NAME

tc – troff(1) output interpreter

SYNOPSIS

tc [-t] [-o *list*] [-a *n*] [-e] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tc utility interprets its input (standard input default) as output from troff(1). The standard output of tc is intended for a TEKTRONIX 4015 (a 4014 terminal with ASCII and APL character sets). The various typesetter sizes are mapped into the 4014's four sizes. The entire troff(1) character set is drawn using the 4014's character generator, using overstruck combinations where necessary.

The tc utility accepts the following options:

-t Does not wait between pages (for directing output into a file).

-o *list* Prints only the pages enumerated in *list*. The list consists of pages and page ranges (such as 5–17) separated by commas. The range *n*- goes from *n* to the end. The range -*n* goes from the beginning to and including page *n*.

-a *n* Sets the aspect ratio to *n*. The default is 1.5.

-e Does not erase before each page.

file Specifies the file to be interpreted.

A typical usage of tc follows:

```
troff file | tc
```

At the end of each page, tc waits for a new line (empty line) from the keyboard before continuing on to the next page. In this wait state, the following commands are recognized:

!*cmd* Sends *cmd* to the shell.

e Inverts the state of the screen erase.

-*n* Skips backward *n* pages.

a*n* Sets the aspect ratio to *n*.

? Prints a list of available options.

BUGS

When using `tc`, font distinctions are lost.

The `tc` command needs a `-w` option to wait for input to arrive.

SEE ALSO

`nroff`(1), `troff`(1)

NAME

`tee` – Duplicates output

SYNOPSIS

`tee` [`-a`] [`-i`] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tee` utility is a filter that copies standard input to standard output, making a copy in zero or more files. `tee` does not buffer its output.

The `tee` utility accepts the following options and operands:

`-a` Appends the output to each respective *file* rather than overwriting it.

`-i` Ignores the `SIGINT` signal.

files Files to be duplicated.

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 `tee` utility exits with one of the following values:

0 No standard input was successfully copied to all output files.

>0 An error occurred.

EXAMPLES

The `tee` utility often is used in shell scripts to redirect data to more than one place simultaneously. In this example, you can simultaneously view the output of the `news`(1) utility and save the information in a file.

```
news | tee mynews
```

SEE ALSO

`sh`(1)

NAME

telnet – User interface to the TELNET protocol

SYNOPSIS

/usr/ucb/telnet [-8] [-E] [-K] [-L] [-S *tos*] [-X *atype*] [-a] [-c] [-d] [-e *escapechar*] [-k *realm*] [-l *user*] [-n *tracefile*] [-r] [-x] *host* [*port*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The telnet utility communicates with another host by using TELNET protocol. If you invoke telnet without the *host* argument, it enters command mode, indicated by its prompt (telnet>). In this mode, it accepts and executes the commands listed following. If you invoke telnet with arguments, it performs an open command with those arguments.

The telnet utility accepts the following options:

-8 Specifies an 8-bit data path. This causes an attempt to negotiate the TELNET BINARY option on both input and output.

-E Stops any character from being recognized as an escape character.

-K Specifies no automatic login to the remote system.

-L Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

-S *tos* Sets the IP Type-of-Service (TOS) option for the telnet connection to the value *tos*, which can be a numeric TOS value or a symbolic TOS name found in the /etc/iptos file.

-X *atype* Disables the atype type of authentication.

-a Specifies automatic login. See the toggle autologin command on this man page for more information.

-c Disables the reading of the user's .telnetrc file. (See the toggle skiprc command on this man page.)

-d Sets the initial value of the debug toggle to TRUE.

-e *escapechar*
Sets the initial telnet esape character to *escapechar*. If you omit *escapechar*, there is no escape character.

-k *realm* If Kerberos authentication is being used, the -k option requests that telnet obtain tickets for the remote host in realm *realm* instead of the remote host's realm, as determined by krb_realmofhost(3K).

`-l` *user* When connecting to the remote system, if the remote system understands the TELNET `ENVIRON` option, *user* is sent to the remote system as the value for the TELNET environment variable `USER`. This option implies the `-a` option. You also can use this option with the `open` command.

`-n` *tracefile* Opens *tracefile* for recording trace information. (See the `set tracefile` command).

`-r` Specifies a user interface similar to `rlogin`(1B). In this mode, the escape character is set to the tilde (~) character, unless modified by the `-e` option.

`-x` Turns on encryption of the data stream if possible. This option is not available outside the United States and Canada.

host Indicates the official name, an alias, or the Internet address of a remote host.

port Indicates a port number (address of an application). If you do not specify a number, the default `telnet` port is used. Port names are mapped to port numbers through the `/etc/services` file. Usually, when you specify a port number, `telnet` does not send out any initial TELNET option negotiation. If the port number/name is preceded by a minus sign, the initial TELNET option negotiation is sent.

When in `rlogin` mode, a line of the form `~.` disconnects from the remote host; `~` is the `telnet` escape character. Similarly, the line `~^Z` suspends the `telnet` session. The line `~^]` escapes to the normal `telnet` escape prompt.

After a connection has been opened, `telnet` attempts to enable the TELNET `LINEMODE` option. If this fails, `telnet` reverts to one of two input modes: either character-at-a-time or line-by-line, depending on what the remote system supports.

When `LINEMODE` is enabled, character processing is done on the local system, under the control of the remote system. When input editing or character echoing will be disabled, the remote system relays that information. The remote system also relays changes to any special characters that occur on the remote system, so that they can become available on the local system.

Using the character-at-a-time mode, most text typed is immediately sent to the remote host for processing.

Using line-by-line mode, all text is echoed locally, and (usually) only completed lines are sent to the remote host. You can use the local echo character (initially ^E) to turn off and on the local echo (used to enter a password without the password being echoed).

If the `LINEMODE` option is enabled, or if the `localchars` toggle is `TRUE` (the default for line-by-line), the user's `quit`, `intr`, and `flush` characters are trapped locally, and they are sent as TELNET protocol sequences to the remote side. If `LINEMODE` was ever enabled, the user's `susp` and `eof` are also sent as TELNET protocol sequences, and `quit` is sent as a TELNET `ABORT` rather than `BREAK`. There are options (see `toggle autoflush` and `toggle autosynch` on this man page) that cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and to flush previous terminal input (see `quit` and `intr`).

While connected to a remote host, you can enter `telnet` command mode by typing the `telnet` escape character (initially ^]). In command mode, the usual terminal editing conventions are available.

