NAME

man – Displays or prints online man page information

SYNOPSIS

man [-l] [-M *path*] [-q] [-t] [-s *section*] [-T *macro*] *title* ...
man [-M *path*] -f *filename* ...
man [-M *path*] -i *keyword* ...
man [-M *path*] -k *pattern* ...

IMPLEMENTATION

UNICOS systems

STANDARDS

POSIX, XPG4
BSD extensions (-f, -M, -t, and -T)
CRI extensions (-i and -q)

DESCRIPTION

The man utility displays online man pages that you select by title, or it can display entries about a particular topic by using the -f, -i, and -k options.

The man utility accepts the following options:

-l — Lists all pages found matching the *title*(s) given the search path. The search path is a colon-separated list of directories in which man page subdirectories may be found.

-M *path* — Changes the directory path that man searches for man page information (default is /usr/man). The search path is a colon-separated list of directories in which man page subdirectories may be found (for example, /usr/local:/usr/man). When you use this option with the -f, -i, or -k option, the -M option must appear first.

-q — Displays the quick-reference entry for the specified command manual entry. Currently, quick-reference entries are provided only for man pages in sections 1, 2, 3, and 8.

-s *section* ... — Specifies sections of the manual for man to search. The directories searched for *title*(s) is limited to those specified by *section*(s). *section* is an Arabic number or one of the words 'new', 'local', 'old', 'public', or 'quick'. A section may be followed by a one-letter classified. To specify multiple sections, separate each section with a comma.

-t — Processes the specified man page entries through troff(1), with the output piped to lpr -n. Use of the -t option on a system where the unformatted source versions of the man pages are not available causes an error.

`-T` *macro* Specifies a different macro package to use when processing `nroff` source versions of man pages. By default, the Cray Research man page macros (`-muc`), defined in `/usr/lib/tmac/tmac.uc`, are used. The format of the macro option argument can be either a full path name or the actual option to be used in the `nroff` or `troff` command line (for example, `-man` and `/usr/lib/tmac/tmac.an` are both valid and are equivalent).

`-f` *filename* ... Displays from the `whatis` file (table of contents), one-line summaries that contain any of the specified file names. The leading path name components are stripped from each file name before the search is performed.

`-i` *keyword* ... Displays from the `index` file, formatted, three-line summaries that contain any of the specified keywords.

`-k` *pattern* ... Displays from the `whatis` file (table of contents), one-line summaries that contain the specified pattern. Grep-style pattern matching characters may be used in *pattern* (see `grep`(1)).

title ... Pipes output through `cat`(1).

section Specifies the section number of manual to be searched.

When you specify *section* and *title* (*section* being an Arabic number, such as `3`, or one of the words `new`, `local`, `old`, or `public`), `man` searches that section of the manual for the specified title. A section number may be followed by a one-letter classifier (for example, `1b`, indicating a utility originating from BSD source code in section 1). If you omit *section*, `man` will search all sections of the manual.

If the standard output is not a terminal, or if you specify the `-` flag, `man` will pipe its output through `cat`(1). Otherwise, `man` pipes its output through `more -s` to handle paging and underlining on the screen (see `more`(1)).

If both the `nroff`(1) source code and preformatted pages are available, the one with the most recent modification time will be displayed. If the source is used and if the user has appropriate permission, the newly formatted page will be installed in the directory of preformatted pages so that it will not have to be formatted the next time it is accessed.

NOTES

The `man` utility displays bold text by overstriking the normal font. If the bold font is specified, it must be the same height and width as the normal font.

The usage of '`man` *section title*' becomes obsolescent, which should be removed from future releases.

ENVIRONMENT VARIABLES

The `man` utility uses the following environment variables:

`MANPATH` When set, its value is used as the search path for man page searching. The `-M` option overrides this variable.

PAGER	When set, its value is used as the utility for displaying the requested man pages. By default, `more -s` is used.
TCAT	When set, its value is the name of the utility used to print `troff` output. By default, `lpr -n` is used.
TROFFCMD	When set, its value is used as the format utility called by the `-t` option. By default, `troff` is used.

FILES

`/usr/man/man?/*`	Directories containing unformatted man pages.
`/usr/man/cat?/*`	Directories containing preformatted pages.
`/usr/lib/tmac/tmac.uc`	Cray Research man page macros.
`/usr/man/index`	File that contains cross-reference information on all UNICOS man page entries; used by the `-i` option.
`/usr/man/whatis`	File that contains all man page entry name lines; used by the `-f` and `-k` options.

SEE ALSO

`apropos`(1), `cat`(1), `col`(1), `grep`(1), `more`(1), `nroff`(1), `troff`(1), `whatis`(1)

NAME

mesg – Permits or denies messages

SYNOPSIS

```
mesg [y | n]
```

Obsolescent version: may not be supported in future releases:

```
mesg -y
mesg -n
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The mesg utility controls whether other users are allowed to send messages by using write(1), talk(1B), or other utilities to a terminal device. The terminal device affected is determined by searching for the first terminal in the sequence of devices associated with standard input, standard output, and standard error, respectively. With no operands, mesg reports the current state without changing it.

Processes with appropriate privileges may be able to send messages to the terminal device independent of its current state.

The mesg utility accepts the following operands:

y Grants permission to other users to send messages to the terminal device.

n Denies permission to other users to send messages to the terminal device.

The obsolescent version accepts the following options:

-y Equivalent to the y operand.

-n Equivalent to the n operand.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to bypass all protections on the terminal device.

If the PRIV_SU configuration option is enabled, the super user is allowed to override all protections on the terminal device.

EXIT STATUS

The mesg utility exits with one of the following values:

0 Receiving messages is allowed.

1 Receiving messages is not allowed.

>1 An error occurred.

EXAMPLES

The following example first determines your current message setting and then changes it. User input is shown in bold font:

```
$ mesg
is y
$ mesg n
$ mesg
is n
```

FILES

/dev/tty*	Device files

SEE ALSO

write(1), talk(1B)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

mkdir – Creates a directory

SYNOPSIS

mkdir [-L *level*[,*compartment*[,*compartment*[,...]]] [-m *mode*] [-p] *directories*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-L option)

DESCRIPTION

The mkdir utility creates specified directories in mode 777 (unless altered by umask(1)). Standard entries . (for the directory itself) and .. (for its parent) are made automatically. mkdir cannot create these entries by name.

The mkdir utility requires write permission in the parent directory.

The owner ID of the new directory is set to the process' real user ID. The group ID of the new directory is set to the group ID of the parent directory.

The mkdir utility accepts the following options:

-L *level*[,*compartment*[,*compartment*[,...]]
: Allows the directory to be relabeled to the specified security label. Choices for *level* and *compartments* are defined by your site security administrator. When the -L option is specified with the -p option, only the last directory in the path is labeled at the requested label. All intermediate directories are created at the user's active label. The behavior of a mkdir request requires that your active security label must be the same as the security label of the parent directory. Also, if the -L option is specified, the directory is relabeled to the specified security label. If the relabeling fails, the directory's label remains at your active label. The requested label must dominate your active label.

-m *mode*
: Lets you specify the *mode* to be used for new directories. You can find choices for modes in chmod(1). In the *symbolic_mode* strings, the op characters + and - are interpreted relative to an assumed initial mode of a=rwx; + adds permissions to the default mode, - deletes permissions from the default mode.

-p
: Creates a directory by creating all nonexisting parent directories first.

directories
: Specifies the name of the directory you are creating.

`mkdir` can fail for the following reasons:

- Your security level is not within the upper and lower security levels and/or authorized compartments of the file system.
- The new directory does not conform to the security level hierarchy rules (that is, the security level of the new directory must be equal to or greater than the security level of the parent directory). In addition, the security compartments of the directory to be created must be a proper superset of the parent directory compartments.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to create any directory.
`sysadm`	Allowed to create any directory, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to create any directory.

EXIT STATUS

The `mkdir` utility returns exit code 0 when all directories are successfully made; otherwise, it prints a diagnostic message and returns a nonzero exit code.

EXAMPLES

To create subdirectory structure `ltr/jd/jan`, enter the following:

```
mkdir -p ltr/jd/jn
```

SEE ALSO

`chmod`(1), `rm`(1), `sh`(1), `umask`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`mkfifo` – Makes FIFO special files

SYNOPSIS

`mkfifo` [`-m` *mode*] *file* ...

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `mkfifo` utility creates the first in, first out (FIFO) special files specified by the operands, in the order specified.

The `mkfifo` utility accepts the following options and operand:

`-m` *mode* Sets the file permission bits of the newly created FIFO to the specified *mode* value. The *mode* argument is the same as the *mode* operand defined for the `chmod`(1) utility. In the *symbolic_mode* strings, the *op* characters `+` and `-` are interpreted relative to an assumed initial mode of `a=rw`.

file Specifies a path name of the FIFO special file to be created.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to create any file.
`sysadm`	Allowed to create any file, subject to security label restrictions on the file's path. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to create any file.

EXIT STATUS

The `mkfifo` utility exits with one of the following values:

0 All of the specified FIFO special files were created successfully.

>0 An error occurred.

SEE ALSO

`chmod`(1)

`mkfifo`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

more, page – File perusal filter for CRT viewing

SYNOPSIS

more [-ceisu] [-n *number*] [- *linenum*] [- *lines*] [-/ *pattern*] [-p *command*] [-t *tagstring*] [*file...*]

Obsolescent versions; may not be supported in future releases:

more [-ceisu] [-n *number*] [+*command*] [-t *tagstring*] [*file...*]
more [-ceisu] [-n *number*] [+*number*] [-t *tagstring*] [*file...*]
more [-cdefilsu] [-n *number*] [-p *command*] [-t *tagstring*] [*file...*]
more [-ceisu] [-*number*] [-p *command*] [-t *tagstring*] [*file...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The more utility is a filter that lets you examine continuous text one screenful at a time. It typically pauses after each screenful. If you then press <return>, one more line is displayed. If you press the space bar, another screenful is displayed. Other possibilities are listed later in this section.

The more utility accepts the following options:

-c Draws each page by beginning at the top of the screen and erasing each line just before drawing on it. This avoids scrolling the screen, making it easier to read while more is writing. If the terminal cannot clear to the end of a line, this option is ignored.

-d (Obsolescent) Prompts you with the Press space to continue, 'q' to quit message at the end of each screenful. This is useful when you are using more as a filter in a setting, such as a class, in which many users may be unsophisticated.

-e Exits immediately after writing the last line of the last file in the argument file list.

-f (Obsolescent) Counts logical, rather than physical screen lines (that is, it does not fold long lines).

-i Performs pattern matching in searches, ignoring case.

-l (Obsolescent) Does not treat the <form-feed> character specially. If you do not specify this option, more will pause after any line that contains a <form-feed> character, as if the end of a screenful was reached. Also, if a file begins with a <form-feed>, the screen will be cleared before the file is printed.

`-n` *number*
`-`*number* (Obsolescent)
Specifies the number of lines per screenful. *number* is a positive decimal integer. The `-n` option overrides any value obtained from the environment.

`-p` *command*
`+`*command* (Obsolescent)
For each file examined, initially executes the `more` utility in the *command* argument. If *command* is a positioning command, such as a line number or a regular expression search, the current position is set to represent the final results of the command, without writing any intermediate lines of the file.

`-s` Squeezes multiple blank lines from the output, producing only one blank line. This option maximizes the useful information present on the screen.

`-t` *tagstring* Writes the screenful of the file containing the tag named by the *tagstring* argument. If both the `-t` *tagstring* and `-p` *command* option are specified, the `-t` *tagstring* is processed first. The file containing the tag is selected by `-t`, and then the command is executed.

`-u` Suppresses underline processing. Usually, `more` handles underlining such as that produced by `nroff`(1) in a manner appropriate to the particular terminal: if the terminal can perform underlining or has a stand-out mode, `more` will output appropriate escape sequences to enable underlining or stand-out mode for underlined information in the source file.

`+`*linenum* Starts up at line *linenum*.

If you invoke `more` as `page`, the screen is cleared before each screenful is printed (but only if a full screenful is being printed), and $k - 1$ rather than $k - 2$ lines are printed in each screenful (k is the number of lines the terminal can display).

The `more` utility looks in the `/usr/lib/terminfo` directory to determine terminal characteristics and the default window size defined by `TERM` environment variable. On a terminal that can display 24 lines, the default window size is 23 lines.

The `more` utility looks in the `MORE` environment variable to preset any desired flags. For example, if you view files by using the `-c` mode of operation, the `sh`(1) utilities will cause all invocations of `more`, including invocations by other programs, to use this mode. Only flags that are preceded with a hyphen (`-`) are recognized when using the `MORE` environment variable.

```
MORE=-c
export MORE
```

When `more` is reading from a file rather than a pipe, a percentage is displayed along with the *file* prompt. This specifies the fraction of the file (in characters, not lines) that has been read so far.

You may type other sequences when `more` pauses, and their effects are as follows (*i* is an optional integer argument, defaulting to 1):

[*i*]`f`
[*i*]`<CONTROL-f>` Moves forward *i* lines; default is one screenful.

[*i*]b
[*i*]<CONTROL-b> Moves backward *i* lines; default is one screenful.

[*i*]<space>
[*i*]j
[*i*]<newline> Scrolls forward *i* lines. The default for <space> is one screenful; for j and <newline>, one line.

[*i*]k Scrolls backward *i* lines; default is 1.

[*i*]d
[*i*]<CONTROL-d> Scrolls forward *i* lines; default is one half of the screen size. If *i* specify, *i* becomes the new default for subsequent d and u commands.

[*i*]s Skips *i* lines and prints a screenful of lines.

[*i*]u
[*i*]<CONTROL-u> Scrolls backward *i* lines; default is one half of the screen size. If you specify *i*, it becomes the new default for subsequent d and u commands.

[*i*]g Goes to line *i* in the file; default is 1 (beginning of file).

[*i*]G Goes to line *i* in the file; default is the end of the file.

r
<CONTROL-L> Redraws screen.

R Refreshes the screen, discarding any buffered input.

m*letter* Marks the current position with the specified *letter*; *letter* represents the name of one of the lowercase letters of the character set.

'*letter* Returns to the position that was previously marked with *letter*.

'' Returns to the position from which the last large movement command was executed (a large movement is defined as any movement of more than a screenful of lines). If no such movements were made, returns to the beginning of the file.

[*i*]/[!]*pattern*<newline>
Searches forward in the file for the *i*th line that contains *pattern*. The *i* defaults to 1. If the ! character is included, the lines that do not contain *pattern* are not searched.

[*i*]?[!]*pattern*<newline>
Searches backward in the file for the *i*th line that contains *pattern*. The *i* defaults to 1. If you include the character !, the lines that do not contain *pattern* are not searched.

[*i*]n Repeats the previous search for the *i*th line (default is 1) that contains the last pattern (or not containing the last pattern, 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 pattern (or does not contain the last pattern if the previous search was /! or ?!).

`:e` [*filename*]`<newline>`
: Examines a new file. If you omit *filename*, the current file from the list of files in the command line is reexamined.

[*i*]`:n`
: Examines the next file; if you specify *i*, the *i*th next file is examined.

[*i*]`:p`
: Examine the previous file; if you specify *i*, the *i*th previous file is examined.

`:t` *tagstring*`<newline>`
: Displays the file beginning with the line that contains *tagstring*.

`v`
: Invokes an editor to edit the current file being examined. The name of the editor is taken from the `EDITOR` environment variable or defaults to `vi`.

`=`
`<CONTROL-g>`
: Displays the current line number.

`h`
: Help command; provides a description of all `more` commands.

`q`
`:q`
`ZZ`
: Exits `more`.

The commands take effect immediately; that is, it is not necessary to press `<RETURN>`.

The terminal is set to `noecho` mode by this program so that the output can be continuous.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `more` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

The use of `more` in the previewing of `nroff`(1) output is as follows:

```
nroff -ms doc.n | more -s
```

FILES

`/usr/lib/terminfo` Terminal database

SEE ALSO

`cat`(1), `nroff`(1), `pg`(1), `sh`(1)

NAME

msgi – Sends informative message to operator

SYNOPSIS

msgi *msg_string*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The msgi utility lets you send an informative message to the operator.

The msgi utility accepts the following operand:

msg_string Specifies the text of the message that you want to sent.

The operator cannot reply to informative messages. All messages are logged in the order in which they are received.

Messages are not issued to operators to let them know that there is outstanding information. To see the msgi messages, operators must run the infd(8) command within the oper(8) command.

NOTES

You must enclose shell special characters or strings that contain shell special characters in quotation marks (single or double); if these are not enclosed in quotation marks, the shell will misinterpret the special characters. For example, the following messages contain a shell special character, the semicolon:

```
msgi "Tape ABC is not on the system; waiting for mount."

msgi Tape ABC is not on the system";" waiting for mount.
```

EXAMPLES

The operator executes the following oper(8) command, which starts the infd(8) command:

```
oper infd
```

The user sends the following message:

```
msgi from Al: See Bob for tape XYZ
```

The operator sees the following message:

```
Command: infd  Page: 1 [delay 10] Fri Jun  4 09:39:38 1993

from Al: See Bob for tape XYZ
              .
              .
              .
> _
```

FILES

`/usr/spool/msg/msglog.log`	Log file

SEE ALSO

`msgr`(1)

`infd`(8), `msgd`(8), `msgdaemon`(8), `msgdstop`(8), `oper`(8), `rep`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

msgr – Sends action message to operator

SYNOPSIS

msgr *msg_string*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The msgr utility lets you send an action message to the operator and receive a reply from the operator.

The msgr utility accepts the following operand:

msg_string Specifies the text of the message that you want to sent.

The sending process is suspended until a reply is received from the operator. The reply is then printed to standard output. If you interrupt the process by using <CONTROL-c> while waiting for a reply, the action message will be canceled. All messages are logged in the order in which they are received.

To see these messages, the operator must run the msgd(8) command within the oper(8) command. The operator responds with the rep(8) command.

NOTES

You must enclose shell special characters or strings that contain shell special characters in quotation marks (single or double); if these are not enclosed in quotation marks, the shell will misinterpret the special characters. For example, the following messages contain shell special characters, a semicolon and question mark:

```
msgr "Please mount tape ABC; OK?"

msgr Please mount tape ABC";" OK"?"
```

EXAMPLES

The operator executes the oper(8) command (which runs msgd(8) if no command argument is provided):

```
$ oper
```

The user sends the following message:

```
$ msgr Please mount tape ABC located on shelf 29
```

The operator sees the message and responds by using the rep(8) command:

```
Command: msgd  Page: 1 [delay 10] Fri Jun  4 09:43:48 1993

  Msg #   Time    System Messages
  =====   =====   ===============

   799    09:43   From us1:  Please mount tape ABC located on shelf 29
                     .
                     .
                     .
> rep 799 Tape ABC found and mounted
> _
```

The user receives the reply:

```
$ msgr Please mount tape ABC located on shelf 29
Tape ABC found and mounted
$ _
```

FILES

/usr/spool/msg/msglog.log	Log file

SEE ALSO

msgi(1)

infd(8), msgd(8), msgdaemon(8), msgdstop(8), oper(8), rep(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

NAME

mt – Issues commands to a magnetic tape drive

SYNOPSIS

/usr/ucb/mt [-f *tapename*] *command* [*count*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The mt utility issues commands to a magnetic tape drive. If *tapename* is not specified, the TAPE environment variable is used; if TAPE does not exist, the utility terminates. *tapename* must reference a character-special tape device. By default, mt performs the requested operation once. Operations may be performed multiple times by specifying *count*.

The mt utility accepts the following option and arguments.

-f *tapename* Specifies the character-special tape device to be activated.

command Specifies the command to execute on the tape device. Only as many characters as are required to uniquely identify a command need to be specified. Valid commands are as follows:

bsf [*count*]	Skips back over *count* file marks; the default is 1.
eof, weof	Writes *count* end-of-file marks at the current position on the tape.
fsf [*count*]	Skips forward over *count* file marks; the default is 1.
offline, rewoffl	Rewinds the tape and places the tape unit offline (*count* is ignored).
rewind	Rewinds the tape (*count* is ignored).
status	Prints status information about the tape unit.

count Specifies the number of files to skip over or the number of end-of-file marks to write.

NOTES

The mt utility can be used only for tape drives that are not configured up (UP).

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
secadm, sysadm	Allowed to use this command.

If the PRIV_SU configuration option is enabled, the super user is allowed to use this command.

EXIT STATUS

The mt utility exits with one of the following values:

0 The operation was successful.

1 The command was unrecognized.

2 The operation failed.

FILES

/dev/tape/*device_name*
Tape device node

SEE ALSO

dd(1)

ioctl(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

environ(7) (available only online)

Tape Subsystem User's Guide, Cray Research publication SG–2051

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`mtdump` – Examines unformatted dump of multitasking history buffer

SYNOPSIS

`mtdump` [`-f` *form*] [`-t` *tsks*] [`-d` *data*] [`-a` *act*] [`-T`] [`-L`] [`-E`] [`-B`] [`-C`] [`-U`] [`-i` *intr*] [`-V`] *file*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `mtdump` command examines the unformatted dump of the multitasking history buffer and generates reports according to the options used. The only required argument is *file*; if no other options are specified, the report generated is a summary of all information contained in the dump. The default is the same as specifying `-f totals`.

You can generate reports in five primary formats, which display all the information in the buffer; you can select specific buffer entries, which display only portions of the available information; or you can select any combination of these formats, using the appropriate options.

Generating the five primary reports is specified by the following `-f` options. You can specify one or more `-f` options, each of which is separated by commas (with no blank spaces) and listed in any order. For each report you request, you will receive a complete listing, in the *form* format, of the entire multitasking history buffer file. Every report requested rewinds the file and displays the data in the requested *form*.

`-f` *form* Displays the multitasking history trace buffer in the specified format. Valid *form* arguments are as follows:

- `totals` Displays a summary of all events in multitasking history trace buffer, including the total number of occurrences of each kind of event. This is the default.
- `chron` Displays a chronological listing of all entries in the file.
- `sync` Displays synchronization points, with a separate column for each of up to 16 user tasks. If an event in the file does not involve task synchronization, it is not listed. The listing is grouped by task identifier.
- `cpu` Displays logical CPU use, with a separate column for each of up to 16 logical CPUs. A listing is made of the activities in each logical CPU. When the state of a logical CPU changes, a symbol appears in that CPU's column on the report, reflecting the new status of the CPU. This listing is grouped by CPU (not task). A map is produced, assigning each logical CPU to a UNICOS process ID.
- `status` Displays the status of each of up to 16 user tasks in uniform time intervals. You can change this interval using the `-i` *intr* option. Listing is by task identifier. Listing by a uniform time interval allows you to see the multitasking execution by a snapshot in time.

The following options let you list selected groups of buffer entries. If these options are specified, `mtdump` searches for and displays only those multitasking history buffer entries that match the specified criteria. The requested information is displayed in chronological order, in a format similar to the `chron` *form* previously described. If you specify any of the following options, with no `-f` option, no primary report is displayed; a single report is generated containing the selected items. Selected options can be combined to narrow the criteria for search. Note, however, that if you make the selection process too narrow, it is possible that no records will be found that meet the desired criteria. An informative message will be issued if no records are found that match the specified criteria.

`-t` *tsks* Selects one or more internal task identifiers, in decimal, separated by commas, for which buffer entries should be listed; maximum of 10 task identifiers is allowed.

`-d` *data* Selects one or more action-dependent data values, in octal, separated by commas, to be searched for; maximum of 10 entries for all data values is allowed.

`-a` *act* Selects one or more action codes of buffer entries, in decimal, separated by commas, to be listed, with a maximum of 40 actions allowed. Valid action codes are 0 through 127 (decimal). The default is to list entries for all action codes unless one or more of the `-t`, `-d`, `-T`, `-L`, `-E`, `-B`, `-C`, `-U`, or `-V` options has been used.

The following options allow you to specify a range of action codes to be searched for and displayed. Using the following options is simpler than trying to specify many values with the `-a` option.

If you specify one or more of these options, and also select actions by number (using `-a` *act*), the effect is cumulative. For example, if you specify `-a 34`, and also specify the `-B` option (barriers), records with action codes 33 through 37 are displayed. If you select the `-C` and `-B` options, records with action codes 21 through 37 are displayed.

`-T` Lists actions involving tasks; these include task starts, completions, waits, and tests. (Action codes 0 through 5.)

`-L` Lists actions involving locks. (Action codes 6 through 12.)

`-E` Lists actions involving events. (Action codes 13 through 20.)

`-B` Lists actions involving barriers. (Action codes 33 through 37.)

`-C` Lists actions involving logical CPUs. (Action codes 21 through 32.)

`-U` Lists actions involving user codes. (Action codes 64 through 127.)

Other options are as follows:

`-i` *intr* Specifies the number of clock periods, in decimal, for each time interval to be used in the `status` format display. The default is 1,000,000. This option has no effect on other format displays.

`-V` Displays the current `mtdump` version number, and a brief copyright message.

file Specifies the file containing the unformatted dump of the multitasking history trace buffer. Specifying *file* is required.

SEE ALSO

BUFTUNE(3F) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

performance(7) (available only online)

Guide to Parallel Vector Applications, Cray Research publication SG–2182

NAME

mv – Moves files or a directory

SYNOPSIS

mv [-f] [-i] *files target*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

Use the mv utility to do any of the following:

- Move (rename) one file
- Move one or more files from a directory to another existing directory
- Rename a directory
- Move a directory

To move one or more files, specify the current file name(s) (*files*) and the new name for the file (*target*). Do not use the same name for *files* and *target*. (Be careful when using shell metacharacters.) If *target* is not a directory, only one file may be specified before it; if it is a directory, more than one file may be specified. If *target* does not exist, mv creates a file named *target*. If *target* exists and is not a directory, its contents are overwritten. If *target* is a directory, the *files* are moved to that directory. If *target* is a link to another file with links, the other links will remain and *target* will become a new file.

If mv determines that the mode of *target* forbids writing, it will print the mode (see chmod(1)), prompt for a response, and read the standard input for one line. If the line begins with y, mv will execute if permissible; if not, the utility will exit. When the standard input is not a terminal, no messages concerning mode restrictions are given.

The mv utility accepts the following options and operands:

-f Moves the *files* without prompting even if it is writing over an existing *target*. This is the default if the standard input is not a terminal. Any previous occurrences of the -i option are ignored.

-i Prompts for confirmation whenever the move would overwrite an existing *target*.

files Files to be moved.

target Destination of the moved file. A y answer means that the move should proceed. Any other answer prevents mv from overwriting the *target*. Any previous occurrences of the -f option are ignored.

Specifying both `-f` and `-i` is not an error. The last option specified determines the behavior of `mv`.

To move files from a directory to another existing directory, supply the file name(s) (*files)* and the directory name (*target*).

To rename a directory, specify the current directory name (*files*) and a new directory name (*target*). The new directory name must be unique and the two directories must have the same file system root. A file system that is not locally mounted must have root write permission for the directory rename to succeed.

Files with set-user-ID (SUID) and set-group-ID (SGID) permissions have those permissions removed on a move when *files* and *target* reside on different file systems, unless you have `suidgid` permission.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to move any file or directory.
`sysadm`	Allowed to move any file or directory, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to move any file or directory.

If *files* and *target* reside on different file systems, `mv` must copy the file and delete the original. In this case, any linking relationship with other files is lost.

If a user tries to move a restart file to a different file system, it is no longer marked as a restart file and cannot be restarted.

EXIT STATUS

The `mv` utility exits with one of the following values:

0 All input files were moved successfully.

>0 An error occurred.

EXAMPLES

Example 1: Use the following command line to move file `yesterday` to `today`:

```
mv yesterday today
```

Example 2: Use the following command line to move all files in your working directory that begin with letter `r` to subdirectory `movehere`:

```
mv r* movehere
```

Example 3: Use the following command line to rename directory `olddir` to `newdir` (`newdir` does not already exist):

```
mv olddir newdir
```

SEE ALSO

`chmod`(1), `cp`(1), `cpio`(1), `ln`(1), `rm`(1), `sh`(1)

`chmod`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

mvf – Provides a multivolume tape filter

SYNOPSIS

/bin/mvf -i [-a] [-b *blocksize*] [-c *vol_count*] [-C *copy_size*] [-d *digits*] [-f *file_id*] [-l *label_option*] [-n *buffers*] [-O *offset*] [-t] [-T *count*] [-v] [-V *volser_Base*] *device_name* [:*vsn*[,*vsn*]... [*device_name* [:*vsn*[,vsn]]]

/bin/mvf -o [-b *blocksize*] [-c *vol_count*] [-d *digits*] [-f *file_id*] [-l *label_option*] [-n *buffers*] [-t] [-T *count*] [-v] [-V *volser_Base*] *device_name* [:*vsn*[,*vsn*]... [*device_name* [:*vsn*[,vsn]]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The mvf command is a filter that enables you to use multiple tape volumes as though a program were reading or writing to a very large tape volume. It accepts input from standard input (stdin) for an output operation to tape and writes to standard output (stdout) for an input operation from tape.

You must select the I/O direction by using either the -i option for input from tape or -o option for output to tape and the tape device path name (*device_name*). You may also specify volume serial numbers (VSNs) (*vsn*) following the device name.

If you specify the -V and -d options to construct the VSNs from a base VSN and a field width, mvf takes the base VSN and increments the relevant width argument as each new volume is required.

For an input operation, reading continues until the volume count (-c option) is exhausted or until all of the VSNs have been read.

For an output operation, writing continues until an EOF is detected on input or until the volume count (-c option) is exhausted. The output operation is allowed to extend to one more volume than is specified in the volume count specification.

The mvf command accepts the following options, arguments, and operands:

-a	Accepts read errors from the tape device. If read errors occur, the data written to stdout is unpredictable.
-b *blocksize*	Specifies a block size in bytes. The default is 98304.
-c *vol_count*	Specifies a count of volumes to process. The default is 1.
-C *copy_size*	Specifies the amount of data to copy to stdout for an input operation.
-d *digits*	Specifies the size of the digit field. See the -V option.
-f *file_id*	Specifies a file ID to be associated with all volumes written with either IBM or ANSI standard labels.

`-i` Reads from tape and writes to `stdout`. You cannot specify the `-i` option with the `-o` option.

`-l` *label_option* Sets the type of label processing to use. *label_option* may be any of the following:

`sl` Specifies IBM standard labels.

`al` Specifies ANSI standard labels.

`blp` Specifies bypass label processing. `blp` skips any label processing and reads or writes nonlabeled tapes. The tape is terminated by the device during close processing and is positioned so that a new file can be added to it.

To unload the tape after `blp` use, you must use the `mt`(1B) command or physically unload the tape device.

If you omit the `-l` option, `mvf` reads the label to verify that a nonlabeled tape has been loaded.

If you load a tape and its tape type differs from the type you specified, `mvf` reformats the tape with the specified label type.

`-n` *buffers* Specifies a number of asynchronous I/O buffers to use for each tape device. The default is 6.

`-o` Reads from `stdin` and writes to tape. You cannot specify the `-i` option with the `-o` option.

`-O` *offset* Specifies an offset into the tape record to extract data for an input operation.

`-t` Displays timing information.

`-T` *count* Sets the count of internal trace buffers to use for debugging and automatically dumps them at exit time to `stderr`.

`-v` Displays portions of the tape labels and performs a limited set of label consistency operations on labeled tapes. The default is no label consistency checking.

`-V` *volser_Base* Specifies the base VSN for internally generated VSN lists. See the `-d` option.

device_name [`:`*vsn*[`,`*vsn*]] Specifies character-special device names and a list of VSNs. At least one device name is mandatory, followed optionally by a colon and a list of VSNs separated by commas. When you specify multiple tape devices, `mvf` processes data on one device until it reaches the end of the tape. At that time, the tape device is unloaded, and the operation is continued on the next device.

For operations that require multiple tapes, device switching saves the rewind unload time and consequently maximizes throughput.

Specifying multiple tape devices enables input or output to continue without having to wait for the current device to rewind and unload.

EXAMPLES

The following examples illustrate different uses of the `mvf` command.

Example 1: This example shows how to back up the `/v` file system onto a tape without a tape label:

```
cd /v find . -depth -print | cpio -o | mvf -o /dev/tape/cart59
```

Example 2: This example shows how to restore the `/v/xyz/test.f` file from a previous backup of the `/v` file system:

```
mvf -i /dev/tape/cart59 | cpio -i ./xyz/test.f
```

Example 3: The following example shows how to back up the `/u` file system. It uses four tape volumes and ANSI label processing:

```
cd /u find . -depth -print | cpio -o | \
mvf -ol al  /dev/tape/cart59:q11010,q11011,q11012,q11013 \
```

Example 4: This example provides an alternative way to back up `/u`. You use two tapes mounted on two devices; the second device overlaps its I/O with the unload of the first device.

