols   5.1 (Berkeley) 4/17/92
#
ip      0    IP         # internet protocol, pseudo protocol number
icmp    1    ICMP       # internet control message protocol
ggp     3    GGP        # gateway-gateway protocol
tcp     6    TCP        # transmission control protocol
egp     8    EGP        # exterior gateway protocol
pup     12   PUP        # PARC universal packet protocol
udp     17   UDP        # user datagram protocol
hmp     20   HMP        # host monitoring protocol
xns-idp 22   XNS-IDP    # Xerox NS IDP
rdp     27   RDP        # "reliable datagram" protocol
```

FILES

/etc/protocols Specifies the pathname of the file.

RELATED INFORMATION

Functions: **endprotoent(3)**, **getprotobyname(3)**, **getprotobynumber(3)**, **getprotoent(3)**, **setprotoent(3)**.

NAME

queuedefs - Describes queues for the **at**, **batch**, and **cron** commands

DESCRIPTION

The **queuedefs** file describes the characteristics of the queues managed by the **cron** demon. Each noncomment line in this file describes one queue.

The format of a line is as follows:

*q***.**[*njob***j**][*nice***n**][*nwait***w**]

The fields in this line are:

q The name of the queue:

- **a** is the default queue name for jobs started by **at**
- **b** is the default queue name for jobs started by **batch**
- **c** is the default queue name for jobs run from a **crontab** file

njob The maximum number of jobs that can be run simultaneously in that queue. If more than *njob* jobs are ready to run, only the first *njob* jobs will be run; the others will be run when currently running jobs terminate. The default value is **100**.

nice The **nice( )** function value to give to all jobs in that queue that are not run with a user ID that has appropriate privileges. The default value is **2**.

nwait The number of seconds to wait before rescheduling a job that was deferred either because more than *njob* jobs were running in that job's queue or because the systemwide limit for the number of jobs executing has been reached. The default value is **60**.

Lines beginning with **#** are comments and are ignored.

EXAMPLES

```
#
#
a.4j1n
b.2j2n90w
```

This file specifies that the **a** queue, for **at** command jobs, can have up to 4 jobs running simultaneously; those jobs will run with a **nice** value of 1. Because no *nwait* value was given, if a job cannot be run because too many other jobs are running, **cron** waits 60 seconds before trying again to run it.

The **b** queue, for **batch** command jobs, can have up to 2 jobs running simultaneously; those jobs will run with a **nice** value of 2. If a job cannot be run because too many other jobs are running, **cron** waits 90 seconds before trying again to run it.

All other queues can have up to 100 jobs running simultaneously; they are run with a **nice** value of 2, and if a job cannot be run because too many other jobs are running, **cron** waits 60 seconds before trying again to run it.

FILES

/var/adm/cron/queuedefs

The queue description file for **at**, **batch**, and **cron**.

RELATED INFORMATION

Commands: **crontab(1)**, **cron(8)**.

Functions: **nice(2)**.

NAME

resolv.conf - Describes BIND 4 Domain Name System resolver configuration file

DESCRIPTION

The configuration file **/etc/resolv.conf** provides an explicit default domain name for the Domain Name System (DNS) to use, and identifies name servers on other processors. Each entry in the file is a directive that consists of a keyword followed by one or more values:

keyword *value*

The **/etc/resolv.conf** file can contain the following directives:

nameserver *address*
: The Internet address of a name server, in standard dot notation. Multiple name server addresses may be listed. The resolver queries the name servers in the order they are listed in the file, stopping when it receives a response, or moving to the next in the list if the query times out. If the resolver reaches the end of the name server list without receiving a response, it will start from the beginning of the list and query each name server again, until a maximum number of retries is reached. If **/etc/resolv.conf** contains no nameserver directives, the resolver uses the loopback address. Therefore, a name server must be running on the processor on which the file resides.

domain *name*
: The default domain to append to names that do not contain a domain, and the default domain name to be used in searches. No trailing spaces are allowed after the value in *name*.

  If **/etc/resolv.conf** does not contain a domain directive, then the resolver uses the the hostname for the processor, but removes the first part of the name. For example, if the host name is set to "`yojimbo.dev1.anyfirm.com`," the resolver uses the name "`dev1.anyfirm.com`."

search *name name ...*
: The explicit search order that you want the resolver to use. The **search** keyword can accept up to six domain names as values.

  The resolver will perform its search using the order specified after the **search** keyword.

Fields are separated by one or more spaces or tab characters. If **/etc/resolv.conf** contains a search and a domain directive, the resolver will use whichever directive comes first in the file.

EXAMPLES

Example lines from a **/etc/resolv.conf** file are shown below:

```
domain          dev1.anyfirm.com
nameserver      123.456.78.90
nameserver      123.456.78.91
```

RELATED INFORMATION

Files: **hosts(4)**, **networks(4)**, **protocols(4)**, **resolv.conf(5)**, **services(4)**.

Commands: **gethostbyaddr(3)**, **gethostbyname(3) setnetent(3).**

NAME

.rhosts - Describes individual user files for trusted remote hosts and users

SYNOPSIS

~/.rhosts

DESCRIPTION

The **/etc/hosts.equiv** and **.rhosts** files provide the "remote authentication" database for the **rsh** command. The files specify remote hosts and users that are considered trusted. Only trusted users from the remote hosts are allowed to access the local system. The **/etc/hosts.equiv** file applies to the entire system, while individual users can maintain their own **.rhosts** files in their home directories. To maintain system security, these files must be carefully created and maintained. The remote authentication procedure determines whether a user from a remote host should be allowed to access the local system with the identity of a local user. This procedure first checks the **/etc/hosts.equiv** file and then checks the **.rhosts** file in the home directory of the local user who is requesting access. The **rsh** command fails if the remote authentication procedure fails.

Each entry in the file has the form:

hostname [*username*]

If the entry is in the form:

hostname

then all users from the named host are trusted. That is, they are allowed to access the system with the same user name as they have on the remote system. This form of entry can be used in both the **/etc/hosts.equiv** and **.rhosts** files.

If the entry is in the form:

hostname username

then the named user from the named host is allowed to access the system. This form can be used in individual **.rhosts** files to allow remote users to access the system as a different local user. If this form is used in the **/etc/hosts.equiv** file, the named remote user is allowed to access the system with the same capabilities as any local user.

FILES

/etc/hosts.equiv

~/.rhosts

NOTES

A *hostname* value used in the **/etc/hosts.equiv** and **.rhosts** files must be the official name of the host, not one of its nicknames.

Super ID access is handled as a special case. Only the **.rhosts** file is checked when access is attempted for the super ID. To help maintain system security, the **/etc/hosts.equiv** file is not checked.

RELATED INFORMATION

Commands: **rsh(1)**.

Files: **hosts(4)**, **hosts.equiv(4)**.

STANDARDS CONFORMANCE

This file is an extension to the XPG4 Version 2 specification.

NAME

services - Contains information about Internet services

DESCRIPTION

The **/etc/services** file contains information about services available through the Internet. A service entry consists of a service name followed by a port number and protocol, and it can optionally contain aliases for the service. Each entry takes the following form:

service_name *port/protocol* *aliases*

The fields contain the following information:

service_name	The official name for the service.
port/protocol	The port number and protocol through which the service is provided, for example, 513/TCP.
aliases	These are alternate names that can be used to request the service.

Fields are separated by one or more space or tab characters. Comments begin with the number sign (#). Routines that search the **/etc/services** file do not interpret characters from the beginning of a comment to the end of the line. A host name can contain any printable character except a field delimiter, newline character, or comment character (#).

EXAMPLES

Example lines from a **services** file are shown below:

```
echo            7/udp
smtp            25/tcp          mail
```

RELATED INFORMATION

Commands: **endservent(3)**, **getservbyname(3)**, **setservent(3)**.

Files: **hosts(4)**, **networks(4)**, **protocols(4)**.

Section 12. Administrator Commands and Files

This section contains reference pages for miscellaneous OSS files from the **cat7** directory and administrator command reference pages from the **cat8** directory. See the *Open System Services System Calls Reference Manual* for other reference pages you might expect to find in this section.

NAME

copyoss - Copies the contents of **pax** archive files from the Guardian environment to the OSS file system

SYNOPSIS

[**gtacl -c** ´] [**$***tsv***.***tsvsvl.*]**copyoss**
[**?** | [*filename* | *subvolname*] ...]
[´]

gtacl -c ´ is a required prefix of this command for users of the OSS shell, but this prefix must be omitted by users of the HP Tandem Advanced Command Language (TACL) command interpreter. Users of the OSS shell must end the command line with an apostrophe (single quotation mark) (´), but this apostrophe must be omitted by users of TACL.

FLAGS

The **copyoss** command has the following operands:

$*tsv***.***tsvsvl* Specifies the disk volume and subvolume where the OSS product installation files are located.

At many sites, COPYOSS is copied to $SYSTEM.ZOSSUTL.

? Requests the display of help text for the command.

This is the default used if no value is specified for *filename* or *subvol*.

filename Specifies the Guardian filename of the **pax** archive file whose content is to be copied to the OSS file system. The specified file must have a file code of 0 or 180.

When **copyoss** is used from the OSS shell, *filename* should be fully qualified.

To avoid unexpected side effects, *filename* should be fully qualified when COPYOSS is used from any subvolume other than ZOSSUTL. If you enter a value for a Guardian file identifier that is not recognized as valid for a file in the current subvolume, COPYOSS will try to interpret the value as a subvolume name.

subvolname Specifies the Guardian subvolume name of the Guardian subvolume whose content is to be copied to the OSS file system. When *subvolname* is specified, **copyoss** copies the content of all **pax** archives within the specified subvolume unless their file identifiers begin with the reserved letters ZFB or ZPG.

When **copyoss** is used from the OSS shell, *subvolname* should be fully qualified.

DESCRIPTION

The **copyoss** command invokes a TACL routine that calls the Guardian **pinstall** utility. The **copyoss** utility is present on your system only if, when you last installed the OSS product set, you retained the target subvolume from the installation or copied the COPYOSS file to a known location such as the $SYSTEM disk. In contrast, the **pinstall** utility is installed on the $SYS*nn* disk and is always present on your system.

copyoss attempts to process only files with file codes of 0 or 180; other files within the subvolume are ignored.

copyoss processes files within a subvolume in alphabetic order. If versions of archive files are maintained within *subvolname* by the Distributed Systems Management/Software Configuration Manager (DSM/SCM) product with Guardian file identifiers that begin with ZFB or ZPG, those files are ignored by **copyoss**.

If you specify more than one operand for COPYOSS, you can mix subvolume names and filenames.

EXAMPLES

1. To copy all the files in the **pax** archives distributed with the OSS product to the OSS file system, enter the following two commands at TACL prompts:

   ```
   VOLUME $SYSTEM.ZOSSUTL
   RUN COPYOSS ZOSSUTL
   ```

   where $SYSTEM.ZOSSUTL is the name of the target subvolume from an installation of the OSS product set.

2. To copy only the Java servlet files from the NonStop Java Server product **pax** archive to the OSS file system, enter the following at a TACL prompt:

   ```
   RUN $tsv1.ZOSSUTL.COPYOSS $tsv2.subvol.T0094PAX
   ```

   where $*tsv1* is the name of the disk volume where the most recent OSS product installation files are kept, $*tsv2.subvol* identifies the disk volume and target subvolume containing the installation files from the NonStop Java Server product, and T0094PAX is the Guardian file identifier of the specific **pax** archive file containing the Java servlet files.

3. To copy only the contents of the two **pax** archives T8626MAN and T8629MNN to the OSS file system when ZOSSUTL contains a copy of COPYOSS, enter either of the following at TACL prompts:

   ```
   VOLUME $SYSTEM.ZOSSUTL
   RUN COPYOSS T8626MAN T8629MNN
   ```

   or:

   ```
   RUN $SYSTEM.ZOSSUTL.COPYOSS $SYSTEM.ZOSSUTL.T8626MAN $SYSTEM.ZOSSUTL.T8629MNN
   ```

4. To copy all of the current OSS product archive content and the Java **pax** archive content to the OSS file system in one operation, enter the following at TACL prompts:

   ```
   VOLUME $SYSTEM.ZOSSUTL
   RUN COPYOSS ZOSSUTL $tsv2.subvol.T0094PAX
   ```

   where the ZOSSUTL subvolume contains multiple product **pax** archives, $*tsv2.subvol* identifies the disk volume and target subvolume containing the installation files from the NonStop Java Server product, and T0094PAX is the Guardian file identifier of the specific **pax** archive file containing the Java servlet files.

FILES

$*tsv*.*tsvsvl*.COPYOSS
: Contains the COPYOSS routine.

NOTES

On systems where DSM/SCM is used to install HP product files from the ZOSSUTL subvolume and maintain those files in the OSS file system, do not use **copyoss** to install HP product files from the ZOSSUTL subvolume.

On systems where DSM/SCM is not used to install HP product files from the ZOSSUTL subvolume and maintain those files in the OSS file system, do not use **copyoss** on files with Guardian file identifiers that begin with ZFB or ZPG.

RELATED INFORMATION

Commands: **gtacl(1)**, **pax(1)**, **pcleanup(8)**, **pinstall(1)**.

Files: **tar(4)**.

STANDARDS CONFORMANCE

The **copyoss** command is an extension to the XPG4 Version 2 specification.

NAME

cron - Runs the system clock daemon

SYNOPSIS

cron [**-f**]

FLAGS

-f Run the **cron** process in the foreground.

This flag is valid for systems running:

- J06.05 and later J-series RVUs
- H06.16 and later H-series RVUs
- G06.33 and later G-series RVUs
- J06.03, J06.04, or H06.03 through H06.15 RVUs and have installed SPR T8626H03^ACU
- G06.29 through G06.32 RVUs and have installed SPR T8626G07^ACV

DESCRIPTION

The **cron** daemon runs shell commands at specified dates and times. Because the **cron** process exits only when killed or when the system stops, only one **cron** daemon should exist on the system at any given time.

In systems that support the **-f** flag, you can start the **cron** process as a persistent generic process. Add the **cron** process to $ZZKRN and associate it with the OSH process. Configure the **cron** process as follows:

- Set the STARTUPMSG attribute to:

  "**-osstty -name /G/***process_name* -p /bin/cron -f"

  where *process_name* is a Guardian process name of the persistent **cron** process.

- Set the ASSOCPROC attribute to the same name as you used for the persistent **cron** process in the STARTUPMSG attribute.
- The attibutes HOMETERM, INFILE, and OUTFILE are required.

For information about the rules that apply to the Guardian process name, see "Environment Variables." For an example of adding the **cron** process as a persistent process, see "Examples." For information about configuring a generic process, see the *SCF Reference Manual for the Kernel Subsystem*.

Commands that are to run according to a regular or periodic schedule are found within the **crontab** files. Commands that are to run only once are found within the **at** files. You submit **crontab** and **at** file entries by using the **crontab** and **at** commands.

During process initialization and when **cron** detects a change, it examines the **crontab** and **at** files. This strategy reduces the overhead of checking for new or changed files at regularly scheduled intervals.

The **cron** command creates a log of its activities as a file named **log** in the directory **/var/adm/cron**. When the log file size exceeds 2.5 MB (2,621,440 bytes), the log file is closed and renamed to save its information. A new log file named **log** is created in the same directory. The renamed log file's filename has the format

log_*yyyymmddhhnnss*

where the variable information denotes the year, month, day, hour, minute, and second at which the file was renamed.

The **cron** daemon starts each job with the following process attributes stored with the job by the invoking process:

- Effective and real user IDs
- Effective and real group IDs
- Supplementary groups

Environment Variables

cron runs as a named process if the **CRON_NAMED** environment variable is set and exported before the **cron** command is entered. The environment variable has the form:

CRON_NAMED=/G/*process_name*

where *process_name* conforms to the rules for Guardian process names described in the **pathname(5)** reference page, available either online or in the *Open System Services System Calls Reference Manual*. The following additional rules apply:

- The process name must be specified in OSS pathname format, so the **$** is omitted
- The first character cannot be a Z

When **CRON_NAMED** is not defined or is not exported, **cron** starts as an unnamed process and nothing prevents the undesirable situation of accidentally running multiple copies.

EXAMPLES

1. These SCF commands configure and start the **cron** process as a persistent process:

```
-> ASSUME $ZZKRN
-> ADD PROC #cro,                                          &
     NAME $cro,                                            &
     CPU FIRST,                                            &
     ASSOCPROC $cro1,                                      &
     PRIORITY 100,                                         &
     STARTUPMSG "-osstty -name /G/cro1 -p /bin/cron -f",&
     PROGRAM $system.system.osh,                           &
     HOMETERM $ZHOME,                                      &
     INFILE $NULL,                                         &
     OUTFILE $ZHOME,                                       &
     STARTMODE manual,                                     &
     AUTORESTART 3
-> START PROC #cro
```

2. This set of commands starts **cron** as the named process $CRON if another copy of **cron** is not already running with the same name:

```
export CRON_NAMED=/G/CRON
cron
```

FILES

/bin/cron **cron** daemon code file.

/var/adm/cron/queuedefs
queue description file for **at**, **batch**, and **cron**.

/var/adm/cron/log
Most recent **cron** history information.

NOTES

Only one copy of **cron** should be running at a given time. HP recommends starting **cron** as a named process and always using the same name for that process to ensure this situation.

RELATED INFORMATION

Commands: **crontab(1)**.

Files: **queuedefs(4)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 Version 2 specification.

NAME

dig - BIND 9 Domain Name System (DNS) server lookup utility

SYNOPSIS

/etc/dns_secure/dig [*global_queryopt* ...]
[**@***server*]
[**-b** *address*[**#***port*]]
[**-c** *class*]
[**-f** *filename1*]
[**-h**]
[**-k** *filename2*]
[**-p** *port#*]
[**-t** *type*]
[**-x** *addr*]
[**-y** *name***:***key*]
[**-4** | **-6**]
[*name*]
[*type*]
[*class*]
[*queryopt* ...]

FLAGS

When no command line flags, arguments, or options are given, **dig** performs an **NS** query for the **.** (root) server.

-b *address*[**#***port*]
Sets the source IP address of the query to *address*. *address* must be a valid address on one of the host's network interfaces or **0.0.0.0** or **::**. An optional port can be specified by appending **#***port*.

-c *class*
Overrides the default query class (**IN** for Internet). *class* is a mnemonic for any valid class, such as **HS** for Hesiod records or **CH** for CHAOSNET records.

-f *filename1*
Makes **dig** operate in batch mode by reading a list of lookup requests to process from the file *filename1*. The file contains one or more queries, one per line. Each entry in the file should be organized in the same way it would be presented as a query to **dig** using the command-line interface.

-h
Prints a brief description of command-line flags and options.

-k *filename2*
Signs the DNS queries sent by **dig** and their responses using transaction signatures (TSIG) in the TSIG key file *filename2*.

When using TSIG authentication with **dig**, the name server that is queried needs to know the key and algorithm that is being used. In BIND 9, this is done by providing appropriate key and server statements in the **named.conf** file.

-p *port#*
Queries a nonstandard port number. *port#* is the port number that **dig** sends its queries to instead of the standard DNS port number 53.

This option is used to test a name server that has been configured to listen for queries on a nonstandard port number.

-t *type*
Sets the query type to *type*. *type* can be any valid query type which is supported in BIND 9. The default query type is **A**, unless the **-x** flag is supplied to indicate a reverse lookup.

A zone transfer can be requested by specifying a *type* of **AXFR**.

When an incremental zone transfer (**IXFR**) is required, set *type* to **ixfr=N**. The incremental zone transfer contains the changes made to the zone since the serial number in the zone's **SOA** record was **N**.

-x *addr* Simplifies reverse lookups (mapping addresses to names). *addr* is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address.

When this flag is used, you do not need to provide the *name*, *class*, and *type* arguments. **dig** automatically performs a lookup for a name like **11.12.13.10.in-addr.arpa** and sets the query type and class to **PTR** and **IN** respectively. By default, IPv6 addresses are looked up using nibble format under the **IP6.ARPA** domain. To use the older RFC1886 method using the **IP6.INT** domain, specify the **-i** flag.

Bit string labels (RFC2874) are experimental and are not attempted.

-y *name***:***key* Signs DNS queries sent by **dig** and their responses using transaction signatures (TSIG), where *name* is the name of the TSIG key and *key* is the actual key. The key is a base-64 encoded string, typically generated by the **dnssec-keygen** utility. Be cautious when using the **-y** flag on multi-user systems because the key can be visible in the output from the **ps** command or in the shell's history file.

When using TSIG authentication with **dig**, the name server that is queried needs to know the key and algorithm that is being used. In BIND 9, this is done by providing appropriate key and server statements in the **named.conf** file.

You can also sign the DNS queries sent by **dig** and their responses using transaction signatures (TSIG) byspecifing a TSIG key file using the **-k** flag.

-4 | **-6** The **-4** flag forces **dig** to only use IPv4 query transport. The **-6** flag forces dig to only use IPv6 query transport.

Arguments

global_queryopt and *queryopt*

dig provides query options that affect the way in which lookups are made and the results displayed. Some of these set or reset flag bits in the query header, some determine which sections of the answer get printed, and others determine the timeout and retry strategies. Each query option is identified by a keyword preceded by a plus sign (+).

Some keywords set or reset an option; these can be preceded by the string **no** to negate the meaning of that keyword.

Other keywords assign values to options like the timeout interval; these have the form *+keyword=value*.

The query options are:

+[**no**]**tcp** Use [do not use] TCP when querying name servers. The default behavior is to use UDP unless an **AXFR** or **IXFR** query is requested, in which case a TCP connection is used.

+[**no**]**vc** Use [do not use] TCP when querying name servers. This alternate syntax to +[**no**]**tcp** provides backwards compatibility. **vc** stands for "virtual circuit".

+[no]ignore Ignore [do not ignore] truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

+domain=*somename*
Set the search list to contain the single domain *somename*, as if specified in a **domain** directive in the **/etc/resolv.conf** file, and enable search list processing as if the **+search** option were given.

+[no]search Use [do not use] the search list defined by the **searchlist** or **domain** directive in the **/etc/resolv.conf** file (if any). The search list is not used by default.

+[no]defname Deprecated; treated as a synonym for **+[no]search**.

+[no]aaonly | **+[no]aaflag**
Set [do not set] the **aa** flag in the query.

+[no]adflag Set [do not set] the **AD** (authentic data) bit in the query. The **AD** bit currently has a standard meaning only in responses, not in queries, but the ability to set the bit in the query is provided for completeness.

+[no]cdflag Set [do not set] the **CD** (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

+[no]cl Display [do not display] the class when printing the record.

+[no]ttlid Display [do not display] the TTL when printing the record.

+[no]recurse Set [do not set] the **RD** (recursion desired) bit in the query. This bit is set by default, which means **dig** normally sends recursive queries. Recursion is automatically disabled when the **+nssearch** or **+trace** query options are used.

+[no]nssearch Search for [do not search for] authoritative name servers for the zone containing the name being looked up and display the **SOA** record that each name server has for the zone.

+[no]trace Set [do not set] tracing of the delegation path from the root name servers for the name being looked up. Tracing is disabled by default.

When tracing is enabled, **dig** makes iterative queries to resolve the name being looked up. **dig** follows referrals from the root servers, showing the answer from each server that was used to resolve the lookup.

+[no]cmd Set [do not set] printing of the initial comment in the output, which identifies the version of **dig** and the query options that have been applied. This comment is printed by default.

+[no]short Provide [do not provide] a terse answer. The default is to print the answer in a verbose form.

+[no]identify Show [do not show] the IP address and port number that supplied the answer when the **+short** option is enabled. If short form answers are requested, the default is to not show the source address and port number of the server that provided the answer.

+[no]comments
Display [do not display] comment lines in the output. The default is to print comments.

+[no]stats Print [do not print] statistics such as: when the query was made, the size of the reply, and so on. The default behavior is to print the query statistics.

+[no]qr Print [do not print] the query as it is sent. By default, the query is not printed.

+[no]question Print [do not print] the question section of a query when an answer is returned. The default is to print the question section as a comment.

+[no]answer Display [do not display] the answer section of a reply. The default is to display it.

+[no]authority Display [do not display] the authority section of a reply. The default is to display it.

+[no]additional
Display [do not display] the additional section of a reply. The default is to display it.

+[no]all Set [or clear] all display flags.

+time=*t1* Set the timeout for a query to *t1* seconds. The default timeout is 5 seconds. An attempt to set *t1* to less than 1 results in a query timeout of 1 second being applied.

+tries=*t2* Set the number of times to try UDP queries to a server to *t2* instead of the default, 3. If *t2* is less than or equal to 0 (zero), the number of tries is silently rounded up to 1.

+retry=*t3* Set the number of times to retry UDP queries to a server to *t3* instead of the default, 2. Unlike **+tries**, this does not include the initial query.

+ndots=*d* Set the number of dots that have to appear in *name* to *d* for it to be considered an absolute name. The default value is that defined using the **ndots** statement in the **/etc/resolv.conf** file, or 1 if no **ndots** statement is present.

Names with fewer dots are interpreted as relative names and are searched for in the domains listed in the **search** or **domain** directive in the **/etc/resolv.conf** file.

+bufsize=*b* Set the UDP message buffer size advertised using **EDNS0** to *b* bytes. The maximum and minimum sizes of this buffer are 65535 and 0 respectively. Values outside this range are rounded up or down appropriately.

+[**no**]**multiline** Print [do not print] records like the **SOA** records in a verbose multiline format with human-readable comments. The default is to print each record on a single line, to facilitate machine parsing of the **dig** output.

+[**no**]**fail** Try [do not try] the next server if you receive a **SERVFAIL** response. The default is to not try the next server, which is the reverse of normal stub resolver behavior.

+[**no**]**besteffort** Display [do not display] the contents of messages which are malformed. The default is to not display malformed answers.

+[**no**]**dnssec** Request [do not request] that DNSSEC records be sent by setting the DNSSEC OK bit (**DO**) in the **OPT** record in the **additional** section of the query.

+[**no**]**sigchase** Chase [do not chase] DNSSEC signature chains. Requires that **dig** be compiled with **-DDIG_SIGCHASE**.

+**trusted-key**=*key* Specify a trusted key to be used with **+sigchase**. Requires that **dig** be compiled with **-DDIG_SIGCHASE**.

+[**no**]**topdown** Perform [do not perform] a top-down validation when chasing DNSSEC signature chains. Requires that **dig** be compiled with **-DDIG_SIGCHASE**.

name Specifies the relative or absolute name of the server to return information for.

type Specifies the type of information to be returned.

class Specifies the class of information to be returned.

DESCRIPTION

dig is a tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name servers that were queried. Most DNS administrators use **dig** to troubleshoot DNS problems because of its flexibility, ease of use, and clarity of output.

Although **dig** is normally used with command-line arguments, it also has a batch mode of operation for reading lookup requests from a file. A brief summary of its command-line arguments and options is printed when the **-h** option is given.

Unlike earlier versions, the BIND 9 implementation of **dig** allows multiple lookups to be issued from the command line. Unless it is told to query a specific name server, **dig** tries each of the servers listed in the **/etc/resolv.conf** file.

You can set per-user defaults for **dig** in the file **$HOME/.digrc**. This file is read and any options in it are applied before the command line values are processed.

Simple Usage

A typical invocation of **dig** looks like:

dig @*server name type*

where:

server is the name or IP address of the name server to query. This can be an IPv4 address in dotted-decimal notation or an IPv6 address in colon-delimited notation. When the supplied server argument is a hostname, **dig** resolves that name before querying that name server. If no server argument is provided, **dig** reads

/etc/resolv.conf and queries the name servers listed there. The reply from the name server that responds is displayed.

name is the name of the resource record that is to be looked up.

type indicates what type of query is required: **ANY**, **A**, **MX**, **SIG**, and so forth. *type* can be any valid query type. If no *type* argument is supplied, **dig** performs a lookup for an **A** record.

Multiple Queries

The BIND 9 implementation of **dig** supports specifying multiple queries on the command line (in addition to supporting the **-f** batch file option). Each of those queries can be supplied with its own set of flags, options, and query options. In this case, each query argument represents an individual query in the command-line syntax described above. Each consists of any of the standard options and flags, the name to be looked up, an optional query type and class, and any query options that should be applied to that query.

You can also supply a global set of query options, which should be applied to all queries. These global query options must precede the first set of name, class, type, options, flags, and query options supplied on the command line. Any global query options (except the **+**[**no**]**cmd** option) can be overridden by a query-specific set of query options.

EXAMPLES

The following example of a multiple query shows how **dig** can be used from the command line to make three lookups:

- An **ANY** query for **www.isc.org**
- A reverse lookup of **127.0.0.1**
- A query for the **NS** records of **isc.org**

dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr

A global query option of **+qr** is applied, so that **dig** shows the initial query it made for each lookup. The final query has a local query option of **+noqr**, which means that **dig** does not print the initial query when it looks up the **NS** records for **isc.org**.

FILES

/etc/named.conf

The default **named** server configuration file. This file defines the recognized values for *class* in its **zone** and **view** statements.

/etc/resolv.conf

The default domain name server resolver file.

$HOME/.digrc

The local file to override default values for **dig** use. This file can contain command line flags and query options, with one flag or query option specification per line.

RELATED INFORMATION

Commands: **dnssec_named(8)**, **named(8)**, **dnssec-keygen(8)**.

Files: **named.conf(4)**, **resolv.conf(5)**.

Documents: *BIND Administrator Reference Manual*, RFC1035.

NAME

dnssec-keygen - Runs the BIND 9 secure domain name server DNSSEC key generation tool

SYNOPSIS

/etc/dns_secure/dnssec-keygen
-a *algorithm*
-b *keysize*
-n *nametype*
[**-c** *class*]
[**-e**]
[**-f** *flag*]
[**-g** *generator*]
[**-h**]
[**-k**]
[**-p** *protocol*]
[**-r** *randomdev*]
[**-s** *strength*]
[**-t** *type*]
[**-v** *level*]
name

FLAGS

-a *algorithm* ... Selects the cryptographic algorithm to be used. The value of *algorithm* must be one or more of:

RSAMD5	Specifies RSA. This value is an alternative to **RSASHA1**.
RSASHA1	Specifies RSA. This value is required to implement a secure DNSSEC name server algorithm.
DSA	Specifies DSA. This value is recommended to implement a secure DNSSEC name server algorithm.
DH	Specifies Diffie Hellman. Using this value automatically sets the **-k** flag.
HMAC-MD5	Specifies HMAC-MD5. This value is required for transaction signatures (TSIG). Using this value automatically sets the **-k** flag.

These values are case-insensitive.

-b *keysize* Specifies the number of bits in the key. The choice of key size depends on the algorithm used:

- RSAMD5/RSASHA1 keys must be between 512 and 2048 bits.
- Diffie Hellman keys must be between 128 and 4096 bits.
- DSA keys must be between 512 and 1024 bits and an exact multiple of 64.
- HMAC-MD5 keys must be between 1 and 512 bits.

-n *nametype* Specifies the owner type of the key. The value of *nametype* must be one of:

ZONE Specifies a DNSSEC zone key (**KEY**/**DNSKEY**).

HOST Specifies a key associated with a host (**KEY**).

ENTITY Specifies a key associated with a host (**KEY**).

USER Specifies a key associated with a user (**KEY**).

OTHER Specifies a **DNSKEY**.

These values are case-insensitive.

-c *class* Indicates that the DNS record containing the key should have the specified class. If this flag is not specified, class **IN** is used.

-e If generating an RSAMD5/RSASHA1 key, use a large exponent.

-f *flag* Set the specified flag in the **flag** field of the KEY/DNSKEY record. The only recognized flag is KSK (Key Signing Key) DNSKEY.

-g *generator* If generating a Diffie Hellman key, use this generator. Allowed values are 2 and 5.

If no *generator* is specified, a known prime from RFC 2539 is used if possible; otherwise, the default is 2.

-h Prints a short help summary of the flags and values to **dnssec-keygen**.

-k Generates **KEY** records rather than **DNSKEY** records.

-p *protocol* Sets the protocol value for the generated key. The protocol is a number between 0 and 255. The default is 3 (DNSSEC). Other possible values for this argument are listed in RFC 2535 and its successors.

-r *randomdev* Specifies the source of randomness. If the operating system does not provide a **/dev/random** or equivalent device, the default source of randomness is keyboard input. (The OSS environment does not have a **/dev/random** device.)

randomdev specifies the name of a character device or file containing random data to be used instead of the default. The special value **keyboard** indicates that keyboard input should be used.

-s *strength* Specifies the strength value of the key. The strength is a number between 0 and 15, and currently has no defined purpose in DNSSEC.

-t *type* Indicates the use of the key. *type* must be one of:

AUTHCONF Use for data authentication and data encryption. This is the default.

NOAUTHCONF Do not use for data authentication or data encryption.

NOAUTH Do not use for data authentication.

NOCONF Do not use for data encryption.

-v *level* Sets the debugging level.

Operands

name Specifies the domain name for which the security information should be generated.

DESCRIPTION

dnssec-keygen generates keys for secure DNS, as defined in RFC 2535. It can also generate keys for use with TSIG (transaction signatures), as defined in RFC 2845.

Generated Keys

When **dnssec-keygen** completes successfully, it prints a string of the form **K***nnnn***.+***aaa***+***iiiii* to the standard output, where:

nnnn is the key name.

aaa is the numeric representation of the algorithm.

iiiii is the key identifier (or footprint).

This is an identification string for the key it has generated.

dnssec-keygen creates two files, with names based on the printed string. **K***nnnn***.+***aaa***+***iiiii***.key** contains the public key, and **K***nnnn***.+***aaa***+***iiiii***.private** contains the private key.

The **K***nnnn***.+***aaa***+***iiiii***.key** file contains a DNS **KEY** record that can be inserted into a zone file (directly or with a $INCLUDE statement).

The **K***nnnn***.+***aaa***+***iiiii***.private** file contains algorithm-specific fields. For security reasons, this file does not have general read permission.

Both files are generated for symmetric encryption algorithms such as HMAC-MD5, even though the public and private keys are equivalent.

EXAMPLE

To generate a 768-bit DSA key for the domain **example.com**, issue the following command:

dnssec-keygen -a DSA -b 768 -n ZONE example.com

This command prints a string of the form:

```
Kexample.com.+003+26160
```

In this example, **dnssec-keygen** creates the files **Kexample.com.+003+26160.key** and **Kexample.com.+003+26160.private**.

RELATED INFORMATION

Commands: **dnssec-signzone(8)**.

Documents: *BIND 9 Administrator Reference Manual*, RFC 2535, RFC 2845, RFC 2539.

NAME

dnssec-signzone - Runs the BIND 9 secure domain name server DNSSEC zone signing tool

SYNOPSIS

/etc/dns_secure/dnssec-signzone
[**-a**]
[**-c** *class*]
[**-d** *directory*]
[**-e** *end_time*]
[**-f** *output_file*]
[**-g**]
[**-h**]
[**-k** *key*]
[**-l** *domain*]
[**-i** *interval*]
[**-n** *nthreads*]
[**-o** *origin*]
[**-p**]
[**-r** *randomdev*]
[**-s** *start_time*]
[**-t**]
[**-v** *level*]
[**-z**]
zonefile
[*key* [...]]

FLAGS

-a Verify all generated signatures.

-c *class* Specifies the DNS class of the zone.

-k *key* Treat the specified key as a key signing key, ignoring any key flags. This flag may be specified multiple times.

-l *domain* Generate a **DLV** set in addition to the key (**DNSKEY**) and **DS** sets. The domain name is appended to the name of the records.

-d *directory* Look for keyset files in *directory* as the current directory.

-e *end_time* Specify the date and time when the generated **RRSIG** records expire. As with *start_time*, an absolute time is indicated in YYYYMMDDHHMMSS notation. A time relative to the start time is indicated with **+***N*, which is *N* seconds from the start time. A time relative to the current time is indicated with **now+***N*. If no *end_time* is specified, 30 days from the start time is used as a default.

-g Generate **DS** records for child zones from keyset files. Existing **DS** records are removed.

-s *start_time* Specify the date and time when the generated **RRSIG** records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in YYYYMMDDHHMMSS notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by **+***N*, which is *N* seconds from the current time. If no *start_time* is specified, the current time minus 1 hour (to allow for clock skew) is used.

-f *output-file* The name of the output file containing the signed zone. The default is to append **.signed** to the input file.

-h Prints a short help summary of the flags and values to **dnssec-signzone**.

-i *interval* When a previously signed zone is passed as input, records may be re-signed. The interval option specifies the cycle interval as an offset from the current time (in seconds). If an **RRSIG** record expires after the cycle interval, it is retained. Otherwise, it is considered to be expiring soon, and it is replaced.

The default cycle interval is one quarter of the difference between the signature end and start times. If neither *end_time* nor *start_time* are specified, **dnssec-signzone** generates signatures that are valid for 30 days, with a cycle interval of 7.5 days. Therefore, if any existing **RRSIG** records are due to expire in less than 7.5 days, they would be replaced.

-n *nthreads* Specifies the number of threads to use. By default, one thread is started for each detected processor.

-o *origin* Specifies the zone origin. If not specified, the name of the zone file is assumed to be the origin.

-p Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

-r *randomdev* Specifies the source of randomness. If the operating system does not provide a **/dev/random** or equivalent device, the default source of randomness is keyboard input. (The OSS environment does not have a **/dev/random** device.)

randomdev specifies the name of a character device or file containing random data to be used instead of the default. The special value **keyboard** indicates that keyboard input should be used.

-t Print statistics at completion.

-v *level* Sets the debugging level.

-z Ignore a **KSK** flag on a key when determining what to sign.

Operands

zonefile The name of the file containing the zone to be signed.

key ... The keys used to sign the zone. These keys are expressed in the form **K***nnnn***.+***aaa***+iiiii** as generated by **dnssec-keygen**.

If no keys are specified, the default values used are all zone keys that have private key files in the current directory.

DESCRIPTION

dnssec-signzone signs a zone. It generates **NSEC** and **RRSIG** records and produces a signed version of the zone file. The security status of delegations from the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a keyset file for each child zone.

EXAMPLE

The following command signs the **example.com** zone with the DSA key generated by the example in the **dnssec-keygen(8)** reference page. The zone's keys must be in the zone. If there are keyset files associated with child zones, they must be in the current directory.

```
dnssec-signzone -o example.com      db.example.com Kexample.com.+003+26160
```

In this example, **dnssec-signzone** creates the file **db.example.com.signed**. This file should be referenced in a **zone** statement in a **named.conf** file.

RELATED INFORMATION

Commands: **dnssec-keygen(8)**.

Documents: *BIND 9 Administrator Reference Manual*, RFC 2535.

NAME

lwresd - Starts the secure BIND 9 lightweight resolver demon

SYNOPSIS

/etc/dns_secure/lwresd
[**-C** *config_file*]
[**-d** *debug_level*]
[**-f**]
[**-g**]
[**-n** *ncpus*]
[**-P** *port1*]
[**-p** *port2*]
[**-s**]
[**-t** *directory*]
[**-T** *tcpip_process_name*]
[**-u** *user*]
[**-v**]

FLAGS

-C *config_file* Use *config_file* as the resolver configuration file instead of the default, **/etc/resolv.conf**. To ensure that reloading the configuration file continues to work after the server has changed its working directory because of a possible **directory** option in the configuration file, *config_file* should be an absolute pathname.

-d *debug_level* Set the server's debug level to *debug_level*. Debugging traces from **lwresd** become more verbose as the debug level increases.

-f Run the server in the foreground (that is, do not run as a demon).

-g Run the server in the foreground and force all logging to **stderr**.

-n *ncpus* Create *ncpus* worker threads to take advantage of multiple processors. If not specified, **lwresd** tries to determine the number of processors present and create one thread per processor. If it is unable to determine the number of processors, a single worker thread is created.

-P *port1* Listen for lightweight resolver queries on port *port1*. If this flag is not specified, the default is port 921.

-p *port2* Send DNS lookups to port *port2*. If this flag is not specified, the default is port 53.

This flag provides a way of testing the lightweight resolver server with a name server that listens for queries on a nonstandard port number.

-s Write memory usage statistics to **stdout** on exit.

This option is mainly of interest to BIND 9 developers and might be removed or changed in a future release.

-t *directory* Make the specified directory the current directory after processing the command line arguments, but before reading the configuration file.

Caution: This option should be used in conjunction with the **-u** option, because changing the root directory for a process running as the super ID does not enhance security on most systems; the way **chroot( )** is defined allows a process with root user privileges to escape a chroot jail.

-T *tcpip_process_name*
Start **lwresd** using the transport provider named *tcpip_process_name*. The process name must be specified as an OSS pathname for a Guardian process; that is, **/G/ZTC0** is a valid specification.

This flag overrides the default behavior of starting with the process $ZTC0 as the transport provider process.

-u *user* Change the user ID to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

-v Report the version number and exit.

DESCRIPTION

lwresd is the server providing name lookup services to clients that use the BIND 9 lightweight resolver library. It is a stripped-down, caching-only name server that answers queries using the BIND 9 lightweight resolver protocol rather than the DNS protocol.

lwresd listens for resolver queries on a UDP port on the IPv4 loopback interface, 127.0.0.1. This means that **lwresd** can only be used by processes running on the local machine. By default, UDP port number 921 is used for lightweight resolver requests and responses.

Incoming lightweight resolver requests are decoded by the server, which then resolves them using the DNS protocol. When the DNS lookup completes, **lwresd** encodes the answers in the lightweight resolver format and returns them to the client that made the request.

If **/etc/resolv.conf** contains any **nameserver** entries, **lwresd** sends recursive DNS queries to those servers. This is similar to the use of forwarders in a caching name server. If no **nameserver** entries are present, or if forwarding fails, **lwresd** resolves the queries autonomously starting at the root name servers, using a built-in list of root server hints.

FILES

/etc/resolv.conf
The default configuration file.

RELATED INFORMATION

Commands: **dnssec_lwresd(8)**, **named(8)**.

NAME

named - Starts the secure BIND 9 Internet domain name server

SYNOPSIS

/etc/dns_secure/named
[**-4** | **-6**]
[**-c** *config_file*]
[**-d** *debug_level*]
[**-f**]
[**-g**]
[**-n** *ncpus*]
[**-p** *port*]
[**-s**]
[**-T** *tcpip_process_name*]
[**-t** *directory*]
[**-u** *user*]
[**-v**]
[**-x** *cache_file*]

FLAGS

-4 Use IPv4 addresses even if the node supports IPv6 addresses. If neither **-4** nor **-6** are specified, both types of addresses can be used.

This flag is not recognized by the nonsecure version of **named**.

-6 Use IPv6 addresses even if the node also supports IPv4 addresses. If neither **-4** nor **-6** are specified, both types of addresses can be used.

This flag is not recognized by the nonsecure version of **named**.

-c *config_file* Use *config_file* as the configuration file instead of the default, */etc/named.conf*. To ensure that reloading the configuration file continues to work after the server has changed its working directory because of a possible **directory** option in the configuration file, *config_file* should be an absolute pathname.

-d *debug_level* Set the server's debug level to *debug_level*. Debugging traces from **named** become more verbose as the debug level increases.

-f Run the server in the foreground (that is, do not run it as a demon).

-g Run the server in the foreground and force all logging to *stderr*.

-n *ncpus* Create *ncpus* worker threads to take advantage of multiple processors. If not specified, **named** tries to determine the number of processors present and creates one thread per processor. If it is unable to determine the number of processors, a single worker thread is created.

-p *port* Listen for queries on port *port*. If not specified, the default is port 53.

-s Write memory usage statistics to *stdout* on exit.

This option is mainly of interest to BIND 9 developers and might be removed or changed in a future release.

-t *directory* Change the working directory to *directory* after processing the command line arguments, but before reading the configuration file.

Caution: This option should be used in conjunction with the **-u** option, because changing the root directory for a process running as the super ID does not enhance security on most systems; the way **chroot( )** is defined allows a process

with root user privileges to escape a chroot jail.

-T *tcpip_process_name*
Start the domain name server so that it uses the transport provider process with the process name *tcpip_process_name*. The process name must be specified as an OSS pathname for a Guardian process; that is, **/G/ZTC0** is a valid specification.

This flag overrides the default behavior of starting with the process $ZTC0 as the transport provider process. This flag is not recognized by the nonsecure version of **named**.

-u *user* **setuid( )** to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

-v Report the version number and exit.

-x *cache_file* Load data from *cache_file* into the cache of the default view.

Caution: This option must not be used. It is only of interest to BIND 9 developers and might be removed or changed in a future release.

DESCRIPTION

named is a domain name system (DNS) server, part of the BIND 9 distribution from the Internet Software Consortium (ISC). For more information on the DNS, see RFCs 1033, 1034, and 1035.

When invoked without arguments, **named** reads the default configuration file */etc/named.conf*, reads any initial data, and listens for queries.

Signals

In routine operation, signals should not be used to control the nameserver; **rndc** should be used instead.

SIGHUP Force a reload of the server.

SIGINT, **SIGTERM**
Shut down the server.

The result of sending any other signals to the server is undefined.

Configuration

The **named** configuration file **named.conf** is too complex to describe in detail here. A complete description is provided in the *BIND 9 Administrator Reference Manual*.

FILES

/etc/named.conf
The default configuration file.

/etc/resolv.conf
The default resolver configuration file.

/var/run/named.pid
The default process-id file.

RELATED INFORMATION

Commands: **dnssec_rndc(8)**, **lwresd(8)**, **named(8)**, **rndc(8)**.

Files: **resolv.conf(5)**.

Documents: *RFC 1033*, *RFC 1034*, *RFC 1035*, *BIND 9 Administrator Reference Manual*.

NAME

nsupdate - Starts the secure BIND 9 dynamic domain name system (DNS) update utility

SYNOPSIS

/etc/dns_secure/nsupdate

[**-d**]
[**[-y** *keyname:secret* **] [-k** *keyfile* **]**]
[**-v**]
[**filename**]

DESCRIPTION

nsupdate is used to submit Dynamic DNS Update requests (as defined in RFC2136) to a BIND 9 domain name server. This allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record.

Zones that are under dynamic control via **nsupdate** or a DHCP server should not be edited by hand. Manual edits could conflict with dynamic updates and cause data to be lost.

The resource records that are dynamically added or removed with **nsupdate** have to be in the same zone. Requests are sent to the zone's master server. This is identified by the **MNAME** field of the zone's **SOA** record.

The **-d** flag makes **nsupdate** operate in debug mode. This mode provides tracing information about the update requests that are made and the replies received from the name server.

Transaction signatures can be used to authenticate the Dynamic DNS updates. These use the **TSIG** resource record type described in RFC2845. The signatures rely on a shared secret that should only be known to **nsupdate** and the name server. Currently, the only supported encryption algorithm for **TSIG** is **HMAC-MD5**, which is defined in RFC 2104. Once other algorithms are defined for **TSIG**, applications will need to ensure they select the appropriate algorithm as well as the key when authenticating each other. For instance, suitable **key** and **server** statements would be added to */etc/named.conf* so that the name server can associate the appropriate secret key and algorithm with the IP address of the client application that will use TSIG authentication. **nsupdate** does not read */etc/named.conf*.

nsupdate uses the **-y** or **-k** flag to provide the shared secret needed to generate a **TSIG** record for authenticating Dynamic DNS update requests. These flags are mutually exclusive. With the **-k** flag, **nsupdate** reads the shared secret from the file *keyfile*, whose name is of the form **K***name***.+157.+***random***.private**. For historical reasons, the file *Kname***.+157.+***random***.key** must also be present. When the **-y** flag is used, a signature is generated from *keyname:secret. keyname* is the name of the key, and *secret* is the base64 encoded shared secret. Use of the **-y** flag is discouraged because the shared secret is supplied as a command line argument in clear text. This may be visible in the output from **ps**(1) or in a history file maintained by the user's shell.

By default, **nsupdate** uses UDP to send update requests to the name server. The **-v** flag makes **nsupdate** use a TCP connection. This may be preferable when a batch of update requests is made.

Input Format

nsupdate reads input from *filename* or standard input. Each command is supplied on exactly one line of input. Some commands are for administrative purposes. The others are either update instructions or prerequisite checks on the contents of the zone. These checks set conditions that some name or set of resource records (RRset) either exists or is absent from the zone. These conditions must be met if the entire update request is to succeed. Updates are rejected if the tests for the prerequisite conditions fail.

Every update request consists of zero or more prerequisites and zero or more updates. This allows a suitably authenticated update request to proceed if some specified resource records are

present or missing from the zone. A blank input line (or the **send** command) causes the accumulated commands to be sent as one Dynamic DNS update request to the domain name server.

The command formats and their meaning are as follows:

server *servername* [*port*]
Sends all dynamic update requests to the name server *servername*. When no server statement is provided, **nsupdate** sends updates to the master server of the correct zone. The **MNAME** field of that zone's **SOA** record identifies the master server for that zone. *port* is the port number on *servername* where the dynamic update requests get sent. If no port number is specified, the default DNS port number of 53 is used.

local *address* [*port*]
Sends all dynamic update requests using the local *address*. When no local statement is provided, **nsupdate** sends updates using an address and port chosen by the system. *port* can additionally be used to make requests come from a specific port. If no port number is specified, the system assigns one.

zone *zonename* Specifies that all updates are to be made to the zone *zonename*. If no *zone* statement is provided, **nsupdate** attempts to determine the correct zone to update based on the rest of the input.

key *name secret*
Specifies that all updates are to be **TSIG** signed using the *keyname keysecret* pair. The **key** command overrides any key specified on the command line via **-y** or **-k**.

prereq nxdomain *domain_name*
Requires that no resource record of any type exists with name *domain_name*.

prereq yxdomain *domain_name*
Requires that *domain_name* exists (has as at least one resource record, of any type).

prereq nxrrset *domain_name* [*class*] *type*
Requires that no resource record exists of the specified *type*, *class* and *domain_name*. If *class* is omitted, **IN** (internet) is assumed.

prereq yxrrset *domain_name* [*class*] *type*
This requires that a resource record of the specified *type*, *class* and *domain_name* must exist. If *class* is omitted, **IN** (internet) is assumed.

prereq yxrrset *domain_name* [*class*] *type data...*
The *data* from each set of prerequisites of this form sharing a common *type*, *class*, and *domain-name* are combined to form a set of RRs. This set of RRs must exactly match the set of RRs existing in the zone at the given *type*, *class*, and *domain_name*. The *data* is written in the standard text representation of the resource record's **RDATA**.

update delete *domain_name* [*ttl*] [*class*] [*type* [*data...*]]
Deletes any resource records named *domain_name*. If *type* and *data* is provided, only matching resource records are removed. The internet class is assumed if *class* is not supplied. The *ttl* is ignored, and is only allowed for compatibility.

update add *domain_name ttl* [*class*] *type data...*
Adds a new resource record with the specified *ttl*, *class*, and *data*.

show Displays the current message, containing all of the prerequisites and updates specified since the last send.

send Sends the current message. This is equivalent to entering a blank line.

Lines beginning with a semicolon are comments, and are ignored.

EXAMPLES

The examples below show how **nsupdate** could be used to insert and delete resource records from the **example.com** zone. Notice that the input in each example contains a trailing blank line so that a group of commands are sent as one dynamic update request to the master name server for **example.com**.

1.