The following `telnet` commands are available: type only enough of each command to identify it uniquely (also do this for arguments to the `mode`, `set`, `toggle`, `unset`, `slc`, and `display` commands).

`auth` *arguments...*
The `auth` command manipulates the information sent through the TELNET `AUTHENTICATE` option. Valid arguments for the `auth` command are as follows:

`disable` *type* Disables the specified type of authentication. To obtain a list of available types, use the `auth disable?` command.

`enable` *type* Enables the specified type of authentication. To obtain a list of available types, use the `auth enable?` command.

`status` Lists the current status of the various types of authentication.

`close` Closes a TELNET session and returns to command mode.

`display` [*argument...*]
Displays all, or some, of the `set` and `toggle` values.

`encrypt` *arguments...*
The `encrypt` command manipulates the information sent through the TELNET `ENCRYPT` option.

Note: Because of export controls, Cray Research does not support the TELNET `ENCRYPT` option outside the United States and Canada.

Valid arguments for the `encrypt` command are as follows:

`disable` *type*[`input|output`]
Disables the specified type of encryption. If you omit `input` and `output`, both input and output are disabled. To obtain a list of available types, use the `encrypt disable?` command.

`enable` *type*[`input|output`]
Enables the specified type of encryption. If you omit `input` and `output`, both input and output are enabled. To obtain a list of available types, use the `encrypt enable?` command.

`input` This is the same as the `encrypt start input` command.

`-input` This is the same as the `encrypt stop input` command.

`output` This is the same as the `encrypt start output` command.

`-output` This is the same as the `encrypt stop output` command.

`start`[`input`|`output`]
: Attempts to start encryption. If you omit `input` and `output`, both input and output are enabled. To obtain a list of available types, use the `encrypt enable?` command.

`status` Lists the current status of encryption.

`stop`[`input`|`output`]
: Stops encryption. If you omit `input` and `output`, encryption is on both input and output.

`type` *type* Sets the default *type* of encryption to be used with later `encrypt start` or `encrypt stop` commands.

`environ` [*argument...*]
: The `environ` command manipulates the variables sent through the TELNET `ENVIRON` option. The initial set of variables is taken from the user's environment, with only the `DISPLAY` and `PRINTER` environment variables being exported by default. If you use the `-a` or `-l` options, the `USER` environment variable is also exported. Valid arguments for the `environ` command are as follows:

`define` *variable value*
: Defines the variable *variable* to have the value *value*. Any variables that this command defines are exported automatically. Enclose *value* in single or double quotation marks to include tabs and spaces.

`export` *variable*
: Marks the variable *variable* to be exported to the remote side.

`list` Lists the current environment variables. Those marked with a `*` are sent automatically; other variables are sent only if explicitly requested.

`undefine` *variable*
: Removes *variable* from the list of environment variables.

`unexport` *variable*
: Marks the variable *variable* to not be exported unless explicitly asked for by the remote side.

`?` Prints out help information for the `environ` command.

`logout` Sends the TELNET `LOGOUT` option to the remote side. This command is similar to a `close` command; however, if the remote side does not support the `LOGOUT` option, nothing happens. If, however, the remote side does not support the `LOGOUT` option, this command should cause the remote side to close the TELNET connection. If the remote side also supports the concept of suspending a user's session for later reattachment, the `logout` argument indicates that you should terminate the session immediately.

`mode` *type* The *type* option is one of several options to select; consider the state of the TELNET session when selecting. The remote host is asked for permission to go into the requested mode. If the remote host can enter that mode, the requested mode is entered.

`character` Disables the TELNET `LINEMODE` option, or, if the remote side does not understand the `LINEMODE` option, it enters the character-at-a-time mode.

`line` Enables the TELNET `LINEMODE` option, or, if the remote side does not understand the `LINEMODE` option, it attempts to enter line-by-line mode.

`isig` (`-isig`)
Attempts to enable (disable) the `TRAPSIG`1 mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`edit` (`-edit`)
Attempts to enable (disable) the `EDIT` mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`softtabs(-softtabs)`
Attempts to enable (disable) the `SOFT_TAB` mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`litecho(-litecho)`
Attempts to enable (disable) the `LIT_ECHO` mode of the `LINEMODE` option. This requires that you enable the `LINEMODE` option.

`?` Prints out help information for the `mode` command.

`open` *host* [[`-q`]*port*]
Opens a connection to the specified host. If you do not specify a port number, `telnet` attempts to contact a TELNET server at the default port. The host specification can be either a host name (see `hosts`(5)) or an Internet address that is specified in the dot notation (see `inet`(3C)). You can use the `-l` option to specify the user name to be passed to the remote system through the `ENVIRON` option. The `-q` option can perform automatic login (see the `toggle autologin` command on this man page). When connecting to a nonstandard port, `telnet` omits any automatic initiation of TELNET options. When the port number is preceded by a minus sign, the initial option is negotiated. After establishing a connection, if the `skiprc` variable is not enabled (see the `toggle skiprc` command on this man page), the `.telnetrc` file in the user's home directory is opened. Lines that begin with a # symbol are comment lines. Blank lines are ignored. Lines that begin without white space are the start of a machine entry. The first item on the line is the name of the machine to which a connection is being made. The remainder of the line, and successive lines that begin with white space are assumed to be `telnet` commands and are processed as if they are typed manually in the `telnet` command prompt.

The special machine name `DEFAULT` specifies commands that should be executed for all machines. There can be more than one entry for a machine; all matches with the specified machine name will be executed.

`quit` Closes any open TELNET session and exits `telnet`. An end-of-file command (in command mode) also closes a session and exits.

`send` *arguments*

Sends one or more special character sequences to the remote host. You can specify the following arguments (you can specify more than one argument at a time):

`abort`	Sends the TELNET ABORT (ABORT processes) sequence.
`ao`	Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output from the remote system to the user's terminal.
`ayt`	Sends the TELNET AYT (Are You There) sequence, to which the remote system might or might not choose to respond.
`brk`	Sends the TELNET BRK (Break) sequence, which can have significance to the remote system.
`ec`	Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.
`el`	Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line being entered currently.
`eof`	Sends the TELNET EOF (end-of-file) sequence.
`eor`	Sends the TELNET EOR (end-of-record) sequence.
`escape`	Sends the current `telnet` escape character (initially ^]).
`ga`	Sends the TELNET GA (Go Ahead) sequence, which probably has no significance to the remote system.
`getstatus`	Sends the TELNET STATUS SEND suboption if the remote side understands the TELNET STATUS option. The response is not seen unless the `options` variable was set.
`ip`	Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.
`nop`	Sends the TELNET NOP (No OPeration) sequence.
`susp`	Sends the TELNET SUSP (SUSPend process) sequence.
`synch`	Sends the TELNET SYNCH sequence, which causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work if the remote system is a 4.2BSD system; if it does not work, a lowercase `r` might be echoed on the terminal).
`?`	Prints out help information for the `send` command.