```
cd /u find . -depth -print | cpio -o | \
mvf -o -l al /dev/tape/cart10:q11010,q11012 \
/dev/tape/cart59:q11011,q11013
```

The volume usage sequence for this example is as follows:

```
write q11010 on /dev/tape/cart10
unload q11010, write q11011 on /dev/tape/cart59
unload q11011, write q11012 on /dev/tape/cart10
unload q11012, write q11013 on /dev/tape/cart59
unload q11013
```

Example 5: This example performs a level 0 dump of `*/ptmp` to a set of volumes.

```
/etc/dump -t0 -f- /ptmp | mvf -ol sl  302:000600,000602,000604
```

It produces the following screen output:

```
mvf: Mount 000600 on 302 OUTPUT mode
dump (/ptmp to -): Date of this level 0 dump: Thu Apr  6 13:44:43 1995
dump (/ptmp to -): Dumping /ptmp
dump (/ptmp to -): to -
dump (/ptmp to -): mapping (Pass I) [regular files]
dump (/ptmp to -): mapping (Pass II) [directories]
dump (/ptmp to -): estimated 229048 sectors on 0.00 volume(s).

dump (/ptmp to -): dumping (Pass III) [directories]
mvf: Mount 000601 on 303 OUTPUT mode
dump (/ptmp to -): dumping (Pass IV) [regular files]
mvf: 6808 blocks out on 302
mvf: Mount 000602 on 302 when Rewind/Unload is Complete.
mvf: 6814 blocks out on 303
mvf: Mount 000603 on 303 when Rewind/Unload is Complete.
mvf: 6816 blocks out on 302
mvf: Mount 000604 on 302 when Rewind/Unload is Complete.
mvf: 6826 blocks out on 303
mvf: Mount 000605 on 303 when Rewind/Unload is Complete.
dump (/ptmp to -): un-mapping (Pass V) [changed files]
dump (/ptmp to -): dump has completed, 229048 blocks
mvf: 28631 blocks on 5 cartridges
```

In this case, the last volume, 000605, was requested, but was not actually used.

NOTES

The `mvf` command does not reject duplicated options; it uses the last valid option of a particular type.

FILES

`/dev/tape/`*device_name*	Tape device node

SEE ALSO

`mt`(1B)

`tpinit`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Tape Subsystem User's Guide, Cray Research publication SG–2051

Tape Subsystem Administration, Cray Research publication SG–2307

NAME

nasa – Adds ASA carriage control characters for printing

SYNOPSIS

nasa [-t *tabspace*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nasa utility provides the function opposite to the asa(1) utility. It transforms typical UNICOS command text output to make it suitable for output to line printers that require ASA carriage control characters.

It processes either the *file* whose name is given as an argument or the standard input if you do not specify a file name. <tab> characters are expanded to the appropriate number of spaces to position a subsequent character at the next <tab> stop.

Other transformations performed are as follows:

line feed	Becomes <newline>, <space>
carriage return	Becomes <newline>, +
form feed	Becomes <newline>, 1

The nasa utility accepts the following option:

-t *tabspace* Specifies *tabspace* number of character <space>s between <tab> stops. Default is 8.

file File to be converted.

The nasa utility forces the first line to start on a new page by starting its output with a 1. <backspace> characters (ASCII code 8) are properly compensated for, to preserve the column position of subsequent <tab> characters, but the destination printer may not accept them.

NOTES

The nasa utility complements asa(1) only in that it converts raw data for printing on asa-style printers. nasa cannot undo the damage done to a file by inadvertently running it through asa(1).

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Allowed to transform any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.

sysadm	Allowed to transform any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to transform any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXAMPLES

Example 1: The following example is a standard shell script preparing the output of a job for printing on a line printer requiring carriage control:

```
(
    set -x                  # print command names as they are executed
    UNICOS commands
          .
          .
          .
    application | asa
    echo '\f'                  # form feed
          .
          .
          .
) 2>&1 | nasa > tracefile
```

Example 2: The following example prepares output to be printed and sends it to file datafile. Output is not sent to the printer.

```
ls -la | nasa >datafile
```

SEE ALSO

asa(1), expand(1), unexpand(1)

NAME

netstat – Displays network status

SYNOPSIS

```
/usr/ucb/netstat [-a] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat [-A] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -g [-s] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -H [-f address_family] [system] [core]
/usr/ucb/netstat -i [-t] [-s] [-a] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -m [-s] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -q [-a] [-i] [-v] [-n] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat -r [-v] [-n] [-k] [-A] [-f address_family] [system] [core]
/usr/ucb/netstat -s [-i] [-r] [-v] [-k] [-f address_family] [system] [core]
/usr/ucb/netstat [-I interface] [-n] [-k] [-f address_family] interval [system] [core]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The netstat command symbolically displays the contents of various network-related data structures. The default display, which is for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and, optionally, the internal state of the protocol.

You can combine the following options and arguments with any of the main options (the list of main options follows this list):

-v
: Indicates verbose mode. You can repeat the -v option to obtain additional information. The option that accompanies the -v option determines the amount of additional information that can be displayed.

-n
: Shows Internet addresses as numbers. (Usually netstat interprets addresses and tries to display them symbolically.)

-k
: The -k option is provided for debugging purposes. It forces netstat to use /dev/kmem to obtain data, rather than use system calls.

-f *address_family*
: Limits statistics or address control block reports to those of the specified *address_family*. The address families that are recognized are inet (for AF_INET), trace (for AF_TRACE), and unix (for AF_UNIX).

system, *core*
: Obtains information from a system dump instead of the operating system. When a nonexistent system or core file is specified on the command line, netstat prints an error message and exits.

Main options:

`-a` Modifies the default display. The `-a` option indicates that all sockets will be shown (usually, sockets that server processes use are not shown).

If `-v` is used, the receive and send queue lengths are omitted, and the machine through which the connection is being routed is shown. If a second `-v` option is used, more information about the state of each TCP connection is printed (maximum segment size, send and receive windows, send and receive next sequence numbers, send and receive urgent pointers, unacknowledged send pointer, and retransmission sequence number).

`-A` Modifies the default display. With the `-A` option, the address of any associated protocol control blocks also is shown. This is used for debugging.

If `-v` is used, the receive and send queue lengths are omitted, and the machine through which the connection is being routed is shown. If you use a second `-v` option, more information about the state of each TCP connection is printed (maximum segment size, send and receive windows, send and receive next sequence numbers, send and receive urgent pointers, unacknowledged send pointer, the peers offered segment size, and retransmission sequence number).

`-g [-s]` Shows information related to multicast (group address) routing. By default, the `-g` option shows the Internet Protocol (IP) multicast virtual interface and routing tables. If the `-s` option is also present, it shows multicast routing statistics.

`-H` Shows statistics about sizes of packets that are read and written. These statistics are not subdivided for each interface. Packet size statistics are not maintained for CRAY J90 Ethernet, FDDI, or ATM interfaces; the `-H` option does not apply under these circumstances.

`-i [-t] [-s] [-a]`
Provides a table of cumulative statistics for transferred packets and errors for each interface that was autoconfigured. (Those interfaces that are statically configured into a system but are not located at boot time are not shown.) The network address (currently Internet-specific) of the interface and the maximum transmission unit (mtu) in bytes also are displayed. For secure systems, if the `-v` option is specified, the level and compartment range are displayed.

The `-t` option displays the timer value. If the interface is not using the timer field, it has a value of 0. With the `-v` option, the configuration and usage of the interface queues are displayed, and the flags are set for the interface. The queues are displayed in an *x/y* format; *x* is the number of packets currently on the queue, and *y* is the maximum number of packets that are allowed on the queue. The number of packets dropped because of the lack of queue space also is displayed.

The `-s` option displays the interface statistics in long format for each interface, and all of the interface statistics are printed with a short explanation.

The `-a` option shows multicast addresses currently in use. Multicast addresses are shown on separate lines following the interface address with which they are associated.

`-m [-s]` Shows statistics that the memory management routines record. (The network manages a private share of memory buffers called *mbufs* (pronounced em-bufs).) With one `-v` option, a list of the mbuf headers for all mbuf clusters in the system is shown. With two `-v` options, a list of the mbuf headers for all mbufs in the system, including those that are part of an mbuf cluster, is shown. With three `-v` options, a list of the mbuf headers for all mbuf clusters in the system is shown. The contents of the mbuf also are printed. Some types of mbufs are recognized and are printed as their structures; other types are printed as hexadecimal bytes. Besides the information shown by three `-v` options, four `-v` options show all mbufs in the system, including those that are part of an mbuf cluster. The `-s` option displays a histogram of mbuf request sizes.

`-q [-a] [-i]` Shows mbuf usage on the kernel IP input, IP reassembly, and raw input queues. If `-a` is specified, any mbufs on TCP reassembly queues also are shown.

If `-i` is specified, any mbufs on any of the interface output queues also are shown.

If `-v` is specified, a count and a list of the mbufs being used on each queue are printed. With `-vvk` options specified, up to the first 256 bytes of data are printed for each mbuf, relative to the offset in the mbuf. With `-vvvk` options, the first 256 bytes of data for each mbuf, or up to the offset, also are printed. It should be noted that the `-k` option is necessary to print the mbuf data.

Because of the transitory nature of these queues, if `-q` is used on a running system, it can easily become confused and give erroneous results. This option is best used on a core dump.

`-r [-k] [-A]` Indicates the available routes and status of each route. Each route consists of a destination host or network and a gateway to use for forwarding packets. The flags field indicates whether the route is for a host or a network, and whether an intermediate gateway or router is used (see flags below). If the route does not use a gateway, the gateway column may list the address for the interface used when sending to the destination, or may contain a link-layer address.

Network routes have an associated mask that specifies which parts of an address are matched. If destination is printed numerically and the mask is not the obvious value (e.g. if a route to an Internet Class B network has a mask other than 16 bits), the mask is indicated in one of two ways: If the mask is contiguous from the most-significant bit to the end, the usual case for subnets, a slash and the number of bits in the mask are appended to the network value. Otherwise, an ampersand (&) and a numeric representation of the mask are appended.

The flags field shows a collection of information about the route stored as binary choices. The individual flags are discussed in more detail in the `route`(8) and `route`(4P) man pages. The mapping between letters and flags is as follows:

1 Protocol specific routing flag #1
2 Protocol specific routing flag #2
B Discard all packets (during updates)
C Generate new routes on use
D Created dynamically (by redirect)
E Exclusive group ID list for route
G Destination requires forwarding by intermediary
H Host entry (default is net)
L Valid protocol to link address translation
M Modified dynamically (by redirect or mtu discovery)
m Route marked for no mtu discovery
N Route marked for no forwarding
R Host or net unreachable
S Manually added
T Route marked for exact or better type-of-service matching
U Route usable
X External daemon translates protocol to link address

Direct routes are created for each interface that is attached to the local host. The gateway field for direct routes shows the address of the outgoing interface. Some interfaces, such as Ethernet, also use link-level routes (e.g., see `arp`(8)). In those cases, the direct network route has the `C` flag set (for cloning), which causes individual host routes to be created on demand for hosts on that network. The host routes contain link-level *gateway* entries with their link-level addresses, and the `L` (link-level) flag is set. The `refcnt` field shows the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface used for the route. If the `-v` option is also specified, additional information is provided. The *Admin MTU* field specifies the *mtu* the administrator set when the route was created using the `route`(8) command. The *Path MTU* field specifies the *mtu* discovery path *mtu* for the route. If *Path MTU* discovery has changed since the route was first installed, an asterisk (*) marks the path mtu. If the route was marked for *type-of-service*, security, or a group ID list, that information also is printed as appropriate. The group ID list is printed with a plus or minus (+/-) to indicate that the group ID list is inclusive (+) or exclusive (-). If the `-vv` option is specified, the routing metrics stored with the kernel are displayed. If the `-vvv` option is specified when examining a system dump, the kernel radix table is displayed. See the `net/radix.h` and `net/route.h` files for an explanation of the fields.

The `-A` option can be used when examining a system dump. The `-A` option causes the kernel memory addresses of route data structure to be printed. These addresses can be used to reconstruct the route radix tree.

`-s` [`-i`] [`-r`] Shows per-protocol statistics. To condense information on the screen, `netstat` does not display some entries if they have a 0 value. If `-r` also is used, statistics on the routing tables are given. If `-i` also is used, statistics on the interfaces are shown. If `-v` is used, all entries are displayed, whether or not they have a value of 0. With the `-iv` options, the interface statistics are gathered from the `ifcray` and `ifnet` data structures defined in the `crayif/ifc.h` and `net/if.h` files.

`-I` *interface* Shows information only about this interface; use with an *interval* argument as described with the `-n` option.

interval When `netstat` is invoked with an *interval* argument, it displays a running count of statistics that are related to network interfaces, and it pauses *interval* seconds before refreshing the screen. This display consists of a column that summarizes information for all interfaces, and a column that highlights the interface that has had the most traffic since the system was last rebooted. The first line of each screen of information summarizes information since the system was last rebooted, and subsequent lines of output show values that have accumulated during the preceding interval.

If a socket address specifies a network but no specific host address, address formats are of the form *host*`.`*port* or *network*`.`*port*. When known, the host and network addresses are displayed symbolically according to files `/etc/hosts` (see `hosts`(5)) and `/etc/networks` (see `networks`(5)), respectively. If a symbolic name for an address is unknown, or if the `-n` option is specified, the address is printed in the Internet dot format; see `inet`(3C) for more information about this format. Unspecified or *wildcard* addresses and ports appear as an asterisk (`*`).

BUGS

The errors are not well-defined. Collisions have no meaning for any of the interfaces that Cray Research supports. Some of the kernel tables are very transitory in nature, and `netstat` can become confused when used on a running system; either `???` is printed, or `netstat` might become hung up in an infinite loop.

EXAMPLES

Example 1: The following example shows the current routing entries for family 2 (`inet`).

```
$ netstat -r
Routing tables
Destination      Gateway            Flags     Refs     Use  Interface
default          core02-f20         UG         55   469940  fd0
loopback         localhost          UR          0        0  lo0
localhost        localhost          UH         10     5160  lo0
cray-hyp/24      cool               U           9    34665  np1
cray-fddi/24     link#18            UC          0        0  fd0
haze             0:0:77:85:29:df    UHL         0        6  fd0
gust-ip-fddi     0:0:a9:2:4:f       UHL         0        0  fd0
thunder          0:0:77:85:29:f7    UHL         2       61  fd0
cool-fddi        0:40:a6:d:4e:2     UHL         6     2510  lo0
sn5607-fddi      0:0:77:16:65:41    UHL         2   151945  fd0
ice-ip-fddi      0:0:a9:2:0:7e      UHL         0        0  fd0
ice              0:40:a6:d:4e:3     UHL         0        2  fd0
squall-ip-fddi   0:0:a9:2:1:e9      UHL         0        0  fd0
core01-f20       0:0:c:e:70:58      UHL         0        0  fd0
core02-f20       0:0:c:15:cc:ae     UHL         2        0  fd0
latte-alpha      core02-f20         UGHD        0      438  fd0
cool-030net      cool-030           U           7     2699  np0
lo0_multicast    localhost          UH          0        1  lo0
```

Example 2: The following example shows the currently configured interfaces.

```
$ netstat -ia
Name   Mtu   Network     Address      Ipkts   Ierrs Opkts   Oerrs
np0    16432 <Link>                    3780   0     3777    0
np0    16432 cool-030net cool-030      3780   0     3777    0
np1    16432 <Link>                   36097   0    35744    0
np1    16432 cray-hyp/24 cool         36097   0    35744    0
np2*   16432 <Link>                       0   0        0    0
np3*   16432 <Link>                       0   0        0    0
np4*   16432 <Link>                       0   0        0    0
np5*   16432 <Link>                       0   0        0    0
np6*   16432 <Link>                       0   0        0    0
np7*   16432 <Link>                       0   0        0    0
np8*   16432 <Link>                       0   0        0    0
np9*   16432 <Link>                       0   0        0    0
np10*  16432 <Link>                       0   0        0    0
np11*  16432 <Link>                       0   0        0    0
np12*  16432 <Link>                       0   0        0    0
np13*  16432 <Link>                       0   0        0    0
np14*  16432 <Link>                       0   0        0    0
np15*  16432 <Link>                       0   0        0    0
hi0*   16432 cray-hyp    sn131          7477   0     6163    0
fd0    4352  <Link>                  1039868   0   801090    1
fd0    4352  cray-fddi/2 cool-fddi   1039868   0   801090    1
lo0    65535 <Link>                    7670   0     7670    0
lo0    65535 loopback    localhost     7670   0     7670    0
```

FILES

`crayif/ifc.h`	Kernel include file that defines the `ifcray` data structure
`net/if.h`	Kernel include file that defines the `ifnet` data structure
`net/radix.h`	File that defines fields in the kernel radix table
`net/route.h`	File that defines fields in the kernel radix table

SEE ALSO

`inet`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`hosts`(5), `networks`(5), `protocols`(5), `services`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

newacct – Changes account ID

SYNOPSIS

```
newacct [-a[user]]
newacct [-l]
newacct [account-name]
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The newacct utility changes the account ID of the calling shell.

The newacct utility accepts the following options:

-a [*user*]
: Outputs the valid account names and account IDs for *user*. If *user* is not specified, the owner of the current process is the default *user*.

-l
: Prints the current account name.

account-name
: Validates whether you have permission to change the account ID to the requested value. If you omit this argument, newacct changes the account ID of the calling process to the default (primary) account ID of the user.

NOTES

If the fair-share scheduler has been activated on your system in the share by account ID mode, newacct also attaches the current session or job to the new account resource group. This may change the amount of system resources that you are allowed.

EXAMPLES

To list the account names and account IDs for jdoe, enter the following:

```
newacct -a jdoe
```

FILES

/etc/udb
: User validation file that contains user control limits

SEE ALSO

`acctid`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`getpwent`(3C), `id2nam`(3C), `putpwent`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`passwd`(5), `udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

newaliases – Rebuilds the sendmail(8) alias database

SYNOPSIS

```
newaliases
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The newaliases utility rebuilds the random access data base for the mail aliases file /usr/lib/aliases. It must be run each time this file is changed for the change to take effect.

The newaliases utility is identical to sendmail -bi.

The newaliases utility exits with a value of 0, if successful, and >0 when an error occurs.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	Shell-redirected I/O is not subject to security label restrictions.

If the PRIV_SU configuration option is enabled, for the super user, shell-redirected I/O is not subject to security label restrictions.

SEE ALSO

aliases(5) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

sendmail(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

newgrp – Changes to a new group

SYNOPSIS

newgrp [-l] [*group*]

Obsolescent version; may not be supported in future releases:

newgrp [-] [*group*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The newgrp utility changes a user's real and effective group ID. The user remains logged in and the current directory is unchanged. The user is always given a new shell, replacing the current shell with newgrp, regardless of whether it terminated successfully or due to an error condition (that is, unknown group).

Exported variables retain their values after invoking newgrp; however, all unexported variables are either reset to their default value or set to null. System environment variables (such as PS1, PS2, PATH, MAIL, and HOME), unless exported by the system or explicitly exported by the user, are reset to default values. For example, a user has a primary prompt string (PS1) other than $ (default) and has not exported PS1. After an invocation of newgrp, successful or not, the user's PS1 will be set to the default prompt string $. The shell command export (see sh(1)) is the method to export variables so that they retain their assigned value when invoking new shells.

With no arguments, newgrp changes the user's group IDs (real and effective) back to the group specified in the user's password file entry. This is a way to exit the effect of an earlier newgrp utility.

The newgrp utility accepts following option and operand:

-l
- (Obsolescent)
  Changes the environment to what would be expected if the user actually logged in again as a member of the new group.

group Identifies a group name from /etc/group or a numeric group ID. Both specify the group ID to which the real and effective group IDs are set.

If the group has a password and the user is not listed in `/etc/group` as being a member of that group, a password is required.

NOTES

On Cray Research systems, you cannot use the `newgrp` utility to create new files and directories under a different group. It also does not have an effect on UNICOS accounting or security features. The utility is provided for POSIX and XPG4 compliance.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to change to any group. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to change to any group. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to change to any group. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

If `newgrp` succeeds in creating a new shell execution environment, whether or not the group identification was changed successfully, the exit status is the exit status of the shell. Otherwise, the `newgrp` utility exits with the following value:

>0 An error occurred.

FILES

`/etc/group`	System's group file
`/etc/passwd`	System's password file

SEE ALSO

`login`(1), `sh`(1),

`group`(5), `passwd`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

news – Prints news items

SYNOPSIS

news [-a] [-n] [-s] [*items*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The news utility keeps users informed of current events. By convention, these events are described by files in the /usr/news directory.

When invoked without arguments, news prints the contents of all current files in /usr/news, most recent first, with each preceded by an appropriate header. news stores the ‘‘currency’’ time as the modification date of a file named .news_time in the user’s home directory (the identity of this directory is determined by environment variable $HOME); only files more recent than this currency time are considered ‘‘current.’’ When *items* is specified, the modification time of .news_time is not updated.

The news utility accepts the following options:

-a Prints all items, regardless of currency. In this case, the stored time is not changed.

-n Reports the names of the current items without printing their contents and without changing the stored time.

-s Reports the number of current items that exist, without printing their names or contents, and without changing the stored time.

items When specified, the modification time of .news_time is not updated. It is useful to include such an invocation of news in your .profile file, .cshrc file, or in the system’s /etc/profile or /etc/cshrc file to report whether there is news.

All other arguments are assumed to be specific news items to be printed. These items must match the names of files found in the /usr/news directory. If an item is not found, news continues with the next item in the list.

If an interrupt signal is received during the printing of a news item, printing stops and the next item is started. Another interrupt within 1 second of the first causes the program to terminate.

FILES

/etc/.cshrc	C shell default start-up file
/etc/profile	Korn shell default start-up file
/usr/news/*	Directory containing news item files

SEE ALSO

`cshrc`(5), `profile`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`nfsid` – Performs NFS authorization functions to NFS servers

SYNOPSIS

`nfsid` [[*options*] [*host* ...]] ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `nfsid` utility uses Kerberos authentication to present your credentials to an network file system (NFS) server, which allows your workstation to access files that reside in file systems exported only for Kerberos authenticated access. The default permissions for a client workstation that is attempting to access an NFS file system are those of the user "nobody," meaning that a program that is running on the workstation usually would not be allowed to modify any NFS server files. `nfsid` sends a Kerberos authentication ticket to the server, which records an association between a workstation's IP address and the user's user ID (UID).

The `nfsid` utility accepts the following arguments:

options Specifies the following arguments to modify the default behavior of the `nfsid` utility. All arguments are processed in the following order. Thus, if a `-u` option is followed by a `-m` option, the `-m` option takes precedence.

`-map` or `-m`	Sets the mapping function to `map user`. This is the default; it establishes a mapping to a server. It requires that the user be authenticated.
`-unmap` or `-u`	Sets the mapping function to `unmap user`. This argument removes a mapping from a server.
`-purge` or `-p`	Sets the mapping function to `purge host`. This argument removes all mappings associated with the user's host.
`-purgeuser` or `-r`	Sets the mapping function to `purge user`. This argument removes all mappings associated with the user on the host. It requires that the user be authenticated.
`-verbose` or `-v`	Displays verbose information about the mapping operation. This is the default.
`-quiet` or `-q`	Indicates that verbose information must not be displayed.
`-debug` or `-d`	Prints debugging information. This argument is not usually useful to users.

host Specifies a list of one or more hosts (either names or Internet addresses).

EXIT STATUS

If `nfsid` is executed with only one *host* argument, the exit status is one of the following:

Exit Status	Description
0	No error encountered
1	Bad arguments
3	Internal fatal error
10	Kerberos failure
11	Host communication failure
12	Authentication failure
13	No reserved ports available
21	Host name could not be resolved

If an error is encountered while one host in the list is being manipulated, processing continues with the other hosts. The exit status is returned after all hosts are attempted.

EXAMPLES

The following example establishes a mapping for the user on the host `CHARON.MIT.EDU` and deletes a mapping for the user on the host at Internet address `18.72.0.6`.

```
nfsid -m CHARON.MIT.EDU -u 18.72.0.6
```

LIMITATIONS

It is important to understand that the current implementation of NFS is not entirely secure when it uses the Kerberos authentication, including `nfsid`, `mountd`, and the UNICOS kernel. The current implementation assumes that the client hosts (those machines on which the users run `nfsid`) are single-user systems. Also, it assumes that no machine on the same subnet as the client host can be reconfigured to run with the same Internet address as the client host.

The kernel caches user ID/IP address pairs as directed by the `mountd` server, which responds to `nfsid` requests. When the client makes an NFS request to the server, the user ID/IP address pair from the NFS header is checked against the entries in the kernel. The problem with this method is that there is nothing to prevent any other user on the same client from creating the same NFS request, and filling in a false UID field. If this request is issued from the same IP address as that with which the client is registered in the server kernel, the server cannot detect that this request is not genuine. There is nothing in the NFS request itself that proves to the server that the request is genuine.

This is not to say that the `nfsid` scheme is worthless. It limits potential attacks to periods when the user ID/IP address pair is mapped in the server kernel. However, it is important to realize that the security that `nfsid` offers is limited when it is run from a multiuser host.

NAME

nice – Invokes a utility that has an altered scheduling priority

SYNOPSIS

nice [-n *increment*] *utility* [*arguments*]

Obsolescent version: may not be supported in future releases:

nice [-*increment*] *utility* [*arguments*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The nice utility invokes a *utility*, requesting that it be run with a different CPU scheduling priority.

If you omit *increment*, a default value of 10 is used.

The nice utility accepts the following options:

-n *increment*
-*increment* (Obsolescent)
: Specifies how the system scheduling priority of the executed utility is adjusted. The *increment* argument is a positive or negative decimal integer used to modify the system schedule priority of the executed utility.

  A positive *increment* value causes a lower or unchanged system scheduling priority. Any user can lower scheduling priority. A negative *increment* value causes higher or unchanged system scheduling priority. Only an appropriately authorized user can raise scheduling priority.

  If the requested *increment* would raise or lower the system scheduling priority of the executed utility beyond the nice value limits, then the limit whose value was exceeded is used. Nice values range from 0 (highest priority) through 39 (lowest priority).

utility [*arguments*]
: Specifies the utility or script to run at lowered priority.

NOTES

The csh(1) utility has a built-in nice utility that has slightly different characteristics. See csh(1).

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to raise or lower the scheduling priority for any file.
`sysadm`	Allowed to raise or lower the scheduling priority for any file, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to raise or lower the scheduling priority for any file.

EXIT STATUS

If you invoke *utility*, the exit status of `nice` is the exit status of *utility*; otherwise, the exit status is one of the following:

1–125 An error occurred in the `nice` utility.

126 The utility specified by *utility* was found, but it cannot be invoked.

127 The utility specified by *utility* cannot be found.

SEE ALSO

`csh`(1), `renice`(1), `sh`(1)

`nice`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

nl – Line-numbering filter

SYNOPSIS

nl [-b *type*] [-d *xx*] [-f *type*] [-h *type*] [-i *incr*] [-l *num*] [-n *format*] [-p] [-s *sep*] [-v *start*] [-w *width*] [*file*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The nl utility reads lines from *file* or standard input if *file* is not named and reproduces the lines on standard output. Lines are numbered on the left according to the command options in effect.

The nl utility views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page, which consists of a header, a body, and a footer section. Empty sections are valid. Different line-numbering options are independently available for header, body, and footer (for example, no numbering of header and footer lines while numbering blank lines only in the body).

The start of a logical page section is signaled by input lines containing nothing but the following delimiter character(s):

Line Contents	Start Of
\:\:\:	Header
\:\:	Body
\:	Footer

Unless otherwise specified, nl assumes that text being read is in a single logical page body.

Command options may appear in any order and may be intermingled with an optional file name. Only one file may be specified.

The nl utility accepts the following options:

-b *type* Specifies the logical page body lines to be numbered. Recognized types and their meanings are as follows:

- a Numbers all lines.
- t Numbers only lines with printable text; default.
- n Does not number lines.

p*string* Numbers only lines that contain the regular expression specified in *string*.

`-d` *xx* The delimiter characters specifying the start of a logical page section may be changed from the default characters (\:) to two user-specified characters. When only one character is entered, the second character remains the default character (:). To enter a backslash, use two backslashes.

`-f` *type* Same as `-b` *type* except for footer. Default for logical page footer is `n` (no lines numbered).

`-h` *type* Same as `-b` *type* except for header. Default *type* for logical page header is `n` (no lines numbered).

`-i` *incr* The increment value used to number logical page lines. Default is 1.

`-l` *num* The number of blank lines to be considered as one. For example, -l2 results in only the second adjacent blank being numbered (when the appropriate `-ha`, `-ba`, and/or `-fa` option is set). Default is 1.

`-n` *format* The line-numbering format. Default is `rn` (right-justified). Recognized *format* values are as follows:

`ln` Left-justified, leading zeros suppressed.

`rn` Right-justified, leading zeros suppressed.

`rz` Right-justified, leading zeros kept.

`-p` Does not restart numbering at logical page delimiters.

`-s` *sep* The character(s) used in separating the line number and the corresponding text line. Default *sep* is a tab.

`-v` *start* The initial value used to number logical page lines. Default is 1.

`-w` *width* The number of characters to be used for the line number. Default is 6.

file File to be filtered.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to read from any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to read from any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to read from any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `nl` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

The following command line numbers `file1`, starting at line number 10, with an increment of 10:

```
nl -v10 -i10 -d!+ file1
```

The logical page delimiters are `!+`.