```
# nsupdate
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
>
```

Any A records for **oldhost.example.com** are deleted and an A record for **newhost.example.com** with IP address 172.16.1.1 is added. The newly-added record has a one-day TTL (86400 seconds)

2.

```
# nsupdate
> prereq nxdomain nickname.example.com
> update add nickname.example.com 86400 CNAME somehost.example.com
>
```

The prerequisite condition gets the name server to check that there are no resource records of any type for **nickname.example.com**. If there are, the update request fails. If this name does not exist, a **CNAME** for it is added. This ensures that when the **CNAME** is added, it cannot conflict with the long-standing rule in RFC1034 that a name must not exist as any other record type if it exists as a **CNAME**. (The rule has been updated for DNSSEC in RFC2535 to allow **CNAME**s to have **SIG**, **KEY**, and **NXT** records.)

FILES

/etc/resolv.conf
Default resolver configuration file, used to identify default name server

K*name***.+157.+***random***.key**
base-64 encoding of **HMAC-MD5** key created by the **dnssec-keygen** utility.

K*name***.+157.+***random***.private**
base-64 encoding of **HMAC-MD5** key created by the **dnssec-keygen** utility.

RELATED INFORMATION

Commands: **dnssec_named(8)**, **named(8)**, **dnssec-keygen(8)**.

Documents: **RFC2136**, **RFC3007**, **RFC2104**, **RFC2845**, **RFC1034**, **RFC2535**.

NAME

rndc - Starts the secure BIND 9 Internet domain name server control utility

SYNOPSIS

/etc/dns_secure/rndc
[**-c** *config_file*]
[**-k** *key_file*]
[**-s** *server*]
[**-p** *port*]
[**-V**]
[**-y** *key_id*]
command

FLAGS

-c *config_file* Use *config_file* as the configuration file instead of the default, */etc/rndc.conf*.

-k *key_file* Use *key_file* as the key file instead of the default, */etc/rndc.key*. The key in */etc/rndc.key* will be used to authenticate commands sent to the server if the *config_file* does not exist.

-s *server* *server* is the name or address of the server which matches a server statement in the configuration file for **rndc**. If no server is supplied on the command line, the host named by the **default-server** clause in the option statement of the configuration file is used.

-p *port* Send commands to TCP port *port* instead of BIND 9's default control channel port, 953.

-V Enable verbose logging.

-y *keyid* Use the key *keyid* from the configuration file. *keyid* must be known by **named** with the same **algorithm** and **secret** string in order for control message validation to succeed. If no *keyid* is specified, **rndc** first looks for a key clause in the server statement of the server being used, or if no server statement is present for that host, it then looks for the **default-key** clause of the options statement. Note that the configuration file contains shared secrets which are used to send authenticated control commands to name servers. It should therefore not have general read or write access.

Operands

command For the complete set of commands supported by **rndc**, see the *BIND 9 Administrator Reference Manual* or run **rndc** without arguments to see its help message.

DESCRIPTION

rndc controls the operation of a BIND 9 domain name server. If **rndc** is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

rndc communicates with the name server over a TCP connection, sending commands authenticated with digital signatures. In the nonsecure version of **rndc** and **named**, the only supported authentication algorithm is **HMAC-MD5**, which uses a shared secret on each end of the connection. This provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a key_id known to the server.

rndc reads a configuration file to determine how to contact the name server and decide what algorithm and key it should use.

NOTES

rndc does not yet support all the commands of the BIND 8 **ndc** utility.

There is currently no way to provide the shared secret for a **key_id** without using the configuration file.

Several error messages could be clearer.

RELATED INFORMATION

Commands: **dnssec_named(8)**, **dnssec_nsupdate(8)**, **named(8)**, **nsupdate(8)**.

Files: **named.conf(4)**.

Documents: *BIND 9 Administrator Reference Manual*.

NAME

ftp server - Services FTP connection requests

DESCRIPTION

The FTP server for the OSS environment is a program called FTPSERV that runs as a process in the Guardian environment. FTPSERV is called to service each FTP connection request for either environment.

FTPSERV is invoked by the Guardian LISTNER process each time a connection to an FTP server process is requested. The **inetd** process is not used to initiate an FTP server (such as the **ftpd** process used on many UNIX systems) in the OSS environment. (An **ftp** client is supported as a process in the OSS environment.) If the PORTCONF file that invokes FTPSERV contains the parameter **-t***timerval*, inactive connections time-out after *timerval* seconds.

Accessing Either Guardian or OSS File Systems

The FTP server allows a client to access either the Guardian file system or the OSS file system. Toggle between the two file systems using the **quote** command from the FTP client followed by the string "Guardian" or "OSS". For example:

```
quote Guardian
```

accesses the Guardian file system, and

```
quote OSS
```

accesses the OSS file system.

Once the user has logged in, the user can determine the file system with a **pwd** command. If the result is either **/G/vol/subvol** or an OSS pathname, the OSS file system is being accessed. If the result is **$vol.subvol**, the Guardian file system is being used.

FTP Client User Command Requests

The FTP server currently supports requests for the following FTP client user commands when the OSS file system is used. These commands are not case-sensitive. For the syntax of each command, enter **remotehelp** *command_name* from an FTP client process.

ABOR	abort previous comand
APPE	append to a file
CWD	change working directory
DELE	delete a file
HELP	display help information
LIST	list files in a directory (**ls -lg**)
MKD	make a directory
MODE	specify data transfer mode
NLST	list files in directory (**ls**)
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory

QUIT	terminate the session
RETR	retrieve a file
RMD	remove a directory
RNFR	specify *rename_from* filename
RNTO	specify *rename_to* filename
SITE	provide system-specific services
STOR	store a file
STRU	specify data transfer structure (store unique)
TYPE	specify data transfer type
USER	specify user name
XPWD	print the current working directory

The remaining FTP commands specified in RFC 959 are recognized but not implemented.

The FTP server aborts an active file transfer only when the **ABOR** command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in RFC 959.

The FTP server interprets file names according to the "globbing" conventions used by **ksh**. This allows users to use the metacharacters '*** ? [] {}**'.

Using SITE Commands

SITE commands are used by an FTP server to provide services specific to the system on which the server runs. Such services are those essential to file transfer but not so generally needed that they must be implemented as commands in the protocol. FTPSERV supports the following **SITE** commands:

CHMOD [*nnn*]
: When the OSS file system is used, the **SITE CHMOD** command sets or checks the security of a destination file. *nnn* is a three-digit octal value for access permissions.

  If *nnn* is a valid value, all files transferred after this command have their security set to the specified value. If *nnn* is an invalid value, the error is reported to the user and the current valid security setting is unchanged.

  If the command is specified without an *nnn* value, the current security setting is displayed.

  If the **SITE CHMOD** command is not specified, all files transferred have their security set to 666 (**-rw-rw-rw**).

HELP
: When either file system is used, the **SITE HELP** command describes each server service and the syntax for its command.

NOCRLF { **ON** | **OFF** }
: When the Guardian file system is used, the **SITE NOCRLF** command enables (**ON**) or disables (**OFF**) carriage return and linefeed translation by the server during file transfers.

  If the **SITE NOCRLF** command is not used, carriage return and linefeed translation occurs by default.

SHOWOPEN { **ON** | **OFF** }

When the Guardian file system is used, the **SITE SHOWOPEN** command displays (**ON**) or does not display (**OFF**) an open flag (an **O** next to the filecode field) when the FTP client's **DIR** command is used and a listed file has at least one current open.

If the **SITE SHOWOPEN** command is not used, no open flags are displayed.

Authenticating Users

The FTP server authenticates users according to the following rules:

- Using the **User_Authenticate** call to validate a user. If an initial OSS directory is configured, OSS is the default file system. Use the **quote Guardian** command to access the Guardian file system. If an OSS directory is not configured, Guardian is the default file system.
- Using the **User_Getinfo_** call to set up the initial directory of both the OSS and Guardian file systems for the user.
- Failure of either of the above results in the user not logging in. The FTP server supports Safeguard security configuration.

The Guardian SAFECOM program used for setting-up FTP user IDs is described below.

Adding FTP user IDs to the System Using the Guardian SAFECOM Program

The following Guardian TACL macro shows how to add an FTP user ID that defaults to an OSS directory.

```
?TACL MACRO
== This macro adds FTP user IDs to the system
#FRAME
#SET #INLINEPREFIX>
INLECHO ON

safecom /INLINE/
>add user admin.ftp, 168,10, password ftp
>alter user admin.ftp, guardian default security NNNN
>alter user admin.ftp, guardian default volume $data2.q9552
>add alias ftp, 168,19, password ftp
>alter alias ftp, guardian default security NNNN
>alter alias ftp, guardian default volume $data2.q9552
>alter alias ftp, initial-program /bin/ksh
>alter alias ftp, initial-directory /user/q9552
>info user super.ftp, detail
>info alias ftp, detail
>exit

#UNFRAME
```

FTP Replies

The FTP server reply consists of a three digit number (transmitted as three alphanumeric characters) followed by text. Programmatic applications can use the number to determine what state to enter next. Users can interpret the text. A reply contains a 3-digit code, followed by Space <SP>, followed by one line of text (where some maximum line length has been specified), and terminated by the Telnet end-of-line code. The first digit of the three digit code denotes whether the response is good, bad or incomplete. There are five values for the first digit of the reply code:

1yz Positive Preliminary reply. The requested action is being initiated; expect another reply before proceeding with a new command.

2yz Positive Completion reply. The requested action completed successfully. A new request may be initiated.

3yz Positive Intermediate reply. The command was accepted, but the requested action is pending, waiting for further information. The user should send another command specifying the information. This reply is used in command sequence groups.

4yz Transient Negative Completion Reply. The command was not accepted. The requested action did not take place, but the error condition is temporary. Request the action again.

5yz Permanent Negative Completion Reply. The command was not accepted. The requested action did not take place.

The second digit indicates the following types of information:

x0z Syntax: syntax errors, syntactically correct commands that don't fit any functional category, or unimplemented commands.

x1z Information: replies to requests for information, such as status or help.

x2z Connections: replies referring to the control and data connections.

x3z Authentication and accounting: replies for the login process and accounting procedures.

x4z Undefined

x5z File system: replies indicate the status of the Server file system regarding the requested transfer or other file system action.

The third digit describes a more detailed meaning of the function categories, specified by the second digit. The list of replies in **Reply Codes by Function Groups** describe values for the third digit. Note that the text associated with each reply is recommended, rather than mandatory, and may even change according to the command with which it is associated.

Reply Codes by Function Groups

The following is a list of reply codes specified in RFC959. Not all codes listed below are generated by the OSS FTP server.

200 Command okay.

500 Syntax error, command unrecognized.

501 Syntax error in parameters or arguments.

202 Command not implemented, superfluous at this site.

502 Command not implemented.

503 Bad sequence of commands.

504	Command not implemented for that parameter.
211	System status, or system help reply.
212	Directory status.
213	File status.
214	Help message.
120	Service ready in *nnn* minutes.
220	Service ready for new user.
221	Service closing control connection. Logged out if appropriate.
421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
125	Data connecton already open; transfer starting.
225	Data connection open; no transfer in progress.
425	Can't open data connection.
226	Closing data connection. Requested file action successful (for example, file transfer or file abort).
426	Connection closed; transfer aborted.
227	Entering Passive Mode (h1,h2,h3,h4,p1,p2).
230	User logged in, proceed.
530	Not logged in.
331	User name okay, need password.
332	Need account for login.
532	Need account for storing files.
150	File status okay; about to open data connection.
250	Requested file action okay, completed.
350	Requested file action pending further information.
450	Requested file action not taken. File unavailable (for example, file busy).
550	Requested action not taken. File unavailable (for example, file not found, no access).
451	Requested action aborted. Local error in processing.
551	Requested action aborted. Page type unknown.
452	Requested action not taken. Insufficient storage space in system.

552 Requested file action aborted. Exceeded storage allocation (for current directory or dataset).

553 Requested action not taken. File name not allowed.

RELATED INFORMATION

Commands: **ftp(1)**, **sh(1)**.

NAME

inetd - The Internet superserver

SYNOPSIS

/usr/ucb/inetd
[**-d**]
[**-L**]
[**-R** *rate*]
[**-W** *process_name*]
[**-f**]
[*configfile*]

FLAGS

-d Turns on debugging.

-L Turns on load balancing. When load balancing is performed, **inetd** runs external server service programs on all available processors in cyclic order, or on the set of processors specified for an external service in the *Proc* entry of an **inetd** configuration file.

The default action is to run all external server service programs on the processor used to run **inetd**.

-R *rate*

Specifies the maximum number of times per minute a service can be invoked. The default value is 40.

-W *process_name*

Specifies the NonStop operating system process name to assign to the running process. The value used as *process_name* must conform to the naming rules for NonStop operating system process names and must be specified as the OSS pathname for a Guardian named process; that is, it must be specified in the form **/G/***process_name*, where the dollar sign (**$**) is omitted.

The default action is to run the process as an unnamed process.

-f Run the **inetd** process in the foreground. For an example of configuring and starting **inetd** as a persistent process, see "Examples."

This flag is valid for systems running:

- J06.05 and later J-series RVUs
- H06.16 and later H-series RVUs
- G06.33 and later G-series RVUs
- J06.03, J06.04, or H06.03 through H06.15 RVUs and have installed SPR T9660H01^AAJ
- G06.20 through G06.32 RVUs and have installed SPR T9660H01^AAK

configfile

Specifies the pathname of the configuration file to be used for this invocation of the process. If this operand is omitted, the default pathname of **/etc/inetd.conf** is used.

DESCRIPTION

The **inetd** process should be run immediately after loading the OSS product files into the OSS environment. It listens for connections on certain Internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. After the program completes the request, it continues to listen on the socket (except in some cases that are described later in this section). Essentially, **inetd** allows one server process to invoke several others, reducing load on the system.

Upon execution, **inetd** reads its configuration information from a configuration file (either *configfile* in the command line or **/etc/inetd.conf**). There must be an entry in each field of the configuration file (except as noted below). The fields must be separated by a tab or a space. Comments are denoted by a **#** (number sign) at the beginning of a line.

The fields of the configuration file are as follows:

SrvName SockType ProtoName Wait/Nowait UserName [Proc] SrvPath SrvArgs

SrvName
The name of a valid service in the **/etc/services** file. For **internal** services (discussed later in this section), the service name must be the official name of the service; that is, the first entry in **/etc/services**).

SockType
One of **stream**, **dgram**, or **raw**, depending on whether the socket is a stream, datagram, or raw socket.

ProtoName
A valid protocol as given in **/etc/protocols**. Examples are **tcp** or **udp**.

Wait/Nowait
Applicable to datagram sockets only. (Other sockets should have a **nowait** entry in this space.)

If a datagram server connects to its peer, freeing the socket so **inetd** can receive further messages on the socket, it is said to be a multithreaded server, and should use the **nowait** entry.

For datagram servers that process all incoming datagrams on a socket and eventually time out, the server is defined as singlethreaded, and should use a **wait** entry.

UserName
The username of the user for whom the server should run. This allows for servers to be given less permission than the super ID.

Proc The processor number of one or more processors on which to run the corresponding external server process. This field is optional and can be omitted.

This field is used to perform load-balancing of server processes among the processors within a node. When this field is omitted, **inetd** assigns each server it starts to the next available processor in the node. This field is ignored unless the **-L** flag is specified when **inetd** is started.

The number specification must be enclosed in square brackets (**[** and **]**). If more than one processor number is used, the numbers can be specified as a range (using a dash between the lowest and highest processor numbers in the range), as a comma-separated list, or as a combination of those two formats. The current processor (the processor on which **inetd** runs) can be specified using empty brackets (**[]**). Blanks cannot appear within the brackets.

When **inetd** is started without specifying the **-L** flag, the behavior is the same as when

empty brackets are used with the **-L** flag; that is, all external service programs run on the processor used by **inetd**.

SrvPath

The pathname of the server program that is to be executed by **inetd** when a service request is found on its socket. If **inetd** provides a service internally, this entry should be **internal**.

SrvArgs

The command-line arguments that the server process must execute. For services that **inetd** provides internally, this field should be left empty.

The arguments to a *SrvPath* entry should be specified just as they normally are for a shell command line, starting with *argv*[*0*], which is the name of the program.

The **inetd** process can provide several trivial services itself, without using external server programs. These services are **echo**, **discard**, **chargen** (character generator), **daytime** (human-readable time), and **time** (machine-readable time, in the form of the number of seconds since midnight January 1, 1900). All of these services are **tcp** based.

The **inetd** process rereads its configuration file when it receives a hangup signal, **SIGHUP**. Services may be added, deleted, or modified when the configuration file is reread.

EXAMPLES

1. To start **inetd** as a named process and restrict it to satisfying approximately 10 service requests per minute, enter:

   /usr/ucb/inetd -R 10 -W /G/INETD /etc/inetd.conf &

2. To start **inetd** as a named process and perform load balancing for the **rexecd** service using processors 2 through 4, enter:

   /usr/ucb/inetd -R 10 -W /G/INETD -L &

   using an **/etc/inetd.conf** configuration file that contains the following entry:

   exec stream tcp nowait super.super [2-4] /bin/rexecd

3. To start **inetd** as a foreground process, enter:

   /usr/ucb/inetd -f

4. These SCF commands configure and start the **inetd** process as a persistent process:

```
SCF-> ADD PROCESS $ZZKRN.OSSINT,                                    &
    NAME $OINT,                                                     &
    ASSOCPROC $INET0,                                               &
    STARTUPMSG                                                      &
    "-osstty -name /G/INET0 -p /usr/ucb/inetd -f -L /etc/inetd.conf", &
    PROGRAM $system.system.osh,                                     &
    HIGHPIN ON,                                                     &
    PRIMARYCPU 1,                                                   &
    HOMETERM $ZHOME,                                                &
    TYPE OTHER,                                                     &
    STARTMODE manual,                                               &
    STOPMODE standard,                                              &
    AUTORESTART 5,                                                  &
    USERID SUPER.SUPER
SCF-> ADD PROCESS $ZZKRN.OSSINT, (param SOCKETˆTRANSPORTˆNAME $ZTC1)
SCF-> START PROCESS $ZZKRN.OSSINT
```

FILES

/usr/ucb/inetd
Specifies the command path.

/etc/inetd.conf
Contains information on the services used for the Internet sockets in the system.

/etc/services
Contains the names of official and unofficial Internet services used in the system.

/etc/protocols
Contains the names of the Internet protocols implemented in the system.

NOTES

The OSS sockets transport provider process used by **inetd** can be specified in either of the following ways:

1. Set the Guardian PARAM SOCKETˆTRANSPORTˆNAME in the HP Tandem Advanced Command Language (TACL) session that subsequently starts the OSS shell session in which **inetd** is started. This specification has the format:

   `PARAM SOCKETˆTRANSPORTˆNAME` *process_name*

   For example:

   ```
   PARAM SOCKETˆTRANSPORTˆNAME $ZTC1
   ```

2. Set the environment variable **SOCKET_TRANSPORT_NAME** in the OSS shell session from which **inetd** is started. This specification has the format:

   `EXPORT SOCKET_TRANSPORT_NAME=\`*process_name*

   For example:

   ```
   EXPORT SOCKET_TRANSPORT_NAME=\$ZTC1
   ```

When the PARAM is specified, it becomes the corresponding environment variable when the OSS shell is started.

RELATED INFORMATION

Miscellaneous: **rshd(8)**.

Files: **services(4)**.

STANDARDS CONFORMANCE

The following are HP extensions to traditional UNIX implementations of **inetd**:

- The **-f**, **-L**, and **-W** parameters of the **inetd** command line
- The *Proc* field of the configuration file entries
- The use of NonStop operating system DEFINE statements

NAME

lwresd - Starts the nonsecure BIND 9 lightweight resolver demon

SYNOPSIS

/etc/dns923/lwresd
[**-C** *config_file*]
[**-d** *debug_level*]
[**-f**]
[**-g**]
[**-n** *ncpus*]
[**-P** *port1*]
[**-p** *port2*]
[**-s**]
[**-t** *directory*]
[**-T** *tcpip_process_name*]
[**-u** *user*]
[**-v**]

FLAGS

-C *config_file* Use *config_file* as the resolver configuration file instead of the default, **/etc/resolv.conf**. To ensure that reloading the configuration file continues to work after the server has changed its working directory because of a possible **directory** option in the configuration file, *config_file* should be an absolute pathname.

-d *debug_level* Set the server's debug level to *debug_level*. Debugging traces from **lwresd** become more verbose as the debug level increases.

-f Run the server in the foreground (that is, do not run as a demon).

-g Run the server in the foreground and force all logging to **stderr**.

-n *ncpus* Create *ncpus* worker threads to take advantage of multiple processors. If not specified, **lwresd** tries to determine the number of processors present and create one thread per processor. If it is unable to determine the number of processors, a single worker thread is created.

-P *port1* Listen for lightweight resolver queries on port *port1*. If this flag is not specified, the default is port 921.

-p *port2* Send DNS lookups to port *port2*. If this flag is not specified, the default is port 53.

This flag provides a way of testing the lightweight resolver server with a name server that listens for queries on a nonstandard port number.

-s Write memory usage statistics to **stdout** on exit.

This option is mainly of interest to BIND 9 developers and might be removed or changed in a future release.

-t *directory* Make the specified directory the current directory after processing the command line arguments, but before reading the configuration file.

Caution: This option should be used in conjunction with the **-u** option, because changing the root directory for a process running as the super ID does not enhance security on most systems; the way **chroot( )** is defined allows a process with root user privileges to escape a chroot jail.

-T *tcpip_process_name*
Start **lwresd** using the transport provider named *tcpip_process_name*. The process name must be specified as an OSS pathname for a Guardian process; that is, **/G/ZTC0** is a valid specification.

This flag overrides the default behavior of starting with the process $ZTC0 as the transport provider process.

-u *user* Change the user ID to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

-v Report the version number and exit.

DESCRIPTION

lwresd is the server providing name lookup services to clients that use the BIND 9 lightweight resolver library. It is a stripped-down, caching-only name server that answers queries using the BIND 9 lightweight resolver protocol rather than the DNS protocol.

lwresd listens for resolver queries on a UDP port on the IPv4 loopback interface, 127.0.0.1. This means that **lwresd** can only be used by processes running on the local machine. By default, UDP port number 921 is used for lightweight resolver requests and responses.

Incoming lightweight resolver requests are decoded by the server, which then resolves them using the DNS protocol. When the DNS lookup completes, **lwresd** encodes the answers in the lightweight resolver format and returns them to the client that made the request.

If **/etc/resolv.conf** contains any **nameserver** entries, **lwresd** sends recursive DNS queries to those servers. This is similar to the use of forwarders in a caching name server. If no **nameserver** entries are present, or if forwarding fails, **lwresd** resolves the queries autonomously starting at the root name servers, using a built-in list of root server hints.

FILES

/etc/resolv.conf
The default configuration file.

RELATED INFORMATION

Commands: **named(8)**.

NAME

merge_whatis - Creates and updates the **whatis** database file used by the **apropos**, **man**, and **whatis** commands

SYNOPSIS

merge_whatis
[*MANPATH_entry*]

FLAGS

MANPATH_entry
Specifies the absolute pathname of the directory in which the **whatis** database file should be located. If this operand is omitted, the default directory expected by the **man** command (**/usr/share/man**) is used.

DESCRIPTION

The **merge_whatis** command allows you to create or replace the **whatis** database file that corresponds to a set of installed reference (man) pages. To execute the **merge_whatis** command for reference pages distributed by HP, your user ID must be the super ID. (This restriction is imposed by the security of the directories that contain the files processed by the command.)

Use this command after you install an HP product for which reference pages are added to your system or updated on your system. You can also use the **merge_whatis** command after adding reference pages that are written on site or by other software vendors, provided the database file file-naming convention and placement shown under **FILES** later in this reference page are met.

You can determine whether this command needs to be executed by recording the last-modification dates for all files in the **whatis.frag** directory after each installation. After the next installation, check the current dates against the dates you recorded from the previous installation. If the last-modification date for any file changes after an installation, then the **merge_whatis** command should be used.

EXAMPLES

1. Using the command to update the **/usr/share/man/whatis** file:

```
merge_whatis
```

FILES

/usr/share/man/whatis
Contains the default database used by the **apropos**, **man**, and **whatis** commands.

/usr/share/man/whatis.frag/whatis.*fragment*
Contains the database fragment files used to create the **/usr/share/man/whatis** file.

For HP products, the *fragment* value is normally the Tandem product number (T number) of the corresponding **pax** archive file that contains the product code. Such **pax** archive files are installed from **/G/***tsvvol***/zossutl/***fragment***man**.

*MANPATH_entry***/whatis**
Contains an alternate database used by the **apropos**, **man**, and **whatis** commands. The **merge_whatis** command can create this database file in any directory that contains directories of reference pages, provided the directory named **whatis.frag** also exists in the directory specified by the *MANPATH_entry* operand.

*MANPATH_entry***/whatis/whatis.frag/whatis.***fragment*
Contains the database fragment files used to create the *MANPATH_entry***/whatis** file. This set of files must be present in the directory specified by the

MANPATH_entry operand when that operand is used.

The values used for *fragment* must not include an asterisk (*****).

NOTES

Any existing **whatis** file is saved as **whatis_old**.

DIAGNOSTICS

The **merge_whatis** command issues the following error messages to the standard output file:

`Usage: merge_whatis [MANPATH_entry]`
Too many operands were specified.

Check for extra blanks in the input line and reenter the command.

MANPATH_entry `does not exist.`
The pathname specified as the *MANPATH_entry* operand cannot be found.

Check the specified pathname for typographical errors and reenter the command. Do not use `.` or `..` in the specified pathname.

EXIT VALUES

If too many operands are specified, or if a specified pathname does not exist, the **merge_whatis** command fails and returns a nonzero value.

RELATED INFORMATION

Commands: **apropos(1)**, **man(1)**, **whatis(1)**.

STANDARDS CONFORMANCE

This utility is an HP extension to the XPG4 Version 2 specification.

NAME

named - Starts the nonsecure BIND 9 Internet domain name server

SYNOPSIS

/etc/dns923/named
[**-c** *config_file*]
[**-d** *debug_level*]
[**-f**]
[**-g**]
[**-n** *ncpus*]
[**-p** *port*]
[**-s**]
[**-t** *directory*]
[**-u** *user*]
[**-v**]
[**-x** *cache_file*]

FLAGS

-c *config_file* Use *config_file* as the configuration file instead of the default, */etc/named.conf*. To ensure that reloading the configuration file continues to work after the server has changed its working directory because of a possible **directory** option in the configuration file, *config_file* should be an absolute pathname.

-d *debug_level* Set the server's debug level to *debug_level*. Debugging traces from **named** become more verbose as the debug level increases.

-f Run the server in the foreground (that is, do not run it as a demon).

-g Run the server in the foreground and force all logging to *stderr*.

-n *ncpus* Create *ncpus* worker threads to take advantage of multiple processors. If not specified, **named** tries to determine the number of processors present and creates one thread per processor. If it is unable to determine the number of processors, a single worker thread is created.

-p *port* Listen for queries on port *port*. If not specified, the default is port 53.

-s Write memory usage statistics to *stdout* on exit.

This option is mainly of interest to BIND 9 developers and might be removed or changed in a future release.

-t *directory* Change the working directory to *directory* after processing the command line arguments, but before reading the configuration file.

Caution: This option should be used in conjunction with the **-u** option, because changing the root directory for a process running as the super ID does not enhance security on most systems; the way **chroot( )** is defined allows a process with root user privileges to escape a chroot jail.

-u *user* **setuid( )** to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

-v Report the version number and exit.

-x *cache_file* Load data from *cache_file* into the cache of the default view.

Caution: This option must not be used. It is only of interest to BIND 9 developers and might be removed or changed in a future release.

DESCRIPTION

named is a domain name system (DNS) server, part of the BIND 9 distribution from the Internet Software Consortium (ISC). For more information on the DNS, see RFCs 1033, 1034, and 1035.

When invoked without arguments, **named** reads the default configuration file */etc/named.conf*, reads any initial data, and listens for queries.

Signals

In routine operation, signals should not be used to control the nameserver; **rndc** should be used instead.

SIGHUP Force a reload of the server.

SIGINT, **SIGTERM**
Shut down the server.

The result of sending any other signals to the server is undefined.

Configuration

The **named** configuration file **named.conf** is too complex to describe in detail here. A complete description is provided in the *BIND 9 Administrator Reference Manual*.

FILES

/etc/named.conf
The default configuration file.

/etc/resolv.conf
The default resolver configuration file.

/var/run/named.pid
The default process-id file.

RELATED INFORMATION

Commands: **dnssec_named(8)**, **dnssec_rndc(8)**, **rndc(8)**, **lwresd(8)**.

Files: **resolv.conf(5)**.

Documents: *RFC 1033*, *RFC 1034*, *RFC 1035*, *BIND 9 Administrator Reference Manual*.

NAME

newusers - Updates and creates new users in batch.

SYNOPSIS

newusers [*options new_users_file*]

FLAGS

-h Display this help and exit.

DESCRIPTION

The **newusers** command reads a file of user name and clear-text password pairs and uses this information to update a group of existing users or to create new users. Each line is in the format as shown below:

user_type:name:passwd:memberNum:groupNum:description:dir

user_type
The type of user, user or alias, as indicated by U or A respectively.

name The name of the user or alias. It can be the name of a new user or alias or the name of an existing user or alias (or a user created before by **newusers**). For an existing user, the user's information will be changed, otherwise a new user will be created. In the absence of a fully qualified user name, a username will be derived as explained in **useradd**(8) or **usermod**, respectively.

passwd The new clear-text value of the password that Safeguard will encrypt.

This option is only valid if Safeguard is running on the system.

memberNum
The member number of the user. If the field is empty for a new user, a new (unused) member number will be derived automatically as explained in **useradd**(8). If this field contains a number, it must match the existing value for an existing user specified or derived from *name*. This must be an unused number for a new user.

groupNum
The primary group number of the user. If the field is empty for a user, the group number is derived automatically as explained in **useradd**(8).

description
A description of the user. Maximum length of 80 characters.

This option is only valid if Safeguard is running on the system.

dir The home directory of the user. If this field does not specify an existing directory, the specified directory is created with ownership set to the user being created or updated and its primary group.

This option is only valid if Safeguard is running on the system.

NOTES

- The **newusers** command does not create parent directories of the new user's home directory. If the parent directories do not exist, the **newusers** command fails to create the home directory and sends a message to STDERR informing the user of the failure.
- If the **newusers** command fails to create the home directory, it does not halt or return a failure to the calling shell. It will continue to process the batch of new users specified.
- If the home directory of an existing user is changed, **newusers** does not move or copy the content of the old directory to the new location. This must be done manually.

- The **newusers** command is intended to be used in a large system environment where many accounts are updated at the same time. When any operation on any of the entries fails, a corresponding warning message is displayed on STDERR and the command processes the next entry.

CAVEATS

- The input file must be protected because it contains unencrypted passwords.
- The passwords must respect the system password policy.

CONFIGURATION

The following configuration variables in **/etc/login.defs** change the behavior of this tool:

GROUP
: When a partial name is specified as LOGIN, the group name and corresponding group number will be taken from the GROUP variable in the **/etc/default/users** file. The GROUP variable is specified in the format GROUP=groupName,groupNumber. For example, GROUP=TEST,99.

USER
: When a LOGIN name is not specified with the **-A** option, the LOGIN name is taken from the USER variable in the **/etc/default/users** file. The USER variable is specified in the format USER=group- Name.memberName. For example, USER=TEST.USER1.

CREATE_HOME (boolean)
: Indicates if a home directory must be created by default for new users.

  Only valid on systems running Safeguard.

PASS_MAX_DAYS (number)
: The maximum number of days a password may be used. If the password is older than this, a password change is forced. The valid value for this is **-1** or **1** through 32767. If not specified, **-1** is assumed (which disables the restriction).

  Only valid on systems running Safeguard.

UMASK (number)
: The file mode creation mask is initialized to this value. If not specified, the mask is initialized to 022. The **useradd** and **newusers** commands use this mask to set the mode of the home directory when they create files.

  Only valid on systems running Safeguard.

FILES

/etc/default/users
: Default values for account creation.

/etc/skel/
: Directory containing default files.

/etc/login.defs
: Default values for login configuration.

EXIT VALUES

The **newusers** command exits with the following values:

0	success
1	invalid command syntax
2	operation on one or more entries failed

RELATED INFORMATION

login.defs(5), users(5), useradd(8), usermod(8), userdel(8).

NAME

nsupdate - Starts the nonsecure BIND 9 dynamic domain name system (DNS) update utility

SYNOPSIS

/etc/dns923/nsupdate

[**-d**]
[**[-y** *keyname:secret* **] [-k** *keyfile* **]**]
[**-v**]
[**filename**]

DESCRIPTION

nsupdate is used to submit Dynamic DNS Update requests (as defined in RFC2136) to a BIND 9 domain name server. This allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record.

Zones that are under dynamic control via **nsupdate** or a DHCP server should not be edited by hand. Manual edits could conflict with dynamic updates and cause data to be lost.

The resource records that are dynamically added or removed with **nsupdate** have to be in the same zone. Requests are sent to the zone's master server. This is identified by the **MNAME** field of the zone's **SOA** record.

The **-d** flag makes **nsupdate** operate in debug mode. This mode provides tracing information about the update requests that are made and the replies received from the name server.

Transaction signatures can be used to authenticate the Dynamic DNS updates. These use the **TSIG** resource record type described in RFC2845. The signatures rely on a shared secret that should only be known to **nsupdate** and the name server. Currently, the only supported encryption algorithm for **TSIG** is **HMAC-MD5**, which is defined in RFC 2104. Once other algorithms are defined for **TSIG**, applications will need to ensure they select the appropriate algorithm as well as the key when authenticating each other. For instance, suitable **key** and **server** statements would be added to */etc/named.conf* so that the name server can associate the appropriate secret key and algorithm with the IP address of the client application that will use TSIG authentication. **nsupdate** does not read */etc/named.conf*.

nsupdate uses the **-y** or **-k** flag to provide the shared secret needed to generate a **TSIG** record for authenticating Dynamic DNS update requests. These flags are mutually exclusive. With the **-k** flag, **nsupdate** reads the shared secret from the file *keyfile*, whose name is of the form **K***name***.+157.+***random***.private**. For historical reasons, the file *Kname***.+157.+***random***.key** must also be present. When the **-y** flag is used, a signature is generated from *keyname:secret. keyname* is the name of the key, and *secret* is the base64 encoded shared secret. Use of the **-y** flag is discouraged because the shared secret is supplied as a command line argument in clear text. This may be visible in the output from **ps**(1) or in a history file maintained by the user's shell.

By default, **nsupdate** uses UDP to send update requests to the name server. The **-v** flag makes **nsupdate** use a TCP connection. This may be preferable when a batch of update requests is made.

Input Format

nsupdate reads input from *filename* or standard input. Each command is supplied on exactly one line of input. Some commands are for administrative purposes. The others are either update instructions or prerequisite checks on the contents of the zone. These checks set conditions that some name or set of resource records (RRset) either exists or is absent from the zone. These conditions must be met if the entire update request is to succeed. Updates are rejected if the tests for the prerequisite conditions fail.

Every update request consists of zero or more prerequisites and zero or more updates. This allows a suitably authenticated update request to proceed if some specified resource records are

present or missing from the zone. A blank input line (or the **send** command) causes the accumulated commands to be sent as one Dynamic DNS update request to the domain name server.

The command formats and their meaning are as follows:

server *servername* [*port*]
Sends all dynamic update requests to the name server *servername*. When no server statement is provided, **nsupdate** sends updates to the master server of the correct zone. The **MNAME** field of that zone's **SOA** record identifies the master server for that zone. *port* is the port number on *servername* where the dynamic update requests get sent. If no port number is specified, the default DNS port number of 53 is used.

local *address* [*port*]
Sends all dynamic update requests using the local *address*. When no local statement is provided, **nsupdate** sends updates using an address and port chosen by the system. *port* can additionally be used to make requests come from a specific port. If no port number is specified, the system assigns one.

zone *zonename* Specifies that all updates are to be made to the zone *zonename*. If no *zone* statement is provided, **nsupdate** attempts to determine the correct zone to update based on the rest of the input.

key *name secret*
Specifies that all updates are to be **TSIG** signed using the *keyname keysecret* pair. The **key** command overrides any key specified on the command line via **-y** or **-k**.

prereq nxdomain *domain_name*
Requires that no resource record of any type exists with name *domain_name*.

prereq yxdomain *domain_name*
Requires that *domain_name* exists (has as at least one resource record, of any type).

prereq nxrrset *domain_name* [*class*] *type*
Requires that no resource record exists of the specified *type*, *class* and *domain_name*. If *class* is omitted, **IN** (internet) is assumed.

prereq yxrrset *domain_name* [*class*] *type*
This requires that a resource record of the specified *type*, *class* and *domain_name* must exist. If *class* is omitted, **IN** (internet) is assumed.

prereq yxrrset *domain_name* [*class*] *type data...*
The *data* from each set of prerequisites of this form sharing a common *type*, *class*, and *domain-name* are combined to form a set of RRs. This set of RRs must exactly match the set of RRs existing in the zone at the given *type*, *class*, and *domain_name*. The *data* is written in the standard text representation of the resource record's **RDATA**.

update delete *domain_name* [*ttl*] [*class*] [*type* [*data...*]]
Deletes any resource records named *domain_name*. If *type* and *data* is provided, only matching resource records are removed. The internet class is assumed if *class* is not supplied. The *ttl* is ignored, and is only allowed for compatibility.

update add *domain_name ttl* [*class*] *type data...*
Adds a new resource record with the specified *ttl*, *class*, and *data*.

show Displays the current message, containing all of the prerequisites and updates specified since the last send.

send Sends the current message. This is equivalent to entering a blank line.

Lines beginning with a semicolon are comments, and are ignored.

EXAMPLES

The examples below show how **nsupdate** could be used to insert and delete resource records from the **example.com** zone. Notice that the input in each example contains a trailing blank line so that a group of commands are sent as one dynamic update request to the master name server for **example.com**.

1.

```
# nsupdate
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
>
```

Any A records for **oldhost.example.com** are deleted and an A record for **newhost.example.com** with IP address 172.16.1.1 is added. The newly-added record has a one-day TTL (86400 seconds)

2.

```
# nsupdate
> prereq nxdomain nickname.example.com
> update add nickname.example.com 86400 CNAME somehost.example.com
>
```

The prerequisite condition gets the name server to check that there are no resource records of any type for **nickname.example.com**. If there are, the update request fails. If this name does not exist, a **CNAME** for it is added. This ensures that when the **CNAME** is added, it cannot conflict with the long-standing rule in RFC1034 that a name must not exist as any other record type if it exists as a **CNAME**. (The rule has been updated for DNSSEC in RFC2535 to allow **CNAME**s to have **SIG**, **KEY**, and **NXT** records.)

FILES

/etc/resolv.conf
Default resolver configuration file, used to identify default name server.

K*name***.+157.+***random***.key**
Contains base-64 encoding of **HMAC-MD5** key created by the **dnssec-keygen** utility.

K*name***.+157.+***random***.private**
Contains base-64 encoding of **HMAC-MD5** key created by the **dnssec-keygen** utility.

RELATED INFORMATION

Commands:

dnssec_named(8), **named(8)**, **dnssec-keygen(8)**.

Documents: **RFC2136**, **RFC3007**, **RFC2104**, **RFC2845**, **RFC1034**, **RFC2535**.

NAME

Pcleanup - Moves or removes obsolete OSS files

SYNOPSIS

Pcleanup -i | -r { source | target } | -?
Pcleanup -m [-?]

FLAGS

-i Displays a list of all obsolete OSS files, as identified within the remove-list files in the **/etc/install_obsolete** directory.

-m Moves all obsolete files specified within the remove-list files in the **/etc/install_obsolete** directory to the **/etc/install_obsolete** directory but does not delete them.

-r { **source** | **target** }
When **source** is specified, removes all obsolete files from their locations as given in the remove-list files within the **/etc/install_obsolete** directory. When **target** is specified, removes all obsolete files from **/etc/install_obsolete** (intended for use after using **Pcleanup -m**.)

-? Displays usage information for the command.

DESCRIPTION

The **Pcleanup** utility removes or moves all obsolete OSS files after an interim product modification (IPM) or a new release of OSS has been installed on a system.

The **pinstall** utility installs OSS files into the OSS file system but it does not remove files that were made obsolete by the IPM or release. The **Pcleanup** utility can provide the file removal function.

The **Pcleanup** utility uses a set of remove-list files. These remove-list files are included in the **pax** archive files for OSS products. The remove-list files are named **/etc/install_obsolete/PINSTALL_REMOVE_LIST_***filename*, where *filename* is the Guardian file identifier of the **pax** archive file the remove-list file applies to. Usually *filename* contains a product number.

Each remove-list file contains a list of absolute pathnames for the obsolete files in their originally installed locations. Each pathname is on a separate line, terminated by a carriage return.

The remove-list files are not removed by the **Pcleanup** utility.

EXAMPLES

1. The following command removes the obsolete files from the locations given in the remove lists.

   Pcleanup -r source

2. The following command moves obsolete files from the locations given in the remove-list files to the directory **/etc/install_obsolete**.

   Pcleanup -m

3. The following command removes obsolete files from **/etc/install_obsolete**, to which those files were moved using the **-r** flag.

   Pcleanup -r target

NOTES

On systems where the Distributed Software Management/Software Configuration Manager (DSM/SCM) product is used to install HP product files from the ZOSSUTL subvolume and maintain those files in the OSS file system, do not use **Pcleanup** with any option other than **-i** or **-?**. Moving or removing files installed by DSM/SCM can invalidate the DSM/SCM database used for file maintenance.

RELATED INFORMATION

Commands: **pax(1)**, **pinstall(1)**.

STANDARDS CONFORMANCE

The **Pcleanup** utility is an extension to the XPG4 standards.

NAME

pcleanup - See the **Pcleanup(8)** reference page

DESCRIPTION

The first character of the **Pcleanup** command must be uppercase.

NAME

portmap - Maps TCP/IP ports to Remote Procedure Call (RPC) program numbers

SYNOPSIS

[**ADD DEFINE =TCPIPˆHOSTˆFILE, FILE** *hostfile*]
[**ADD DEFINE =TCPIPˆRESOLVERˆNAME, FILE** *resconffile*]
[**ADD DEFINE =TCPIPˆPROCESSˆNAME, FILE** *process*]
[*node*.]**PORTMAP / NAME $ZPM***n***, NOWAIT /** [*param*] [**,** *param*] . . .

FLAGS

hostfile Specifies the Guardian filename of the TCP/IP host definition file. This file contains a list of valid host names, aliases for those names, and the corresponding Internet Protocol (IP) addresses for those hosts.

This value must be specified if a domain name resolver is not running.

This value cannot be specified if the =TCPIPˆRESOLVERˆNAME DEFINE is specified.

For information on creating a TCP/IP host definition file, refer to the *TCP/IP (Parallel Library) Configuration and Management Manual* or the *TCP/IP Configuration and Management Manual*.

resconffile Specifies the Guardian filename of the TCP/IP domain name resolver configuration file. This file contains a list of valid name servers for resolution between host names and Internet Protocol (IP) addresses for those hosts.

This value cannot be specified if the =TCPIPˆHOSTˆFILE DEFINE or the **TCPIP** value for the *param* option is specified.

For information on creating a TCP/IP domain name resolver configuration file, refer to the *TCP/IP (Parallel Library) Configuration and Management Manual* or the *TCP/IP Configuration and Management Manual*.

process Specifies the Guardian process name of the standard process that provides an IP address for the portmapper process.

This value should be specified if the **TCPIP** value for the *param* option is not specified.

The value you specify depends on whether you use conventional TCP/IP or parallel library TCP/IP.

- For conventional TCP/IP, the standard server process is usually named $ZTC*n*, where *n* by convention is the digit used in the input/output process (IOP) name associated with the controller being served. For example, by convention $ZTC0 is the server for the controller that is accessed through $LAN0.
- For parallel library TCP/IP, the standard server process is usually named $ZSAM*n*, where *n* by convention is the digit used in the input/output process (IOP) name associated with the controller being served. For example, by convention $ZSAM0 is the server for the controller that is accessed through $LAN0.

Most HP NonStop system software assumes that the correct default TCP/IP server process name is $ZTC0. If only one TCP/IP server is configured, it should use that process name, so the usual value to use for *process* with the first portmapper process to be started is $ZTC0.

The process name used for the TCP/IP server process is a convention and is not

enforced by HP software. The value used does not need to contain any specific collection of letters or the digit *n*.

node Specifies the Expand node name of the NonStop server node on which to run the portmapper process.

If this value is omitted, the portmapper process is run on the local NonStop server node.

n The value you specify depends on whether you use conventional TCP/IP or parallel library TCP/IP.

- For conventional TCP/IP, by convention, this number matches the number used in the process name of the TCP/IP server process that provides the correct IP address.

  This value is usually 0, corresponding to the assumed default TCP/IP process name of $ZTC0.

- For parallel library TCP/IP, you must specify the fourth and fifth characters of the process name for the TCPSAM process.

The correct number to specify depends on the IP address that the portmapper process should be registered with. Refer to the **NOTES** section of this reference page for a procedure that can help you choose a number based on the IP address configured for a TCP/IP server process.

The process name used for the portmapper process is a convention and is not enforced by HP software. The value used does not need to contain any specific collection of letters or the digit *n*.

param Is one of the following options:

[**BACKUPCPU**] *processor*
Specifies the processor number in the range 0 through 15 of the processor that should run the backup process of the portmapper process pair. The value used should not specify the processor that runs the primary portmapper process of the process pair.

The keyword **BACKUPCPU** can be omitted only if this is the first value for the *param* option specified in the command.

If this option is omitted, the portmapper process does not run as a process pair.

TCPIP *pname* Specifies the Guardian process name of the standard process that provides access to the portmapper process.

If this option is omitted, the value specified in the =TCPIPˆPROCESSˆNAME DEFINE is used.

If this option is specified, the value used must meet the same requirements as those described for =TCPIPˆPROCESSˆNAME DEFINE values.

If both this option and the DEFINE are omitted, the default name for the TCP/IP server process is $ZTC0.

COLLECTOR *cname*

Specifies the Guardian process name of the Event Management Service (EMS) server process that collects event messages from the portmapper process.

If this option is omitted, the primary EMS collector ($0) is used.

In addition to the NOWAIT option, other TACL RUN command options (such as CPU and PRI) are also valid in this command. Refer to the *TACL Reference Manual* for a description of the RUN command.

DESCRIPTION

The portmapper process is a Guardian server process that converts host port numbers to Remote Procedure Call (RPC) program numbers. The portmapper process must be running for successful execution of server processes that make remote procedure calls, such as the Network File System (NFS).

When an RPC server process is started, that process tells the portmapper process what host port number it is listening to and what RPC program numbers it is prepared to serve. When a client process wants to make an RPC call to a given program number, that client first contacts the portmapper process on the system that should receive the call to determine the host port number to which RPC packets should be sent.

EXAMPLES

1. The following example starts the portmapper process as a process pair on the local node. This example assumes that the user's PMSEARCHLIST variable does not include the ZNFS subvolume and that the =ZTCPIPˆPROGRAMˆNAME DEFINE has not been declared:

```
VOLUME $SYSTEM.ZNFS
PORTMAP / NAME $ZPM0, CPU 1, NOWAIT / BACKUPCPU 0, TCPIP $ZTC0
```

FILES

$SYSTEM.ZNFS.PORTMAP

The Guardian filename of the program file for the portmapper program.

$SYSTEM.ZTCPIP.HOSTS

The Guardian filename of the default TCP/IP host definition file for the NonStop server node. This file must be created by a site administrator.

$SYSTEM.ZTCPIP.RESCONF

The Guardian filename of the default TCP/IP domain name resolver configuration file for the NonStop server node. This file must be created by a site administrator.

NOTES

The Guardian portmapper program corresponds to the program that is located at **/usr/etc/portmap** on some UNIX systems.

On UNIX systems, standard RPC servers are started by the **inetd** utility, so the portmapper process must be started before **inetd** is started. On an HP NonStop server, RPC servers are not started by **inetd**. However, the portmapper process must still be started before an RPC server is started.

Binding to a Single Subnet

On systems running H06.24 and later H-series RVUs or J06.13 and later J-series RVUs, you can bind a portmapper process to an IP address by setting the TACL PORTMAPˆBINDˆIP parameter before you start the portmapper process. The IP address must be an IPv4 address in dotted-decimal

format.

For example, to bind the portmapper to IP address 192.168.10.10, enter this TACL command before you start the portmapper process:

PARAM PORTMAPˆBINDˆIP 192.168.10.10

Determining the portmapper Process for a TCP/IP Process

For conventional TCP/IP, when more than one TCP/IP server is running, the portmapper process for a specific IP process can be determined as follows:

- By convention, each portmapper process name contains the same number as the corresponding TCP/IP server process name.
- Each TCP/IP server is associated with a host name and an IP address (called a Host ID in Subsystem Control Facility [SCF] displays). The corresponding portmapper process returns information for that IP address.
- The IP address for each TCP/IP server process (and, if naming conventions were followed, for its corresponding portmapper process) can be determined by entering the following SCF command:

  ```
  INFO PROCESS $ZTC*, DETAIL
  ```

For parallel library TCP/IP, only one portmapper process can be started for the entire TCP/IP subsystem, regardless of how many TCPSAM processes are running.

- By convention, each portmapper process name contains the fourth and fifth characters of the TCPSAM process name.
- Even though only one portmapper process can be started, it serves all the IP addresses associated with the parallel library TCP/IP subsystem.

When starting a copy of the portmapper process, the digit to use in its process name can be determined by the same method:

- Use an SCF command to determine the IP address for each TCP/IP server process and the digit from the correct corresponding TCP/IP process name.

DIAGNOSTICS

In addition to the following messages, the portmapper program generates EMS error messages for problems encountered during socket or file input or output.

`portmap: backup CPU` *nn* `is not valid`
: The specified backup processor does not exist, or it is the same as the processor used for the primary portmapper process.

`portmap:` *name* `is not a valid tcpip process`
: The process specified by *name* is the wrong process type to function as a valid TCP/IP server process. Specify a valid process name for a process of device type 48, device subtype 0.

`portmap:` *name* `is not a valid collector process`
: The process specified by *name* is not an existing EMS collector process.

`portmap: process must be named`
: The portmapper program was started as an unnamed process. The portmapper program must run as a named process.

`portmap: process` *name* `does not exist`
: The process name specified by *name* for the EMS collector or TCP/IP server process does not identify a running process.

RELATED INFORMATION

Commands: **inetd(8)**, **rpcinfo(8)**.

STANDARDS CONFORMANCE

This command is an HP extension to the XPG4 Version 2 specification.

The HP implementation of the portmapper program depends upon an RPC implementation that conforms to the Defense Data Network (DDN) Request for Comments (RFC) 1057.

NAME

rexecd - Starts the remote execution server

SYNOPSIS

rexecd

DESCRIPTION

rexecd is the local server for the **rexec( )** function available on remote UNIX systems. The server provides remote execution facilities with authentication based on user names and passwords.

rexecd listens for service requests at the port indicated in the **exec** service specification of the **/etc/services** file. When a service request is received, the following steps occur:

1. The server reads characters from the socket up to a **NUL** byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is nonzero, it is interpreted as the port number of a secondary stream to be used for the standard error file. A second connection is then created to the specified port on the client's machine.
3. A **NUL**-terminated user name of at most 80 characters is retrieved on the initial socket.
4. A **NUL**-terminated, unencrypted password of at most 80 characters is retrieved on the initial socket.
5. A **NUL**-terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
6. **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fails, the connection is aborted with a diagnostic message returned.
7. A **NUL** byte is returned on the initial socket, and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

FILES

/etc/inetd.conf The default file containing the configuration and startup information for the **rexecd** process.

NOTES

To start the **rexecd** process, add the following entry to the configuration file for the **inetd** process:

```
exec   stream  tcp     nowait  super.super    /bin/rexecd
```

(The default configuration file for **inetd** is **/etc/inetd.conf**.) Then refresh the **inetd** configuration by restarting **inetd** or by sending it a **SIGHUP** signal.

DIAGNOSTICS

Except for the last message listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (zero is returned in step 7, above, upon successful completion of all the steps prior to the command execution).

username too long
The name is longer than 80 characters.

password too long
The password is longer than 80 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect.
The user name and password could not be validated.

No remote directory.
The **chdir( )** function call to the home directory failed.

Try again. A **fork( )** function call by the server failed.

*shellname***: ...** The user's login shell, identified as *shellname*, could not be started. This message is returned on the connection associated with the standard error file, and is not preceded by a flag byte.

RELATED INFORMATION

Commands: **inetd(8)**.

Files: **services(4)**.

STANDARDS CONFORMANCE

This command is an extension to the XPG4 version 2 specification.

NAME

rndc - Starts the nonsecure BIND 9 Internet domain name server control utility

SYNOPSIS

/etc/dns923/rndc
[**-c** *config_file*]
[**-k** *key_file*]
[**-s** *server*]
[**-p** *port*]
[**-V**]
[**-y** *key_id*]
command

FLAGS

-c *config_file* Use *config_file* as the configuration file instead of the default, */etc/rndc.conf*.

-k *key_file* Use *key_file* as the key file instead of the default, */etc/rndc.key*. The key in */etc/rndc.key* will be used to authenticate commands sent to the server if the *config_file* does not exist.

-s *server* *server* is the name or address of the server which matches a server statement in the configuration file for **rndc**. If no server is supplied on the command line, the host named by the **default-server** clause in the option statement of the configuration file is used.

-p *port* Send commands to TCP port *port* instead of BIND 9's default control channel port, 953.

-V Enable verbose logging.

-y *keyid* Use the key *keyid* from the configuration file. *keyid* must be known by **named** with the same **algorithm** and **secret** string in order for control message validation to succeed. If no *keyid* is specified, **rndc** first looks for a key clause in the server statement of the server being used, or if no server statement is present for that host, it then looks for the **default-key** clause of the options statement. Note that the configuration file contains shared secrets which are used to send authenticated control commands to name servers. It should therefore not have general read or write access.

Operands

command For the complete set of commands supported by **rndc**, see the *BIND 9 Administrator Reference Manual* or run **rndc** without arguments to see its help message.

DESCRIPTION

rndc controls the operation of a BIND 9 domain name server. If **rndc** is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

rndc communicates with the name server over a TCP connection, sending commands authenticated with digital signatures. In the nonsecure version of **rndc** and **named**, the only supported authentication algorithm is **HMAC-MD5**, which uses a shared secret on each end of the connection. This provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a key_id known to the server.

rndc reads a configuration file to determine how to contact the name server and decide what algorithm and key it should use.

NOTES

rndc does not yet support all the commands of the BIND 8 **ndc** utility.

There is currently no way to provide the shared secret for a **key_id** without using the configuration file.

Several error messages could be clearer.

RELATED INFORMATION

Commands: **dnssec_named(8)**, **dnssec_nsupdate(8)**, **named(8)**, **nsupdate(8)**.

Files: **named.conf(4)**.

Documents: *BIND 9 Administrator Reference Manual*.

NAME

rpcinfo - Reports or changes Remote Procedure Call (RPC) information

SYNOPSIS

[**ADD DEFINE =ZRPCˆRPCˆFILE, FILE** *rpcfile*]
[**ADD DEFINE =TCPIPˆHOSTˆFILE, FILE** *hostfile*]
[**ADD DEFINE =TCPIPˆRESOLVERˆNAME, FILE** *resconffile*]
[**ADD DEFINE =TCPIPˆPROCESSˆNAME, FILE** *process*]

[*node*.]**RPCINFO -p** [**-u** | **-t**] [*target*]

[*node*.]**RPCINFO** [**-n** *portnum*] { **-u** | **-t** }
host program [*version*]

[*node*.]**RPCINFO -b** *program version*

[*node*.]**RPCINFO -d** *program version*

FLAGS

rpcfile Specifies the Guardian filename of the RPC program definition file. This file contains a list of valid RPC program numbers and program names.

This value should be specified if the default RPC program definition file for the HP NonStop server node is not used or was not created. If no RPC program definition file can be found, the RPCINFO output will not contain a program name.

The RPC program definition file is an EDIT file and can be created using any Guardian text editor. Each line in the file is either a comment line beginning with a pound sign (#) or an entry for one program. Program entries contain the program name to be reported by RPCINFO, the program number, and possible aliases for the program name. For example:

```
#
#       rpc   1.16   89/12/27
#
portmapper      100000  portmap sunrpc
```

The fields of each program-entry line are separated by blanks.

hostfile Specifies the Guardian filename of the TCP/IP host definition file. This file contains a list of valid host names, aliases for those names, and the corresponding Internet Protocol (IP) addresses for those hosts.

This value must be specified if a domain name resolver is not running.

For information on creating a TCP/IP host definition file, refer to the *TCP/IP (Parallel Library) Configuration and Management Manual* or the *TCP/IP Configuration and Management Manual*.

resconffile Specifies the Guardian filename of the TCP/IP domain name resolver configuration file. This file contains a list of valid name servers for resolution between host names and Internet Protocol (IP) addresses for those hosts.

This value cannot be specified if the =TCPIPˆHOSTˆFILE

DEFINE is specified.

For information on creating a TCP/IP domain name resolver configuration file, refer to the *TCP/IP (Parallel Library) Configuration and Management Manual* or the *TCP/IP Configuration and Management Manual*.

process Specifies the Guardian process name of the standard process that provides an IP address for the portmapper process.

The value you specify depends on whether you use conventional TCP/IP or parallel library TCP/IP.

- For conventional TCP/IP, the standard server process is normally named $ZTC*n*, where *n* by convention is the digit used in the input/output process (IOP) name associated with the controller being served. For example, by convention $ZTC0 is the server for the controller that is accessed through $LAN0.
- For parallel library TCP/IP, the standard server process is usually named $ZSAM*n*, where *n* by convention is the digit used in the input/output process (IOP) name associated with the controller being served. For example, by convention $ZSAM0 is the server for the controller that is accessed through $LAN0.

Most HP NonStop system software assumes that the correct default TCP/IP server name is $ZTC0. If only one TCP/IP server is configured, it should use that process name, so the usual value to use for *process* with RPCINFO is $ZTC0.

The process name used for the TCP/IP server process is a convention and is not enforced by HP software. The value used does not need to contain any specific collection of letters or the digit *n*.

node Specifies the Expand node name of the HP NonStop server node on which to run the RPCINFO utility.

If this value is omitted, the RPCINFO utility is run on the local node.

-p [-u | **-t**] [*target*]

Requests that the portmapper process on the host specified by *target* list all registered RPC programs.

The *target* value can be specified either as the name or the dotted-decimal form of the IP address of a host within a reachable TCP/IP network.

The *target* value is case-sensitive; that is, the specified name must match the case of the host name within the TCP/IP host definition file. For example, specifying **FORTY.TANDEM.COM** does not match an entry of **forty.tandem.com**.

If no value is specified for the *target* argument, information is reported for the HP NonStop server node specified or implied by the value of *node*.

The associated flags specify the TCP/IP client protocol used to contact the portmapper process, as follows:

-t Use the TCP protocol.

-u Use the UDP protocol.

If neither flag is specified, TCP protocol is used.

-n *portnum* { **-u** | **-t** }
Makes an RPC call using the port identified within the TCP/IP host definition file by *portnum* and reports whether a response is received. The RPC call is made to procedure 0 of a targeted RPC program that is identified by the other specified options.

If this flag is not specified, the RPC call is made to the program using the first port specified for the targeted program in the RPC program definition file.

The associated flags specify the TCP/IP client protocol used to contact the targeted program, as follows:

-t Use the TCP protocol.

-u Use the UDP protocol.

One of these flags must be specified, as indicated by the { } notation.

host Specifies the host on which the targeted program runs.

The *host* value can be specified either as the name or the dotted-decimal form of the IP address of a host within a reachable TCP/IP network.

Note that the *host* value is not the same as the *node* value.

The *host* value is case-sensitive; that is, the specified name must match the case of the host name within the TCP/IP host definition file. For example, specifying **FORTY.TANDEM.COM** does not match an entry of **forty.tandem.com**.

program Specifies the targeted program.

The *program* value can be specified either as the name of a program within the RPC program definition file or as the program number of a program defined in that file.

The *program* value is case-sensitive; that is, the specified name must match the case of the program name within the RPC program definition file. For example, specifying **PORTMAPPER** does not match an entry of **portmapper**.

version Specifies the version of the targeted program as defined within the RPC program definition file.

If no value is specified for *version* in those cases where *version* is not required, RPCINFO attempts to identify all registered versions of the targeted program. RPCINFO first calls version 0 (zero), which is assumed not to exist. If version 0 exists, RPCINFO calls a very high version number instead.

-b Makes an RPC broadcast to procedure 0 of a targeted RPC program that is identified by the other specified options and reports whether a response is received.

-d Deletes the registration of a targeted RPC program that is identified by the other specified options. This flag can be specified only by the super ID (255,255 in the Guardian environment, 65535 in the OSS environment).

DESCRIPTION

The RPCINFO utility is a Guardian program that reports or modifies the status of RPC services accessible from the local HP NonStop server node.

The program names reported do not have a one-to-one relationship to a specific process. When multiple server processes are running on the same HP NonStop server node, the report includes only one entry for each version of the server program and each TCP/IP client protocol supported by the program.

EXAMPLES

1. The following example displays the names and TCP/IP or UDP port numbers of all RPC services registered on the local node:

   ```
   ADD DEFINE =ZRPC^RPC^FILE, FILE $SYSTEM.ZRPC.RPC
   RPCINFO -p
   ```

   The output contains one line for each version of an RPC program and for each client protocol supported by the program. The program line shows the RPC program number, the program version, the client protocol used by the program, the port number, and the program name. For example:

   ```
   program  vers   proto   port
   100000      2     tcp    111     portmapper
   ```

2. The following example displays the names and TCP/IP or UDP port numbers of all RPC services registered on the remote node **forty.tandem.com**:

   ```
   ADD DEFINE =ZRPC^RPC^FILE, FILE $SYSTEM.ZRPC.RPC
   RPCINFO -p forty.tandem.com
   ```

3. The following example displays the names of all HP NonStop server nodes that are running version 2 of the HP portmapper program, as configured in the default RPC program definition file:

   ```
   RPCINFO -b portmapper 2
   ```

   In this example, the output contains the IP address of each node contacted during the search for a node that runs version 2 of the portmapper program and the corresponding name of the host. The listing looks something like the following:

   ```
   111.222.333.444 forty.tandem.com
   111.222.555.666 loc247.tandem.com
   program 100000 version 2 ready and waiting
   ```

4. The following example deletes the registration for version 1 of the RPC program identified by the program number 1073741824:

   ```
   RPCINFO -d 1073741824 1
   ```

5. The following example shows typical RPC program definition file information for a node that has the Network File System (NFS) running and the file $SYSTEM.ZRPC.RPC defined:

   ```
   RPCINFO -p

   program  vers  proto    port
   100000      2    udp     111    portmapper
   100000      2    tcp     111    portmapper
   100003      2    udp    2049    nfs
   100005      1    udp     740    mountd
   150001      1    udp     808    pcnfsd
   150001      2    udp     808    pcnfsd
   150001      1    tcp     811    pcnfsd
   150001      2    tcp     811    pcnfsd
   ```

   In this example, the local HP NonStop server node has eight registered RPC programs. These include:

   - The portmapper program, which has one entry for each supported client protocol.

     The portmapper process on HP NonStop servers is called $ZPM*nn* within the Guardian environment. The portmapper process appears as **portmapper** when viewed from a UNIX system. Refer to the **portmap(8)** reference page for additional information about the portmapper program on HP NonStop servers.

     The portmapper program shown in the report supports both UDP and TCP/IP client protocols.

   - The NFS server headpin process, which indicates that the NFS subsystem is running.

     The headpin process on HP NonStop servers is unnamed within the Guardian environment but appears as **nfs** when viewed from a UNIX system.

     The headpin process shown in the report supports the UDP client protocol.

   - The mounting process for NFS filesets.

     The mounting process on HP NonStop servers is called the NFS Manager with the process name $ZNFS within the Guardian environment but appears as **mountd** when viewed from a UNIX system.

     The mounting process shown in the report supports the UDP client protocol.

   - One copy of the NFS printing and authentication server for each version of the server and for each supported client protocol.

     The NFS printing and authentication server on HP NonStop servers is called $PCD*n* within the Guardian environment but appears as **pcnfsd** when viewed from a UNIX system. (This

server is used only by TCP/IP nodes that do not run the OSS environment or a UNIX operating system.)

6. The following example uses the TCP client protocol to request information about the program identified as **portmapper** on the host **forty.tandem.com**:

```
RPCINFO -t forty.tandem.com portmapper
```

In this example, the output shows the routing used for the request, in the form of the IP address and host names for each node along the route, as follows:

```
111.222.333.444 aardvark.tandem.com
111.222.555.666 (unknown)
program 100000 version 2 ready and waiting
```

FILES

$SYSTEM.SYSTEM.RPCINFO
The Guardian filename of the program file for the RPCINFO utility.

$SYSTEM.ZRPC.RPC
The Guardian filename of the default RPC program definition file for the HP NonStop server node. This file must be created by a site administrator.

$SYSTEM.ZTCPIP.HOSTS
The Guardian filename of the default TCP/IP host definition file for the HP NonStop server node. This file must be created by a site administrator.

$SYSTEM.ZTCPIP.RESCONF
The Guardian filename of the default TCP/IP domain name resolver configuration file for the HP NonStop server node. This file must be created by a site administrator.

NOTES

The information reported for an RPC program on a node that is not an HP NonStop server might vary slightly from the description in this reference page.

The Guardian RPCINFO program corresponds to the program that is located at **/usr/etc/rpcinfo** on some UNIX systems.

DIAGNOSTICS

In response to user errors, RPCINFO terminates abnormally and issues the following diagnostic messages:

```
Usage: rpcinfo -p [ -u | -t ]  [ host ]
            rpcinfo [ -n portnum ] -u host program [ version
            ]
            rpcinfo [ -n portnum ] -t host program [ version
            ]
            rpcinfo -b program version
            rpcinfo -d program version
```

Indicates a syntax error or specification of an unrecognized option.

`rpcinfo:` *aaaaaaaa* `is unknown service`
Indicates that the value specified by *aaaaaaaa* does not correspond to a registered RPC program on the host to which you made the request.

Additionally, RPCINFO returns messages related to RPC call error conditions. For example, the following message is returned if an RPC call times out during a call to the portmapper process:

```
rpcinfo: RPC: Portmapper failure - RPC: Timed out
program 100000 is not available
```

RELATED INFORMATION

Commands: **portmap(8)**.

STANDARDS CONFORMANCE

This command is an HP extension to the XPG4 Version 2 specification.

The HP implementation of RPCINFO depends upon an RPC implementation that conforms to the Defense Data Network (DDN) Request for Comments (RFC) 1057.

NAME

rshd - Starts the remote shell (demon) server process

SYNOPSIS

rshd

DESCRIPTION

The **rshd** process is the server process for the **rsh** utility.

NOTES

To start the **rshd** process, you need to add the following entry to the configuration file for the **inetd** process:

shell stream tcp nowait root /bin/rshd

(The default configuration file for **inetd** is **/etc/inetd.conf**.) Then refresh the **inetd** configuration by restarting **inetd** or by sending it a **SIGHUP** signal.

FILES

/etc/inetd.conf	The default file containing the configuration and startup information for the **rshd** process.

RELATED INFORMATION

Commands: **inetd(8)**, **rsh(1)**.

STANDARDS CONFORMANCE

This command is an HP extension to the XPG4 Version 2 specification.

NAME

useradd - Creates a new user or alias, or updates default new user or alias information.

SYNOPSIS

useradd [*OPTION*] *LOGIN*
useradd -*A ALIAS_NAME* [*OPTION*] [*LOGIN*]
useradd -*D*
useradd -*D* [*OPTION*]

FLAGS

The following flags are only valid on systems running Safeguard. If these flags are used on a system not running Safeguard, a diagnostic message is reported on STDERR and the command exits with the exit status of 15.

-A *ALIAS_NAME*
Add user alias with name *ALIAS_NAME*.

-c *COMMENT*
Any text string of maximum length of 80 characters. It is generally a short description of the login, and is currently used as the field for the user's full name.

-d *HOME_DIR*
The new user is created using *HOME_DIR* as the value for the user's login directory. The default is to append the LOGIN name (in the form 'groupName.memberName', specified or derived) to *BASE_DIR* and use that as the login directory name. The directory *HOME_DIR* need not exist, but is not created if it is missing.

-e *EXPIRE_DATE*
The date on which the user account will expire. The date is specified in the format "YYYY-MM-DD". If not specified, *useradd* uses the default expiry date specified by the EXPIRE variable in **/etc/default/users**, or an empty string (no expiry) by default.

-f *INACTIVE*
The number of days after a password expires until the account is expired. This value ranges from 0 through 32767. A value of 0 disables the account as soon as the password has expired. If not specified, **useradd** uses the default inactivity period specified by the INACTIVE variable in the **/etc/default/users** file, or none. If none, the PASSWORD-EXPIRY-GRACE attribute specified in the Safeguard configuration record is used.

-k *SKEL_DIR*
The skeleton directory, which contains files and directories to be copied in the user's home directory, when the home directory is created by **useradd**. This option is only valid if the **-m** option is specified. If this option is not set, the skeleton directory is defined by the SKEL variable in **/etc/default/users** or, by default, **/etc/skel**.

-K *KEY=VALUE*
Overrides **/etc/login.defs** defaults (UMASK, PASS_MAX_DAYS and others). Example: **-K** UMASK=023 can be used to specify the umask value. You can specify multiple **-K** options. For example: **-K** UMASK=023 **-K** PASS_MAX_DAYS=90.

-m Create the user's home directory if it does not exist. The files and directories contained in the skeleton directory (which can be defined with the **-k** option) are copied to the home directory. By default, no home directories are created.

-M Do not create the user's home directory, even if the system wide setting from **/etc/login.defs** (CREATE_HOME) is set to yes.

-p *PASSWORD*
The login password. By default the value is set to null (no password is required to log on).

The following flags are always valid.

-b *BASE_DIR*
The default base directory for the system if **-d** *HOME_DIR* is not specified. *BASE_DIR* is concatenated with the account name, that is, 'groupName.memberName' to define the home directory. If the **-m** option is not used, *BASE_DIR* must exist. If this option is not specified, *useradd* uses the base directory specified by the HOME variable in **/etc/default/users**, or **/home** by default. If Safeguard is not running on the system, this option is not applicable. In these cases, a diagnostic message is reported on STDERR and the command exits with the exit status of 15.

-D Print or change default **useradd** configuration. See the "Changing the default values" subsection.

-h Display this help and exit.

DESCRIPTION

When invoked without the **-D** option, the **useradd** command creates a new user account using the values specified on the command line and the default values from the system. Depending on command-line options, the **useradd** command updates system files and might also create the new user's home directory and copy initial files.

You can specify the LOGIN name in a fully qualified form 'groupName.memberName:groupNum,memberNum' (no white spaces); or a partial name containing only the memberName. When a partial name is used, the group name and corresponding group number will be taken from the GROUP variable in the **/etc/default/users** file. memberNum will be the first unused number starting from 254 in the descending order, in the range of 254 - 0. *memberNum* 255 is reserved for SUPER *memberName*.

In the **useradd -A** alias [OPTION] [LOGIN] form, a user alias is added. The LOGIN in this case, specified or derived as described previously, must be an existing user. If [LOGIN] is not specified, the USER variable in the **/etc/default/users** file is used.

Changing the Default Values

When invoked with only the **-D** option, **useradd** displays the current default values. When invoked with **-D** plus other options, **useradd** will update the default values for the specified options. Valid default-changing options are:

-b *BASE_DIR*
The path prefix for a new user's home directory. If the **-d** option is not used when creating a new account, the user's name is affixed to the end of *BASE_DIR* to form the new user's home directory name. This option sets the HOME variable in **/etc/default/users**.

-e *EXPIRE_DATE*
The date on which the user account will expire. This option sets the EXPIRE variable in **/etc/default/users**.

Only valid on systems running Safeguard.

-f *INACTIVE*
The number of days after a password has expired before the account expires. This option sets the INACTIVE variable in **/etc/default/users**.

Only valid on systems running Safeguard.

NOTES

- You can only use partial LOGIN when the Safeguard subsystem is running.
- The **-p** option is not recommended because the password will be visible by users listing the processes.

Only valid on systems running Safeguard

- The password must respect the system password policy.
- The system administrator is responsible for placing the default user files in the **/etc/skel/** directory (or any other skeleton directory specified in **/etc/default/users** or on the command line).
- The **-D** option changes the default values specified in **/etc/default/users** file. If this command is executed by someone other than super.super, such as SOA or other delegates, super.super must verify they have write access to the **/etc/default/users** file.

CAVEATS

- Usernames can be up to 17 characters in length.
- Alias names can be 32 characters in length.
- Group number and member number can be 0 - 255.

CONFIGURATION

The following configuration variables in **/etc/login.defs** change the behavior of this tool:

GROUP
: When a partial name is specified as LOGIN, the group name and corresponding group number will be taken from the GROUP variable in the **/etc/default/users** file. The GROUP variable is specified in the format GROUP=groupName,groupNumber. For example, GROUP=TEST,99.

USER
: When a LOGIN name is not specified with the **-A** option, the LOGIN name is taken from the USER variable in the **/etc/default/users** file. The USER variable is specified in the format USER=group- Name.memberName. For example, USER=TEST.USER1.

CREATE_HOME (boolean)
: Indicates if a home directory must be created by default for new users.

  Only valid on systems running Safeguard.

PASS_MAX_DAYS (number)
: The maximum number of days a password may be used. If the password is older than this, a password change is forced. The valid value for this is **-1** or **1** through 32767. If not specified, **-1** is assumed (which disables the restriction).

  Only valid on systems running Safeguard.

UMASK (number)
: The file mode creation mask is initialized to this value. If not specified, the mask is initialized to 022. The **useradd** and **newusers** commands use this mask to set the mode of the home directory when they create files.

  Only valid on systems running Safeguard.

FILES

/etc/default/users
: Default values for account creation.

/etc/skel/
: Directory containing default files.

/etc/login.defs
Default values for login configuration.

EXIT VALUES

The **useradd** command exits with the following values:

0 success

1 cannot set password

2 invalid command syntax

3 invalid argument to option

4 specified member (name or number) already in use

6 specified group (name or number) does not exist

7 incorrect LOGIN

9 LOGIN already in use

10 alias already in use

11 LOGIN does not exist

12 cannot create group

13 insufficient privilege

14 cannot create home directory

15 Safeguard is not available, invalid option

RELATED INFORMATION

login.defs(5), users(5), newusers(8), userdel(8), usermod(8).

NAME

userdel - Deletes a user account and removes related files.

SYNOPSIS

userdel [*OPTION*] *LOGIN*
userdel -*A ALIAS_NAME* [*OPTION*]

FLAGS

-A *ALIAS_NAME*
Delete user alias with name *ALIAS_NAME*.

This option is only valid if Safeguard is running on the system. If Safeguard is not running, a diagnostic message is reported on STDERR and the command exits with the exit status of 15.

-f This option forces the removal of the user account, even if the user is still logged in. It also forces **userdel** to remove the user's home directory, even if another user uses the same home directory. **Warning**: This option can leave your system in an inconsistent state.

-h Display help message and exit.

-r Files in the user's home directory will be removed along with the home directory itself. Search for files located in other file systems and delete manually.

DESCRIPTION

The **userdel** command modifies the system account files, deleting all entries that refer to the user name LOGIN. The named user must exist.

You can specify the LOGIN name in a fully qualified form 'groupName.memberName:groupNum,memberNum' (no white spaces); or a partial name containing only the memberName. When a partial name is used, the group name and corresponding group number will be taken from the GROUP variable in the **/etc/default/users** file.

In the **userdel -A** alias [OPTION] form, the specified alias is deleted.

NOTES

Manually check all file systems to ensure that no files remain owned by this user.

FILES

/etc/default/users
Default values for account creation.

EXIT VALUES

The **userdel** command exits with the following values:

0 success

2 invalid command syntax

6 specified user or alias does not exist

7 specified user is associated with one or more aliases

8 user or alias currently logged in

11 insufficient privilege

12 cannot remove home directory

15 Safeguard is not available, invalid option

RELATED INFORMATION

login.defs(5), users(5), newusers(8), useradd(8), usermod(8).

NAME

usermod - Modifies a user account.

SYNOPSIS

usermod [*OPTION*] *LOGIN*
usermod *-A ALIAS_NAME* [*OPTION*]

FLAGS

The following flags are only valid on systems running Safeguard. If these flags are used on a system not running Safeguard, a diagnostic message is reported on STDERR and the command exits with the exit status of 15.

-A *ALIAS_NAME*
Modify user alias with name *ALIAS_NAME*.

-c *COMMENT*
Any text string of 80 characters maximum length. Generally a short description of the login. Currently used as the field for the user's full name.

-d *HOME_DIR*
The user's new login directory. If the **-m** option is given, the contents of the current home directory are moved to the new home directory, which is created if it does not already exist.

-e *EXPIRE_DATE*
The date on which the user account will expire. The date is specified in the format "YYYY-MM-DD".

-f *INACTIVE*
The number of days after a password expires until the account is expired. This value ranges from 0 through 32767. A value of 0 disables the account when the password expires, unless the PASSWORD-EXPIRY-GRACE attribute in the Safeguard configuration record specifies a different value.

-L Locks or freezes the user account and temporarily suspends the users ability to log on to a system. You cannot use this option with **-p** or **-U**.

-m Move the content of the user's home directory to the new location. This option is only valid in combination with the **-d** option.

-p *PASSWORD*
The password.

-U Unlock or thaw a user's account. You cannot use this option with **-p** or **-L**.

The following flag is always valid.

-h Display this help and exit.

DESCRIPTION

The **usermod** command modifies the system account files to reflect the changes that are specified on the command line.

You can specify the LOGIN name in a fully qualified form 'groupName.memberName:groupNum,memberNum' (no white spaces); or a partial name containing only the *memberName*. When a partial name is used, the group name and corresponding group number is taken from the GROUP variable in the **/etc/default/users** file.

In the **usermod -A** *ALIAS_NAME* [OPTION] form, a user alias is modified.

NOTES

- The **-p** option is not recommended because the password will be visible by users listing the processes. This option is only valid on systems running Safeguard.
- The password must respect the system password policy.

CAVEATS

- You must verify the named user is not executing any processes when this command is being executed if the user's name, or the user's home directory is being changed.
- You must change the owner of any crontab files or at jobs manually.

CONFIGURATION

The following configuration variables in **/etc/default/users** change the behavior of this tool:

GROUP

When a partial name is specified as LOGIN, the group name and corresponding group number will be taken from the GROUP variable in the **/etc/default/users** file. The GROUP variable is specified in the format GROUP=groupName,groupNumber. For example, GROUP=TEST,99.

FILES:

/etc/default/users

Default values for account creation.

EXIT VALUES

The **usermod** command exits with the following values:

0	success
1	cannot set password
2	invalid command syntax
3	invalid argument to option
4	LOGIN does not exist
5	alias does not exist
6	specified group (name or number) does not exist
8	user or alias currently logged in
11	insufficient privilege
12	cannot move home directory contents to new location
15	Safeguard is not available, invalid option

RELATED INFORMATION

login.defs(5), users(5), newusers(8), useradd(8), userdel(8).

Permuted Index

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z