`set` *argument value*

`unset` *arguments...*

The `set` command sets any one of several `telnet` variables to a specific value or to `TRUE`. The special value `off` turns off the function associated with the variable; this is equivalent to using the `unset` command, which disables or sets to `FALSE` any of the specified functions. Use the `display` command to interrogate variables. The variables that can be set or unset, but not toggled, are listed following. Also, any of the variables for the `toggle` command can be explicitly set or unset by using the `set` and `unset` commands.

`ayt` If TELNET is in `localchars` mode, or `LINEMODE` is enabled, and the `status` character is typed, a TELNET AYT sequence (see `send ayt` preceding) is sent to the remote host. The initial value for the `Are You There` character is the terminal's `status` character.

`echo` If in line-by-line mode, this is the value (initially `^E`) that toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for example, for entering a password).

`eof` If TELNET is operating in `LINEMODE` or line-by-line mode, entering this character as the first character on a line sends this character to the remote system. The initial value of the EOF character is the terminal's `eof` character.

`erase` If TELNET is in *localchars* mode (see `toggle localchars` on this man page), and if `telnet` is operating in character-at-a-time mode, when this character is typed, a TELNET EC sequence (see `send ec` listed previously) is sent to the remote system. The initial value for the erase character is the terminal's `erase` character.

`escape` TELNET escape character (initially ^[), which causes entry into `telnet` command mode (when connected to a remote system).

`flushoutput`
If TELNET is in *localchars* mode (see `toggle localchars` on this man page) and the `flushoutput` character is typed, a TELNET AO sequence (see `send ao` listed previously) is sent to the remote host. The initial value for the flush character is the terminal's `flush` character.

`forw1`
`forw2` If TELNET is operating in `LINEMODE`, these are the characters that, when typed, cause partial lines to be forwarded to the remote system. The initial value for the forwarding characters are taken from the terminal's `eol` and `eol2` characters.

`interrupt` If TELNET is in *localchars* mode (see `toggle localchars` on this man page) and the `interrupt` character is typed, a TELNET IP sequence (see `send ip` listed previously) is sent to the remote host. The initial value for the interrupt character is the terminal's `intr` character.

`kill` If TELNET is in *localchars* mode (see `toggle localchars` on this man page), and if `telnet` is operating in character-at-a-time mode, when this character is typed, a TELNET EL sequence (see `send el` listed previously) is sent to the remote system. The initial value for the kill character is the terminal's `kill` character.

`lnext` If TELNET is operating in `LINEMODE` or line-by-line mode, this character is the terminal's `lnext` character. The initial value for the `lnext` character is the terminal's `lnext` character.

`quit` If TELNET is in *localchars* mode (see `toggle localchars` on this man page) and the `quit` character is typed, a TELNET BRK sequence (see `send brk` listed previously) is sent to the remote host. The initial value for the quit character is the terminal's `quit` character.

`reprint` If TELNET is operating in `LINEMODE` or line-by-line mode, this character is the terminal's `reprint` character. The initial value for the reprint character is the terminal's `reprint` character.

`rlogin` This is the `rlogin` escape character. If set, the normal TELNET escape character is ignored unless it is preceded by this character at the beginning of a line. This character, at the beginning of a line followed by a `.` character closes the connection; when followed by a `^Z` character, it suspends the `telnet` utility. The initial state is to disable the `rlogin` escape character.

`start` If the TELNET `TOGGLE-FLOW-CONTROL` option is enabled, this character is the terminal's `start` character. The initial value for the start character is the terminal's `start` character.

`stop` If the TELNET `TOGGLE-FLOW-CONTROL` option is enabled, this character is the terminal's `stop` character. The initial value for the stop character is the terminal's `stop` character.

`susp` If TELNET is in `localchars` mode, or `LINEMODE` is enabled, and the `suspend` character is typed, a TELNET SUSP sequence (see `send susp` previously) is sent to the remote host. The initial value for the suspend character is the terminal's `suspend` character.

`tracefile` If it is set to a `-` symbol, tracing information is written to standard output (the default). This is the file to which the output, caused by `netdata` or `option` tracing being TRUE, is written.

`worderase` If TELNET is operating in `LINEMODE` or line-by-line mode, this character is the terminal's `worderase` character. The initial value for the word erase character is the terminal's `worderase` character.

`?` This displays the legal `set` (`unset`) commands.

`slc` *state* The `slc` command (Set Local Characters) sets or changes the state of the special characters when the TELNET `LINEMODE` option was enabled. Special characters are characters that are mapped to TELNET command sequences (such as `ip` or `quit`) or line editing characters (such as `erase` and `kill`). By default, the local special characters are exported. Values for *state* are as follows:

`check` Verifies the current settings for the current special characters. The remote side is requested to send all of the current special character settings, and if any discrepancies exist with the local side, the local side switches to the remote value.

`export` Switches to the local defaults for the special characters. The local default characters are those on the local terminal at the time when TELNET was started.

`import` Switches to the remote defaults for the special characters. The remote default characters are those on the remote system at the time when the TELNET connection was established.

`?` Prints help information for the `slc` command.

`status` Shows the current status of TELNET. This includes the peer to which you are connected, as well as the current mode.

`toggle` *arguments...*
Toggles (between `TRUE` and `FALSE`) various flags that control how TELNET responds to events. These flags can be set explicitly to `TRUE` or `FALSE` by using the `set` and `unset` commands listed previously. You can specify more than one argument. You can interrogate the state of these flags by using the `display` command. Valid arguments are as follows:

`authdebug` Turns on debugging information for the authentication code.

`autoflush` If `autoflush` and `localchars` are both `TRUE`, when the `ao`, `intr`, or `quit` characters are recognized (and transformed into TELNET sequences; see the preceding `set` command for details), TELNET refuses to display any data on the user's terminal until the remote system acknowledges (through a TELNET `TIMING MARK` option) that it has processed those TELNET sequences. If the terminal user has not done an `stty noflsh`, the initial value for this toggle is `TRUE`; otherwise they are `FALSE` (see `stty`(1)).