SEE ALSO

`pr`(1)

NAME

nlimit – Queries and modifies resource limits

SYNOPSIS

nlimit [-a *action*] [-c *cat*] [-i *id*] [-l *value*] [-q] *resource* [-t *type*]

nlimit [-q]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nlimit utility allows the inquiry and setting of resource limits. Currently, only CPU time is supported as a valid resource.

The output contains all pertinent limit values, and when applicable, the action the system will take upon reaching a hard limit and the amount of resource consumed. In addition, when a limit value is unlimited, the output will print the message unlimited.

The nlimit utility accepts the following options:

-a *action* The action to take upon reaching a hard limit. When using the -a option, the -t option must be set to hard. Valid actions are as follows:

term or terminate	Terminate process
check or checkpoint	Checkpoint process and then terminate
null	Do not change action

-c *cat* Identifies which category of resource is to be queried or set. Acceptable values are as follows:

proc or process	Process limits
sess or session	Session limits
user or uid	User limits
sessprocs	Default process limits for the session

-i *id* The *pid*, *sid*, or *uid* correlating to the *cat* argument. Zero indicates the current *pid*, *sid*, or *uid*. The *cat* option of sessprocs requires an *sid*.

-l *value* The value of the new limit to be set. Acceptable values are as follows:

n	Positive integer. For CPU resources, this value represents seconds. (See the CAUTIONS section.)
unlimited	This specifies unlimited resources.
-1	Minus one. Do not change current limit. For use with changing the hard action but not changing the limit value.

-q Quiet mode. No headers will be printed with the output.

resource The resource you want to inquire about or change. Valid options are as follows:

`cpu` CPU resources. (See the CAUTIONS section.)

`-t` *type* Identifies the type of limit to be set or viewed. Valid options are as follows:

`abs` or `absolute`	Absolute limit (only the super user can set this limit)
`hard`	Hard limit
`soft`	Soft limit

Only an appropriately authorized user can increase resource limits. Only an appropriately authorized user can set the resource limits of another user, process, or session.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	**Action**
`system`, `secadm`	Allowed to raise or lower the resource limits of any user, process, or session.
`sysadm`	Allowed to raise or lower the resource limits of any user, process, or session, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to raise or lower the resource limits of any user, process, or session.

CAUTIONS

The CPU time limit does not apply when running as root.

EXAMPLES

The following examples show the inquiry and setting of resources. User input is shown in bold type.

Example 1: The following example shows the inquiry of all CPU limits.

```
$ nlimit cpu
RESOURCE   CATEGORY    ABSOLUTE        HARD        SOFT       ACTION    USED
-------------------------------------------------------------------------------
cpu:        process   unlimited         600         600   checkpoint       0
cpu:        session   unlimited       14400       14400    terminate     112
cpu:      sessprocs           0         600           0   checkpoint       0
cpu:            uid   unlimited   unlimited   unlimited    terminate   32935
```

Example 2: The following example shows the resetting of a CPU hard limit for process 1674.

```
$ nlimit -c process -i 1674 -t hard -l 2000 cpu
```

Example 3: The following example shows the inquiry of process 1674 (notice the change in hard limit value from the previous `nlimit` command).

```
$ nlimit -c process -i 1674 cpu
CATEGORY    ABSOLUTE        HARD        SOFT       ACTION    USED
-------------------------------------------------------------------
process    unlimited        2000         600   checkpoint      91
```

FUNCTIONAL DESCRIPTION

This section provides an overview of resource limits. Resource limit control is an extension of standard UNIX; therefore, it is not in effect unless explicitly specified. This ensures compatibility across UNIX systems and upward compatibility among UNICOS releases. To enable the resource limits feature, a user must explicitly use the `nlimit` utility, or the `nlimit`(3C) or `NLIMIT`(3F) library routines. CPU limits can be applied to sessions or processes running in foreground, background, or submitted through the Network Queuing Environment (NQE). Limits can also be applied against a user ID (UID).

RESOURCE LIMIT TYPES

There are three CPU limit types: absolute, hard, and soft. The system administrator defines the absolute limit; it cannot be modified by the user. Users define hard and soft limits. Each limit can be applied to three different categories: user, session, and process. Another category, `sessprocs`, can only have a hard limit defined.

Absolute Limit Type

The absolute limit type is the system administrator-defined limit and is the maximum amount of a system resource available to the user. This limit could be the maximum of the system resource or some system administrator-defined value less than the maximum system resource. Reaching this limit will cause the user program to terminate.

Hard Limit Type

The hard limit is user-defined, and can be set equal to or less than the absolute limit value. Associated with the hard limit is a user-definable default action for when the hard limit is exceeded. The action is either "terminate" or "checkpoint and terminate." By default, if a user process reaches a hard limit, the program will terminate. The hard limit action determines whether the program will attempt to be checkpointed before termination.

Soft Limit Type

The soft limit is user-defined and can be set equal to or less than the hard limit value. Upon hitting a soft limit, a `SIGINFO` signal is sent to the program. The user can register the program to catch the signal, using `getinfo`(2), query the reason for the signal, and then take some action based on the reason for the signal.

RESOURCE LIMIT ACTIONS

This section describes the actions taken when the absolute, hard, and soft limits are reached for processes, sessions, and users.

Process Absolute Limit

If the absolute limit is exceeded on a process, that process is sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and is terminated.

Session Absolute Limit

If the absolute limit is exceeded on a session, then all processes related to that session are sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and are terminated. If the user is logged on interactively and their login shell is part of the session that exceeded the limit, the login shell will be terminated and the user will be logged off the system.

User Absolute Limit

If the absolute limit is exceeded on a user, then all processes owned by that user are sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and are terminated. If the user is logged on interactively, the user will be logged off the system. User access to the system will be denied if the user absolute limit has been reached. The system administrator must either raise the absolute limit or set the user's accumulated CPU to a lower value.

Process Hard Limit

If the hard limit is exceeded on a process, the default hard action is checked. If the default hard action is to terminate, the process is sent the `SIGCPULIM` signal followed by the `SIGKILL` signal and is terminated. If the default hard action is to checkpoint and terminate, then the system attempts to checkpoint that process before termination. If a checkpoint file cannot be created, a standard core file will be created. For Fortran users wanting a checkpoint file by using the hard action of checkpoint, set the environment variable `TRACEBK` to 0 prior to running your application.

Session Hard Limit

If the hard limit is exceeded on a session, the default hard action is checked. If the default hard action is to terminate, all processes related to that session are sent the `SIGCPULIM` signal followed by the `SIGKILL` and are terminated. If the default hard action is to checkpoint and terminate, then the system attempts to checkpoint all processes related to the session before termination. Because checkpoint creates a file called `core`, it is possible when checkpointing multiple processes in the same directory to overwrite previous `core` files. If the user is logged on interactively and their login shell is part of the session that exceeded the limit, the login shell will be terminated and the user will be logged off the system.

`sessprocs` Hard Limit

The `sessprocs` limit lets the user set up default hard process limits per session. If a `sessprocs` limit is set, all processes created in this session have the hard and soft limit set to the `sessprocs` limit. Two values can be set by `sessprocs`: the hard limit and the hard limit action. Because this sets up process hard limits, the action resulting from hitting a `sessprocs` limit is the same as hitting a process hard limit.

User Hard Limit

If the hard limit is exceeded on the user category, the default hard action is checked. If the default hard action is to terminate, all processes owned by that user are sent the SIGCPULIM signal followed by the SIGKILL signal and are terminated. If the default hard action is to checkpoint and terminate, the system attempts to checkpoint all processes owned by the user before termination. If the user is logged on interactively, the user will then be logged off the system. User hard limits are valid only as long as the user has an active process on the system.

Process Soft Limit

If the soft limit is exceeded on a process, the SIGINFO signal is sent to that process. This SIGINFO signal can be trapped by the user. It is up to the user to catch the SIGINFO signal by registering for the receipt of this signal. If the user has registered for the SIGINFO signal, control is returned to the user program. If the user has not registered for the SIGINFO signal, no signal is sent and the process continues to execute until the hard limit is reached or the session completes. There could be multiple reasons for the posting of a SIGINFO signal. Therefore, after the soft limit signal has been received, the user should query the reason for the soft signal by using the getinfo(2) system call.

Session Soft Limit

If the soft limit is exceeded on a session, all processes related to that session that have registered for the SIGINFO signal are sent the SIGINFO signal.

User Soft Limit

If the soft limit is exceeded on the user category, all processes owned by that user having registered for SIGINFO are sent the signal. If the user is logged on interactively, the user will not notice this limit being reached. User soft limits are valid only as long as the user has an active process on the system.

DEFAULT LIMITS

Absolute limits (process, session, and UID) are set at login time from values in the user database (UDB). The UID limits are in effect only if the fair-share scheduling feature is enabled. The process, session, and UID hard and soft limits are set equal to the absolute limit at login time. The sessprocs hard limit value is set equal to the process absolute limit. The default hard action is to terminate.

TYPE CHECKING PRECEDENCE

The absolute limit takes precedence over the hard and soft limit. The hard limit takes precedence over the soft limit if they are set equal. This means if all limits are the same (soft = hard = absolute) and the limits are exceeded, the absolute action will be followed (terminate). If a hard limit is specified but no soft limit, the soft limit is set equal to the hard limit. If no default hard action is specified, the default hard action will be to terminate.

CATEGORY CHECKING PRECEDENCE

Process limits will be checked before session limits and session limits checked before user limits.

EXPLICITLY SETTING LIMITS

Two ways exist to view, set, and modify limits: through the `nlimit`(1) utility and through the `nlimit`(3C) and `NLIMIT`(3F) library interface. The hard and soft limit values can be raised and lowered as long as the limit does not exceed the absolute limit.

IMPLICITLY SETTING DEFAULT LIMITS UPON LOGIN

Although the user must explicitly define the hard and soft limits for them to be in effect, the `nlimit` utility can be added to the `.login`, `.cshrc`, or `.profile` scripts. This lets the user set up default limits without having to explicitly define them upon every login. Another means for implicitly setting limits is to use the `sessprocs` category. The setting of a `sessprocs` hard limit implies that all processes created in the session will have this limit.

LIMIT CASCADING

Any new process created will inherit their parents session and UID limits. Process limits will be inherited either from their parent or from the `sessprocs` value. If the `sessprocs` limit is unlimited, the child will inherit the parent process limits. If the `sessprocs` limit is not unlimited, the child will inherit the `sessprocs` limit.

RESTARTING A LIMIT-INDUCED RESTART FILE

To restart a checkpoint file that was created due to reaching a hard limit, the current process limits must be greater than the process limits of the checkpoint file.

ACCUMULATED RESOURCE USAGE

The `nlimit` utility and the `nlimit`(3C) and `NLIMIT`(3F) library routines can return the current amount of CPU time accumulated for a process, session, and user.

SEE ALSO

`limit`(1)

`getlim`(2), `setlim`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`NLIMIT`(3F) in the *Application Programmer's Library Reference Manual*, Cray Research publication SR–2165

`nlimit`(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

UNICOS Shells Ready Reference, Cray Research publication SQ–2116

NAME

nm – Prints name list

SYNOPSIS

nm [-a] [-A] [-c] [-e] [-f] [-g] [-L] [-m] [-n] [-o] [-P] [-t *format*] [-u] [-v] [-V] [-x] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-e and -V options)
BSD extensions (-n)
CRI extensions (-a, -c, -f, and -m options)

DESCRIPTION

The nm utility prints the name list (entries and external files) of each object *file* specified on the command line.

There are two forms of output. The -P flag results in the POSIX standard portable output, described later. Absent that flag, the traditional UNICOS output is produced. Each symbol name is preceded by its value or its size (blanks if undefined) and by its type (see the end of this section).

The nm utility accepts the following options:

-a Prints the variable names and the label names found in module or common block debug tables.

-A Causes the full pathname or library name to be prepended to the symbol name.

-c Adds secondary identification characters to reflect Cray Research enhancements to relocatable table formats. For a description of these characters, see the end of this section.

-e Prints only the defined symbols.

-f Produces full output. Writes redundant synbols that are normally suppressed.

-g Prints only global symbols. If the -g option is not present, the local block names (for example, X$DATA) are printed.

-L **What is this?**

-m Appends the module name to the file name. (Not effective if the -P option is selected.)

-n Sorts numerically rather than alphabetically.

-o Writes numeric values in octal (equivalent to -t o).

-P Prints in the POSIX standard portable format.

`-t` *format*
Writes each numeric value in the specified format. The format is dependent on the single character used as the *format* option-argument:

`d` Writes the offset in decimal.

`o` Writes the offset in octal.

`x` Writes the offset in hexadecimal.

`-u` Prints only undefined symbols.

`-v` Sorts output by value instead of alphabetically.

`-V` Prints the version of the program on `stderr`.

`-x` Prints numeric values in hexidecimal (equivalent to `-t x`).

In the POSIX portable output, the symbol name is followed by a character describing its type (see below), its value and, if appropriate, by its size. Both the number and size are represented in the number base determined by the `-t` flag: `-t d` produces decimal, `-t o` octal, and `-t x` hexadecimal. If `-t` is not specified, the number and size are represented in hexadecimal.

If the `-A` flag is specified, the symbol name is preceded by the file name, followed by a library member name enclosed in square brackets if appropriate, followed by a colon.

Symbols are written sorted by symbol name, collated as determined by the current locale, unless the `-v` flag is specified, in which case symbols are sorted by value.

If the `-u` flag is specified, only undefined symbols are written.

Symbol types are:

`A` Global absolute
`B` Global bss segment symbol
`b` Local bss segment symbol
`C` Common block
`D` Global data segment symbol
`d` Local data segment symbol
`L` Block resides in local memory
`T` Global text segment symbol
`t` Local text segment symbol
`U` Undefined

If the `-c` option was specified, the following characters describe additional information on relocatable formats:

`b` A bss block or common block.
`e` A common block that is also an entry point.
`s` A soft external.
`x` An external passed as an address by a Fortran routine.
`z` A bss block or common block that may be preset to 0.

EXIT STATUS

The nm utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following example searches two libraries for a specific global symbol:

```
nm -go /lib/lib1.a /lib/lib2.a | grep symbol_name
```

Example 2: This example lists all the references that must be satisfied from libraries after the files in a development directory have been successfully compiled into relocatables:

```
nm -g *.o | sort | uniq
```

SEE ALSO

ar(1) for archive and library maintainer for portable archives
bld(1) to maintain relocatable libraries
lorder(1) to find ordering relation for an object library

ar(5) for archive file format
relo(5) for relocatable object table format under UNICOS in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014.

NAME

nmake – Maintains and updates programs

SYNOPSIS

nmake [-b] [-c] [-d *n*] [-f *makefile*] [-g *makefile*] [-h] [-i] [-j *n*] [-k] [-l] [-m] [-n] [-o] [-q] [-r] [-s] [-t] [-u] [-v] [-x] [-A] [-D*name*[=*value*]] [-F] [I*directory*] [-M] [-O] [-R] [-S] [-T] [-U*name*] [-V] [-X] [*targets*]

nmake -b -f -l -r *base*.ms

nmake -l -f /dev/null

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

This man page describes the function of /usr/bin/nmake. The nmake utility reads the input *makefile* and executes shell commands (see sh(1)) to update one or more *targets*. Each makefile contains a sequence of entries that specify dependencies. An entry consists of a nonempty list of targets and a list of dependency rules for the target, separated by a colon (for example, *target : dependency rules*). A target is updated if at least one of its dependencies was modified since the target was last modified or if the target does not exist.

The nmake utility interacts only with sh(1); it does not function with csh(1).

The makefiles are passed through cpp(1) or /usr/lib/make/cpp (for system builds) and are compiled into object files before any actions are taken. If a makefile was not modified since the last nmake, the corresponding object file is used. Each nmake object file is placed in *base*.mo; *base* is the base name (suffix deleted) of the corresponding makefile.

You can find further guidelines for using this utility for system builds in *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303.

The nmake utility accepts the following options.

Built-in attributes that match the regular expression:

```
.[.A-Z][.A-Z0-9]*
```

User attributes that match the regular expression:

```
.[.A-Z][.a-z0-9]*
```

Intermediate targets that match the regular expression:

```
.[.a-z][.a-z0-9]*
```

Use `${!:-$$}` to generate temporary file names.

Options are preceded by a `-` or `+` and can appear anywhere on the command line. `+` turns off the corresponding option. Command-line options override options specified in the makefile. Options not in the following list can be interpreted by local versions of the built-in rules. The options are as follows:

`-b` Does not use the built-in rules.

`-c` Forces the makefiles to be compiled into one `nmake` object file. `nmake` exits after the files are compiled.

`-d` *n* Provides a step-by-step dump of `nmake`'s actions. The optional number argument, *n*, selects the dump level; higher levels produce more output. The debug levels and the objects that are traced are as follows:

0 No trace
1 Basic `nmake` trace
2 Major actions
3 Coshell messages
4 State variables, `M` information
5 Makefile input lines
6 Directory and archive scan information
7 Detailed hash table information
8 Built-in rule input lines
9 Lexical analyzer states

`-f` *makefile* Reads the descriptions in *makefile*. If the `-f` option is not specified, the files specified by the dependencies of the `.MAKEFILES` rule are tried in order from left to right. If *makefile* is `-`, the standard input is read. More than one `-f` option can appear; the files are read in order from left to right. If no `-f` option is specified and no default *makefile* is found, suffix `.mk` is appended to the base name of the first argument in the *makefile* dependency files and this file is read. If no *makefile* can be found, an error occurs.

`-g` *makefile* Treats *makefile* like a global makefile, similar to the `-f` option. This means that the default *makefile*s are still tried if no `-f` option appears.

`-h` Does not automatically search files with suffixes marked by `.SEARCH` for `#include` header file dependencies.

`-i` Ignores command error codes from the shell.

`-j` *n* Specifies that `nmake` can execute up to *n* command update jobs concurrently. If omitted, *n* defaults to 1 (when using the install tool, the default is 3).

`-k` If the update commands for the current target return nonzero status, `nmake` continues working on targets that do not depend on the current target. This is the same functionality as the `.DONTCARE` built-in rule.

`-l` Lists variable and rule definitions after the *makefile*s are read. The targets are neither checked nor updated, only listed.

`-m` Moves each target to the directory of the corresponding primary source file, which is determined by using the implicit suffix rules. Usually, targets are placed in the current directory. Archive members and targets with explicit directory names are not moved.

`-n` Traces and prints, but does not execute, the target update commands. This option also inhibits *makefile* compilations.

`-o` Outputs intermediate dependency information.

`-q` Does not update any targets but exits with a value of 0 if the targets are up-to-date; otherwise, it exits with a value of –1.

`-r` Lists the detailed status of each rule after all targets are made. If the `-l` option is also set, the listing occurs before any targets are made and `nmake` exits without making any targets.

`-s` Executes but does not print the update commands. This is the same functionality as the `.SILENT` built-in rule.

`-t` Touches the modify date of targets, bypassing the update commands. Only existing targets are touched. Targets that have state variable dependencies can be touched only by use of this option. This is the same functionality as the `.TOUCH` built-in rule.

`-u` Does not force file bindings to be unique and does not warn about duplicate files.

`-v` Lists variable assignments. Useful in conjunction with the `-d` option.

`-x` Does not check implicit file dependencies generated by a previous `nmake`.

`-A` Accepts any existing targets as being up-to-date, even when the targets have state variable dependencies; however, targets out-of-date with file dependencies are always updated.

`-D`*name*[=*value*]
Defines *name* as though defined by a `#define` directive. If [=*value*] is omitted, *name* is defined as 1. No space is allowed between `-D` and *name*.

`-F` Forces all targets to be updated.

`-I`*directory* Passes parameters to `cpp` when *makefile*s are read. `nmake` first searches for `#include` files that are enclosed in quotation marks and do not begin with a / symbol in the current directory; the directories specified by the `-I` option are searched next, and then the directories specified on the standard list are searched. No space is allowed between `-I` and *directory*.

`-M` Causes `nmake` to act as if no command-line target was made.

`-O` Overrides explicit command update blocks by applying only implicit command update blocks.

`-R` Forces the input makefiles to be read rather than loading the corresponding `nmake` object files.

`-S` Ignores previous state variable definitions.

`-T` Enables testing code. Do not use this option unless you are familiar with the source code and you know what the current test is.

`-U`*name* Removes any initial definition of *name*; *name* is a predefined reserved name. The `-U` option overrides the `-D` option. The `#assert` directive makes reserved preprocessor names obsolete. No space is allowed between `-U` and *name*.

`-V` Lists the current `nmake` version.

`-X` Clears the special rules `.SOURCE.a` and `.SOURCE.h`.

targets The file to be written to.

`-b -f -l -r`
Lists the contents of the state variable file.

`-l -f /dev/null`
Lists the default rule and variable definitions.

Makefiles

You can specify commands in a makefile by starting the command line with <TAB> or <SPACE>, or by enclosing the command with braces { }; neither the space nor the tab can precede an initial target name.

Both the list of dependency rules and the shell command list can be empty. Comments are the same as in the C language. A # in column 1 is interpreted by `cpp`; otherwise, `nmake` treats text (including a space or a tab) between the # symbol and the newline character as a comment.

A dependency rule can appear as a target in more than one entry; the dependencies for successive entries are simply appended. Only one shell command list may be specified for a given target. Rule names that contain the special characters `:`, `{`, `}`, #, `=`, and `+` must be enclosed in double quotation marks. The dependency rules for a target are scanned in order from left to right.

Rule Binding

The `nmake` utility binds each rule to either a file or a state variable in the process of updating targets. Rule names are bound to file names by the dependencies of the built-in `.SOURCE` and `.SOURCE.`*x* rules (see the Special Rules subsection). The dependencies of these rules are the directories that are to be scanned when searching for files. `nmake` warns when the same file is found in more than one directory, but it continues with the first file found.

A *state variable* is a variable whose time of modification and whose definition was stored from one invocation of `nmake` to the next. State variables have two basic forms: `(`*variable*`)` is a makefile variable (see the Variables subsection) and *file*`(`*variable*`)` is a variable that corresponds to a `#define` statement in *file*. The following command line specifies that `x.o` (the target) depends on the definition of the `DEBUG` variable from the `header.h` file and that it also depends on the definition of `MACHINE` (a variable) from the current makefile:

```
x.o : header.h(DEBUG) (MACHINE)
```

Usually, state variables are not included in automatic variable expansions (`nmake` maintains automatic variables; see the Automatic Variables subsection). State variable definitions are stored in *base*`.ms` in the current directory; *base* is the name of the first makefile in the argument list.

If a state variable is defined by a `#define` statement that is part of a conditional `#if, #ifdef`, or `#ifndef`, only the first encountered definition is used.

Variables

Entries in makefiles can contain variable definitions of the following form:

variable = value

In this example, *value* can be composed of several elements. Subsequent appearances of `$(`*variable*`)` are replaced by *value*. `$$(`*variable*`)` is replaced by `$(`*variable*); otherwise, `$` is passed untouched. The *value* is not expanded until `$(`*variable*`)` is encountered. If *value* contains another variable, the contained variable is expanded before *value* is determined.

Use `:=` instead of `=` to expand the *value* immediately; use `+=` to expand and append *value* to the current value of *variable*. You can modify command-line variable assignments only by using the `+=` assignment operator.

Variable definitions come from many sources. The precedence order (highest to lowest) is as follows:

1. Automatic definitions
2. Dynamic definitions
3. Command-line definitions
4. Makefile definitions
5. Environment definitions
6. Built-in definitions

Automatic definitions are maintained by `nmake` (see the Automatic Variables subsection). *Dynamic definitions* occur when targets are updated. *Command-line* arguments of the form *variable=value* are considered command-line definitions. *Makefile definitions* are included in the individual makefiles that are used with the `nmake` utility. *Environment definitions* are read from the inherited environment (see `env`(1)). *Built-in definitions* associate attributes with targets and dependency rules.

Variables in target and dependency names are expanded once when the makefile is read and once when the target or dependency rule is bound. Variables in shell command lists are expanded each time the commands are executed. If `$` is the first character on a line, the corresponding variable is always expanded when the makefile is read. This allows conditional makefile input, using edit operators such as `T=N` or `T=V`.

A variable name is a sequence of letters, digits, underscores, and periods. Variables with names that contain a period (`.`) should not conflict with environment variables set by `sh`(1); typically, such variables are used in built-in rule specifications. To avoid conflicts with the built-in rules, do not define uppercase variable names with a `.` as the first character.

Editing Variable Values

You can edit variable values during expansion by using the built-in edit operators. The general syntax for the editing operator is `$(`*variable*`:`*op*`[=`*arg*`]:...)`. The operators are applied to each space-separated element of the expanded *value* in which the newline character is treated as a separate element.

You can edit the following components of variable file names:

`M` *machine*	Edits all characters up to and including the last `!` (inserts *null* if no `!` appears).
`D` *directory*	Edits all characters after the last `!`, up to and including the last `/` (inserts *null* if no `/` appears).
`P` *prefix*	Edits all characters after the last `/`, up to and including the first `.` symbol. (inserts *null* if less than two `.` appear or if `.` is the first character).
`B` *base*	Edits all characters after the first `.` symbol, up to but not including the last `.` symbol.
`S` *suffix*	Edits all characters from the last `.` symbol to the end (inserts *null* if no `.` appears).

You can edit variables by using the `$(`*variable*`:`*edit*`)` syntax, in which *edit* is a concatenation of *c:* or *c*=*new* causes *variable* to be edited. *c* specifies the `M`, `D`, `P`, `B`, or `S` component, and *new* specifies a new value for the component.

If component letters are omitted, the corresponding component is deleted from the substituted value. The following variable name is used for substitution in the following examples:

```
FILES = a.c dir/s.x.c bozo!.profile
```

Following are examples of the use of the `B` and the `M` edit operators:

```
$(FILES:B:S=.o)      Results in a.o x.o .profile.o
$(FILES:M)           Results in bozo!
```

Other edit operators and examples based on the previous variable name are as follows:

`A[!]=a[|`*b...*`]`

Selects components that [do not] have the user attribute *a* [or *b...*]. An example of this is as follows:

```
$(FILES:A=.c:)       Results in a.c dir/s.x.c
$(FILES:A!=.c:)      Results in a.c dir/s.x.c boxo!.profile
```

`C`<*delim*>*old*<*delim*>*new*<*delim*>

Substitutes every occurrence of string *old* with string *new* for each delimiter-separated element. <*delim*> can be any delimiter character. If `^` is the first character in *old*, it represents the beginning of each element of *variable*'s expanded value; if # is the last character, it represents the end of each element. If *old* is null, `&` in *new* is replaced by the current element. `C/` may be abbreviated as `/`. An example of this is the following substitution:

```
$(FILES:c/ /,/:)      Results in a.c,dir/s.x.c,bozo!.profile
```

`E=`*message* Outputs *message* as an error on `stderr`, and `nmake` exits with nonzero exit status. Messages are in the following format:

`nmake:`*(error)message*

`F=`*format* Expands tokens according to *format*, which can be a concatenation of the following:

`L` Converts the token to lowercase.

`U` Converts the token to uppercase, as in the following example:

`$(FILES:F=U%.3s:)` Results in `A.C DIR BOZ`

This example uses the `%[`−`][`*n*`][.`*m*`]`*c* `printf`(3C) style formatting; *c* can be any of the following print conventions: `s`, `d`, `o`, `x`, and `u`.

`G=.`*s* Selects elements that can generate files with the *.s* suffix by using the implicit rules (see the Implicit Rules subsection for details); that is, if the implicit rule **.s : *.y* exists, token *x.y* is selected.

`I=`*message* Displays *message* on `stderr`; these are informational messages that do not interrupt `nmake` processing. Messages are in the following format:

`nmake:`*(information)message*

`N[ ! ]=`*pattern* Selects only tokens [not] matching the shell file match *pattern*. An example of this is the following substitution:

`$(FILES:N=*.c)` Results in `a.c dir/s.x.c`

`T=`*type*
`T=`*type*`[ ! ]=`*pattern*
`T=`*type*`?`*return*

Selects tokens specified by *type*. `[ ! ]=`*pattern* specifies that the substituted tokens must also [not] match the *pattern*; if there is no match, a null string is returned; otherwise, *type* can be one of the following:

`A` Selects each token that can be bound to an archive.

`B` Translates multiple space-character sequences to a single space.

`D` Expands each token that can be bound to a *state variable*, using the state variable definition. The expanded definitions can be used as arguments to `cc`(1).

`E` Similar to `T=D` except that the expanded definitions are of the format *x=y*; *x* is the state variable, and *y* is its value.

`F` Selects each token that can be bound to a file by using the *bound* file name.

N Expands the null string if *variable* has a null value; otherwise, # is expanded. This can be used to specify conditional makefile input.

O Selects each token that is bound neither to a file nor to a *state variable*.

S Selects each token that can be bound to a *state variable*.

T Selects each token that appears as a target in the *makefiles* input.

V Expands the null string if *variable* has a nonnull value; otherwise, # is expanded. This can be used to specify conditional makefile input. The *value* of *variable* is substituted without expansion.

W=*message* Outputs *message* as a warning on `stderr`. Messages are of the format:

`nmake:`*(warning)message*

X=*cross* Expands the cross product of the tokens in *variable* and the tokens in *cross*.

The following examples show the use of the `T=`*type* edit operator. The original variables, before substitution, are as follows:

```
FILES=x.c (DEBUG) (TEST) libc.a
TEST =
DEBUG = 1
```

The variables after substitution are as follows:

```
$(FILES:T=D:)      Results in -DDEBUG
$(FILES:T=F:)      Results in x.c  /lib/libc.a
$(FILES:T=S:)      Results in (DEBUG) (TEST)
$(TEST:T=V:)       Results in #
$(FILES:T=V:)      Results in null string
$(FILES:T=E:)      Results in DEBUG=1 TEST=
```

Only one of each operator can occur for each edit expansion. The operators are evaluated in the following order: `X`, `V`, `G`, `N`, `A`, `T`, `C`, `M`, `D`, `P`, `B`, `S`, and `F`.

Using the $(*var1* | *var2*...) syntax causes the value of the first nonnull variable (left to right) to be substituted. If the last variable name is enclosed in double quotation marks (`""`), and if all preceding variables have null values, this string is used as the value. The standard `\` character constants are interpreted within the string. $(*file(variable)*) causes the value of state variable *file(variable)* to be substituted as in the following example:

```
FILES = a.h b.h c.h x.c y.c z.c
HEADERS = $(FILES:N=*.h)

$(HEADERS:/^/-I/:) becomes -Ia.h -Ib.h -Ic.h
$(HEADERS:/ /:/:) becomes a.h:b.h:c.h
```

Automatic Variables

The nmake utility maintains the following variables:

`$(-)` The concatenation of the option flags presented to nmake (with the preceding –).

`$(-`*x*`)` This variable expands to 1 if the option is set; otherwise, it is null.

`$(=)` The list of variable assignments in the nmake utility arguments.

`$(?)` The list of nmake utility arguments that were not made yet. After this expansion, all arguments are treated as if they were already made. `$(?)` and `$(=)` are useful in recursive nmake calls.

`$(<)` The current target name.

`$(>)` The list of all file dependencies of the current target that are out-of-date with the target.

`$(*)` The list of all file dependencies of the current target.

`$(&)` The list of all file and state variable dependencies of the current target.

`$(%)` If the current target is a state variable, `$(%)` is the state variable value; otherwise, it is the unbound rule name.