`autoencrypt`
`autodecrypt`
When the TELNET `ENCRYPT` option is negotiated, by default the actual encryption (decryption) of the data stream does not start automatically. The `autoencrypt` (`autodecrypt`) command states that encryption of the output (input) stream should be enabled as soon as possible.
Note: Because of export controls, Cray Research does not support the TELNET `ENCRYPT` option outside the United States and Canada.

`autologin` If the remote side supports the TELNET `AUTHENTICATION` option, TELNET attempts to use it to perform automatic authentication. If the `AUTHENTICATION` option is not supported, the user's login name is propagated through the TELNET `ENVIRON` option. This command is the same as specifying the `-a` option on the `open` command.

`autosynch` If `autosynch` and `localchars` are both TRUE, when either the `intr` or `quit` character is typed (see `set` listed previously for descriptions of the `intr` and `quit` characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure should cause the remote system to begin deleting all previously typed input until both of the TELNET sequences are read and acted upon. The initial value of this toggle is `FALSE`.

`binary` Enables or disables the TELNET `BINARY` option for both input and output.

`inbinary` Enables or disables the TELNET `BINARY` option on input.

`outbinary` Enables or disables the TELNET `BINARY` option on output.

`crlf` If `crlf` is `TRUE`, carriage returns are sent as `<CR><LF>`. If `crlf` is `FALSE`, carriage returns are sent as `<CR><NUL>`. The initial value for this toggle is `FALSE`.

`crmod` Toggles carriage return mode. When you enable this mode, most carriage return characters received from the remote host are mapped into a carriage return followed by a line feed. This mode does not affect those characters that the user types; it affects only those received from the remote host. This mode is not very useful, unless the remote host sends only a carriage return, but never a line feed. The initial value for this toggle is `FALSE`.

`debug` Toggles socket-level debugging (useful only to the super user). The initial value for this toggle is `FALSE`.

`encdebug` Turns on debugging information for the encryption code.

`localchars`
If this value is TRUE, the `flush`, `interrupt`, `quit`, `erase`, and `kill` characters (see `set` listed previously) are recognized locally, and they are transformed into appropriate TELNET control sequences (respectively `ao`, `ip`, `brk`, `ec`, and `el`; see `send` listed previously). The initial value for this toggle is `TRUE` in line-by-line mode, and `FALSE` in character-at-a-time mode. When you enable the `LINEMODE` option, the value of `localchars` is ignored, and it is assumed to be always `TRUE`. If `LINEMODE` was ever enabled, `quit` is sent as `abort`, and `eof` and `suspend` are sent as `eof` and `susp` (see `send` listed previously).

`netdata` Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is `FALSE`.

`options` Toggles the display of some of the internal TELNET protocol processing (related to TELNET options). The initial value for this toggle is `FALSE`.

`prettydump`
When you enable the `netdata` toggle, and `prettydump` is enabled, the output from the `netdata` command is formatted in a more user-readable format. Spaces are inserted between each character in the output, and the beginning of any TELNET escape sequence is preceded by a `*` symbol to aid in locating them.

`skiprc` When the `skiprc` toggle is `TRUE`, TELNET skips the reading of the `.telnetrc` file in the user's home directory when connections are opened. The initial value for this toggle is `FALSE`.

`termdata` Toggles the display of all terminal data (in hexadecimal format). The initial value for this toggle is `FALSE`.

`verbose_encrypt`
When the `verbose_encrypt` toggle is `TRUE`, TELNET prints out a message each time encryption is enabled or disabled. The initial value for this toggle is `FALSE`.
Note: Because of export controls, Cray Research does not support data encryption outside the United States and Canada.

`?` Displays the legal `toggle` commands.

`z` Suspends `telnet`. This command works only when the user is using the `csh`(1) command.

`![`*command*`]`
Executes a single command in a subshell on the local system. If you omit *command*, an interactive subshell is invoked.

`?[`*command*`]`
Gets help. With no arguments, TELNET prints a help summary. If you specify a command, TELNET prints the help information for only that command.

NOTES

On some remote systems, you must turn off echo manually when in line-by-line mode.

In line-by-line mode or `LINEMODE`, the terminal's `eof` character is recognized (and sent to the remote system) only when it is the first character on a line.

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
`admin`	Allowed to use `telnet` over a restricted interface.

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 use `telnet` over a restricted interface.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to use `telnet` over a restricted interface.

ENVIRONMENT VARIABLES

TELNET uses at least the `HOME`, `SHELL`, `DISPLAY`, and `TERM` environment variables. To propagate other environment variables to the other side, use the TELNET `ENVIRON` option.

FILES

`~/.telnetrc`	User's `telnet` directory

SEE ALSO

`csh`(1), `rlogin`(1B), `rsh`(1), `stty`(1)

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

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

`kerberos`(7) (available only online)

Software Overview for Users, Cray Research publication SG–2052

NAME

`test` – Performs a conditional evaluation

SYNOPSIS

`test` *expr*
`[` *expr* `]`

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-a`, `-G`, `-h`, `-k`, `-L`, `-O`, and `-S` primaries)
CRI extensions (`-m` and `-M` primaries)

DESCRIPTION

The `test` utility evaluates expression *expr* and, if its value is true, returns an exit status of 0 (true); otherwise, a nonzero (false) exit status is returned. Without arguments, `test` also returns a nonzero exit status.

You must specify all operators and elements of primaries as separate arguments.

The following primaries are used to construct *expr*:

`-b` *file* True if *file* exists and is a block special file.

`-c` *file* True if *file* exists and is a character special file.

`-d` *file* True if *file* exists and is a directory.

`-e` *file*
`-a` *file* True if *file* exists.

`-f` *file* True if *file* exists and is a regular file.

`-g` *file* True if *file* exists and its set-group-ID flag is set

`-G` *file* True if *file* exists and its group matches the effective group of this process.

`-h` *file* True if *file* exists and is a symbolic link.

`-k` *file* True if *file* exists and its sticky bit is set. This is always false; there is no sticky bit in UNICOS.

`-m` *file* True if *file* exists and is a migrated file (type IFOFL).

`-M` *file* True if *file* exists and is a migrated file (has a DMF handle).

`-n` *string* True if the length of the string *string* is nonzero.

`-O` *file* True if *file* exists and is owned by the effective user ID of this process.

`-p` *file* True if *file* exists and is a fifo (named pipe) special file.

`-s` *file* True if *file* exists and has a size greater than 0.

`-r` *file* True if *file* exists and can be read.

`-t` *fildes* True if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.

`-u` *file* True if *file* exists and its set-user-ID flag is set.

`-w` *file* True if *file* exists and can be written. True indicates that only the write flag is on.

`-x` *file* True if *file* exists and can be executed. True indicates that only the execute flag is on. If *file* is a directory, true indicates that *file* can be searched.

`-z` *string* True if the length of string *string* is 0.

string True if *string* is not the null string.

s1 `=` *s2* True if strings *s1* and *s2* are identical.

s1 `!=` *s2* True if strings *s1* and *s2* are *not* identical.

n1 `-eq` *n2* True if the integers *n1* and *n2* are algebraically equal.

n1 `-ne` *n2* True if the integers *n1* and *n2* are algebraically not equal.

n1 `-gt` *n2* True if the integer *n1* is algebraically greater than the integer *n2*.

n1 `-ge` *n2* True if the integer *n1* is algebraically greater than or equal to the integer *n2*.

n1 `-lt` *n2* True if the integer *n1* is algebraically less than the integer *n2*.

n1 `-le` *n2* True if the integer *n1* is algebraically less than or equal to the integer *n2*.

You may combine these primaries with the following operators:

`!` Unary negation operator.

`-a` Binary AND operator.

`-o` Binary OR operator (`-a` has higher precedence than `-o`).

`(` *expr* `)` Parentheses for grouping.

NOTES

The `test` utility is a built-in utility to the standard shell (`sh`(1)). An executable version of `test` is available in `/usr/bin/test`.

Because parentheses are meaningful to the shell, they must be escaped.

CAUTIONS

In the second form of the utility (that is, the one that uses `[]` rather than the word `test`(1)), you must delimit the brackets by blanks.

EXIT STATUS

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

0 *expression* evaluated to true.

1 *expression* evaluated to false or *expression* was missing.

>1 An error occurred.

SEE ALSO

`find`(1), `sh`(1)

NAME

`tftp` – Invokes the trivial file transfer program

SYNOPSIS

`/usr/ucb/tftp` [`-S` *tos*] [*host*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `tftp` utility is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. (See the `connect` command that follows.)

The `tftp` utility accepts the following options:

`-S` *tos* Sets the IP Type-of-Service (TOS) option for the connection to the value *tos*, which may be a numeric TOS value or a symbolic TOS name found in the `/etc/iptos` file.

host Specifies the network name for the host computer.

After `tftp` is running, it issues the `tftp>` prompt and recognizes the following commands:

`connect` *host-name* [*port*]
Sets up *host-name* (and optionally *port*) for transfers. The TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the `connect` command does not actually create a connection, but merely remembers the host that is to be used for transfers. You do not have to use the `connect` command; you can specify the remote host as part of the `get` or `put` command.

`mode` *transfer-mode*
Sets up the mode for transfers; *transfer-mode* can be `netascii` or `octet`. The default is `netascii`. The TFTP protocol defines `netascii` and `octet`. You can use `ascii` as an alias for `netascii`; you can use `binary` and `image` as aliases for `octet`. If *transfer-mode* is not specified, the current mode is printed.

`put` *file*
`put` *localfile remotefile*
`put` *file1 file2 ... fileN remote-directory*
Writes a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the specified host name becomes the default for future transfers. If the *remote-directory* form is used, you must specify at least three source files.

`get` *filename*
`get` *remotename localname*

`get` *file1 file2 . . . fileN*
: Gets a file or set of files from the specified remote file(s), which can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the last host name specified becomes the default for future transfers.

`aput` *file*
`aput` *localfile remotefile*
`aput` *file1 file2 . . . fileN remote-directory*
: Uses the Kerberos-authenticated send file to write a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the specified host name becomes the default for future transfers. If the *remote-directory* form is used, you must specify at least three source files.

`aget` *filename*
`aget` *remotename localname*
`aget` *file1 file2 . . . fileN*
: Uses a Kerberos-authenticated receive file to get a file or set of files from the specified remote file(s), which can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the last host name specified becomes the default for future transfers.

`quit` Exits `tftp`. An EOF command also exits.

`verbose`
: Toggles verbose mode.

`trace` Toggles packet tracing.

`status` Shows current status.

`rexmt` *retransmission-timeout*
: Sets the per-packet retransmission time-out (in seconds).

`timeout` *total-transmission-timeout*
: Sets the total transmission time-out (in seconds).

`ascii` Specifies shorthand for `mode ascii`.

`binary` Specifies shorthand for `mode binary`.

`?` [*command-name . . .*]
: Prints help information for the indicated command.

SEE ALSO

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

NAME

`ticksum` – Provides a time independent checksum of a file

SYNOPSIS

`ticksum` *file*
`ticksum` *file1 file2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

CRI extension

DESCRIPTION

The *time independent* checksum utility, if invoked with one argument, calculates a checksum of the named file and writes it (in hexadecimal) to standard output. For the specific kinds of binary files it recognizes, information such as the time of compilation and the version of the generating product is ignored in calculating the checksum.

If invoked with two arguments, the two files are compared by comparing their checksums calculated as above. A zero exit status indicates that the checksums were the same; a non-zero exit status that the checksums differed.

EXIT STATUS

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

0 Checksum(s) computed; if two files given, the checksums were identical.

>0 Files differ, by the checksum criterion.

SEE ALSO

`cksum`(1)

NAME

`time` – Times a simple command

SYNOPSIS

`time` [`-p`] *utility* [*argument* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (output format)

DESCRIPTION

The *utility* with any *arguments* is executed; after it is complete, `time` writes the elapsed time during the utility, the time spent in the system, and the time spent in execution of the utility to standard error. By default, times are reported in seconds and in the corresponding number of CPU clock cycles.

`time` accepts the following options and arguments:

`-p`	Times are reported only in seconds in a standard and portable format.
utility	Name of the utility to be timed.
argument ...	Arguments specific to the utility being timed.

NOTES

The `csh`(1) utility has a built-in `time` utility with slightly different characteristics. See `csh`(1).