`$(@)` The precommand list for the current target.

`$(#)` The postcommand list for the current target.

`$(MAKE)` The name of the current nmake program.

`$(MAKEFILE)`
The name of the first nonglobal makefile.

Each successive occurrence of <, >, *, &, %, @, or # causes the parent of the current target to be accessed. For example, `$(<<)` is the name of the parent of the current target, and `$(**)` is the list of all of its dependencies. `$(c`*rule*`)` (with c referring to one of the automatic variables) accesses information for *rule* rather than the current target. Also, `$(+`*c*`)` accesses information for the first dependency of the current target. The state variables and the .NOTOUCH, .POST, and .USE rules are not included in <, >, or * automatic variable substitutions.

Following is an example of this use of automatic variables:

```
PROG::g.y  g.l  g.c
   echo (rule $(<*.o)
```

This results in:

```
g.o  g.y  g.l
```

Operators

The built-in rules for nmake also contain operator definitions. An operator specifies actions to be taken when makefiles are read and often allow abbreviated makefile specifications. The :: operator is similar to the : edit operator in specifying dependencies, except that :: specifies source dependencies.

Any ld(1) library flags and libraries with suffix .a can be placed in :: dependency lists only if the directory containing the .a libraries is defined by the .SOURCE.a rule. By default, .SOURCE.a is set to /lib /usr/lib. The LDFLAGS variable (-L) is ignored because nmake expands -lx to libx.a.

Option Generation

The built-in rules automatically generate the proper -D and -I options of cc(1) in the $(CCFLAGS) variable. The -D options are generated from state variable dependencies, and the -I options are generated from the dependencies of the .SOURCE.h rule. State variable dependencies that are specified using the :: operator apply to all dependencies of the corresponding target (global dependencies); otherwise, the dependencies apply only to the individual target of the operator.

The following makefile specifies that program depends on two files, a.o and b.o, and that they in turn depend on .c files and the common file header.h:

```
program : a.o b.o
   cc a.o b.o lib.a -lm -o program
a.o : header.h a.c
   cc -c a.c
b.o : header.h b.c
   cc -c b.c
```

Implicit Rules

The nmake utility infers dependencies for files that do not have explicit update commands. For example, a .c file can be inferred as a dependency for a .o file and can be compiled to produce the .o file.

The rule for creating a file with suffix .s2 that depends on a file (with the same *base* name) with suffix .s1 is specified as an entry for the target rule:

```
*.s2 : .s1
```

The suffixes .s2 and .s1 are appended automatically to .SUFFIXES if they were not specified previously. Any dependencies following *.s1 are transferred to each implicit target when the implicit rule is applied. For example, a rule for making optimized .o files from *.c files is the following:

```
*.o : *.c (CC) (CCFLAGS)
$(CC) $(CCFLAGS) -c -o $(<) $(>)
```

If the current target is a.o, nmake infers the following rule from the *.o : *.c rule:

```
a.o:a.c
cc -c -O -o a.o a.c
```

Using the default *.o : *.c rule, the example can be stated more briefly:

```
program : a.o b.o
  $(CC) $(*) lib.a -lm -o $(<)
a.o b.o : header.h
```

The rule for creating a file with no suffix from a similarly named file with suffix `.s` is specified as an entry for the target rule `* : *.s`. The rule for creating a file with no suffix when no other suffix rule applies is specified as an entry for the target rule `* : *`.

Implicit rules are inferred according to the suffixes listed as the dependencies of the built-in rule `.SUFFIXES`. Suffix order is significant; the first possible name for which both a file and a rule exists is inferred.

Built-in Rules

The following rules are special to `nmake`. Most are used to associate attributes with rules and targets. To avoid conflicts with these built-in rules, do not define uppercase rule names with a period (`.`) as the first character.

`.ARCHIVE` The dependencies of `.ARCHIVE` are suffixes associated with archive files (see `ar`(1) and `bld`(1)). A rule with this suffix is treated as an archive. When used as a dependency, `.ARCHIVE` causes the target to be treated as an archive and the `*.a : *` rule is used to update the target. The `$(ARUPDATE)` variable is set dynamically by `nmake` to contain the proper update sequence for the current archive target (the commands are preceded by a newline character). The default archive suffixes and update commands are as follows:

```
.ARCHIVE : .a

*.a : * (AR) (ARFLAGS)
        $(AR) $(ARFLAGS) $(<) $(>) $(ARUPDATE)
        $(RM) $(RMFLAGS) $(>)
```

`.ATTRIBUTE` When used as a dependency, `.ATTRIBUTE` defines the target as a user attribute. Dependencies of user attributes are treated as suffixes and a rule with one of these suffixes automatically inherits the corresponding user attribute. A rule can have any number of user attributes, up to an implementation-defined limit.

`.CLEAR` When used as a dependency, `.CLEAR` clears the dependency and the command lists for the target.

`.CURRENT` When used as a dependency, `.CURRENT` marks the target as being produced in the current directory. The dependencies of `.CURRENT` are suffixes associated with targets that are produced only in the current directory. The default is `.CURRENT : .a .o`.

`.DEFAULT` This rule is used as a last resort when no other rules can be inferred to make the current target.

`.DONE` This rule is made after all targets are made and has no effect on the update status of the targets. The commands are always executed in the foreground shell (see the Jobs subsection).

`.DONTCARE` When used as a dependency, `.DONTCARE` causes `nmake` to continue if the target cannot be made; otherwise, `nmake` issues an error and exits if a target cannot be made. This is the same functionality as the `-k` command-line option.

`.FOREGROUND` When used as a dependency, `.FOREGROUND` causes the target update commands to be executed in the foreground shell; otherwise, the commands can be executed in a background shell (see the Jobs subsection). Usually, `nmake` computes future dependencies while update commands are being executed; however, `.FOREGROUND` update commands cause `nmake` to block until the commands complete.

`.GLOBAL.`*x* The dependencies of `.GLOBAL.`*x* are inserted onto each target with suffix *.x* immediately before the target is made; *.x* may be the null suffix. For this rule, `$(<)` refers to the current `.GLOBAL.`*x* target, and `$(<<)` refers to the target on which the dependencies are inserted.

`.IMPLICIT` When used as a dependency, `.IMPLICIT` causes the implicit suffix rules to be applied even when update commands were specified for the target; otherwise, the implicit suffix rules are applied only to targets with no explicit update commands.

`.INIT` This rule is made before any other target and has no effect on the update status of the targets. The commands are always executed in the foreground shell (see the Jobs subsection).

`.INSERT` When used as a dependency, `.INSERT` causes the dependency list to be inserted rather than appended to the target dependency list.

`.INTERNAL` This rule is used internally.

`.INTERRUPT` This rule is made when an interrupt signal is caught; it has no effect on the update status of the targets. The commands are always executed in the foreground shell (see the Jobs subsection). `nmake` exits after the commands are executed.

`.MAIN` If no targets are explicitly listed on the command line, the dependencies of `.MAIN` are used as the main targets. If not explicitly specified in the input makefiles, the first dependency of `.MAIN` is set to be the first target encountered that is not a special rule or inference rule.

`.MAKE` When used as a dependency, `.MAKE` causes the command update list to be parsed by `nmake` rather than executed by the shell. Such command lists are always parsed, even with the `-n` option (the `-n` option traces but does not execute the target update command).

`.MAKEFILES` This rule specifies the default makefile names. If no explicit makefile is specified, the following files are tried in order:

```
.MAKEFILES : Nmakefile nmakefile Makefile makefile
```

`.MAKEINIT` This target is made before the `.SOURCE` targets are examined. `.INIT` is made after the `.SOURCE` targets are examined. Do not redefine this rule; however, it is safe to insert or append dependencies to `.MAKEINIT`. Variable assignments within `.MAKEINIT` commands override any command-line variable assignments.

`.NOEXPAND` This attribute is associated with state variables that are not to be expanded by the `:T=D:` edit operator. `.NOEXPAND` is defined using `.ATTRIBUTE`.

`.NOTOUCH` When used as a dependency, `.NOTOUCH` causes the target modify time to remain untouched, even if the corresponding update commands are executed. This allows initialization sequences to be specified for individual rules:

```
main : init header
        echo "executed if header is newer than main"
        init : .NOTOUCH
        echo "always executed for main"
```

`.NULL` When used as a dependency of a target with no suffix or explicit update commands, `.NULL` causes the commands that are associated with the `* : *` rule to be used when updating the target.

`.OPERATOR` When used as a dependency, `.OPERATOR` marks the target as an operator to be applied when makefiles are read. Operator names must consist of 2 characters.

`.OPTIONS` The dependencies of `.OPTIONS` are treated like command-line options. The options take effect immediately when `.OPTIONS` is read. The `-f` and `-g` options have no effect in this case. All options specified on the command line override the `.OPTIONS` specification, except for the `-j` option (which specifies the number of jobs that `nmake` can create concurrently).

`.PARAMETER` When used as a dependency, `.PARAMETER` marks the target as a parameter file that contains only definitions (for example, `#define` definitions) and comments. The modify time of a parameter file is ignored when determining the update status of corresponding targets.

`.POST` When used as a dependency, `.POST` causes the target to be made after the parent target is made. If a `.POST` dependency also has the `.NOTOUCH` attribute, commands are executed only if the target is updated.

`.PRECIOUS` If `nmake` is interrupted, the current target(s) are usually deleted; the dependencies of `.PRECIOUS`, however, are not deleted. If `.PRECIOUS` is specified with no dependencies, all targets are protected. Targets marked with `.ARCHIVE` are always precious.

`.PREFIXES` The dependencies of `.PREFIXES` are file name prefixes of the form *p*. They are used to infer implicit rules. The (left to right) prefix order is important; the first inference rule name for which both a file and a rule exist is inferred. The default prefix rules provide a smooth interface to Source Code Control System files (see `admin`(1)):

```
.PREFIXES : s.
```

`.READONLY` When used as a dependency for `.o` target files, `.READONLY` causes the data parts of the corresponding `.c` source file to be placed in read-only text. `.READONLY` can also be used to place the tables of corresponding `.l` and `.y` source files in read-only text. The commands are tailored to the current host machine. `.READONLY` is implemented as a `.USE` rule, and it will become obsolete when the `const` data attribute becomes a C language standard. Typical usage is as follows:

```
file.o : .READONLY
```

`.SEARCH` The dependencies of `.SEARCH` are suffixes associated with files that are to be searched for implicit file dependencies. When used as a dependency, `.SEARCH` marks the target to be searched for implicit file dependencies. By default, any dependency with a `.SEARCH` suffix or marked with `.SEARCH` is searched automatically for `#include` header file dependencies. If any files are newer than the dependency file, the corresponding target is updated. The default is as follows:

```
.SEARCH : .c .h .y .l .F
```

`.SOURCE` The dependencies of `.SOURCE` are directories to be scanned when searching for files. Because the directories are always searched in left-to-right order, the first directory that contains the file is used. The default is the current directory (.):

```
.SOURCE : .
```

`.SOURCE.`*x* The directories specified by the `.SOURCE.`*x* rule are checked for files with a period (.) as a suffix. If the file is not found, the directories specified by `.SOURCE` are checked. The *.x* suffix must be a dependency of `.SUFFIXES`. Because the directories are always searched in left-to-right order, the first directory that contains the file is used. The default when the `-X` option is disabled is as follows:

```
.SOURCE.a : /lib /usr/lib
.SOURCE.h : /usr/include
```

`.SUFFIXES` The dependencies of `.SUFFIXES` are file name suffixes of the form `.s`, used to infer implicit rules. The (left-to-right) suffix order is important; the first inference rule name for which both a file and a rule exist is inferred. The default suffix list is as follows:

```
SUFFIXES : .o .c .f .F .f90 .F90 .p  .y .l .s .sh .h .a .msg .exp .cat
```

The following list describes the meaning of each suffix:

Suffix	Meaning
`.o`	Relocatable
`.c`	C
`.f`	Fortran
`.F`	Fortran (processed)
`.f90`	CF90 Fortran

.F90	CF90 Fortran (processed)
.p	Pascal
.y	yacc(1)
.l	lex(1)
.s	Assembler
.sh	Shell script
.h	Header files
.a	Libraries
.msg	Message
.exp	Explanatory message
.cat	Catalog message

.TOUCH Each dependency of .TOUCH is touched as though it were already made. The dependencies are touched immediately when .TOUCH is read. This is the same functionality as the -t command-line option.

.UNTOUCH Each dependency of .UNTOUCH is untouched as though it were never made. The dependencies are untouched immediately when .UNTOUCH is read.

.USE When used as a dependency, .USE marks the target as a .USE rule. Any target that has a .USE rule as a dependency is updated using the commands associated with the .USE rule.

.WAIT .WAIT is a synonym for .FOREGROUND.

Command Execution

The nmake and sh(1) commands run as coprocesses. All update commands are sent to a single copy of the shell, keeping the shell environment intact between command executions (see the Jobs subsection for an exception). This includes the effects of cd(1) and shell parameter assignments. sh echoes each command when executed, unless the -s option is specified. Because entire command blocks are sent as a unit, special shell constructs (case, if, for, and while) can cross newline boundaries without newline escape characters indicating a continuation.

Commands returning nonzero status (see intro(1)) stop nmake, unless the -i or -k option is specified.

Interrupt and quit signals cause the current target to be deleted, unless the target is an archive, or a directory, or unless it is a dependency of the special rule .PRECIOUS.

Any command update list that contains the $(MAKE) variable is always executed, even when the -n option is specified. This simplifies the maintenance of a makefile hierarchy because one makefile can invoke other makefiles in the hierarchy, passing along the top-level options and variable assignments by using $(-) and $(=). You can use $(MAKE:) to disable execution when the -n option is specified.

Special Shell Commands

Some special shell commands are provided for the nmake and sh coprocess environment:

. . . The *ellipsis* command separates an update command list into precommands (before . . .) and postcommands (after . . .). The precommands are executed when a target is being updated. The postcommands are stacked (first-in, first-out) and are executed in the foreground shell after the last target is generated. The ellipses must appear as the first command on a line.

`exit` *code*
Removes the shell coprocess, stopping `nmake`.

`ignore` *shell-command*
Causes the exit status of *shell-command* to be ignored.

`_make_` *command data*
Passes messages from `sh`(1) to `nmake`. Do not modify the shell definition of either `_make_` or `make_id_`.

`_make_ error` *exit-code*
Called when an update command returns a nonzero exit code.

`_make_ exit`
Called when an update command block completes.

`_make_ start` *job-id process-id*
Associates a command block with a process ID.

`@ nmake-command`
The `nmake-command` is sent to and executed by `nmake`. This allows variable definitions and even makefiles to be modified dynamically (although some dynamic modifications may not work with the `-n` option). Embedded newline characters can be sent by enclosing the `nmake-command` in either single or double quotation marks.

`silent` *shell-command*
Prevents the shell from printing *shell-command.* If both `silent` and `ignore` are used, `silent` must precede `ignore`.

Jobs

The `-j` option allows `nmake` to update many targets concurrently. The updates are synchronized using the target dependency graph specified in `makefile`. With this option, each update command block is sent to a new subshell (background shell). Background shells inherit the environment of the main shell (foreground shell) and the foreground shell inherits the environment of `nmake`. You can force target update commands to execute in the foreground shell by including the special rule `.FOREGROUND` as a dependency.

Common Actions

When the `::` operator is used, several common action targets are defined automatically. The common action target *xxx* is defined as *.XXX* in the built-in rules. If *xxx* appears as a command-line target and *xxx* was not defined by the input makefiles, the target *.XXX* is made. The common action target must be the first command argument target (from left to right) and can be followed by specific command targets. If no command argument targets are specified, all command targets are affected. The common actions are as follows:

`arch` Creates an `ar`(1) archive of the source files that are listed after each `::` operator. The archive is placed in file *main*.arch; *main* is the base name of the main target rule.

`clean` Deletes all object files that correspond to the current makefile.

`clobber` Executes the `clean` action and also deletes the targets that correspond to the current makefile.

`cpio` Creates a `cpio`(1) archive of the source files listed after each `::` operator. The archive is placed in file *main*.`cpio`; *main* is the base name of the main target rule.

`ctags` Creates a *tags* file for `vi`(1) by using `ctags`(1).

`install` Makes the main target and copies it to the `$(INSTALLDIR)` directory. By default, `$(INSTALLDIR)` is `$(BINDIR)` for executable targets, `$(LIBDIR)` for object archive targets, and `$(MANDIR)`*n* for man page targets; *n* is the manual section number. `BINDIR`, `ETCDIR`, `INCLUDEDIR`, `LIBDIR`, `MANDIR`, and `NLSDIR` are defined as `$(ROOT)/bin`, `$(ROOT)/etc`, `$(ROOT)/include`, `$(ROOT)/man/man`, and `$(ROOT)/lib/nls/En`, respectively. The commands that are associated with the `.DOINSTALL` rule are used to do the copying. You can specify the installation directory for any (non-man page) target by using the `.INSTALL.`*x.* variable.

`lint` Runs `lint`(1) on the input source files. Any `.l` and `.y` source files are preprocessed automatically if necessary.

`lprof` Runs `lprof`(1) (systems other than Cray Research systems that support only System V) on the target command(s). Each command must have been generated using the `-ql` profiling option on the `lprof` command, and the *command*`.cnt` file must exist in the current directory (that is, *command* must have been run at least once).

`print` Prints the source files by passing them through the filter `$(PR)` and listing them with `$(LP)`.

`tar` Creates a `tar`(1) archive of the source files listed after each `::` operator. The archive is placed in file *main*.`tar`; *main* is the base name of the main target rule.

`uarch` Same as `arch` except only the source files modified since the last `uarch` are archived (see also `$(UTIME)` in `ucpio`).

`ucpio` Same as `cpio` except only those source files modified since the last `ucpio` are archived. If `$(UTIME)` is specified, it is assumed to be a file name whose modify time is used to determine which files are to be archived; only files that are newer than this modify time are archived.

`uprint` Same as `print` except only the source files modified since the last `uprint` are printed (see also `$(UTIME)` in `ucpio`). The `-F` option (forcing all targets to be updated) must be set the first time `uprint` is used.

`utar` Same as `tar` except that only the source files modified since the last `utar` are archived (see also `$(UTIME)` in `ucpio`).

NOTES

The default makefile ordering is the exact opposite of make(1). If old makefiles are named makefile and new makefiles are named Nmakefile, nmakefile, or Makefile, both make and nmake can be run without specifying explicit files names.

Because the built-in rules are placed in an nmake object file, performance is not degraded when different built-in rules are specified either by the MAKERULES environment variable or by using the -b and -g options.

The following command line causes all :: targets to be made by default:

```
.MAIN : .CLEAR .ALL
```

Otherwise, only the first :: target is made by default.

The :, :=, +=, operators, and operator lines are expanded when the makefile is read. Any $(*var*) variables that occur in these lines are frozen into the corresponding nmake object file. If the value of a frozen variable changes from one invocation of nmake to the next, either in a command-line definition or in the environment, a warning is issued and the makefile is recompiled automatically. You can defer variable evaluation on these lines by entering $(*var*) as $$(*var*).

It is not possible to specify explicit makefile dependencies; however, implicit makefile dependencies are generated automatically.

Some commands return inappropriate nonzero status; use the ignore command to avoid this.

The nmake utility detects only source files that exist before nmake is executed.

For a given command-name variable $(XX), the default flags are specified by the $(XXFLAGS) variable. For example, the flags for $(CC) are $(CCFLAGS), and the flags for $(YACC) are $(YACCFLAGS). You can use the $(LDLIBRARIES) variable to specify additional libraries to the cc(1) command. You can use the $(LINTLIBRARIES) variable to specify additional libraries to the lint(1) command. The $(LINTLIB) variable specifies the default lint(1) library directory.

The default yacc(1) rules for the *x*.y file produces the *x*.h and *x*.o files, not *y.tab*.h and *y.tab*.o or *y.tab*.c. The *x*.h file is updated only if it differs from the *x*.h file that is generated by the previous yacc. Similarly, the default lex(1) rules for the *x*.l file produce the *x*.o file, not lex.yy.o or *lex.yy*.c. Make the yacc target before any other targets that may depend on the generated header file.

For dynamic dependencies, #include <file> is treated as #include "file".

The nmake -l*x* library expansions can load different libraries than they would if the -l*x* options were passed directly to ld(1).

Because of optimizations, unified input syntax, and new functionality, this version of nmake is not compatible with either the original or the augmented version.

WARNINGS

A warning is issued when the input makefiles must be recompiled. A warning is issued if a rule can be bound to more than one file (unless the `-u` option is set); `nmake` proceeds with the first file found, using the `.SOURCE` and `.SOURCE`.*s* rules.

ENVIRONMENT VARIABLES

The `NPROC` environment variable specifies the number of jobs that `nmake` can execute concurrently. The `ARRANGED` variable specifies the location of the `arrange` daemon script. The built-in rules are compiled into an `nmake` object file whose location is specified by the `MAKERULES` environment variable. `MAKEPP` specifies the path name of the makefile preprocessor. `MAKESHELL` specifies the path name of the shell that is used to update targets. `SRCPATH` is interpreted as a colon-separated list of directories inserted in the `.SOURCE` and `.SOURCE.h` rule dependency lists (the current directory, designated by the period, is always searched first).

Do not redefine the following variables in the `nmake` object file; redefinitions of these environment variables can cause inconsistent system performance:

```
ARRANGED_FILE
ARRANGED_TMOUT
CPPINCLUDE
MAKE
MAKEFILE
ROOT
SHELL
```

BUGS

Syntactically incorrect `sh`(1) commands can cause `nmake` to hang; usually, it is caused by nonterminated strings enclosed in quotation marks.

The `nmake` utility is optimized to work with `sh`(1); it does not work with `csh`(1).

FILES

`Nmakefile, nmakefile, Makefile, makefile`	Default makefiles, which are tried in order
`/usr/lib/make/Makerules.mo`	Standard compiled built-in rules
`/usr/lib/make/cpp`	The `nmake` preprocessor
`/usr/lib/make/arranged`	Daemon used to sort output from one or more `nmake` files
`base.mo`	The `nmake` object file
`base.ms`	The `nmake` state file

SEE ALSO

admin(1), ar(1), cd(1), cpio(1), csh(1), ctags(1), ld(1), lex(1), lint(1), make(1), sed(1), tar(1), vi(1), yacc(1)

cc(1), cpp(1) in the *Cray Standard C Reference Manual*, Cray Research publication SR–2074

UNICOS Configuration Administrator's Guide, Cray Research publication SG–2303

NAME

nohup – Invokes a utility immune to hangups and quits

SYNOPSIS

nohup *utility* [*arguments*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The nohup utility executes *utility*, ignoring hangups and quits. If you do not redirect the output, it will be appended to nohup.out. If nohup.out cannot be created or opened for appending in the current directory, output is appended to the end of file $HOME/nohup.out.

If neither file can be created or opened for appending, *utility* is not invoked.

If the standard error is a terminal, all output written by the named *utility* to its standard error is redirected to the same file descriptor as the standard output.

The exit status of nohup is that of the utility specified by the *utility* operand.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

The csh(1) utility has a built-in nohup utility with slightly different characteristics (see csh(1)).

EXIT STATUS

The nohup utility exits with one of the following values:

126 The *utility* specified was found but could not be invoked.

127 An error occurred in the nohup utility or the *utility* specified could not be found.

Otherwise, the exit status of nohup is that of the utility specified by the *utility* operand.

EXAMPLES

This example puts the cc(1) compilation of example.c in the background, with the output going to nohup.out. User input is shown in courier bold:

```
$ nohup cc example.c &
10793
$ Sending output to nohup.out
```

FILES

nohup.out	File that contains output from *utility*
$HOME/nohup.out	File that contains output from *utility*

SEE ALSO

nice(1)

signal(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

nroff – Text formatting language

SYNOPSIS

nroff [-o*list*] [-n*N*] [-s*N*] [-r*aN*] [-i] [-q] [-z] [-m*name*] [-T*tty_type*] [-e] [-h] [-un] [*file_name(s)*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nroff utility formats text contained in *file_name*(*s*) (standard input by default) for printing on typewriter-like devices and line printers.

An argument consisting of a dash (-) is taken to be a file name corresponding to the standard input. Options may appear in any order, but they must appear before the *file_name*(*s*) argument.

The nroff utility accepts the following options and operands:

-o*list* Prints only pages whose page numbers appear in the *list* of numbers and ranges separated by commas. A range *N*-*M* means pages *N* through *M*; an initial -*N* means from the beginning to page *N*; and a final *N*- means from *N* to the end. (See the BUGS section.)

-n*N* The first page generated begins with the number *N*.

-s*N* Stops every *N* pages. nroff halts *after* every *N* pages (default *N*=1) to allow paper loading or changing, and it resumes on receipt of a line feed or new line (new lines do not work in pipelines, for example, with mm(1)). This option does not work if the output of nroff is piped through col(1). When nroff halts between pages, an ASCII BEL is sent to the terminal.

-r*aN* Sets register *a* to *N*. This must have a 1-character name.

-i Reads standard input after *files* are exhausted.

-q Invokes the simultaneous input-output mode of the .rd request.

-z Prints only messages generated by .tm (terminal message) requests.

-m*name* Prepends to the input *file_name*(*s*) the macro file /usr/lib/tmac/tmac.*name*.

-T*tty_type* Prepares output for specified terminal. Known *tty_types* are as follows:

2631	Hewlett-Packard 2631 printer in regular mode
2631-c	Hewlett-Packard 2631 printer in compressed mode
2631-e	Hewlett-Packard 2631 printer in expanded mode
300	DASI-300 printer

300-12	DASI-300 terminal set to 12-pitch (12 characters per inch)
300s	DASI-300s printer (300S is a synonym)
300s-12	DASI-300s printer set to 12-pitch (12 characters per inch) (300S-12 is a synonym)
37	TELETYPE O Model 37 terminal (default)
382	DTC-382
4000a	Trendata 4000a terminal (4000A is a synonym)
450	DASI-450 (Diablo Hyterm) printer
450-12	DASI-450 terminal set to 12-pitch (12 characters per inch)
832	Anderson Jacobson 832 terminal
8510	C.ITOH printer
`lp`	Generic name for printers that can underline and tab (all text that use reverse line feeds, such as those having tables, that is sent to `lp` must be processed with `col`.)
tn300	GE Terminet 300 terminal
X	Printers equipped with TX print train

`-e` Produces equally spaced words in adjusted lines by using the full resolution of the particular terminal.

`-h` Uses output tabs during horizontal spacing to speed output and reduces output character count. Tab settings are assumed to be every 8 nominal character widths.

`-un` Sets the emboldening factor (number of character overstrikes) for the third font position (bold) to *n*, or to 0 if *n* is missing.

file_name(s) File that contains the text to be formatted.

BUGS

`nroff` uses Eastern Standard Time; therefore, depending on the time of the year and on your local time zone, the date that `nroff` generates may be off by one day from the date you think it is.

When `nroff` is used with the `-o`*list* option inside a pipeline (for example, with one or more `eqn`(1) and `tbl`(1) commands), it may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.

FILES

`/usr/lib/tmac/tmac.*`	Pointers to standard macro files
`/usr/lib/macros/*`	Standard macro files
`/usr/lib/nterm/*`	Terminal driving tables for `nroff`
`/usr/pub/terminals`	List of supported terminals

SEE ALSO

col(1), eqn(1), mm(1), tbl(1)

mm(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

"The Formatter nroff: a Tutorial" in the *UNIX System V Documenter's Workbench Software User's Guide* (Prentice Hall, ISBN 0-13-943598-0)

"nroff/troff: Technical Discussion" in the *UNIX System V Documenter's Workbench Software Technical Discussion and Reference Manual* (Prentice Hall, ISBN 0-13-943580-8)

NAME

nslookup – Queries name servers interactively

SYNOPSIS

```
/usr/ucb/nslookup [-keyword[=value]]
/usr/ucb/nslookup [-keyword[=value]] host-to-find
/usr/ucb/nslookup [-keyword[=value]] -
/usr/ucb/nslookup [-keyword[=value]] - server address
/usr/ucb/nslookup [-keyword[=value]] - server name
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The nslookup utility queries DARPA Internet domain name servers. nslookup can operate in interactive and noninteractive mode. Interactive mode lets users query the name server for information about various hosts and domains or print a list of hosts in the domain. Noninteractive mode is used to print just the name and Internet address of a host or domain.

Interactive mode is entered in the following cases:

- When no arguments are specified (the default name server is used).
- When the first argument is a hyphen (-) and the second argument is the host name of a name server.

Noninteractive mode is used when the name of the host to be looked up is specified as the first argument. The optional second argument specifies a name server.

You can also use the command line to set status information that affects the lookups by preceding the status information keyword by a hyphen (-). See the set command in the next subsection for a complete list of keywords and descriptions.

Interactive Commands

You can interrupt commands at any time by typing your interrupt character (usually <CONTROL-c>). To exit, press <CONTROL-d> (EOF). The command line must consist of fewer than 80 characters. An unrecognized command is interpreted as a host name.

host [*server*] Looks up information for *host* by using the current default server or using *server* if it is specified.

server *domain*
lserver *domain*
Changes the default server to *domain*. lserver uses the initial server to look up information about *domain*; server uses the current default server. When an authoritative answer cannot be found, the names of servers that might have the answer are returned.

`root` Changes the default server to the server for the root of the domain name space. Currently, the `ns.nic.ddn.mil` host is used. (This command is a synonym for the `lserver ns.nic.ddn.mil` command.) The name of the root server can be changed using the `set root` command.

`finger` [*name*] [> *filename*]
`finger` [*name*] [>> *filename*]
Connects with the finger server on the current host. The current host is defined when a previous lookup for a host was successful and it returned address information (see the `set querytype=A` command). The *name* argument is optional. The > and >> characters can be used to redirect output in the usual manner.

`ls` *domain* [> *filename*]
`ls` *domain* [>> *filename*]
`ls -a` *domain* [> *filename*]
`ls -a` *domain* [>> *filename*]
`ls -h` *domain* [> *filename*]
`ls -h` *domain* [>> *filename*]
`ls -m` *domain* [> *filename*]
`ls -m` *domain* [>> *filename*]
`ls -s` *domain* [> *filename*]
`ls -s` *domain* [>> *filename*]
`ls -d` *domain* [> *filename*]
`ls -d` *domain* [>> *filename*]
`ls -t` [*type*] *domain* [> *filename*]
`ls -t` [*type*] *domain* [>> *filename*]
Lists the information available for *domain*. The default output contains host names and their Internet addresses. The `-a` option lists aliases of hosts in the domain. The `-h` option lists CPU and operating-system information for the domain. The `-m` option lists mail exchange information for the domain. The `-s` option lists well-known services for the domain. The `-d` option lists all contents of a zone transfer. The `-t` option lists all records of the specified type for the domain, or for the current `querytype` if *type* is not specified. When output is directed to a file, hash marks are printed for every 50 records received from the server.

`view` *filename* Sorts and lists the output of previous `ls` command(s) by using `more`(1).

`help`
`?` Prints a brief summary of commands.

`set` *keyword*[=*value*]
Changes status information that affects the lookups. Valid keywords are as follows:

`all"` Prints the current values of the various options to `set`. Information about the current default server and host is also printed.

`[no]debug` Turns on debugging mode. More information is printed about the packet sent to the server and the resulting answer. Turning debugging mode off by using `set nodebug` turns off all debugging, including that set by `set d2`. The default is `nodebug`; abbreviation is `[no]deb`.

`[no]d2` Turns on exhaustive debugging mode. Essentially, all fields of every packet are printed. Also turns on normal debugging mode, as if `set debug` had been issued. Turning off exhaustive debugging mode by using `set nod2`, however, leaves normal debugging mode on. The default is `nod2`.

`[no]defname` Appends the default domain name to every lookup. The default is `defname`; abbreviation is `[no]def`.

`[no]search` With `defname`, searches for each name in parent domains of the current domain. The default is `search`; abbreviation is `[no]sea`.

`domain=`*name* Changes the default domain name to *name*. If the `defname` option was set, the default domain name is appended to all lookup requests. The search list is set to parents of the domain with at least two components in their names. The default is value in host name or `/etc/resolv.conf`; abbreviation is `do`.

`querytype=`*value*
The default is `A`; abbreviation is `q`.

`type=`*value* Changes the type of information returned from a query to one of the following:

`A` 10	Host's Internet address (the default).
`CNAME` 10	Canonical name for an alias.
`HINFO` 10	Host CPU and operating system type.
`MB` 10	Mailbox domain name.
`MG` 10	Mail group member.
`MINFO` 10	Mailbox or mail list information.
`MR` 10	Mail rename domain name.
`MX` 10	The mail exchanger.
`NS` 10	Name server for the specified zone.
`PTR` 10	Pointer record for an Internet address in the `in-addr.arpa` domain.
`SOA` 10	Start Of Authority record for the named zone.
`WKS` 10	Well Known Services for the name.

The default is A; abbreviation is q or ty.

[no]recurse Instructs the name server to query other servers if it does not have the information. The default is recurse; abbreviation is [no]rec.

retry=*number* Sets the number of retries to *number*. When a reply to a request is not received within a certain amount of time (changed with set timeout), the request is resent. The retry value controls the number of times a request is resent before it gives up. The default is 4; abbreviation is ret.

root=*host* Changes the name of the root server to *host*. This affects the root command. The default is ns.nic.ddn.mil; abbreviation is ro.

timeout=*number*
Changes the time-out interval for waiting for a reply to *number* seconds. The default is 5; abbreviation is t.

[no]vc Specifies a virtual circuit for sending requests to the server. The default is novc; abbreviation is [no]v.

class=*value* Changes the current query class to one of the following:

IN 10	Internet records
CHAOS 10	CHAOSnet records
HESIOD 10	Hesiod records
ANY 10	Any record class

The default is IN; abbreviation is cl.

port=*number* Changes the number of the port used to connect the name server to *number*. The default is 53; abbreviation is po.

MESSAGES

When the look-up request is not successful, an error message is printed. Possible errors are as follows:

Time-out The server did not respond to a request after a certain amount of time (changed with set timeout=*value*) and a certain number of retries (changed with set retry=*value*).

No information Depending on the query type set with the set querytype command, information about the host was not available, although the host name is valid.

Nonexistent domain
The host or domain name does not exist.

Connection refused

`Network is unreachable`
: The connection to the name or finger server could not be made at the current time. This error commonly occurs with `finger` requests.

`Server failure`
: The name server found an internal inconsistency in its database and could not return a valid answer.

`Refused`
: The name server refused to service the request.

The following error indicates a bug in the program:

`Format error`
: The name server found that the request packet was not in the proper format.

FILES

`/etc/resolv.conf` Initial domain name and name server addresses

SEE ALSO

`more`(1)

RFC 1034, *Domain Names--Concepts and Facilities*, Mockapetris, November 1987
RFC 1035, *Domain Names--Implementation and Specification*, Mockapetris, November 1987

NAME

obc – Invokes the arithmetic language preprocessor

SYNOPSIS

obc [-c] [-l] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX
AT&T extensions (-c option)

DESCRIPTION

The obc utility (formerly, the UNICOS 8.0 version bc(1) utility) is an interactive processor for a language that resembles C but provides unlimited arbitrary-precision arithmetic. It takes input from any files specified and then reads from standard input.

The obc utility is actually a preprocessor for the odc(1) desk calculator utility, which obc invokes automatically unless you specify the -c (compile only) option. In this case, the odc(1) input is sent to standard output instead.

The obc utility accepts the following options and operand:

-c Specifies compile only. The output is sent to the standard output.

-l Defines the math functions and initializes scale to 20, rather than the default, which is 0.

files When using obc, assignment to scale influences the number of digits to be retained on arithmetic operations in the manner of odc(1).

You may use a letter simultaneously as an array, a function, and a simple variable. All variables are global to the program. auto variables are pushed down during function calls. When arrays are used as function arguments or defined as automatic variables, empty brackets must follow the array name.

Syntax and Functions

Comment strings are enclosed in /* and */.

Simple variable names can be any single lowercase letter (designated in the following examples as *l*). Array element names are expressed as *l*[*expression*]. Other acceptable variables are the words ibase (used to set the input number radix), obase (used to set the output number radix), and scale (used to set the number of digits to the right of the decimal).

Operands can be arbitrarily long numbers with optional sign and decimal points. Acceptable forms include (*expression*), `sqrt`(*expression*), `length`(*expression*), `scale`(*expression*), and *l*(*expression*). Both `scale` and `length` may be followed by a number that indicates the number of significant decimal digits.

Operators include `+`, `-`, `*`, `/`, `^`, and `%`; `%` is the remainder, and `^` is the exponent.

The `++` and `--` operators are used for prefix and postfix notation, and they are applied to variables.

Assignment operators include `=+`, `=-`, `=*`, `=/`, `=%`, and `=^`. These are used to change a named variable according to the operator. For example, the following two statements are equivalent:

```
s=+3
s=s+3
```

Statements are of the form *expression* or [*statement* `;` `...` `;` *statement*]. Either a semicolon or `<newline>` character can separate statements. Other statements include the following:

`if (`*expression*`)` *statement*
`while (`*expression*`)` *statement*
`for (`*expression* `;` *expression* `;` *expression*`)` *statement*
null statement
`break`
`quit`

Function definitions are of the following form:

```
define l (l, ... ,l) {
          auto ll ,..., l
          statement ; ... statement
          return (expression)
       }
```

Functions in the `-l` math library include the following:

`s`(*x*)	Sine
`c`(*x*)	Cosine
`e`(*x*)	Exponential
`l`(*x*)	Log
`a`(*x*)	Arctangent
`j`(*n*,*x*)	Bessel function

All function arguments are passed by value.

NOTES

The `&&` and `||` operators are unavailable.

A `for` statement must have all three *expression*s.

The `quit` statement is interpreted when read, not when executed.

EXIT STATUS

The `obc` utility exits with one of the following values:

0 All input files were processed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following is a simple example of `obc` functionality.

```
$ obc
78*67 + 20*58 + 55*69
10181
120*(67+20+55+69)
25320
p=(211*120)
q=10181
p-q
15139
quit
```

Example 2: The following example is input to `obc`, which defines a function that computes an approximate value of the exponential function:

```
scale = 20
define e(x){
        auto a, b, c, i, s
        a = 1
        b = 1
        s = 1
        for(i=1; 1==1; i++){
                a = a*x
                b = b*i
                c = a/b
                if(c == 0) return(s)
                s = s+c
                }
        }
```

Example 3: The following example is input to `obc`, which prints approximate values of the exponential function of the first 10 integers:

```
for(i=1; i<=10; i++) e(i)
```

FILES

`/usr/lib/lib.b` Mathematical library

SEE ALSO

`odc`(1)

NAME

od – Dump files in various formats

SYNOPSIS

od [-A *address_base*] [-j *skip*] [-N *count*] [-t *type_string*] [-v] [*file...*]

Obsolescent version; may not be supported in future releases:

od [-b] [-c] [-d] [-f] [-n *nl*] [-o] [-s] [-x] [-p] [-B] [-C] [-W] [*file*] [[+]*offset*[.][b]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The od utility dumps *file* in one or more formats as selected by the argument. The *file* operand specifies the file(s) to be dumped. If no file argument is specified, the standard input is used. If no options are specified, -t o2 is the default.

The od utility accepts the following options:

-A *address_base*
: Specifies the input offset base. The *address_base* argument is a character. The characters d, o, and x specify that the offset base is written in decimal, octal, or hexadecimal, respectively. The character n specifies that the offset is not written.

-j *skip*
: Jumps over *skip* bytes from the beginning of the input. od seeks past the first *skip* bytes in the concatenated input files. If the combined input does not consist of at least *skip* bytes, od writes a diagnostic message to standard error and exits with a nonzero exit status.

  By default, the *skip* is interpreted as a decimal number. With a leading 0x or 0X, *skip* is interpreted as a hexadecimal number; otherwise, with a leading 0 it is interpreted as an octal number. When you append the character b, k, or m to *offset*, it is interpreted as a multiple of 512, 1024, or 1048576 bytes, respectively.

-N *count*
: Formats no more than *count* bytes of input. By default, *count* is interpreted as a decimal number. With a leading 0x or 0X, *count* is interpreted as a hexadecimal number; otherwise, with a leading 0, it is interpreted as an octal number. If *count* bytes of input (after successfully skipping, if -j is specified) are not available, input that is available is formatted.

`-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. The string may consist of the type specification characters `a`, `c`, `d`, `f`, `o`, `u`, and `x`, specifying named character, character, signed decimal, floating point, octal, unsigned decimal, and hexadecimal, respectively. 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 an optional `F`, `D`, or `L` indicating that the conversion should be applied to an item of type *float*, *double*, or *long double*, respectively. The type specification characters `d`, `o`, `u`, and `x` can be followed by an optional `C`, `S`, `I`, or `L` indicating that the conversion should be applied to an item of type *char*, *short*, *int*, or *long*, respectively. Multiple types can be concatenated within the same *type_string* and multiple `-t` options can be specified. Output lines are written for each type specified in the order in which the type specification characters were given.

For characters and integers, byte sizes of 1, 2, 4, and 8 are supported. For floating-point data types, byte size of 8 and 16 are supported.

`-v` Writes all input data. Without this option, any number of groups of output lines, which would be identical to the immediately preceding group of output lines (except for the byte offsets), are replaced with a line that contains only an asterisk (`*`).

In the obsolescent version, you may specify only one file operand. If no file argument is specified, the standard input is used.

The obsolescent version of `od` accepts the following options:

`-b` Interprets bytes in octal.

`-c` Interprets bytes in ASCII. Certain nongraphic characters appear as C escapes: null=`\0`, backspace=`\b`, `<form-feed>`=`\f`, `<newline>`=`\n`, `<carriage-return>`=`\r`, and `<tab>`=`\t`; others appear as 3-digit octal numbers.

`-d` Interprets words in unsigned decimal (equivalent to `-t u2`).

`-f` Interprets words in floating-point format.

`-n` *nl* Prints *nl* lines, two words per line.

`-o` Interprets words in octal (equivalent to `-t o2`).

`-s` Interprets words in signed decimal (equivalent to `-t d2`).

`-x` Interprets words in hexadecimal (equivalent to `-t x2`).

`-p` Prints parcels rather than words.

`-B` Prints address in bytes.

`-C` If the format is that of `-d`, `-o`, `-s`, or `-x`, prints an additional column that contains the ASCII interpretation of the bytes. A period represents nonprintable characters.

`-W` Prints address in words (default for all except `-b` and `-c`).

offset Specifies the offset in the file where dumping will commence. Usually, this argument is interpreted as octal bytes. If `.` is appended to *offset*, the offset is interpreted in decimal. If *offset* begins with `0x`, the base is hexadecimal. The base generates the address column on the left; thus `+0.` prints decimal addresses. If `b` is appended to *offset*, the offset is interpreted in blocks of 4096 bytes. If the *file* argument is omitted, the *offset* argument must be preceded by `+`.

Dumping continues until the end-of-file is reached, or until *nl* lines have been printed.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to dump any file.
`sysadm`	Allowed to dump any file, subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to dump any file.

The input data is manipulated in blocks of 16 bytes. Each input block is written as transformed by each output type, one per written line, in the order that the output types were specified.

When the specified character type specification is chosen, all nongraphic characters are written as a three-character name.

EXIT STATUS

The `od` utility exits with one of the following values:

0 All input files were processed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example assumes that you know that the file `myfile` contains ASCII characters. It is formatted as 1-byte character data shown on one line, and 8-byte hexadecimal data shown on another line. For easier study and review of the dump, the `od` output is redirected to a file named `mylist`:

```
od -t cx myfile > mylist
```

Example 2: The following example does the same task as Example 1, but it produces slightly different output by using the obsolescent version of `od`:

```
od -Cx myfile > mylist
```

Example 3: The following example dumps `512` bytes of the `textfile` file starting at byte `100`, suppressing the printing of the offset:

```
od -j 100 -N 512 -A n textfile
```

SEE ALSO

General UNICOS System Administration, Cray Research publication SG–2301

NAME

odc – Invokes the desk calculator

SYNOPSIS

odc [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The odc utility (formerly, the UNICOS 8.0 version dc(1) utility) is an arbitrary-precision arithmetic package. Ordinarily it operates on decimal integers, but you may specify an input base, output base, and several fractional digits to be maintained. (See obc(1).) The overall structure of odc is a stacking (reverse Polish) calculator. When an argument is specified, input is taken from *file* until its end and is then taken from standard input. The following constructions are recognized:

number Pushes the value of *number* onto the stack. *number* is an unbroken string of the digits 0 through 9. It may be preceded by an underscore (_) to input a negative number. Numbers may contain decimal points.

file Optional file containing input.

+ - * / % ^
Adds (+), subtracts (-), multiplies (*), divides (/), remainders (%), or exponentiates (^) the top two values on the stack. The two entries are popped off the stack; the result is pushed onto the stack in their places. Any fractional part of an exponent is ignored.

s *x* Pops the value off the top of the stack and stores it in a register named *x*, which can be any character. When the s is capitalized, *x* is treated as a stack and the value is pushed onto it.

l *x* Pushes the value in register *x* onto the stack. Register *x* is not altered. All registers start with 0 value. When the l is capitalized, register *x* is treated as a stack and its top value is popped off and pushed onto the main stack.

c Pops all values off the stack.

d Duplicates the top value on the stack.

f Prints all values of the stack.

i Pops the top value off the stack and uses it as the number radix for further input.

I Pushes the input base onto the top of the stack.

k Pops the top value off the stack and uses that value as a nonnegative scale factor. The appropriate number of decimal places is printed on output and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base will be reasonable if all are changed together.

K	Pushes the value of the scale factor onto the stack.
o	Pops the top value off the stack and uses it as the number radix for further output.
O	Pushes the output base onto the top of the stack.
p	Prints the top value of the stack. The top value remains unchanged.
P	Interprets the top of the stack as an ASCII string, removes it, and prints it.
Q	Pops the top value off the stack. The string execution level is popped by that value.
q	Exits the program. When a string is executed, the recursion level is popped by two.
v	Replaces the top element on the stack with its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.
x	Treats the top element of the stack as a character string and executes it as a string of odc commands.
X	Replaces the number on the top of the stack with its scale factor.
z	Pushes the value of the stack level onto the stack.
Z	Replaces the number on the top of the stack with its length.
[*string*]	Puts the bracketed ASCII string on the top of the stack.
<*x* >*x* =*x* !<*x* !>*x* !=*x*	Pops and compares the top two elements of the stack. Register *x* is evaluated if the values obey the stated relation. The exclamation point indicates negation.
!	Interprets the rest of the line as a UNICOS command. Control returns to odc when the command terminates.
?	Takes a line of input from the input source (usually the terminal) and executes it.
; :	Used by obc(1) for array operations.

MESSAGES

x is unimplemented	The *x* is an octal number.
stack empty	Not enough elements on the stack to do what was requested.
Out of space	Free list is exhausted (too many digits).
Out of headers	Too many numbers are kept.
Out of pushdown	Too many items on the stack.
Nesting depth	Too many levels of nested execution.

EXAMPLES

Example 1: The following is a simple example of odc functionality.

```
$ odc
210
100+
98+
63*
p
25704
quit
```

Example 2: The following example prints the first 10 values of n!:

```
$ odc
[la1+dsa*pla10>y]sy
0sa1
lyx
1
2
6
24
120
720
5040
40320
362880
3628800
quit
```

SEE ALSO

obc(1)

NAME

pack, pcat, unpack – Compresses and expands files

SYNOPSIS

```
pack [-] [-f] files
pcat files
unpack files
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pack utility tries to store the specified files in a compressed form. Wherever possible (and useful), each input file *file* is replaced by a packed file *file*.z with the same access modes, access and modified dates, and owner as those of *file*. The -f option is useful for causing an entire directory to be packed even if some of the files will not benefit. If *file*.z is successfully created, *file* will be removed. Packed files can be restored to their original form by using unpack or pcat.

The pack utility uses Huffman (minimum redundancy) codes on a byte-by-byte basis.

The pack utility accepts the following options:

- If you use the - argument, an internal flag is set, which prints the number of times each byte is used, its relative frequency, and the code for the byte on the standard output. Additional occurrences of - in place of *file* cause the internal flag to be set and reset.

-f Forces the packing of *files*.

files Specifies the files to be compressed or expanded.

The amount of compression obtained 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, it is usually not worthwhile to pack files smaller than three blocks, unless the character frequency distribution is very skewed, which may occur with printer plots or pictures.

Typically, text files are reduced to 60% to 75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

The pack utility returns a value representing the number of files that it failed to compress.

No packing occurs if any one of the following is true:

- The file appears to be already packed.
- The file has links.
- The file is a directory.
- The file cannot be opened.
- No disk storage blocks are saved by packing.
- A file called *file*`.z` already exists.
- The `.z` file cannot be created.
- An I/O error occurred during processing.

Directories cannot be compressed. The `pcat` utility does for packed files what `cat`(1) does for ordinary files, except that `pcat` cannot be used as a filter. The specified files are unpacked and written to the standard output. Thus, to view a packed file named *file*`.z`, enter one of the following:

```
pcat file.z
pcat file
```

To make an unpacked copy, `nnn`, of a packed file named `file.z` (without destroying `nnn`), use the following command:

```
pcat file >nnn
```

The `pcat` utility returns the number of files it was unable to unpack. If one or all of the following are true, failure may occur.

- The file cannot be opened.
- The file does not appear to be the output of *pack*.

The `unpack` utility expands files created by `pack`. For each file specified in the command, a search is made for a file called *file*`.z` (or just *file*, if *file* ends in `.z`). If this file appears to be a packed file, it is replaced by its expanded version. The new file has the `.z` suffix removed from its name, and it has the same access modes, access and modification dates, and owner as those of the packed file.

The `unpack` utility returns a value that is the number of files it was unable to unpack. Failure may occur for the same reasons that it may in `pcat`, as well as for the following reasons:

- A file with the ‘‘unpacked’’ name already exists.
- The unpacked file cannot be created.

NOTES

There are many zeros in Cray Research system executable files. Compression of 50% is possible, but the average is 35%.

SEE ALSO

`cat`(1), `compress`(1)

`chown`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

passwd – Changes login password

SYNOPSIS

passwd [-b] [*name*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The passwd utility changes or installs a password associated with the login *name*.

Ordinary users may change only the password that corresponds to their login *name*.

The passwd utility prompts you for your old password if you have one. (If you did not have a previous password, the system skips the old-password prompt.) The system then provides a prompt for the new password. Because the system does not echo the password to the screen, it provides a second new-password prompt to verify the correctness of your first entry.

The UNICOS system can use machine-generated passwords (if RANDOM_PASS_ON is enabled). If this feature is enabled, you are not allowed to select your own password. Instead, a machine-generated password is presented to you when using the passwd utility (see the EXAMPLES section). You have the choice of accepting the password or requesting another. You may continue to request new passwords by entering a carriage return until a suitable password is presented. Normal password requirement checking is not done when this feature is enabled.

The passwd utility accepts the following option:

-b Allows batch users to change their password. This option cannot be used if the machine-generated password feature is enabled. The old password followed by the new password is read from stdin.

The system administrator is responsible for defining the *aging* time limit for passwords. This means that if you have attempted to use your old password beyond the length of time defined by the system administrator, you may not create a new password. In this case, the system rejects your new password attempt and the passwd utility terminates.

If the old password meets the aging requirements defined by the system administrator, the system checks to ensure that the new password meets the construction requirements that follow. If it does, and if the new password is entered correctly for the first and second prompt, the new password is accepted. (When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the system repeats the prompting cycle.)

Passwords must meet the following requirements (these requirements are relevant only when the machine-generated password feature is not enabled):

- Each password must have at least 6 characters. Only the first 8 characters are significant.

- Each password must contain at least 2 alphabetic characters and at least 1 numeric or special character. In this case, ‘‘alphabetic’’ means mixed-case letters.
- Each password must differ from the user’s login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.
- The new password must differ from the old by at least 3 characters. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.

One whose effective user ID is 0 is called a super user; see `id`(1) and `su`(1). Because super users may change any password, `passwd` does not prompt super users for the old password. Super users are not forced to comply with password aging and password construction requirements. A super user can create a null password by entering a carriage return in response to the prompt for a new password.
NOTE: This is not recommended.

Only a user with an authorized `secadm` category may change another user’s password, according to the rules previously specified for the super user. Additionally, users are not allowed to log in with a null password.

NOTES

If this utility is installed with the default privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the actions shown:

Privilege Text	Action
`chgany`	Allowed to change any password.

EXAMPLES

Example 1: The following example shows how to change your password if you are a batch user:

```
$ passwd -b username << EOF
oldpassword
newpassword
EOF
```

Example 2: The following example shows how to change your password when the machine-generated password feature is enabled:

```
$ passwd
Old password:
Your new password is:  lempamdo
Re-enter new password or (CR) to get another:
```

FILES

`/etc/udb` User validation file containing user control limits

SEE ALSO

`id`(1), `login`(1), `privtext`(1), `su`(1)

`crypt`(3C) in *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

`udb`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

General UNICOS System Administration, Cray Research publication SG–2301

NAME

paste – Merges same lines of files or subsequent lines of a file

SYNOPSIS

```
paste files ...
paste -d list files ...
paste -s [-d list] files ...
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The paste utility concatenates corresponding lines of the given input *files*. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). It can be considered the counterpart of cat(1), which concatenates vertically (that is, one file after the other). In the last form in the SYNOPSIS section, paste combines subsequent lines of the input file (serial merging). In all cases, lines are glued together by using the <tab> character, or by using characters from an optionally specified *list*. Output is to standard output, so that it can be used as the start of a pipe, or as a filter, if - is used in place of a file name.

The paste utility accepts the following options:

-d Replaces the <newline> characters of each file except the last file (or last line in case of the -s option) with a <tab> character. This option allows replacing the <tab> character by one or more alternative characters (see below).

-s Merges subsequent lines rather than one from each input file. Use <tab> for concatenation, unless *list* is specified with the -d option. Regardless of the *list*, the last character of each file is forced to be a <newline> character.

list Replaces the default <tab> with one or more characters immediately following -d as the line concatenation character. The list is used circularly; that is, when exhausted, it is reused. In parallel merging (that is, no -s option), the lines from the last file are always terminated with a <newline> character, not from *list*. The list may contain the special escape sequences: \n (<newline>), \t (<tab>), \\ (backslash), and \0 (empty string, not a null character). If characters have special meaning to the shell, quoting may be necessary (for example, to get one backslash, use -d"\\\\").

- May be used in place of any file name to read a line from standard input. (No prompting occurs.)

files Specifies files in which lines will be merged.

NOTES

The `pr -t -m ...` command works similarly, but creates extra <blank>s, <tab>s, and <newline>s for a nice page layout.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to manage any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to manage any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to manage any input file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `paste` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

`paste: too many files (limit 12)`
When the `-s` option is not specified, you can specify up to 12 input files.

`paste: cannot allocate enough memory`
When processing files with line lengths greater than 2048 bytes, `paste` tries to allocate more space for its buffers. If a memory limit is reached, this error message will appear.

EXAMPLES

Example 1: This example lists files in the directory in one column:

```
ls | paste -d" " -
```

Example 2: This example lists files in the directory in four columns:

```
ls | paste - - - -
```

Example 3: This example combines pairs of lines into lines:

```
paste -s -d"\t\n" file
```

SEE ALSO

cat(1), cut(1), grep(1), pr(1)

NAME

patch – Applies a diff(1) file to an original file

SYNOPSIS

patch [-c | -e | -n] [-b] [-d *dir*] [-D *define*] [-i *patchfile*] [-l] [-N] [-o *outfile*] [-p *number*] [-r *rejectfile*] [-R] *file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The patch utility takes a patch file that contains any of the three forms of difference listings produced by the diff(1) utility and applies those differences to an original file, producing a patched version. By default, the patched version is put in place of the original. This POSIX version produces numbered backup (of the form *.n*). The -b option puts the back-up files in a file that has the same name but with the extension .orig.

To specify where you want to place output, use the -o option; if that file already exists, it is backed up first.

If you omit the -i option, the patch is read from standard input. At startup, patch tries to determine the type of the differences (diffs) listing, unless overruled by a -c, -e, or -n option. Context diffs (old-style and new-style) and normal diffs are applied by the patch program itself, while ed(1) diffs are simply fed to the ed(1) utility through a pipe.

The patch utility tries to skip any leading lines, apply the diff, and then skip any trailing lines. Thus, you can feed an article or message containing a diff listing to patch, and it should work. If the entire diff is indented by a consistent amount, this will be taken into account.

With context diffs, and to a lesser extent with normal diffs, patch detects when the line numbers mentioned in the patch are incorrect, and tries to find the correct place to apply each hunk of the patch. As a first guess, it takes the line number mentioned for the hunk, plus or minus any offset used in applying the previous hunk. If that is not the correct place, patch scans both forward and backward for a set of lines that matches the context given in the hunk.

First, patch looks for a place at which all lines of the context match. If no such place is found, and it is a context diff and the maximum fuzz factor is set to 1 or more, another scan occurs ignoring the first and last line of context. If that fails, and the maximum fuzz factor is set to 2 or more, the first two and last two lines of context are ignored, and another scan is made. (The default maximum fuzz factor is 2.) If patch does not find a place to install that hunk of the patch, it puts the hunk out to a reject file, which is the name of the output file plus the .rej extension. (The rejected hunk will come out in context-diff form whether the input patch was a context-diff or a normal diff.) If the input was a normal diff, many of the contexts will be null.) The line numbers on the hunks in the reject file may differ from the patch file: they reflect the approximate location patch thinks the failed hunks belong in the new file, rather than the old one.

As each hunk is completed, you will be informed whether the hunk succeeded or failed, and the line on which (in the new file) patch thinks the hunk should go. If this differs from the line number specified in the diff, the offset is displayed. A single large offset may indicate that a hunk was installed in the wrong place. patch also indicates whether a fuzz factor was used to make the match.

If you do not specify an original file on the command line, patch tries to determine from the leading lines, the name of the file to edit. In the header of a context diff, the file name is found from lines that begin with *** or ---, with the shortest name of an existing file being used. Only context-diffs have lines like that but if an Index: line is in the leading lines, patch tries to use the file name from that line. The context-diff header takes precedence over an Index line. If no file name can be determined from the leading lines, you will be prompted for the file name to patch.

For example, while in a news interface, you can specify the following line and patch a file in the blurfl directory directly from the article that contains the patch:

```
| patch -d /usr/src/local/blurfl
```

If the patch file contains more than one patch, patch tries to apply each of them as if they came from separate patch files. This means that it is assumed that the name of the file to patch must be determined for each diff listing, and that the lines before each diff listing will be examined for items such as file names and revision level, as mentioned previously.

The patch utility recognizes the following options:

-b	Saves a backup file of the original contents of each modified file, before the differences are applied, in a file of the same name with the .orig extension appended to it.
-c	Interprets the patch file as a context diff (the output of the diff(1) utility when the -c or -C option is specified.)
-d *dir*	Changes the current directory to *dir* before processing.
-D *define*	The #ifdef. . .#endif constructs are used to mark changes. The *define* argument is used as the differentiating symbol.
-e	Interprets the patch file as an ed(1) script.
-i *patchfile*	Reads the patch information from the file named by the path name *patchfile*, rather than the standard input.

`-l` Causes any sequence of blank characters in the diff script to match any sequence of blank characters in the input file. Other characters are matched exactly.

`-n` Interprets the patch file as a normal diff.

`-N` Ignores patches that may be reversed or already applied. By default, already-applied patches are rejected; see the `-R` option.

`-o` *file* 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 being written to *outfile*.

`-p` *number* Deletes *number* path name components from the beginning of each path name for all path names in the patch file that indicate the names of files to be patched. If the path name in the patch file is absolute, the leading slash(es) are considered the first component (for example, `-p 1` removes the leading slashes). Specifying `-p 0` causes the full path name to be used. If you omit the `-p` option, only the base name (the final path name component) is used.

`-r` *rejectfile* Overrides the default reject file name. In the default case, the reject file has the same name as the output file, with the suffix `.rej` appended to it.

`-R` Reverses the sense of the patch script. For example, assume that the diff script was created from the new version to the old version. The `-R` option cannot be used with `ed`(1) scripts. The `patch` utility attempts to reverse each portion of the script before applying it. Rejected differences is saved in swapped format. If this option is not specified, and until a portion of the patch file is successfully applied, `patch` attempts to apply each portion in its "reversed" sense as well as in its normal sense. If the attempt is successful, the user is prompted to determine if the `-R` option should be set.

ENVIRONMENT VARIABLES

`TMPDIR` Directory in which to put temporary files; default is `/tmp`.

NOTES

If you will be sending out patches, you should keep a `patchlevel.h` file, which is patched to increment the patch level as the first diff in the patch file you send out. Make sure you always specify the file names correctly, either in a context diff header, or with an `Index` line. If you are patching something in a subdirectory, tell the patch user to specify a `-p` patch as needed. You can create a file by sending out a diff that compares a null file to the file you want to create. This works only when the file you want to create does not exist already in the target directory. Do not send out reversed patches, because people may wonder whether they already applied the patch. Group related patches into separate easier-to-review files in case of problems.

EXIT STATUS

The `patch` utility exits with one of the following values:

0 Successful completion.

1 One or more lines were written to a reject file.

>1 An error occurred.

MESSAGES

The `Hmm...` message indicates that there is unprocessed text in the patch file and that `patch` is trying to see whether a patch is in that text and, if so, what kind of patch it is.

If any reject files were created, the `patch` utility exits with a nonzero status. When applying a set of patches in a loop, check this exit status so you don't apply a later patch to a partially patched file.

The `patch` utility cannot tell whether the line numbers are off in an `ed`(1) script, and can only detect bad line numbers in a normal diff when it finds a `change` or a `delete` command. A context diff using fuzz factor 3 may have the same problem. Until a suitable interactive interface is added, you should probably do a context diff in these cases to see if the changes made sense. Compiling without errors indicates that the patch probably worked, but not always.

The `patch` utility usually produces the correct results, even when it has to do a lot of guessing. However, the results 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.

BUGS

The `patch` utility could be smarter about partial matches, excessively-deviant offsets and swapped code, but that would take an extra pass. If code has been duplicated (for instance with `#ifdef OLDCODE ... #else ... #endif`), `patch` cannot patch both versions, and it may patch the wrong one and tell you that it succeeded.

If you apply a patch that you have already applied, `patch` treats it as a reversed patch and offers to unapply the patch.

FILES

`$TMPDIR/patch*`

SEE ALSO

`diff`(1), `ed`(1)

NAME

pathchk – Checks path names

SYNOPSIS

pathchk [-p] *pathnames...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pathchk utility checks that one or more path names are valid (for example, they can be used to access or create a file without causing syntax errors) and portable (for example, no file name truncation will result). The -p option provides more extensive portability checks.

By default, the pathchk utility checks each component of each *pathname* operand based on the underlying file system. A diagnostic is written for each *pathname* operand that has one of the following characteristics:

- Is longer than {PATH_MAX} bytes
- Contains any component longer than {NAME_MAX} bytes in its containing directory
- Contains any component in a directory that cannot be searched
- Contains any character in any component that is not valid in its containing directory

It is not an error if one or more components of a *pathname* operand does not exist as long as a file matching the path name specified by the missing components can be created that does not violate any of the preceding checks.

The pathchk utility accepts the following option and operand:

-p Writes a diagnostic for each *pathname* operand rather than performing checks based on the underlying file system:

- Is longer than {_POSIX_PATH_MAX} bytes
- Contains any component longer than {_POSIX_NAME_MAX} bytes
- Contains any character in any component that is not in the portable file name character set

pathnames Denotes path names to be checked.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `pathchk` utility exits with one of the following values:

0 All *pathname* operands passed all checks.

>0 An error occurred.

EXAMPLES

Example 1: The following shell script fragment verifies that all path names in an imported data interchange archive are legitimate and unambiguous on the current system:

```
pax -f archive | xargs pathchk
if [ $? -eq 0 ]
then
        pax -r -f archive
else
        echo Investigate problems before importing files.
        exit 1
fi
```

SEE ALSO

`getconf`(1)

`access`(2), `pathconf`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

pax – Portable archive interchange

SYNOPSIS

pax [-cdnv] [-f *archive*] [-s *replstr*] ... [*pattern* ...]

pax -r [-cdiknuv] [-f *archive*] [-o *options*] ... [-p *string*] ... [-s *replstr*] ... [*pattern* ...]

pax -w [-dituvX] [-b *blocksize*] [-a] [-f *archive*] [-o *options*] ... [-s *replstr*] ... [-x *format*] [*files* ...]

pax -r -w [-diklntuvX] [-p *string*] ... [-s *replstr*] ... [*file* ...] *directory*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pax utility reads, writes, and lists members of archive files and copies directory hierarchies. Archive formats supported are those that conform to the Archive/Interchange File Format specified in *IEEE Std. 1003.1-1990* (POSIX.1). These archive formats include ustar and cpio.

The action to be taken depends on the presence of the -r and -w options. The four combinations of -r and -w are referred to as the four modes of operation: *list*, *read*, *write*, and *copy* modes, corresponding respectively to the four forms shown in the SYNOPSIS section.

Mode **Description**

list In *list* mode (when neither the -r option nor the -w option is specified), pax writes the names of the members of the archive file read from the standard input, with path names that match the specified patterns, to standard output. If a specified file is of type directory, the file hierarchy rooted at that file will also be written out. In *list* mode, if -f is not specified, the standard input will be an archive file.

In this mode, pax lists normal files one per line, hard link path names as:

pathname == *linkname*

and symbolic link path names as:

pathname -> *linkname*

The *pathname* argument is the name of the file being extracted, and *linkname* is the name of a file that appeared earlier in the archive. If *pathname* is a symbolic link file, *linkname* might not exist in the archive.

If the -v option is specified, pax lists normal path names in the same format used by ls(1) with the -l option. Hard links are shown as follows:

<ls -l listing> == *linkname*

Symbolic links are shown as follows:

<ls -l listing> -> *linkname*

read In *read* mode (when -r is specified, but -w is not), pax extracts the members of the archive file with path names that match the specified patterns. If an extracted file is of type directory, the file hierarchy rooted at that file will also be extracted. The extracted files are created relative to the current file hierarchy. In *read* mode, if -f is not specified, the standard input will be an archive file.

The extracted or copied output files are of the archived file type.

The ownership, access and modification times, and file mode of the restored files are discussed under the -p option.

write In *write* mode (when -w is specified, but -r is not), pax writes the contents of the *file* operands to the standard output in an archive format. A *file* operand of type directory includes all of the files in the file hierarchy rooted at the file. In *write* mode, the standard input is used only if no *file* operands are specified. Standard input will be a text file that contain a list of path names, one per line, without leading or trailing <blanks>. Symbolic links are not followed (that is, they are stored on the archive as symbolic links). If the -f option is specified, the archive will be written to the file specified by the argument.

copy In *copy* mode (when both -r and -w are specified), pax copies the *file* operands to the destination *directory*. If no *directory* operands are specified, a list of files to copy, one per line, is read from the standard input. A *file* operand of type directory includes all of the files in the file hierarchy rooted at the file.

The effect of the copy will be as if the copied files were written to an archive file and then subsequently extracted, except that there hard links may be between the original and the copied files.

WARNING: A recursive copy occurs if the destination directory is a subdirectory of one of the source files. It is an error for the file specified by the *directory* operand not to exist, not be writable by the user, or not be a file of type directory.

In *read* or *copy* modes, intermediate directories are created as necessary to preserve the proper file hierarchy of the archive members.

If any specified *pattern* or *file* operands are not matched by at least one file or archive member, pax writes a diagnostic message to standard error for each one that did not match and exits with a nonzero exit status.

The supported archive formats will be detected automatically on input. The default output archive format is the `ustar` format.

If the selected archive format supports the specification of hard-linked files, it is an error if these files cannot be linked when the archive is extracted.

The `pax` utility accepts the following options and operands:

`-r` Reads an archive file from standard input.

`-w` Writes files to the standard output in the specified archive format.

`-a` Appends files to the end of the archive. The `-w` and `-f` options must also be specified. Appending to character-special devices (for example, tapes) is not supported.

`-b` *blocksize* Blocks the output at a positive decimal integer number of bytes per write to the archive file. A `b` suffix multiplies *blocksize* by 512, a `k` suffix multiplies *blocksize* by 1024, and an `m` suffix multiplies *blocksize* by 1,048,576. Devices and archive formats may impose restrictions on blocking. Blocking is determined automatically in *read* mode. Default blocking when creating archives on disk is 32,256 bytes. Default blocking when creating archives on tape depends on the archive format. (See the `-x` option.) The largest *blocksize* allowed is 32,256 bytes.

`-c` Matches all file or archive members except those specified by the *pattern* or *file* operand.

`-d` Causes files of type directory being copied or archived or archive members of type directory being extracted to match only the file or archive member itself and not the file hierarchy rooted at the file.

`-f` *archive* Specifies the path name of the input or output archive, overriding the default standard input (in *list* or *read* mode) or standard output (*write* mode). The `/dev/tty` file is used to write prompts and read responses.

`-i` Interactively rename files or archive members. For each archive member that matches a *pattern* operand or file that matches a *file* operand, a prompt is written to file `/dev/tty`. A line is then read from `/dev/tty`. If this line is blank, the file or archive member will be skipped. If this line consists of one period, the file or archive member will be processed with no modification to its name; otherwise, its name is replaced with the contents of the line. If end-of-file is encountered when reading a response or if `/dev/tty` cannot be opened for reading and writing, the `pax` utility immediately exits with a nonzero exit status.

`-k` Prevents the overwriting of existing files.

`-l` Links files. In *copy* mode, hard links are made between the source and destination file hierarchies whenever possible.

`-n` Selects the first archive member that matches each *pattern* operand. No more than one archive member are matched for each pattern (although members of type directory will still match the file hierarchy rooted at that file).

`-o` *options* Provides information to `pax` to modify the algorithm for extracting or writing files that is specific to the file format specified by `-x`. This option currently does not provide any functionality.

`-p` *string* Specifies one or more file characteristic options (privileges). The *string* argument is a string that specifies file characteristics to be retained or discarded on extraction. The string consists of the specification characters `a`, `e`, `m`, `o`, and `p`. Multiple characteristics can be concatenated within the same string and multiple `-p` options can be specified. The meanings of the specification characters are as follows:

`a` Does not preserve file access times.

`e` Preserves the user ID, group ID, file mode bits, access time, and modification time.

`m` Does not preserve file modification times.

`o` Preserves the user ID and group ID.

`p` Preserves the file mode bits.

In the preceding list, preserve indicates that an attribute stored in the archive will be given to the extracted file, subject to the permissions of the invoking process; otherwise, the attribute will be determined as part of the normal file creation.

If neither the `e` nor the `o` specification character is specified, or the user ID and group ID are not preserved for any reason, `pax` does not set the `S_ISUID` and `S_ISGID` bits of the file mode.

If the preservation of any of these items fails for any reason, `pax` writes a diagnostic message to standard error. Failure to preserve these items affects the final exit status, but does not delete the extracted file.

If file-characteristic letters in any of the *string* option-arguments are duplicated or conflict with each other, the one(s) specified last will take precedence (for example, if you specify `-p eme`, file modification times will be preserved).

`-s` *replstr* Modifies file or archive member names specified by *pattern* or *file* operand according to the substitution expression *replstr*, using the syntax of the `ed`(1) utility. The concepts of *address* and *line* are meaningless in the context of the `pax` utility, and they are not supported. The format is as follows:

```
-s /old/new/[gp]
```

In `ed`(1), *old* is a basic regular expression; and *new* can contain an ampersand, *n* (*n* is a digit) backreferences, or subexpression matching. The *old* string also may contain `<newline>` characters.

Any nonnull character can be used as a delimiter (/ shown here). Multiple `-s` expressions can be specified; the expressions are applied in the order specified, terminating with the first successful substitution. The optional trailing `g` is defined in the `ed`(1) utility and specifies the replacement of every occurrence of *old* with the pattern *new*. The optional trailing `p` causes successful substitutions to be written to standard error. File or archive member names that substitute to the empty string are ignored when reading and writing archives.

If the `-s` option is specified, and the replacement string has a trailing `p`, substitutions will be written to standard error in the following format:

<original path name> `>>` *<new pathname>*

`-t` Causes the access times of the archived files to be the same as they were before being read by `pax`.

`-u` Ignores files that are older (having a less recent file modification time) than a preexisting file or archive member that has the same name. In *read* mode, an archive member that has the same name as a file in the file system will be extracted if the archive member is newer than the file. In *copy* mode, the file in the destination hierarchy is replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer.

`-v` In *list* mode, produces a verbose table of contents; otherwise, writes archive member path names to standard error.

If `-v` is specified in *read*, *write*, or *copy* mode, `pax` writes the path names it processes to the standard error.

`-x` *format* Specifies the output archive format. The `pax` utility recognizes the following formats:

`cpio` The extended `cpio` interchange format. The default *blocksize* for this format for character special archive files is 5120. All *blocksize* values less than or equal to 32,256 that are multiples of 512 are supported.

`ustar` The extended `tar` interchange format. The default *blocksize* for this format for character special archive files is 10,240. All *blocksize* values less than or equal to 32,256 that are multiples of 512 are supported.

Any attempt to append to an archive file in a format different from the existing archive format causes `pax` to exit immediately with a nonzero exit status.

`-X` When traversing the file hierarchy specified by a path name, `pax` will not descend into directories that have a different device ID (see `stat`(2)).

The options that operate on the names of files or archive members (`-c`, `-i`, `-n`, `-s`, `-u`, and `-v`) interact as follows.

In *read* mode, the archive members are selected based on the user-specified *pattern* operands as modified by the `-c`, `-n`, and `-u` options. Then, any `-s` and `-i` options will modify, in that order, the names of the selected files. The `-v` option writes names that result from these modifications.

In *write* mode, the files are selected based on the user-specified path names as modified by the `-n` and `-u` options. Then, any `-s` and `-i` options will, in that order, modify the names of these selected files. The `-v` option writes names that result from these modifications.

If both the `-u` and `-n` options are specified, `pax` will not consider a file selected unless it is newer than the file to which it is compared.

directory Specifies the destination directory path name for *copy* mode.

file Specifies a path name of a file to be copied or archived.

pattern Specifies a pattern that matches one or more path names of archive members. You may specify pattern in the name-generating notation of the pattern matching notation in `sh`(1). If no *pattern* is specified, all members in the archive are selected.

EXIT STATUS

The `pax` utility exits with one of the following values:

0 All files were processed successfully.

>0 An error occurred.

MESSAGES

If `pax` cannot create a file or a link when reading an archive or cannot find a file when writing an archive, or cannot preserve the user ID, group ID, or file mode when the `-p` option is specified, a diagnostic message will be written to standard error and a nonzero exit status will be returned, but processing will continue. In cases in which `pax` cannot create a link to a file, `pax` will not, by default, create a second copy of the file.

If the extraction of a file from an archive is terminated prematurely by a signal or error, `pax` may have only partially extracted the file or (if the `-n` option was not specified) may have extracted a file of the same name as that specified by the user, but which is not the file the user wanted. Additionally, the file modes of extracted directories may have additional bits from the `S_IRWXU` mask set, as well as incorrect modification and access times.

BUGS

Special permissions may be required to copy or extract files.

When getting an `ls -l` style listing on `ustar` format archives, link counts are listed as 0 because the `ustar` archive format does not keep link count information.

EXAMPLES

Example 1: The following command copies the file hierarchy rooted at the current directory to an archive file in the user's temporary directory:

```
pax -w -f $TMPDIR/archive .
```

Example 2: The following command copies the contents of `olddir` to `newdir`:

```
pax -rw olddir newdir
```

Example 3: The following command reads the archive file `pax.out` with all files rooted in `/usr` in the archive extracted relative to the current directory:

```
pax -r -s,/usr/,, -f pax.out
```

SEE ALSO

`ed`(1), `find`(1), `sh`(1), `ls`(1)

`chmod`(2), `stat`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

ISO/IEC 9945-1:1990 (IEEE Std 1003.1-1990), *Information technology – Portable Operating System Interface (POSIX) – Part 1: System Application Program Interface (API)* [*C Language*]

NAME

perfscripts – Generates a file tree containing performance tool files

SYNOPSIS

perfscripts [-V]

IMPLEMENTATION

Cray PVP systems (except CRAY EL series)

DESCRIPTION

The perfscripts utility generates a file tree in your current directory. This tree contains several subtrees that contain scripts that can be used to generate reports from the raw output of the following performance tools:

- flowtrace
- perftrace
- prof(1)
- hpm(1)
- hpmall(8)

Each set of files has a README file, which explains how to use the scripts. In addition, sample data files are included that can be used with these scripts or with the appropriate post-processing tools such as flowview(1), perfview(1), and profview(1).

Each of these performance tools is discussed on its associated man page and in the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182.

You are free to alter these scripts to your own specifications. However, note that Cray Research will support only the script contents as they are initially written in your directory by this utility.

The perfscripts utility executes a shell script to create the directories and write the files. If the script cannot perform these actions, it issues error messages. During normal script execution, you will see a number of messages that look like the following:

```
shar: Extracting "performance/perftrace/report" (2856 characters)
```

The perfscripts utility accepts the following option:

-V Lists version number of perfscripts, along with a copyright notice, before creating the directories for the performance tools.

EXAMPLES

The following example shows an execution of `perfscripts` using the `-V` option:

```
$ perfscripts -V
perfscripts VERSION 70.0

   (c) COPYRIGHT CRAY RESEARCH, INC.

    UNPUBLISHED -- ALL RIGHTS RESERVED UNDER
    THE COPYRIGHT LAWS OF THE UNITED STATES

shar: Creating directory "performance"
shar: Creating directory "performance/flowtrace"
 .....
 .....
 .....
```

SEE ALSO

`csh`(1), `hpm`(1), `prof`(1), `sh`(1)

`flowtrace`(7), `performance`(7), `perftrace`(7) (available only online)

`hpmall`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

Guide to Parallel Vector Applications, Cray Research publication SG–2182

NAME

pg – File perusal filter for CRTs

SYNOPSIS

pg [-*number*] [-c] [-e] [-f] [-n] [-p *string*] [-s] [+*linenumber*] [+/*pattern*/] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The pg utility is a filter that allows the examination of each specified file one screenful at a time on a CRT. (The file name - and/or NULL arguments indicate that pg should read from the standard input.) Each screenful is followed by a prompt. If the user types a <carriage return>, another page is displayed; other possibilities are enumerated below.

This utility is different from previous paginators in that it allows you to back up and review something that has already passed. The method for doing this follows.

To determine terminal attributes, pg scans the terminfo(5) database for the terminal type specified by environment variable TERM. If TERM is not defined, the terminal type dumb is assumed.

The pg utility accepts the following options:

-*number* Specifies the number of lines the window pg is to use instead of the default. (On a terminal that has 24 lines, the default window size is 23.)

-c Homes the cursor and clears the screen before displaying each page. If clear_screen is not defined for this terminal type in the terminfo(5) database, this option is ignored.

-e Prevents pg from pausing at the end of each file.

-f Inhibits pg from splitting lines. Typically, pg splits lines longer than the screen width, but some sequences of characters in the text being displayed (for example, escape sequences for underlining) generate undesirable results.

-n This option causes an automatic end of command as soon as you enter a command letter. Typically, commands must be terminated by a <newline>.

-p *string* Causes pg to use *string* as the prompt. If the prompt string contains a %d, the first occurrence of %d in the prompt will be replaced by the current page number when the prompt is issued. The default prompt string is :.

-s Causes pg to print all messages and prompts in standout mode (usually inverse video).

+*linenumber* Starts at *linenumber*.

+/*pattern*/ Starts up at the first line containing the basic regular expression pattern.

files Files to be displayed.

The responses that may be typed when pg pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment.

Commands that cause further perusal usually take a preceding *address*, which is an optionally signed number indicating the point from which further text should be displayed. This address is interpreted in either pages or lines depending on the command. A signed address specifies a point relative to the current page or line, and an unsigned address specifies an address relative to the beginning of the file. Each command has a default address that is used if none is provided.

The perusal commands and their defaults are as follows:

(+1)<newline> or <blank>
Causes one page to be displayed. The address is specified in pages.

(+1) l With a relative address, causes pg to simulate scrolling the screen, forward or backward, the number of lines specified. With an absolute address, prints a screenful beginning at the specified line.

(+1) d or ^D Simulates scrolling half a screen forward or backward.

The following perusal commands take no address:

. or ^L Causes the current page of text to be redisplayed.

$ Displays the last windowful in the file. Use with caution when the input is a pipe.

The following commands are available for searching for text patterns in the text. The regular expressions described in ed(1) are available. They must always be terminated by a <newline>, even when the -n option is specified.

i/*pattern*/ Searches forward for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately after the current page and continues to the end of the current file, without wrap-around.

i^*pattern*
i?*pattern*? Searches backward for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately before the current page and continues to the beginning of the current file, without wrap-around. The ^ notation is useful for Adds 100 terminals that will not properly handle the ?.

After searching, pg usually displays the line found at the top of the screen. This can be modified by appending m or b to the search command to leave the line found in the middle or at the bottom of the window from now on. The t suffix can be used to restore the original situation.

The user of pg can modify the environment of perusal with the following commands:

*i*n	Begins perusing the *i*th next file in the command line. The *i* is an unsigned number; default value is 1.
*i*p	Begins perusing the *i*th previous file in the command line. The *i* is an unsigned number; default is 1.
*i*w	Displays another window of text. If *i* is present, sets the window size to *i*.
s *filename*	Saves the input in the named file. Only the current file being perused is saved. The white space between the s and *filename* is optional. This command must always be terminated by a <newline>, even when the -n option is specified.
h	Displays an abbreviated summary of available commands.
q or Q	Quits pg.
!*command*	*command* is passed to the shell, whose name is taken from the SHELL environment variable. If this is not available, the default shell is used. This command must always be terminated by a <newline>, even when the -n option is specified.

At any time when output is being sent to the terminal, you can press the quit key (<CONTROL-\>) or the interrupt (<CONTROL-c>) key. This causes pg to stop sending output and to display the prompt. The user may then enter one of the preceding commands in the normal manner. Unfortunately, some output is lost when this is done, because any characters waiting in the terminal's output queue are flushed when the QUIT signal occurs.

If the standard output is not a terminal, pg acts just like cat(1), except that a header is printed before each file (if there is more than one file).

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

While waiting for terminal input, pg responds to <CONTROL-c>, <DEL>, and ^ by terminating execution. Between prompts, however, these signals interrupt the current task of pg and place the user in prompt mode. These should be used with caution when input is being read from a pipe, because an interrupt is likely to terminate the other commands in the pipeline.

Users of the more(1) utility will find that the z and f commands are available and that the terminal /, ^, or ? may be omitted from the searching commands.

EXIT STATUS

The pg utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

If terminal tabs are not set every eight positions, undesirable results may occur.

When you use pg as a filter with another command that changes the terminal I/O options, terminal settings may not be restored correctly.

Nulls in a character string cause the current line to terminate on output.

EXAMPLES

The following is a sample usage of pg for reading system news:

```
news | pg -p "(Page %d):"
```

FILES

/usr/lib/terminfo/?/*	Terminal information database
/tmp/pg*	Temporary file when input is from a pipe

SEE ALSO

cat(1), ed(1), grep(1), more(1), pr(1)

terminfo(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

pr – Prints files

SYNOPSIS

pr [-*column*] [-w [*width*]] [-a] [+*page*] [-d] [-e[*ck*]] [-f] [-F] [-h *header*] [-i[*ck*]] [-l [*length*]] [-L] [-n[*ck*]] [-o [*offset*]] [-p] [-r] [-s[*separator*]] [-t] [*files*]

pr [-m] [-w [*width*]] [+*page*] [-d] [-e[*ck*]] [-f] [-F] [-h *header*] [-i[*ck*]] [-l [*length*]] [-L] [-n[*ck*]] [-o [*offset*]] [-p] [-r] [-s[*separator*]] [-t] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-f and -p options)

DESCRIPTION

The pr utility formats and prints the contents of a file. If *file* is -, or if no files are specified, pr assumes standard input. pr prints the specified files on standard output.

By default, the listing is separated into pages, each headed by the page number, a date and time, and the name of the file. (The date and time is the last modification time, unless input is redirected from standard input; in this case, the current date and time is displayed.) Page length is 66 lines, which includes 10 lines of header and trailer output. The header is composed of 2 blank lines, 1 line of text (which can be altered with -h) and 2 blank lines; the trailer is 5 blank lines. For single-column output, line width may not be set and is unlimited. For multicolumn output, line width may be set, and the default is 72 columns. Diagnostic reports (failed options) are reported at the end of standard output associated with a terminal, rather than interspersed in the output. Pages are separated by a series of <newline> characters rather than <form-feed> characters.

By default, columns are of equal width, separated by at least one <space>; lines that do not fit are truncated. If you specify the -s option, lines are not truncated and columns are separated by the *separator* character.

To produce multicolumn output, use either -*column* or -m; -a must be used only with -*column* and not -m.

The pr utility accepts the following options:

NOTE: Space is not permitted between the following options and their arguments, except for -h, -l, -o, and -w. The -h option requires a <space>.

-*column* Prints *column* columns of output (default is 1). Output appears as if -e and -i are turned on for multicolumn output. Do not use with -m.

`-m` Merges and prints all files simultaneously, one per column. The maximum number of files that may be specified is 8. If a line is too long to fit in a column, it is truncated. Do not use with `-`*column*.

`-w` [*width*] Sets the width of a line to *width* character positions (default is 72). This is effective only for multicolumn output (`-`*column* and `-m`). No line limit exists for single-column output.

`-a` Prints multicolumn output across the page, one line per column. *column* must be greater than 1. If a line is too long to fit in a column, it is truncated.

`+`*page* Begins printing with page numbered *page* (default is 1).

`-d` Double-spaces the output. Blank lines that result from double-spacing are dropped when they occur at the top of a page.

`-e`[*ck*] Expands each input `<tab>` character to character positions *k*+1, 2**k*+1, 3**k*+1, and so on. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. All `<tab>` characters in the input are expanded into the appropriate number of `<space>`s. If *c* (any nondigit character) is given, it is treated as the input `<tab>` character. (Default for *c* is the `<tab>` character.)

`-L` Folds the lines of the input file. When used in multicolumn mode (with the `-a` or `-m` option) lines will be folded to fit the current column's width; otherwise, they will be folded to fit the current line width.

`-F` Uses one `<form-feed>` character for new pages. (Default is to use a sequence of `<newline>` characters.)

`-f` Uses one `<form-feed>` character for new pages. (Default is to use a sequence of `<newline>` characters.) If the standard output is associated with a terminal, it pauses before beginning the first page.

`-h` *header* Uses *header* as the text line of the header to be printed instead of the file name. `-h` is ignored when `-t` is specified, or when `-l` *length* is specified and the value of *length* is 10 or fewer. A `<space>` is required between the `-h` option and its argument.

`-i`[*ck*] In output, replaces white space when possible by inserting `<tab>` characters to character positions *k*+1, 2**k*+1, 3**k*+1, and so on. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any nondigit character) is given, it is treated as the output `<tab>` character (default for *c* is the `<tab>` character).

`-l` [*length*] Sets the length of a page to *length* lines (default is 66). `-l0` is reset to `-l66`. When the value of *length* is 10 or less, `-t` appears to be in effect because headers and trailers are suppressed. By default, output contains 5 lines of header and 5 lines of trailer, leaving 56 lines for user-supplied text. If *length* exceeds 10, *length*–10 lines are left per page for user-supplied text. When *length* is 10 or less, header and trailer output is omitted to make room for user-supplied text.

`-n`[*ck*]	Provides *k*-digit line numbering (default for *k* is 5). The number occupies the first *k*+1 character positions of each column of single-column output or each line of `-m` output. If *c* (any nondigit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a `<tab>`).
`-o` [*offset*]	Offsets each line by *offset* character positions (default is 0). The number of character positions per line is the sum of the width and offset.
`-p`	Pauses before beginning each page if the output is directed to a terminal. (`pr` rings the bell at the terminal and waits for a `<carriage-return>`.)
`-r`	Prints no diagnostic reports on files that will not open.
`-s`[*separator*]	Separates columns by the single-character *separator* instead of by the appropriate number of `<space>` characters. (Default for *separator* is a `<tab>`.) Prevents truncation of lines on multicolumn output, unless `-w` is specified.
`-t`	Does not print the five-line identifying header or the five-line trailer. Quits printing after the last line of each file without spacing to the end of the page. Use of the `-t` options overrides the `-h` option.
files	Specifies the files to print.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `pr` utility exits with one of the following values:

0 All files were written successfully.

>0 An error occurred.

EXAMPLES

Example 1: To print `file1` and `file2` as a double-spaced, three-column listing headed by `file list`, enter the following command line:

```
$ pr -3dh "file list" file1 file2
```

Example 2: To copy `file1` to `file2`, expanding tabs to columns 10, 19, 28, 37, . . ., enter the following command line:

```
$ pr -e9 -t <file1 >file2
```

Example 3: To print `file1` and `file2` simultaneously in a two-column listing, with no header or trailer, where both columns have line numbers, enter the following command line:

```
$ pr -t -n file1 | pr -t -m -n file2 -
```

Example 4: To print `file1` with line numbers 0001 through 9999 and page breaks and headers every 66 lines, enter the following:

```
$ pr -n4 -l66 file1
```

FILES

`/dev/tty*` Terminal devices

SEE ALSO

`cat`(1), `more`(1), `pg`(1)

NAME

printenv – Prints the environment variable values

SYNOPSIS

/usr/ucb/printenv *name*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The printenv utility prints the values of the variables in the environment.

The printenv utility accepts the following operand:

name When *name* is specified, only the value is printed.

EXIT STATUS

If you specify *name* and it is not defined in the environment, printenv returns exit status 1; otherwise, it returns status 0.

SEE ALSO

env(1), sh(1)

NAME

printf – Writes formatted output

SYNOPSIS

printf *format* [*argument* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The printf utility writes formatted operands to the standard output. The *argument* operands are formatted under control of the *format* operand.

The printf utility accepts the following operands:

format A string that describes the format to use to write the remaining operands.

argument The strings to be written to standard output, under the control of *format*.

The *format* operand is used as the *format* string described in printf(3C), except for the following:

- A <space> character in the format string, in any context other than a flag of a conversion specification, is treated as an ordinary character that is copied to the output.
- The following escape sequences are supported:

  \\ Represents the backslash character
  \a Represents the <alert> character (octal 07)
  \b Represents the <backspace> character (octal 010)
  \f Represents the <form-feed> character (octal 014)
  \n Represents the <newline> character (octal 012)
  \r Represents the <carriage-return> character (octal 015)
  \t Represents the <tab> character (octal 011)
  \v Represents the <vertical-tab> character (octal 013)

  In addition, *ddd* is supported; *ddd* is a one-, two-, or three-digit octal number that is written as a byte with the numeric value specified by the octal number.
- The printf utility will not precede or follows output from the d or u conversion specifications with <blank> characters not specified by the *format* operand.
- The printf utility will not precede output from the o conversion specification with zeros not specified by the *format* operand.

- An additional conversion character, `b`, is supported as follows. The argument is taken to be a string that may contain backslash-escape sequences. The following backslash-escape sequences are supported:
  - The escape sequences listed in item 2 are converted to the characters they represent.
  - `\0`*ddd*; *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.
  - `\c`, which is not written and causes `printf` to ignore any remaining characters in the string operand that contains it, any remaining string operands, and any additional characters in the *format* operand.

  The interpretation of a backslash followed by any other sequence of characters is interpreted as the character immediately following the backslash.

  Bytes from the converted string are written until the end of the string or the number of bytes indicated by the precision specification is reached. If the precision is omitted, it will be taken to be infinite, therefore, all bytes up to the end of the converted string are written.
- For each specification that consumes an argument, the next argument operand is evaluated and converted to the appropriate type for the conversion as specified below.
- The *format* operand is reused as often as necessary to satisfy the argument operands. 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 0 argument were supplied. If the *format* operand contains no conversion specifications and *argument* operands are present, the *argument*s are ignored.
- If a character sequence in the *format* operand begins with a `%` character, but does not form a valid conversion specification, the character immediately following the `%` is printed.

The *argument* operands are treated as strings if the corresponding conversion character is `b`, `c`, or `s`; otherwise, it is evaluated as a C constant, as described by the C Standard, with the following extensions:

- A leading plus or minus sign is allowed.
- If the leading character is a single or double quotation mark, the value is the numeric value in the underlying code set of the character following the single or double quotation mark.

If an argument operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a diagnostic message is written to standard error and the utility does not exit with a zero exit status, but it continues processing any remaining operands and writes to standard output the value accumulated at the time the error was detected.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In an administrator privileged shell environment, shell-redirected I/O is not subject to security label restrictions.

If the PRIV_SU configuration option is enabled, and if the user is the super user, shell-redirected I/O is not subject to security label restrictions.

EXIT STATUS

The printf utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following shell script fragment prompts for two responses from the user:

```
printf "\aPlease fill in the following: \nName: "
read name
printf "Phone Number: "
read phone
```

Example 2: The following example reads from a file that contains two integer values per line, right and wrong, then calculates the quotient, printing it out as follows:

```
$ while read right wrong ; do
>      percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
>      printf "%2d right\+%2d wrong\+(%s%%)\n" $right $wrong $percent
> done < $TMPDIR/database_file
```

Example 3: The following example produces the following output:

```
$ printf "%5d%4d\n" 1 21 321 4321 54321
    1  21
  3214321
54321   0
```

SEE ALSO

echo(1), sh(1)

printf(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`privtext` – Gets the privilege text of a file

SYNOPSIS

`privtext` [`-a`] [`-c` *category_list*] *files...*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `privtext` utility displays the privilege text character sequence. The privilege text displayed is the one assigned to you when you invoke the files identified by *files*, based on your current active category.

If you specify the `-c` option, `privtext` displays the privilege text that is assigned to any user who has any of the active categories specified in *category_list*, upon execution of each of the files identified by *files*.

If the `-a` option is specified, *privtext* displays privilege text information for each of your authorized categories.

The `privtext` utility accepts the following options and operands:

`-a`	Displays privilege text for each of the user's authorized categories.
`-c`	Displays privilege text for each active category specified in *category_list*.
category_list	List of active categories. If multiple categories are specified, they must be separated by commas, with no intervening white space.
files	File names for which privilege text is to be retrieved.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Shell-redirected output is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `privtext` utility exits with one of the following values:

0 Privilege text was successfully displayed for all specified commands.

1 An invalid or badly formed option was supplied.

2 Privilege text was displayed for some, but not all commands.

4 Privilege text could not be displayed for any of the specified commands.

EXAMPLES

The following examples assume that you have the `secadm` and `sysadm` categories authorized, the privilege assignment list (PAL) for `othercommand` is empty, and `somecommand` has been assigned the following PAL:

```
a:PRIV_NULL
f:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE
s:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE

system:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:texta
secadm:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:texta
sysadm:PRIV_DAC_OVERRRIDE:textb
other:PRIV_NULL:TEXT_NULL
```

Example 1: The following example shows the retrieval of the privilege text assigned to you if you have no active categories when you invoke `somecommand`:

```
$ privtext somecommand
other:TEXT_NULL
```

Example 2: The following example shows the use of the `-c` option, which displays the privilege text for the specified active category. In this example, the display shows the privilege text assigned to you if you have an active `sysadm` category when you invoke `somecommand`.