EXIT STATUS

The `time` utility exits with a 0 status if the timed *utility* terminates because of a signal. If the utility specified by *utility* could not be found, the exit status is 127. If the utility was found, but cannot be invoked, the exit status is 126. If some other error occurs within the `time` utility, the exit status is 2. Otherwise, the exit status shall be that of the timed *utility*.

SEE ALSO

`csh`(1), `ja`(1), `sar`(1), `timex`(1)

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

NAME

timex – Times a command and reports process data and system activity

SYNOPSIS

timex [-f] [-h] [-k] [-m] [-o] [-p] [-r] [-s] [-t] *command* [*args*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

When you invoke the timex utility, the given *command* is executed; the elapsed time, user time, and system time spent in execution are reported in seconds. Optionally, process accounting data for the *command* and all of its children can be listed or summarized, and total system activity during the execution interval can be reported.

The output of timex is written on standard error.

The timex utility accepts the following options:

-f Prints the fork/exec flag and system exit status columns in the output. This option applies when -p is specified; otherwise, it is ignored.

-h Displays the fraction of total available CPU time consumed by the process during its execution. This "hog factor" is computed as follows:

(*total CPU time*)/(*elapsed time*)

This option applies when -p is specified; otherwise, it is ignored.

-k Displays total kcore-minutes; this is an integral of memory usage over time. One kcore minute is 1024 words used for 1 minute. This option applies when -p is specified; otherwise, it is ignored.

-m Shows mean core size. This option applies when -p is specified; otherwise, it is ignored.

-o Reports the total number of blocks read or written and total characters transferred by *command* and all its children.

-p Lists process accounting records for *command* and all its children. The number of blocks read or written and the number of characters transferred are always reported.

-r Shows CPU factor (*user time*)/(*system-time* + *user-time*). This option applies when -p is specified; otherwise, it is ignored.

-s Reports total system activity (not just that due to *command*) that occurred during the execution interval of *command*. All data items listed in sar(1) are reported.

-t Shows separate system and user CPU times. This option applies when -p is specified; otherwise, it is ignored.

args Arguments for the command being timed.

WARNINGS

Process records associated with *command* are selected from the `/usr/adm/acct/day/pacct` accounting file by inference, because process genealogy is not available. Background processes that have the same user ID, terminal ID, and execution are spuriously included.

EXAMPLES

Example 1: A simple example follows:

```
timex -ops sleep 60
```

Example 2: A terminal session of arbitrary complexity can be measured by timing a subshell, as follows:

```
$ timex -opskmt sh
$ <sub-shell session commands>
   .
   .
   .
$ exit
.$
```

SEE ALSO

`acctcom`(1), `ja`(1), `sar`(1), `time`(1)

NAME

tmpdir – Creates a unique temporary directory

SYNOPSIS

tmpdir *path*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tmpdir utility creates a unique directory in *path*. It is useful when creating secondary job temporary directories on alternate devices. Before creating a directory, tmpdir checks the /etc/tmpdir.users file for authorization. Before exiting normally, tmpdir prints the name of the temporary directory to stdout.

The tmpdir utility accepts the following argument:

path Location of the temporary directory to be created.

Either init(8) or the Network Queuing System (NQS) will delete the files in temporary directories at job end.

NOTES

If tmpdir is executed in a secure system, the temporary directories can be found in the files ${TMPDIR}/.tmpdir[a-z]. The suffix a is added when the first tmpdir is executed. Subsequent executions of tmpdir may not be able to access .tmpdira for write if the user has raised either his security level or compartment settings. Thus, tmpdir will try to open .tmpdirb, .tmpdirc, and so on, looking for a file that it can create. The user can raise the security level and compartment setting a total of 26 times with intervening tmpdir executions. All tmpdir executions at the same security level and compartments will write to the same file.

MESSAGES

tmpdir: TMPDIR environment variable not set
: Either init or NQS did not set TMPDIR, or you unset it before executing tmpdir.

tmpdir: *user* is not in users file
: You are not authorized to use tmpdir.

tmpdir: *path* is not an authorized path
: You are not authorized to create directories in *path*.

tmpdir: too many tmpdir files
: You have changed security levels or compartments and executed tmpdir more than 26 times.

`tmpdir: file not owned by root:` *file*
The `tmpdir` file *file* is not owned by `root`, so it is not a valid `tmpdir` file.

BUGS

Changing the `TMPDIR` environment variable prevents cleanup by NQS or `init`(8).

CAUTIONS

If the `-` argument is used, the `TMPDIR` environment variable for the specified user is set to `JTMPDIR`, which is defined in `/usr/include/tmpdir.h`.

EXAMPLES

The following shell script fragment is used to create a temporary directory:

```
.
.
.
#
# shell script fragment to create a unique temporary directory
#
TMP2=‘tmpdir /ram‘
if [ $? != 0 ]
then
        echo "tmpdir failed" >&2
        exit 1
fi
#
# alias unit 11 an actual filename on /ram
#
assign -a ${TMP2}/fort.11 u:11
.
.
.
```

FILES

`${TMPDIR}/.tmpdir`	List of temporary directories created by `tmpdir`.
`/etc/tmpdir.users`	Authorization file for `tmpdir`.

SEE ALSO

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

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

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

NAME

`touch` – Updates access and modification times of a file

SYNOPSIS

`touch` [`-a`] [`-c`] [`-m`] [`-r` *ref_file*] *files*
`touch` [`-a`] [`-c`] [`-m`] [`-t` *time*] *files*

Obsolescent version; may not be supported in future releases:

`touch` [`-a`] [`-c`] [`-m`] [*date_time*] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `touch` utility updates the access and modification times of each file.

You may specify the time used by the `-t` *time* argument, the corresponding time field(s) of the file referenced by the `-r` *ref_file* argument, or the *date_time* operand. If you omit all of these options, `touch` uses the current time. The file name is created if it does not exist. If you omit all options, `touch` updates the access times and modification times.

The `touch` utility accepts the following options and operands:

`-a` Causes `touch` to update only the access times. The modification time is not changed unless you also specify `-m`.

`-c` Silently prevents `touch` from creating the file if it did not previously exist.

`-m` Causes `touch` to update only the modification times. The access time is not changed unless you also specify `-a`.

`-r` *ref_file* Uses the corresponding time of the file specified by the path name *ref_file*, rather than the current time.