```
$ privtext -c sysadm somecommand
sysadm:textb
```

Example 3: The following example shows the use of the `-a` option, which displays privilege text assigned to all categories authorized for you when you invoke `somecommand`. The `other` category is retrieved to show the case where you do not have any active categories.

```
$ privtext -a somecommand
secadm:texta
sysadm:textb
other:TEXT_NULL
```

Example 4: The following example shows the use of the `-c` option, which displays the privilege text assigned to a user with any of the specified categories active. Use of this option allows you to determine what privilege text would be assigned to an administrative role, not a specific user. Note that you do not need to have the category authorized to retrieve information about the category. Because the PAL for `somecommand` does not have an entry for `netadm`, the match is made from the `other` entry:

```
$ privtext -c netadm,secadm somecommand
netadm:TEXT_NULL
secadm:texta
```

Example 5: The following example shows the use of both the `-a` and `-c` options, which display the privilege text for each category that is authorized for you and for any user with an active `netadm` category (even though the `netadm` category is not authorized for you). The `other` category is retrieved to show the case where you do not have any active categories:

```
$ privtext -a -c netadm,secadm somecommand
netadm:TEXT_NULL
secadm:texta
sysadm:textb
other:TEXT_NULL
```

Example 6: The following example shows the privilege text retrieved for `somecommand` and `othercommand` for each category authorized for you. The `other` category is retrieved to show the case where you do not have any active categories.

```
$ privtext -a somecommand othercommand
# somecommand:
secadm:texta
sysadm:textb
other:TEXT_NULL
# othercommand:
secadm:TEXT_NULL
sysadm:TEXT_NULL
other:TEXT_NULL
```

NAME

procstat – Gathers I/O and process statistics

SYNOPSIS

procstat [-R *rawfile*] [-r *rptfile*] [-m] [-i] [-d] [-V] *commands*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The procstat utility produces information on the current activity of the specified commands. The information always includes process start and exit, and by default includes memory activity and I/O usage (including secondary data segments (SDS) usage). The recommended way to read procstat output is by means of the procview(1) utility; the procstat -r option can also be used to create a report with a format that is less convenient.

The procstat utility accepts the following options:

-R *rawfile* Writes file *rawfile* from which procview can create a report. This is the recommended usage. Either -r or -R (or both) must be specified.

-r *rptfile* Writes a formatted report to *rptfile*.

-m Reports on memory activity, as reported by sbreak (see brk(2)). This information is included by default; this option has the effect of turning off -i unless that is also specified.

-i Reports specifically on I/O activity (including SDS usage). This information is included by default; this option has the effect of turning off -m unless that is also specified.

-d Gathers detailed I/O information on each operation counted by procstat. This option should be used with caution, because it can generate a very large amount of information.

-V Lists the version of procstat on stderr, along with a copyright message.

commands Specifies the commands to be monitored by procstat. Arguments to these commands can be included on the same line. There is no default.

NOTES

The procstat utility does not support macrotasked programs. Autotasked programs are supported only if the NCPUS environment variable is set to 1. The statistics provided by procstat are limited to tracking by process ID, not by multitasking group. With multitasking groups, the statistics for file I/O need to be associated with OS multitasking groups to be correct.

Several more system calls, including ialloc(2), dup(2), fork(2), and ioctl(2) do not have hooks to provide information to procstat. Problems that stem from misuse of the heap may abort when run under procstat, even though they appear to execute correctly in normal circumstances.

I/O waiting times shown for interactive problems will be high for files such as stdin and stdout.

Statistics for READA and WRITEA may be inaccurate.

If a program aborts when run with procstat, it is usually caused by the program's incorrect use of the malloc(3C) subroutine.

BUGS

When a program terminates abnormally, some of the information still in the buffers in child processes is not flushed down the pipe.

The default report does not include all the information in the packets from the child processes.

Asynchronous I/O is not very well handled.

The procstat utility cannot identify certain file types, such as sockets.

EXAMPLES

The following example writes a report in file a.report, for command a.out, which has the arguments arg1 and arg2.

```
$ procstat -R a.raw a.out arg1 arg2
$ procview -L -Sn a.raw > a.report
```

SEE ALSO

procview(1)

brk(2), dup(2), fork(2), ialloc(2), ioctl(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

malloc(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

performance(7) (available only online)

NAME

prs – Prints an SCCS file

SYNOPSIS

prs [-d[*dataspec*]] [-r[*SID*]] [-e] [-l] [-c *date-time*] [-a] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The prs utility prints, on the standard output, parts or all of a Source Code Control System (SCCS) file (see sccsfile(5)) in a user-supplied format. If a directory is named, it behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored.

Options to prs, which may appear in any order, consist of keyletter arguments and file names.

All the described options apply independently to each named file.

The prs utility accepts the following options:

-d[*dataspec*] Specifies the output data specification. The *dataspec* is a string consisting of SCCS file data keywords (see the Data Keywords subsection) interspersed with optional user-supplied text.

-r[*SID*] Specifies the SCCS identification (SID) string of a delta for which information is desired. If no SID is specified, the SID of the most recently created delta is assumed.

-e Requests information for all deltas created earlier than and including the delta designated, using the -r option or the date specified by the -c option.

-l Requests information for all deltas created later than and including the delta designated, using the -r option or the date specified by the -c option.

-c[*date-time*] Cutoff *date-time*. Appears in the form:

```
YY[MM[DD[HH[MM[SS]]]]]
```

Units omitted from the *date-time* default to their maximum possible values. That is, -c7502 is equivalent to -c750228235959. Any number of nonnumeric characters may separate the various 2-digit units of the *date-time* in the form: "-c77/2/2 9:22:25".

`-a` Requests printing of information for both removed deltas. That is, delta type = *R* (see `rmdel`(1)) and existing deltas, that is, delta type = *D*. If the `-a` option is not specified, information for only existing deltas is provided.

files Specifies the SCCS files to print.

Data Keywords

Data keywords (see table 1) specify which parts of an SCCS file are to be retrieved and output. All parts of an SCCS file (see `sccsfile`(5)) have an associated data keyword. There is no limit on the number of times a data keyword may appear in a *dataspec*.

The information printed by `prs` consists of: the user-supplied text and appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the order of appearance in the *dataspec*. The format of a data keyword value is either simple (S), in which keyword substitution is direct, or multiline (M), in which keyword substitution is followed by a carriage return.

User-supplied text is any text other than recognized data keywords. A tab is specified by `\t` and carriage return/`new line` is specified by `\n`. The default data keywords are as follows:

```
":Dt:\t:DL:\nMRs:\n:MR:COMMENTS:\n:C:"
```

Table 1. SCCS Files Data Keywords

Keyword	Data item	File section	Value	Format
`:Dt:`	Delta information	Delta table	See below†	S
`:DL:`	Delta line statistics	"	`:Li:/:Ld:/:Lu:`	S
`:Li:`	Lines inserted by delta	"	*nnnnn*	S
`:Ld:`	Lines deleted by delta	"	*nnnnn*	S
`:Lu:`	Lines unchanged by delta	"	*nnnnn*	S
`:DT:`	Delta type	"	`D`~or~`R`	S
`:I:`	SCCS ID string (SID)	"	`:R:.:L:.:B:.:S:`	S
`:R:`	Release number	"	*nnnn*	S
`:L:`	Level number	"	*nnnn*	S
`:B:`	Branch number	"	*nnnn*	S
`:S:`	Sequence number	"	*nnnn*	S
`:D:`	Date delta created	"	`:Dy:/:Dm:/:Dd:`	S
`:Dy:`	Year delta created	"	*nn*	S
`:Dm:`	Month delta created	"	*nn*	S
`:Dd:`	Day delta created	"	*nn*	S
`:T:`	Time delta created	"	`:Th:::Tm:::Ts:`	S
`:Th:`	Hour delta created	"	*nn*	S
`:Tm:`	Minutes delta created	"	*nn*	S
`:Ts:`	Seconds delta created	"	*nn*	S
`:P:`	Programmer who created delta	"	*logname*	S
`:DS:`	Delta sequence number	"	*nnnn*	S

Table 1. SCCS Files Data Keywords (continued)

Keyword	Data item	File section	Value	Format
:DP:	Predecessor delta seq #	"	*nnnn*	S
:DI:	Seq # of deltas included, excluded, or ignored	"	:Dn:/:Dx:/:Dg:	S
:Dn:	Deltas included (seq #)	"	:DS:~:DS:. . .	S
:Dx:	Deltas excluded (seq #)	"	:DS:~:DS:. . .	S
:Dg:	Deltas ignored (seq #)	"	:DS:~:DS:. . .	S
:MR:	MR numbers for delta	"	*text*	M
:C:	Comments for delta	"	*text*	M
:UN:	User names	User names	*text*	M
:FL:	Flag list	Flags	*text*	M
:Y:	Module type flag	"	*text*	S
:MF:	MR validation flag	"	yes~or~no	S
:MP:	MR validation program name	"	*text*	S
:KF:	Keyword error/warning flag	"	yes~or~no	S
:KV:	Keyword validation string	"	*text*	S
:BF:	Branch flag	"	yes~or~no	S
:J:	Joint edit flag	"	yes~or~no	S
:LK:	Locked releases	"	:R:. . .	S
:Q:	User-defined keyword	"	*text*	S
:M:	Module name	"	*text*	S
:FB:	Floor boundary	"	:R:	S
:CB:	Ceiling boundary	"	:R:	S
:Ds:	Default SID	"	:I:	S
:ND:	Null delta flag	"	yes~or~no	S
:FD:	File descriptive text	Comments	*text*	M
:BD:	Body	Body	*text*	M
:GB:	Gotten body	"	*text*	M
:W:	A form of what(1) string	NA	:Z::M:\t:I:	S
:A:	A form of what(1) string	NA	:Z::Y:~:M:~:I::Z:	S
:Z:	what(1) string delimiter	NA	@(#)	S
:F:	SCCS file name	NA	*text*	S
:PN:	SCCS file path name	NA	*text*	S

† :Dt:~=~:DT:~:I:~:D:~:T:~:P:~:DS:~:DP:

MESSAGES

Error messages from SCCS are printed. Use help(1) for explanations.

EXAMPLES

Example 1:

```
$ prs -d"Users and/or user IDs for :F: are: \n:UN:" s.file
```

May produce the following on the standard output:

```
Users and/or user IDs for s.file are:
xyz 131 abc
```

```
$ prs -d"Newest delta for pgm :M:: :I: Created :D: By :P:" -r s.file
```

May produce the following on the standard output:

```
Newest delta for pgm file: 1.2 Created 92/03/12 By cl
```

Example 2: As a special case:

```
$ prs s.file
```

May produce the following on the standard output for each delta table entry of the D type:

```
s.file:

D 1.2 92/03/12 09:56:16 cl 2 1  00001/00000/00005
MRs:
COMMENTS:
added one more line

D 1.1 92/03/12 09:45:26 cl 1 0  00005/00000/00000
MRs:
COMMENTS:
date and time created 92/03/12 09:45:26 by cl
```

The only keyletter argument allowed to be used with the special case is the -a option.

FILES

/tmp/pr*?????* Temporary working file

SEE ALSO

admin(1), cdc(1), comb(1), delta(1), get(1), help(1), rmdel(1), sact(1), sccsdiff(1), unget(1), val(1), vc(1), what(1)

sccsfile(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

ps – Reports process status

SYNOPSIS

ps [-a] [-A] [-c *corefile*] [-d] [-e] [-f] [-g *pgrplist*] [-G *grplist*] [-j *jidlist*] [-l] [-L] [-m] [-M] [-n *namelist*] [-o *format*] [-p *proclist*] [-t *termlist*] [-u *uidlist*] [-U *uidlist*] [-w]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-c, -n, and -u options)
CRI extensions (-j, -m, -L, and -M options)

DESCRIPTION

The ps utility prints certain information about active processes. Without options, information is printed only about processes associated with the same effective user ID and the same controlling terminal as the invoker. The output is a short listing that consists of only the process ID, terminal identifier, cumulative time, and command name. Otherwise, the options control information that is displayed.

Options that use lists as arguments can have the list specified in one of two forms: a list of identifiers separated from one another by a comma, or a list of identifiers enclosed in double quotation marks and separated from one another by a comma and/or one or more spaces.

The ps utility accepts the following options:

-a	Prints information about all processes, except process group leaders and processes, not associated with a terminal.
-A	Same as -e option.
-c *corefile*	Uses the *corefile* file in place of /dev/mem.
-d	Prints information about all processes, except process group leaders.
-e	Prints information about all processes.
-f	Generates a full listing.
-g *pgrplist*	Restricts listing to data about processes that have process group leaders specified in *pgrplist*.
-G *grplist*	Restricts listing to data about processes whose group ID numbers or group names are given in *grplist*.
-j *jidlist*	Restricts listing to data about processes whose job ID numbers are specified in *jidlist*.
-l	Generates a long listing.

`-L` Same as `-l` option except that the PRI field is represented symbolically.

`-m` (CRAY T3D systems only) Prints information about active CRAY T3D processes and the partition with which they are associated. This option prints the UNICOS process information and the CRAY T3D partition information.

`-M` (CRAY T3D systems only) Prints only information about CRAY T3D partitions.

`-n` *namelist* Takes *namelist* as the name of an alternative system namelist file in place of `/unicos`. Only appropriately authorized users can successfully use this option.

`-o` *format* Produces a listing according to the specified *format*. (*format* is defined in the Format Specification subsection.) If you specify multiple `-o` options, the format specification will be interpreted as the space-separated concatenation of all *format* option arguments. The `-o` option takes precedence over the `-f`, `-l`, `-L`, and `-w` options.

`-p` *proclist* Restricts the listing to data about processes that have process ID numbers specified in *proclist*.

`-t` *termlist* Restricts the listing to data about the processes associated with the terminals specified in *termlist*. You may specify terminal identifiers in one of two forms: the device's file name (such as `tty02`) or, if the device's file name starts with `tty` or `ttyp`, just the identifier following the characters `tty` or `ttyp` (such as `02`, `004`, or `p004`).

`-u` *uidlist* Restricts the listing to data about processes that have user ID numbers or login names in *uidlist*. In the listing, the numerical user ID is printed, unless the `-f` option is used, in which case, the login name is printed.

`-U` *uidlist* Same as `-u` option.

`-w` Generates a wide listing.

The column headings and the meaning of the columns in a `ps` listing follows. The letters `f`, `w`, or `l` indicate the options (full, wide, or long, respectively) that cause the corresponding heading to appear; `all` means that the heading always appears. These options determine only the information provided for a process; they do not determine which processes will be listed.

`ACCTID` (`w`) Account ID of the process.

`ADDR` (`w,l`) Memory address of the process, if resident; otherwise, the disk address.

`C` (`w,l`) Processor utilization.

`CM` (`w`) CPU mask; used to limit a process to one or more specified CPUs.

`CMD` (`all`) Command name.

`CPUTIME` (`w`) Cumulative execution time for the process to a precision of hundredths of seconds.

`F` (`w,l`) Flags (octal) associated with the process. For more information on flags, see `proc.h` in the `/usr/include/sys` directory.

`HIMEM` (`w`) Maximum size (in blocks) that the process has grown.

`JID` (`w`) Job ID of the process.

NI (w,l)	Nice value; used in priority computation.
PID (all)	Process ID of the process; if you know this number and if you are the owner or root, you can kill the process.
PPID (f,w,l)	Process ID of the parent process.
PRI (w,l)	Priority of the process; higher numbers mean lower priority.
S (w,l)	The state of the process: I Intermediate R Running S Sleeping T Stopped W Waiting Z Terminated
SCTIME (w)	Cumulative system call time for the process.
STIME (w)	Starting time of the process. Not supported.
SWAPS (w)	Number of times the process has been swapped.
SZ (w,l)	Size in blocks of the core image of the process.
TIME (all except w)	Cumulative execution time for the process.
TTY (all)	Controlling terminal for the process.
UID (f,w,l)	User ID number of the process owner; the login name is printed under the -f option.
WAITSWAP (w)	Cumulative time that the process has been swapped.
WCHAN (w,l)	The event for which the process is waiting or sleeping; if blank, the process can be run.

When you specify the -m or -M option, you receive the following additional information:

ETIME (all)	Wall-clock execution time for the CRAY T3D process minutes and seconds in the following format: m:s
PES (m, M)	Number of processing elements allocated to the CRAY T3D process.
PRTN (all)	CRAY T3D partition ID of the process; this identifier is used for obtaining partition statistics from the mppstat(8) utility.
SHAPE (w, l)	CRAY T3D partition shape; (XxYxZ) hardware partition only.
STATE (all)	CRAY T3D partition state: Z ZOMBIE A ACTIVE

F FROZEN
E ERROR
U UNKNOWN

TYPE (all) CRAY T3D partition type; hardware or operating system.

When a process has exited and has a parent, but the parent has not waited for it, the process is marked <defunct>.

Format Specification

The -o option allows the output format to be specified under user control. The format specification consists of a list of field names presented as a single argument, separated by a blank or comma. Each field has a default header, which you can override by appending an equals sign and the new text of the header, such as -o **user=FOO**. The rest of the characters in the argument are used as the header text. The fields specified are arranged in columns and are written in the order specified on the command line. The width of a field will be at least as wide as the header text for that field. If the header text is null, such as -o "user=", the field will have no header text. If all header text fields are null, no header line is written.

The following field names are valid for the -o *format* argument:

Field name	Default header	Description
acctid	ACCTID	Account ID of the process.
address	ADDRESS	Memory address of the process if resident; otherwise, the disk address (octal).
addr	ADDRESS	Same as address.
args	COMMAND	Not supported. Required for POSIX and XPG4 compliance.
c	C	Processor utilization. Required for XPG4 compliance.
cmd	CMD	Command name.
comm	COMMAND	Same as cmd. Required for POSIX and XPG4 compliance.
command	COMMAND	Command name (longer version of command name).
cpumask	CPUMASK	CPU mask; used to limit a process to one or more specified CPUs (octal).
cm	CPUMASK	Same as cpumask.
cputime	CPUTIME	Cumulative execution time for the process.
etime	ELAPSED	Not supported. Required for POSIX and XPG4 compliance.
flags	FLAGS	Flags associated with the process (octal).
group	GROUP	Group name associated with the effective group ID of the process. Required for POSIX and XPG4 compliance.
himem	HIMEM	Maximum size in blocks that the process has grown to (decimal).
jid	JID	Job ID of the process.
nice	NI	Nice value; used in priority computation. Required for POSIX and XPG4 compliance.
ni	NI	Same as nice.

Field name	Default header	Description
pcpu	%CPU	Not supported. Required for POSIX and XPG4 compliance.
pid	PID	Process ID of the process. Required for POSIX and XPG4 compliance.
pgid	PGID	Process-group ID of the process. Required for POSIX and XPG4 compliance.
ppid	PPID	Process ID of the parent process. Required for POSIX and XPG4 compliance.
pri	PRI	Priority of the process (decimal).
rgroup	RGROUP	Group name associated with the real group ID of the process. Required for POSIX and XPG4 compliance.
ruser	RUSER	Login name associated with the real user ID of the process. Required for POSIX and XPG4 compliance.
size	SZ	Size (in blocks) of the core image of the process (decimal).
sz	SZ	Same as size.
state	S	The state of the process.
s	S	Same as state.
sctime	SCTIME	Cumulative system call time for the process.
stime	STIME	Starting time of the process. Not supported. Required for XPG4 compliance.
swaps	SWAPS	Number of times the process has been swapped.
time	TIME	Same as cputime. Required for POSIX and XPG4 compliance.
tty	TT	Controlling terminal for the process (if any). Required for POSIX and XPG4 compliance.
tt	TT	Same as tty.
uid	UID	User ID number of the process.
user	USER	Login name associated with the effective user ID of the process. Required for POSIX and XPG4 compliance.
vsz	VSZ	Size (in kilobytes) of the core image of the process (decimal). Required for POSIX and XPG4 compliance.
wchan	WCHAN	The event for which the process is waiting or sleeping.
waitswap	WAITSWAP	Cumulative time that the process has been swapped.

NOTES

Only an appropriately authorized user can see output for processes whose active security label is greater than that of the user.

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`showall`	Allowed to see output for all processes. Allowed to successfully use the `-n` option.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to see output for all processes and is allowed to successfully use the `-n` option.

EXIT STATUS

If `ps` finds processes to report, the exit status is 0; otherwise, it is 1.

BUGS

Some data printed for defunct processes are irrelevant.

When you use the `-f`, `-l`, or `-o` options, output may be wider than the allowed field, which causes the columns not to line up.

FILES

`/unicos`	System namelist
`/dev/mem`	Memory
`/dev/mpp/config`	CRAY T3D configured as part of the system
`/etc/passwd`	Supplies UID information
`/etc/group`	Supplies GID information
`/etc/ps_data`	Internal data structure
`/dev`	Searched to find terminal (tty) names
`/usr/include/sys/proc.h`	Supplies flag information

SEE ALSO

`kill`(1), `nice`(1), `privtext`(1), `renice`(1), `tty`(1)

`mppstat`(8) in the *CRAY T3D Administrator's Guide*, Cray Research publication SG–2507

NAME

`ptyrecon` – Manages pty reconnection

SYNOPSIS

```
ptyrecon -c [pty]
ptyrecon -d [pty]
ptyrecon -e [-t secs] [pty]
ptyrecon -h pty
ptyrecon -s
ptyrecon -S
```

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `ptyrecon` utility enables or disables pseudo tty (pty) disconnection, and reconnects, searches, or hangs up disconnected sessions.

If disconnection is enabled, a session does not disappear on a pty master side close operation (such as `telnet close`), but remains disconnected for a specified amount of time. Users can later search, reconnect to, or hang up disconnected sessions.

Because this utility restores terminal characteristics, it permits the use of either cooked (line) or raw (character) terminal modes, like `vi` to disconnect sessions.

The `ptyrecon` utility accepts the following options:

`-c` [*pty*] Connects to an already disconnected pty. If *pty* is omitted, the current controlling terminal is used.

`-d` [*pty*] Disables reconnection on a pty. If *pty* is omitted, the current controlling terminal is used.

`-e` [`-t` *secs*] [*pty*]
Enables reconnection on a pty. `-t` *secs* indicates the number of seconds the disconnected session is to stay alive. If this argument is omitted, the default `DISTIMEO` in the `sys/ptyrecon.h` file is used. If *pty* is omitted, the current controlling terminal is used.

`-h` *pty* Hangs up the specified pty that is already disconnected.

`-s` Searches and displays owner's disconnected ptys.

`-S` Searches and displays all disconnected ptys (authorized users only).

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to search and display all disconnected ptys.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to search and display all disconnected ptys.

To indicate a pty, use a number (such as 2, 02, or 002).

EXAMPLES

```
birch28% telnet cool
   ...
cool% ptyrecon -e
cool% bc
3 + 2
5
^]
telnet> close
Connection closed.
birch28% telnet cool
   ...
cool% ptyrecon -s
    PTY    UID    PID     TIME     TLIMIT
    000  10208   1629   0:00:51   0:10:00
cool% ptyrecon -c 000
5 + 8
13
quit
cool% logout
Connection closed by foreign host.
birch28%
```

SEE ALSO

`ptyrecon`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

pwd – Displays working directory name

SYNOPSIS

```
pwd
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The pwd utility writes the absolute path name of the current working directory to standard output.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
system, secadm	In a privileged administrator shell environment, allowed to write shell-redirected output to any file.
sysadm	Shell-redirected output is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user can write shell-redirected output to any file.

EXIT STATUS

The pwd utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

`Cannot open ..`
or
`Read error in ..` This message indicates possible file system trouble; you should try the full path name (`cd /`*`name/name/`*), followed by issuing a pwd command. It also may indicate that the `..` directory has been removed or the mode of the `..` directory does not allow reading.

SEE ALSO

cd(1)

NAME

quota – Reports quota information

SYNOPSIS

quota [-A] [-B] [-D] [-E] [-G] [-U] [-b] [-n] [-a *account_name*] [-f *file_quota*] [-g *group_name*] [-i *inode_quota*] [-p *fstab_path*] [-q *quota_file_name*] [-r *level*] [-s *file_system*] [-u *user_name*] [-v]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The quota utility reports on the state of the quota control system for the user or system administrator. Ordinary users are able to see information associated with their user ID (uid) and all authorized account IDs (acids) and group IDs (gids). An appropriately authorized user can see all acids, gids, and uids. A number of special exit codes can be used to determine the state of the quota system in scripts or batch jobs. Output is sent to the stdout file.

The quota utility accepts the following options:

-A Selects all acids. If -E is not present, all acids found in the /etc/udb entry for the user are selected; otherwise, all acids are selected. This option is not valid with -a.

-B Backup format. This option causes the report containing quota, warning, and usage information to be written in quadmin(8) directive format. If this option is used with -E, an ASCII file that represents the entire quota file is written. This option is not allowed with -b or -n.

-D Prints debug output to the standard error file.

-E Super user only (but see -q). Selects every existing id. If -A, -G, or -U is not present, the result is the same as if options -AGU had been presented. If any of the options -A, -G, or -U is present, only those classes of ids are selected.

-G Selects all gids. If -E is not present, all gids found in the /etc/udb entry for the user are selected; otherwise, all gids are selected. This option is not valid with -g.

-U Selects all uids. If -E is not present, only the user's current uid is selected; otherwise, all uids in /etc/udb are selected. This option is not allowed with -u.

-b Back-up format. This option causes the report containing quota and warning information to be written in quadmin(8) directive format. If this option is used with -E, an ASCII file that represents the entire quota file is written. This option is not allowed with -B or -n. Unlike the -B option, which also specifies back-up format, this option does not write out the usage information.

`-n` This option suppresses the report and causes just the name or names of the selected file systems (`-s` option) or quota files (`-q` option) to be written to the `stdout` file. This option is not allowed with `-b` or `-B`.

`-a` *account_name*
Selects the specified account names or `id`s. *account_name* is a list of one or more account names (or `acid`s) separated by commas or white space. Lists separated by white space must be enclosed in quotation marks so that the shell does not break them up into separate options. If the name is not an account name and it is numeric, it is treated as an account ID. The default is the `acid` of the current process, returned from `acctid`(2). This option is not valid with `-A`.

`-f` *file_quota*
Selects quota entries with at least *file_quota* blocks free in the quota. The default for *file_quota* is 0. This option is not allowed with `-r`.

`-g` *group_name*
Selects the specified groups. *group_name* is a list of one or more group names (or `gid`s) separated by commas or white space. Lists separated by white space must be enclosed in quotation marks so that the shell does not break them up into separate options. If the name is not a group name and it is numeric, it is treated as a `gid`. The default is the `gid` returned from `getgid` (see `getuid`(2)). This option is not valid if `-G` is selected.

`-i` *inode_quota*
Selects quota entries with at least *inode_quota* inodes free in the quota. The default for *inode_quota* is 0. This option is not allowed with `-r`.

`-p` *fstab_path*
Specifies an alternate path for `fstab`. Without this option, the default path is `/etc`, but this option allows the user to have a local version of these files for testing or experimentation. This option is intended for test purposes and is not recommended for normal use.

`-q` *quota_file_name*
Selects the named quota file. This option is intended to be used only if quota control is not active on a file system and can be used to bypass the normal `quotactl`(2) access method. Usually, only the super user will have permission to read the quota control file in this manner. If quota control is actively using this file, the content of the report may be inaccurate because the kernel could have information in its tables that have not been written to the file. This option is not valid with the `-s` option unless the `-b` or the `-B` option is also used. Because the quota file permission controls access in this mode, the super-user-only restriction on the other options is not applied.

`-r` *level* Selects quota entries specified by *level*. *level* selects quota entries below warning (`b`), at or above warning but below limit (`w`), or at limit (`l`) usage levels. These flags may be combined. The default is `blw`, which means all quota entries otherwise selected. This option is not allowed with `-f` or `-i`.

`-s` *file_system*
: Selects the named file system. The default is all file systems to which the other selection criteria apply. This option is not valid with the `-q` option unless the `-b` or the `-B` option is also used. This option suppresses the read of the kernel mount table to determine the list of mounted file systems.

`-u` *user_name*
: Selects the specified user names. *user_name* is a list of one or more user names (or `uid`s) separated by commas or white space. Lists separated by white space must be enclosed in quotation marks so the shell does not break them up into separate options. If the name is not a user name and it is numeric, it is treated as a user ID. The default is the `uid` returned from the `getuid`(2) system call. This option is not valid with `-U`. Only the super user may specify a `uid` different from that returned from `getuid`(2).

`-v`
: Reports explicit records, even if they have no current usage.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user who is assigned the following privilege text upon execution of this command is allowed to perform the action shown:

Privilege Text	Action
`showall`	Allowed to see all user, group, and account IDs.

If this utility is installed with a PAL, a user with one of the following active categories is allowed to perform the action shown:

Active Category	Action
`system`, `secadm`, `sysadm`	Allowed to see all user, group, and account IDs.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to see all user, group, and account IDs.