`-t` *time* Uses the specified *time* instead of the current time. The argument is of the form:

[[*CC*]*YY*]*MMDDhhmm*[*.SS*]

Each two digits represents the following:

CC The first two digits of the year (the century).
YY Second two digits of the year.
MM Month of the year (01–12).
DD Day of the month (01–31).

hh Hour of the day (00–23).
mm Minute of the hour (00–59).
SS Second of the minute (00–61).

If neither *CC* or *YY* is given, the current year is assumed. If *YY* is specified but *CC* is not, *CC* will become 19 if *YY* is in the range 69–99. *CC* becomes 20 if *YY* is in the range 00–68.

files Specifies each file path name whose times are to be modified.

date_time Uses the specified *date_time* rather than the current time. The operand is of the form:

MMDDhhmm[*yy*]

MM, *DD*, *hh*, and *mm* are as described for the *time* argument to the `-t` option, and the optional *yy* is interpreted as follows:

- If not specified, the current year is used.
- If *yy* is in the range 69–99, the year 1969–1999 is used.
- If *yy* is in the 00–68, the year 2000–2068 is used.

If you omit the `-r` option, the `-t` option is not specified, at least two operands are specified, and the first operand is an 8- or 10-digit decimal number, the first operand is assumed to be a *date_time* operand; otherwise, the first operand is assumed to be a *file* operand.

Only an appropriately authorized user can update time information for a file owned by another user.

NOTES

The `touch` utility updates the date and time displayed by the `ls`(1) utility.

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`	Update time information of any file.
`sysadm`	Update time information of 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 update time information of any file.

EXIT STATUS

The exit status of `touch` is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

SEE ALSO

`date`(1), `ls`(1)

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

tpcatalog – Catalogs, recatalogs, or deletes a dataset in a front-end catalog

SYNOPSIS

tpcatalog [-d] *path*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpcatalog command catalogs, recatalogs, or deletes a dataset in a front-end catalog.

The tpcatalog command accepts the following option and argument:

-d Deletes dataset from the catalog.

path Specifies the path name, a positional parameter, previously specified on the -P or -p option of the tpmnt(1) utility. By default, a new dataset is cataloged; however, if the dataset already exists in the catalog, it is recataloged.

The tpcatalog command returns an error if the specified *path* has not been accessed yet, or if the *path* is still open. Also, the tpcatalog command returns an error if front-end servicing is not turned on in the tape daemon.

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
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

If tpcatalog completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

FILES

/usr/include/taperr.h Tape daemon error codes

SEE ALSO

`tpmnt`(1)

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

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

NAME

tplist – Lists contents of tape volume

SYNOPSIS

tplist [-b *buffer_size*] [-B] [-c *copy_file*] [-C *copy_file*] [-e *tm_count*] [-g *resource_name*] [-G *resource_name*] [-k] [-n *file_count*] [-p] [-P] [-r] [-R] [-s *skip_count*] [-u] [-v *vilist*] [-V *vilist*] *pathname*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tplist utility displays the contents of one or more tape volumes. The data displayed is selected information from the volume header record, the header 1 record, and the header 2 record, along with the number of records in the data sections.

The tplist program automatically issues rsv(1), tpmnt(1), and rls(1) utilities to access a tape volume. If the tape daemon did not create the files, the first end-of-volume will cause tplist to exit.

The tplist utility accepts the following options:

-b *buffer_size* Sets the read/write buffer size (in bytes) to the specified value. The defaults are as follows:

Computer System	Default
CRAY J90 series	48 kilobytes or 49152 bytes
All other Cray Research systems	128 kilobytes or 131072 bytes

If the default buffer size or the value specified is a value smaller than the physical record size, it is too small to identify the contents of the tape and you will receive an error. If the buffer is too small, increase the buffer size and rerun the command until the command executes and the buffer size is determined.

-B Selects blp as the label option for the internal tpmnt(1) command. For additional information on label options, see the tpmnt(1) man page.

-c *copy_file* Copies the input file to the path *copy_file*.

-C *copy_file* Copies the input file to the path *copy_file* and converts from EBCDIC to ASCII. The -c and the -C options are mutually exclusive.

-e *tm_count* Sets the end-of-volume tape mark count to *tm_count*. The default is 2.

-g *resource_name* Sets the device resource group to *resource_name*.

-G *resource_name* Sets the device resource group to *resource_name* for copy volumes.

`-k`	Allows you to verify the output tapes that result from a `tplist` copy operation after the operation has successfully completed.
`-n` *file_count*	Specifies the number of files to copy. The default is 1.
`-p`	Allows you to specify additional parameters for the primary `tpmnt`(1) utility. Double quotation marks must enclose the parameter list.
`-P`	Allows you to specify additional parameters for the copy or secondary `tpmnt`(1) utility. Double quotation marks must enclose the parameter list.
`-r`	Displays unformatted labels.
`-R`	Retains the volume serial number (VSN) on the output tape if labeled tape copy. If you omit the `-R` option, the VSN from the input tape (as specified by the `-v` option) will be written to the output tape (as specified by the `-V` option).
`-s` *skip_count*	Skips *skip_count* files before listing or copying.
`-u`	Sets the no unload flag to keep the primary volume mounted when `tplist` completes.
`-v` *vilist*	Specifies volume identifier list for one or more input tapes. Same format as for `tpmnt`(1).
`-V` *vilist*	Specifies volume identifier list for one or more tapes for a copy operation. Same format as for `tpmnt`(1).
pathname	Specifies the path name to use for input.

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**
`secadm`	Allowed to use this command.

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

To access Data Migration Facility (DMF) tapes, the user must also have the active category of `datamgr`.

MESSAGES

Messages that originate with the tape daemon or any of its service programs are put in the tape message file (`tape.msg`). Other messages may result from I/O errors or tape system failures, and they are printed on `stderr`.

EXAMPLES

The following examples illustrate different uses of the `tplist` utility.