EXIT STATUS

A set of exit codes is available for use in scripts or batch jobs. The codes marked with `-f`, `-i`, or `-r` options appear only if those options are specified.

Code	Meaning
`0`	Normal completion
`1`	Usage warning; see the `stderr` file
`2`	Usage error
`3`	Internal error
`4`	Access denied
`5`	Quota system not active
`6`	Quota file magic mismatch
`7`	Quota file size is wrong

17 Warning level (-r)
18 Limit level (-r)
19 Both warnings and limit (-r)
20 Insufficient quota (-f, -i)

Report Format

The format of the normal report (when -n is not selected) is shown in this subsection, followed by an example. The portion in the outer bracket is repeated for each file system found to contain information for a selected id; the portion in the inner bracket is repeated for each id on that file system. The flags for quota warning and limit appear only if those conditions exist. The percentage shown with Quota is (Usage / Quota), and the percentage shown with Warning is (Usage / Warning). Percentages greater than 999.9 are shown as ***.*. If the -q option had been specified, quota file would replace file system in the example shown. When a * appears after a File block, Inode quota, or Warning value, the value shown is the default and has not been explicitly set.

The lines beginning with ** appear only if the particular warning, block, or unblock condition applies. The messages dealing with block and unblock times appear only on file systems with soft quotas and are intended to tell the user that, if the present usage levels are maintained, further file space will be denied or permitted at the time shown. These times are a prediction and should be considered approximate, especially if the file space in use changes appreciably.

```
⌈   File system: file_system_name
|
| ⌈ Account: account_name, Group: group_name, User: user_name, Id: 60000
| | ** Account warning time: Mon Feb 12 14:11 CST 1990
| | ** Account quota will [un]block on Mon Feb 12 14:11 CST 1990
| | ** Group warning time: Mon Feb 12 14:11 CST 1990
| | ** Group quota will [un]block on Mon Feb 12 14:11 CST 1990
| | ** User warning time: Mon Feb 12 14:11 CST 1990
| | ** User quota will [un]block on Mon Feb 12 14:11 CST 1990
| |                File blocks (512 bytes)   Inodes
| | Account Quota: 999999999999999* (100.0%) 9999999999* (100.0%) ** LIMIT
| |       Warning: 999999999999999* (100.0%) 9999999999* (100.0%) ** WARNING
| |         Usage: 999999999999999           9999999999
| |   Group Quota: 999999999999999* (100.0%) 9999999999* (100.0%) ** LIMIT
| |       Warning: 999999999999999* (100.0%) 9999999999* (100.0%) ** WARNING
| |         Usage: 999999999999999           9999999999
| |    User Quota: 999999999999999* (100.0%) 9999999999* (100.0%) ** LIMIT
| |       Warning: 999999999999999* (100.0%) 9999999999* (100.0%) ** WARNING
| |         Usage: 999999999999999           9999999999
```

The following is an example of a report showing the state of the quota control system for two users. No warning, block, or unblock conditions apply. Online quotas are in effect.

```
File system: /.fs/f04
 User: user1, Id: 001
                     File blocks (512 bytes)   Inodes
     User Quota:         4000000  (  4.0%)      50000  ( 69.0%)
        Warning:         3600000* (  4.4%)      45000* ( 76.6%)
          Usage:          158952                34477

 User: user2, Id: 002
                     File blocks (512 bytes)   Inodes
     User Quota:         4000000  (  6.9%)      50000  ( 69.0%)
        Warning:         3600000* (  7.6%)      45000* ( 76.6%)
          Usage:          274056                59316

File system: /.fs/f12
 User: user1, Id: 001
                     File blocks (512 bytes)   Inodes
     User Quota:         1300000* (  0.0%)      50000* (  0.0%)
        Warning:         1170000* (  0.0%)      45000* (  0.0%)
          Usage:               1                    1
```

The column labeled File blocks shows the quota block counts and block usage for online quotas. If "aggregate" quotas are in effect on the file system, the column will be labeled Aggregate blocks. In that case, the usage refers to the sum of the data blocks online and the data blocks migrated offline by the Data Migration Facility (DMF).

In the (default) online quotas case, files migrated offline by DMF do not count against the quota block counts. If the administrator has selected aggregate quotas, both online files and offline files contribute to the quota block count.

FILES

`/etc/acid`	Account file containing account ID and account name for each account
`/etc/group`	Group file containing group names and group IDs
`/etc/udb`	User validation file containing user control limits
`/etc/udb.public`	Public version of the `/etc/udb` file
`*/.Quota60`	Quota control file
`/etc/fstab`	File describing file system and swapping partitions used by UNICOS

SEE ALSO

`acctid`(2), `getuid`(2), `quotactl`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

`quadmin`(8), `qudu`(8) in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

quotamon – Monitors UNICOS quota state

SYNOPSIS

quotamon [-s *timeout*] [-p]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The quotamon utility runs as a background process to monitor the state of the quota control features of the UNICOS operating system and report to the user when quota warning and/or limit conditions occur. The messages are written to the stderr file.

The system administrator may elect to start this utility automatically for all users through the /etc/cshrc and /etc/profile files. If running this utility is the user's choice, do not alter /etc/cshrc and /etc/profile, but instead make known to the users that quotamon can be started by placing the utility in the user's home directory .login or .profile file.

The quotamon utility accepts the following options:

-s *timeout* The *timeout* is the amount of time in seconds that should elapse between successive messages about the same event. The default is 30 seconds, but any positive integer from 0 to the maximum allowed integer value is accepted.

-p When the -p option is specified, the process ID of the running quotamon process is printed on standard output.

NOTES

You do not need to start the quotamon process. This feature is automatically provided by the Korn, standard, and C login shells.

BUGS

If a standard error message is issued from a user process and quotamon happens to write a message at the same time because of a quota signal, the messages could be intermixed in the stderr file.

Be careful not to initiate more than one copy of quotamon per job or interactive session. Additional copies wastefully occupy process table and memory space and cause duplicate messages to appear in the stderr file.

EXAMPLES

To start the quota monitor so that messages are not repeated within 60 seconds, enter the following line in the selected file or files mentioned previously. `quotamon` places itself in the background.

```
quotamon -s 60
```

SEE ALSO

`quota`(1)

NAME

rcp – Copies remote files

SYNOPSIS

rcp [-b] [-c *copy_buffer_size*] [-p] [-r] [-s *socket_buffer_size*] [-S *tos*] [-T] *sourcelist destination*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rcp utility copies files between machines. The *sourcelist* argument can refer to remote files, local files, or directories; arguments can consist of either absolute or relative path names.

Remote files are specified in the format *rhost*:*file* where *rhost* is a remote host name or alias (described in hosts(5)). When the login name differs from your login name on a Cray Research system, file names on a remote host are specified as *user*@*rhost*:*file*. If you do not specify a full path name, the path is interpreted relative to your login directory on *rhost*. A path name on a remote host can be quoted (using \, ", or ‘) so that metacharacters are interpreted remotely.

The local file name *file* may not contain a colon (:).

The rcp utility accepts the following options:

-b Displays the buffer sizes for the copy buffer and socket buffer during a transfer.

-c *copy_buffer_size*
Sets the copy buffer size. This buffer is used for reading and writing between the data socket and the source or destination file. rcp automatically chooses a copy buffer size; however, you can alter the selection. The -c option requires an argument. An argument of off turns off copy buffer sizing and the buffer defaults to the size specified by COPYBUFSIZE #define in the tcp_config.h file. An argument of auto sets buffer sizing to automatic and has no effect relative to default operation. A numeric argument sets the buffer to that size. The letter K or k can follow a numeric buffer size to specify a multiple of 1024. The default setting is auto. A size of 0 is synonymous with auto.

-p Preserves in its copies the modification times and access modes of the source files, ignoring the user file creation mode mask (see umask(1)). By default, the mode and owner of the copy of the file are preserved if it already existed; otherwise, the mode of the source file modified by the umask on the destination host is used.

-r Copies each subtree that is rooted at that name when any of the source files are directories. The destination must be a directory.

`-s` *socket_buffer_size*
: Sets the socket buffer size. This kernel buffer is for data transfer in the data socket. `rcp` automatically chooses a socket buffer size; however, you can alter the selection. The `-s` option requires an argument. An argument of `off` turns off socket buffer sizing and the buffer defaults to the default kernel socket buffer size. An argument of `auto` sets buffer sizing to automatic and has no effect relative to default operation. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. The default setting is `auto`. A size of 0 is synonymous with `auto`.

`-S` *tos*
: Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name that is found in the `/etc/iptos` file.

`-T`
: Outputs timing information after the transfer completes.

sourcelist
: Specifies one or more remote files, local files, or directories.

destination
: Specifies the destination file or directory.

The `rcp` utility does not prompt for passwords; your current local user name must exist on *rhost* and allow remote command execution by using `remsh`(1B). `rcp` requires that the `.rhosts` file exists in the `$HOME` directory of the remote host.

The system configuration may require the `/etc/hosts.equiv` and `.rhosts` files each to contain a match for the remote host, and also require the remote user and local user names to match.

The `rcp` utility handles third-party copies in which neither source nor target files are on the current machine. Host names can also take the form *rname*@*rhost* to use *rname* rather than the current user name on the remote host. The destination host name can also take the form *rhost*`.`*rname* to support destination machines that are running the 4.2BSD version of `rcp`.

NOTES

Use `rcp` with discretion in secure mode. For more information, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

BUGS

When only a directory should be legal, the `rcp` utility does not detect all occurrences in which the target of a copy might be a file.

The `rcp` utility is confused by any output that is generated by commands in a `.profile`, `.login`, or `.cshrc` file on the remote host.

EXAMPLES

Example 1: To copy the `proposal` file, which is on a Cray Research system, into file `report` on the remote host `chemistry`, enter the following:

```
rcp proposal chemistry:report
```

Example 2: To copy the `whale` file, which is on the remote host `biology`, into file `mammal` on a Cray Research system, enter the following:

```
rcp biology:whale mammal
```

Example 3: The following example shows how to copy a file to or from an account that has a login other than your login on a Cray Research system.

In this example, the remote file `letter` must be accessed under the login name `tami`. To copy the file from the remote host `engineering` into a file named `memo` on a Cray Research system, enter the following:

```
rcp tami@engineering:letter memo
```

FILES

`$HOME/.rhosts`	File that contains a list of valid users
`/etc/hosts`	TCP/IP host name database

SEE ALSO

`ftp`(1B), `remsh`(1B), `rlogin`(1B), `umask`(1)

`hosts`(5), `rhosts`(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Software Overview for Users, Cray Research publication SG–2052

TCP/IP Network User's Guide, Cray Research publication SG–2009

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

rdist – Maintains identical copies of files over multiple hosts

SYNOPSIS

/usr/ucb/rdist [-n] [-q] [-R] [-h] [-i] [-v] [-w] [-y] [-b] [-D] [-p] [-r] [-s] [-S *tos*] [-f *distfile*] [-d *var=value*] [-m *host*] [*names*]

/usr/ucb/rdist [-n] [-q] [-R] [-h] [-i] [-v] [-w] [-y] [-b] [-D] [-p] [-r] [-s] [-S *tos*] -c *names host*[.*login*][:*dest*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The rdist utility maintains identical copies of files over multiple hosts. It preserves the owner, group, mode, and last modification time of files, when possible, and can update programs that are executing.

rdist reads commands from *distfile* to direct the updating of files and directories.

The rdist utility accepts the following options and arguments:

-n Prints the commands without executing them. This option is used for debugging *distfile*.

-q Specifies quiet mode. The names of files that are being updated usually are printed on standard output. The -q option suppresses this.

-R Removes extraneous files. If a directory is being updated, any files that exist on the remote host but not in the master (local) directory are removed. This option is useful for maintaining truly identical copies of directories.

-h Follows symbolic links. Copies the file to which the link points rather than copying the link itself.

-i Ignores unresolved links. rdist usually tries to maintain the link structure of files being transferred and warns the user if all links cannot be found.

-v Verifies that the files are up-to-date on all hosts. Any files that are out-of-date are displayed, but no files are changed and no mail is sent.

-w Specifies whole mode. The whole file name is appended to the destination directory name. Usually, only the last component of a name is used when renaming files. This preserves the directory structure of the files being copied, rather than flattening the directory structure (for example, renaming a list of files such as (dir1/f1 dir2/f2) to dir3 would create files dir3/dir1/f1 and dir3/dir2/f2 rather than dir3/f1 and dir3/f2).

`-y` Specifies younger mode. If their *mtime* and *size* (see `stat`(2)) disagree, files usually are updated. The `-y` option prevents `rdist` from updating files that are newer than the master copy. Use this mode to prevent newer copies on other hosts from being replaced. A warning message is printed for files that are newer than the master copy.

`-b` Specifies a binary comparison. Performs a binary comparison and updates files if they differ, rather than comparing dates and sizes.

`-D` Invokes debug mode. `rdist` generates debugging information. Use this mode only if you must debug `rdist`.

`-p` Invokes force directory permissions mode. This option forces an update of all remote directories to have the same permissions as the master directory.

`-r` Follows symbolic links on the remote host. If the master is not a symbolic link, a symbolic link usually is overwritten.

`-s` Specifies short name mode. Compares only the first 14 characters of the trailing component of the file name when removing files that use `-R`.

`-S` *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

`-f` *distfile* Specifies the name of the file that contains the commands that direct the updating of the files and directories. If *distfile* is a dash (`-`), the standard input is used. If you omit this option, the program looks in the current directory first for `distfile` and then for `Distfile` to use as the input. If neither `distfile` nor `Distfile` is found, it uses the `-c` option.

`-d` *var=value*
Defines or overrides a variable (*var*) in the *distfile* with *value*. *value* can be an empty string, one name, or a list of names enclosed in parentheses and separated by tabs or spaces.

`-m` *host* Specifies machines to be updated. You can specify multiple `-m` options to limit updates to a subset of the hosts listed in the *distfile*.

names Specifies names of files to be updated or labels of commands to execute. If label and file names conflict, a label is assumed. If you omit this argument, `rdist` updates all the files and directories that are listed in *distfile*.

`-c` *names host*[.*login*][:*dest*]
Forces `rdist` to interpret the remaining arguments as a small *distfile*. The equivalent *distfile* is as follows:

(*names* ...) -> *host*[`.`*login*] `install` [*dest*] ;

Creating the *distfile*

The *distfile* contains a sequence of entries to do the updating that specifies the files to be copied, the destination hosts, and identifies the operations to perform. Each entry has one of the following formats:

<variable name> = *<namelist>*
[*label*:] *<source list>* -> *<destination list>* *<command list>*
[*label*:] *<source list>* :: *<time_stamp file>* *<command list>*

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for distributing files that have changed since a given date.

The elements of the *distfile* are as follows:

<variable name> = <namelist>
Defines the files to be copied.

label Identifies a command for partial updates; labels are optional.

source list and *destination list*
The *source list* specifies a list of files and directories on the local host that will be used as the master copy for distribution. The *destination list* is the list of hosts to which these files will be copied. If the file is out-of-date on the host that is being updated (second format) or the file is newer than the time-stamp file (third format), each file in the source list is added to a list of changes. The source and destination lists have one of the following formats:

<name>
'(' *<zero or more names separated by white space>* ')'

command list The command list consists of zero or more of the following commands:

`install` *options opt_dest_name* ;
Copies out-of-date files and directories. Each source file is copied to each host in the destination list. Directories are recursively copied this way. *opt_dest_name* is an optional parameter to rename files. If an `install` command does not appear in the command list or you do not specify the destination name, the source file name is used. Directories in the path name are created if they do not exist on the remote host. To help prevent disasters, a nonempty directory on a target host is never replaced with a regular file or a symbolic link. However, under the `-R` option, a nonempty directory is removed if the corresponding file name is absent on the master host.

The *options* are `-R`, `-h`, `-i`, `-v`, `-w`, `-y`, `-b`, `-p`, `-r`, and `-s`; they have the same semantics as the options on the command line, except that they apply only to the files in the source list. The login name that is used on the destination host is the same as the local host, unless the destination name is of the format `host.login`.

`notify` *name list* ;
Mails the list of updated files (and any errors that occurred) to the listed names. If `@` does not appear in the name, the destination host is appended to the name (for example, `name1@host`, `name2@host`, ...).

`except` *name list* ;
Updates all of the files in the source list except the files listed in *name list*. This command usually is used to copy everything except certain files in a directory.

`except_pat` *pattern list* ;
Like the `except` command, except that *pattern list* lists regular expressions (see `ed`(1) for details). If one of the patterns matches a string within a file name, that file is ignored. Because \ is a quotation character, you must double it for it to become part of the regular expression. Variables are expanded in *pattern list*, but shell file pattern-matching characters are not expanded. To include a $, it must be escaped with a \.

`special` *name list string* ;
Specifies `sh`(1) commands that will be executed on the remote host after the file in *name list* is updated or installed. If you omit the *name list*, the shell commands are executed for each updated or installed file. The `FILE` shell variable is set to the current file name before executing the commands in *string*; *string* starts and ends with " and can cross multiple lines in *distfile*. You should separate multiple commands to the shell with a semicolon (;). Commands are executed in the user's home directory on the host being updated.

You cannot use the `special` command for rebuilding (for example, private databases) after a program is updated.

New-line characters, tabs, and blanks are used only as separators and are ignored otherwise. Comments begin with a # and end with a new-line character.

Variables to be expanded begin with a $ and are followed either by 1 character or a name enclosed in braces (see the EXAMPLES section).

The shell metacharacters are recognized and expanded (on the local host only) similar to `csh`(1). They can be escaped with a backslash. The ~ character also is expanded similar to `csh`, but it is expanded separately on the local and destination hosts. When you use the `-w` option with a file name that begins with ~, everything except the home directory is appended to the destination name. File names that do not begin with a `/` or ~ use the destination user's home directory as the root directory for the file name.

MESSAGES

A complaint of mismatching `rdist` version numbers can originate from starting your shell (for example, when you are in too many groups).

BUGS

Source files must reside on the local host in which `rdist` is executed.

You must execute a special command after all files in a directory are updated.

Variable expansion works only for name lists; a general macro facility is needed.

The `rdist` utility aborts files that have a negative *mtime* (before January 1, 1970).

A force option is needed to allow replacement of nonempty directories by regular files or symbolic links. A means of updating file modes and owners of otherwise identical files also is needed.

EXAMPLES

Following is an example of a *distfile*:

```
HOSTS = ( matisse arpa.root)

FILES = ( /bin /lib /usr/bin /usr/games
        /usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h}
        /usr/lib /usr/man/man? /usr/ucb /usr/local/rdist )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
        sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont )
${FILES} -> ${HOSTS}
        install -R ;
        except /usr/lib/${EXLIB} ;
        except /usr/games/lib ;
        special /usr/lib/sendmail "/usr/lib/sendmail -bz" ;

srcs:
/usr/src/bin -> arpa
        except_pat ( \\.o\$ /SCCS\$ ) ;

IMAGEN = (ips dviimp catdvi)

imagen:
/usr/local/${IMAGEN} -> arpa
        install /usr/local/lib ;
        notify ralph ;

${FILES} :: stamp.cory
        notify root@cory ;
```

FILES

`distfile`	Input command file.
`/tmp/rdist*`	Temporary file for update lists.
`$HOME/.rhosts`	User file that lists remote host names and logins names of users who are allowed access to the home directory. This file must be present on the remote host.

SEE ALSO

csh(1), ed(1), sh(1)

stat(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

`read` – Reads a line from standard input

SYNOPSIS

`read` [`-p`] [`-r`] [`-s`] [`-u` *n*] *var...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-p`, `-s`, and `-u` options)

DESCRIPTION

The `read` utility reads one line from standard input.

By default, unless the `-r` option is specified, backslash (\) acts as an escape character. If standard input is a terminal device and the invoking shell is interactive, `read` prompts for a continuation line when the following conditions are met:

- The shell reads an input line that ends with a backslash, unless the `-r` option is specified.
- A here document is not terminated after a `<newline>` is entered.

The `read` utility accepts the following options and operand:

`-p` Causes the input line to be taken from the input pipe of a process spawned by the shell by using `|&`.

`-r` Does not treat a backslash character in any special way (considers each backslash to be part of the input line).

`-s` Saves the input as a command in the shell history file.

`-u` *n* Reads from the one-digit file descriptor unit *n*.

var... Specifies the name of an existing or nonexisting shell variable.

The line is split into fields as in the shell; the first field is assigned to the first variable *var1*, the second field to the second variable *var2*, and so on. If fewer *var* operands are specified than there are fields, the leftover fields and their intervening separators are assigned to the last *var*. If fewer fields than *var*s exist, the remaining *var*s are set empty strings.

The setting of variables specified by the *var* operands affect the current shell execution environment.

NOTES

The `read` utility is a built-in utility to the standard shell (`sh`(1)). An executable version of this utility is available in `/usr/bin/read`.

EXIT STATUS

The `read` utility exits with one of the following values:

0 Successful completion.

>0 End-of-file was detected or an error occurred.

EXAMPLES

Example 1: The following command prints 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 < input_file
```

SEE ALSO

`sh`(1), `line`(1)

NAME

regcmp – Compiles regular expressions

SYNOPSIS

regcmp [-] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

Usually, the regcmp utility precludes the need for calling regcmp(3C) from C programs. This saves on both execution time and program size. The regcmp utility compiles the regular expressions in *file* and places the output in *file*.i.

The regcmp utility accepts the following options:

- Output will be placed in *file*.c.

files Files to compile.

The format of entries in *file* is a name (C variable), followed by one or more blanks, followed by a regular expression enclosed in double quotation marks.

The output of regcmp is C source code. Compiled regular expressions are represented as char[] arrays. Thus, the *file*.i files may be included in C programs, or *file*.c files may be compiled and later loaded. In a C program that uses the regcmp output, regex (abc, line) applies the regular expression abc to line.

EXAMPLES

Example 1: The following are examples of the contents that may be found in the input file for regcmp:

```
name        "([A-Za-z][A-Za-z0-9_]*)$0"

telno       "({0,1}([2-9][01][1-9])$0){0,1} *"
            "([2-9][0-9]{2})$1[ -]{0,1}"
            "([0-9]{4})$2"
```

Example 2: In a C program that uses the regcmp output, the following will apply the regular expression named telno to line.

```
regex(telno, line, area, exch, rest)
```

SEE ALSO

regcmp(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

remsh, rsh – Invokes a remote shell

SYNOPSIS

/usr/ucb/remsh *host* [-d] [-l *username*] [-n] [-S *tos*] [*command*]
/usr/ucb/rsh *host* [-d] [-l *username*] [-n] [-S *tos*] [*command*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The remsh and rsh utilities connect to *host* and execute *command*. The command copies its standard input to the remote command; copies the standard output of the remote command to its standard output; and copies the standard error of the remote command to its standard error. Interrupt, quit, and terminate signals are propagated to the remote command; usually the remote shell terminates when the remote command terminates. remsh requires that the .rhosts file exists in the $HOME directory of the remote host.

The remsh and rsh utilities accept the following options:

-d Sets the SO_DEBUG option on the sockets for the standard output and standard error.

-l *username* Specifies the account name (*username*) to use when loggin in to the remote machine. The default is the current account name. The remote name must be authorized for automatic authentication (as described in rlogin(1B)) to the originating account; no provision is made for specifying a password with a command.

-n Redirects the input of remsh or rsh to /dev/null if no input is desired.

-S *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the /etc/iptos file.

command Specifies the shell command to be executed. If you omit *command*, rather than executing a single command, you will be logged in on the remote host by using rlogin(1B).

Shell metacharacters that are not enclosed in quotation marks are interpreted on the local machine; metacharacters that are enclosed in quotation marks are interpreted on the remote machine. Thus, the following command line appends the remote file remotefile to the local file localfile:

```
remsh otherhost cat remotefile >> localfile
```

The following command line appends remotefile to remote file otherremotefile:

```
remsh otherhost cat remotefile ">>" otherremotefile
```

Host names are given in hosts(5). Each host has one standard name that is the first name specified in the file. This name is rather long and unambiguous; therefore, each host name may have one or more nicknames. The host names for local machines are also commands in the /usr/hosts directory; when you put this directory in your search path, you can omit rsh and remsh.

NOTES

The remsh utility is called rsh on UNIX 4.2 BSD systems.

The rsh(1) utility, the restricted shell, is supported in /bin. The /bin/rsh and /usr/ucb/rsh are two entirely different utilities.

If the remote system is using certain security features, the system configuration can require the /etc/hosts.equiv and .rhosts files each to contain a match for the originating host, and requires the remote user and local user names to match. If the system is configured this way, the -l option is not allowed.

FILES

$HOME/.rhosts	On remote machine
/etc/hosts	TCP/IP host name database

SEE ALSO

rlogin(1B)

setsockopt(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

hosts(5), rhosts(5) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

TCP/IP Network User's Guide, Cray Research publication SG–2009

Software Overview for Users, Cray Research publication SG–2052

UNICOS Networking Facilities Administrator's Guide, Cray Research publication SG–2304

NAME

renice – Sets system scheduling priorities of running processes

SYNOPSIS

```
renice [-n increment] [-g] ID ...
renice [-n increment] -j ID ...
renice [-n increment] -p ID ...
renice [-n increment] -u ID ...
```

Obsolescent version; may not be supported in future releases:

```
renice nice_value pid ... [-g gid ...] [-j jobid ...] [-p pid ...] [-u user ...]
renice nice_value [-g gid ...] [-j jobid ...] [-p pid ...] [-u user ...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-j option)

DESCRIPTION

The renice utility requests that the system scheduling priorities of one or more running processes be changed. When a process ID is reniced (the default), the applicable processes are specified by their IDs. When a process group is reniced, the request applies to all processes in the process group.

If the requested *increment* (or *nice_value*) raises or lowers the system scheduling priority of the executed utility beyond the system limits, then the limit whose value was exceeded is used. The *increment* (or *nice_value*) refers to the system's nice value. The nice value can range from 0 through 39; 0 represents the highest priority, and 39 represents the lowest priority.

The renice utility the following options and operands:

-n *increment* Specifies how the system scheduling priority of the specified process(es) is adjusted.

-g Interprets all arguments (or just the *gid* arguments in the obsolescent version) as process group IDs.

-j Interprets all arguments (or just the *jobid* arguments in the obsolescent version) as job IDs. The *increment* argument is a positive or negative integer that modifies the system scheduling priority of the specified process(es). If no -n *increment* is specified, the default *increment* is 4.

-p Interprets all arguments (or just the *pid* arguments in the obsolescent version) as process IDs. If no options are specified, the -p option is the default.

`-u` Interprets all arguments (or just the *user* arguments in the obsolescent version) as users. If a user exists with a user name equal to the operand, the user ID of that user is used; otherwise, the operand represents an unsigned integer and interprets it as the numeric user ID of the user.

ID Specifies a process ID, a job ID, a process group ID, or user name or user ID, depending on the option selected.

In the obsolescent version, the `-g`, `-j`, `-p`, and `-u` options can each take multiple arguments. A *pid* of 0 means the current process, a *jobid* of 0 means the current job, a *gid* of 0 means the current process group, and a *user* of 0 means the current user.

nice_value (Obsolescent) Saves the value specified as the actual system scheduling priority, rather than as an increment to the scheduling priority. Specifying a scheduling priority higher than that of the existing process may require super-user privileges.

NOTES

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to set the scheduling priority of any process. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to set the scheduling priority of any process subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to set the scheduling priority of any process. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The `renice` utility exits with one of the following values:

0 All processes had their priorities successfully changed.

>0 An error occurred.

SEE ALSO

`nice`(1), `ps`(1)

`nice`(2), `nicem`(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

NAME

resize – Sets terminal settings current window size

SYNOPSIS

resize [-c] [-u] [-s [*row col*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The resize utility writes to its standard output commands for setting the TERM and TERMCAP environment variables to indicate the current size of a window from which the command is run.

The resize utility accepts the following options:

-c	Causes the commands to be formed appropriately for csh(1) rather than sh(1).
-u	Causes the commands to be formed appropriately for sh (the standard shell) rather than csh.
-s [*row col*]	Uses Sun tty escape sequences. A new *row* and *column* size may be specified; the window will resize appropriately. If -s is not specified, VT102 escape sequences are used.

BUGS

The -u must appear to the left of -s, if both are specified.

The -c must appear to the left of -s, if both are specified.

There should be some global notion of display size; termcap and terminfo need to be analyzed in the context of window systems.

EXAMPLES

The following aliases, when executed as a command, reset the environment of the current shell:

```
alias xs       ´eval `resize`´
alias xrs       ´set noglob; eval `resize -s \!\*`´
```

The following example sets your TERM to sun, exports TERM, and then inserts the size of your sun screen into the environment:

```
TERM=sun;export TERM;eval ‘resize -s‘
```

FILES

`~/.shrc`	User's alias for the command

SEE ALSO

`csh`(1), `sh`(1), `tset`(1B)

NAME

`restart` – Recovers a process, multitask group, or job from a restart file

SYNOPSIS

`restart` [`-i`] [`-d`] [`-f`] [`-w`] *file*

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The `restart` utility recovers the process, multitask group, or job defined in the named restart file.

The `restart` utility accepts the following options:

`-i` Causes `restart` to print the ID returned by the `restart`(2) system call on standard output.

`-d` Gives the restarted processes the Distributed Computing Environment (DCE) credentials of the current process rather than using the ones stored in the restart file.

`-f` Forces recovery, even if one or more of the files referenced by the restart image has changed.

`-w` Causes `restart` to wait until the restarted process or job completes.

file Specifies the restart file. The *file* operand is required.

NOTES

The following restrictions apply to processes and jobs that are to be restarted:

- Only an appropriately authorized user may checkpoint or restart another user's job.
- Processes using online tapes cannot be checkpointed or restarted.
- If a user attempts to copy a restart file, it is no longer marked as a restart file and is not restartable.
- If a user attempts to move a restart file to a different file system, it is no longer marked as a restart file and is not restartable.
- If an interactive session is checkpointed and later recovered with a `restart` utility, all processes that are part of the session which performed the restart are terminated.

If this utility is installed with a privilege assignment list (PAL), a user with one of the following active categories is allowed to perform the actions shown:

Active Category	Action
`system`, `secadm`	Allowed to restart any file.
`sysadm`	Allowed to restart any file, subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to restart any file.

EXAMPLES

The following example illustrates how to use the restart utility to recover a process that has been checkpointed.

The user is running a.out in the background and enters a chkpnt(1) utility to checkpoint the process. The -p option specifies the PID and the -k option kills the process (pid.23634). The user then issues an ls(1) command to list information about pid.23634; the capital R at the beginning of the long listing indicates a restart file has been created.

```
unicos$ a.out &
23634

unicos$ chkpnt  -p  23634  -k

unicos$ ls  -l  pid.23634
Rr--------  1  (id) (group) (size) (date) pid.23634
```

Later, the user enters a restart utility to recover the process. The ps(1) listing confirms the process has been reactivated.

```
unicos$ restart  pid.23634

unicos$ ps
   PID     TTY  TIME COMMAND
  23634   p031  0:13 a.out
  23510   p031  0:00 sh
  23758   p031  0:00 ps
```

SEE ALSO

chkpnt(1), chkpnt_util(1), ps(1)

chkpnt(2), restart(2) in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301