Example 1: The `tplist` utility lists the contents of the tapes `U00600` and `U00601`:

```
tplist -v u00600:u00601 pathname
```

Example 2: The `tplist` utility lists the contents of a round tape `scrsl`:

```
tplist -g TAPE -v scrsl pathname
```

Example 3: The `tplist` utility copies the first 20 files from `u00600` to `mycopy`:

```
tplist -v u00600 -c y -V mycopy -R -n 20 x
```

FILES

`/usr/include/taperr.h`	Tape daemon error codes

SEE ALSO

`rls`(1), `rsv`(1), `tpmnt`(1)

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

tpmnt – Requests a tape mount for a tape file

SYNOPSIS

tpmnt [-a] [-b *block-size*] [-B] [-c *path-name*] [-C *catop1*[:*catop2*]] [-d *density*] [-D *device-name*]
[-f *file-identifier*] [-F *record-format*] [-g *device-group-name*] [-h *file-access-mode*]
[-H *volume-access-mode*] [-i on | off] [-I] [-j *CRL-volume-ID*] [-J *CRL-volume-ID-file*]
[-k *file-password*] [-K *volume-password*] [-l *label-type*] [-L *record-length*] [-m *front-end id*] [-M]
[-n] [-o] [-O *offset-of-first-volume-id*] [-p *path-name*] [-P *path-name*] [-q *file-sequence-number*]
[-Q *file-sequence-number*] [-r *ring-option*] [-R *path-name*] [-s] [-S *volume-set-name*]
[-t *retention-period*] [-T] [-u] [-U] [-v *ivid*[=*evid*][=*fid*][/*part*][:*ivid*[=*evid*][=*fid*][/*part*]...]
[-V *volume-identifier-file*] [-w] [-W *pool-name*] [-x *expiration-date*] [-X]
[-y *volume-set-expiration-date*] [-z]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpmnt utility requests the system operator to mount the specified tape volume and defines the characteristics of a tape file for the tape subsystem.

The -h, -H, -j, -J, -k, -K, -S, -W, -X, and -y options are valid only when the Cray/REELlibrarian (CRL) is in use. These options are documented at the end of the options list.

The tpmnt utility accepts the following options:

-a
Indicates that data is added to the end of an existing file. If the file does not exist, a new one is created. After the program opens the file, the data is positioned after the last block of the file. You cannot use this option with the -o or -n option.

-b *block-size*
Specifies the maximum block length in bytes (represented in decimal). The default maximum block size is specified in the tape configuration. If you specify this option, the specified *block-size* is used, rather than the value in the label of an existing tape file. If you are creating a new file and you do not specify this option, an installation default is used. (The default is specified in the tape configuration file.) If you do not use this option when writing to an existing file, the value of the *block-size* field in the label is used. When you are reading or creating a tape file, you should specify the largest block to be written.

For tapes read on ER90 devices, this option specifies the maximum block size. The value can exceed the actual data block size. However, an error is returned on the I/O request if *block-size* is less than the actual data block size. For tapes created on ER90 devices, this option specifies the size of all blocks to be written to tape. If this option is not specified when creating a new tape, an installation default is used. If writing to an existing file, the existing block size is used. This option is not valid for byte-stream tape files.

`-B` ER90 volumes only. For output tapes, this specifies that a blocked file section be created. For input files, this option is not used. For output tapes, a byte-stream file section is created by default. For input tapes, read requests are issued based on the mode in which the volume was created.

`-c` *path-name* Specifies a concatenated tape file. A previously executed `tpmnt` utility that used the `-p` or `-P` option established tape file *path-name*. If you specify this option, tape files will be concatenated to *path-name*. To concatenate several tape files, use sequential `tpmnt` utility with identical values for the `-c` option. The order of the commands determines the order in which the files are mounted. To read concatenated tape files, the tape subsystem automatically switches between files. The `-c`, `-p`, and `-P` options are mutually exclusive.

`-C` *catop1*[:*catop2*] Specifies catalog options to be implemented for the tape file, specified in the *file-identifier* of the `-f` option, on a servicing front end.

The following catalog options are available with the `-C` option:

`SAVE` Catalogs or recatalogs file in the front-end catalog

`DEL` Deletes file from the front-end catalog

`NO` Performs no operation on the front-end catalog

If the tape access executes normally, the catalog option specified in *catop1* is implemented. If the tape access terminates abnormally, the catalog option specified in *catop2* is implemented. If you specify *catop1* alone, *catop2* defaults to `NO`. If you omit the `-C` option, no operation is performed on the front-end catalog.

`-d` *density* Specifies the density of the tape volume. Density may be 1600 b/i or 6250 b/i. You cannot use this option if IBM 3480-type drives are specified with the `-D` or `-g` option (`tpmnt` will abort).

`-D` *device-name* Requests a specific device name for the mount.

If the requested device is available, it will be used to mount all volumes of a job. If the requested device is busy (assigned to another user), the request will be queued until the device is available. If the requested device is not configured up, the mount request will fail. If the volume serial number (VSN) requested is premounted and idle on another drive, that drive will be used instead of the drive requested. When automatic volume recognition (AVR) is enabled, this option is not available.

`-f` *file-identifier* Identifies the file on the volume that was recorded or will be recorded in the HDR1 label. You can specify the file identifier in lowercase or uppercase letters; however, the system automatically converts lowercase letters to uppercase letters (ANSI standard requirement). If the result is longer than 17 characters, the last 17 characters are used. If, after any trailing blanks are removed, the result is less than 17 characters, it is padded with blanks. The system administrator configures whether the *file-identifier* is needed and checked. If you omit the *file-identifier* with this option, the file name portion of the path name specified in the `-p` or `-P` option is used as the *file-identifier*. This option is ignored for unlabeled tapes.

For CRL the format of the *file-identifier* may be the format of a CRL file name. See the `rlfedit`(1) command for information on CRL file name syntax.

`-F` *record-format* Specifies the value of the record format field in the HDR2, EOV2, and EOF2 labels of ANSI and standard IBM labels. Record format may be one of following:

`F` Fixed length (for both ANSI label and IBM standard label)

`D` Variable length with zoned decimal length indicator (for ANSI label)

`U` Undefined length (for both ANSI label and IBM standard label)

`V` Variable length (for standard IBM label)

This value is used when creating labels. The UNICOS operating system does not verify that the data format in the file agrees with this value, and there is no data conversion when the tape is written or read (see `assign`(1) for data conversion). If you omit the `-F` option when creating a new file, the default record format is `U`. If you omit the `-F` option when writing over an existing file, the record format in the label is used.

You may specify an optional second character. This character indicates the attributes of the data, and it may be one of the following:

`B` Blocked records

`S` Spanned or standard records

`R` Blocked and spanned or standard records

`-g` *device-group-name*

Specifies the name of the group to which the requested device belongs. If you omit *device-group-name*, it defaults to an installation-specified name. The device group name (*dgn*) field of `tpstat`(1) shows the allowable parameters.

`-i on | off` Specifies the status of the Improved Data Recording Capability (IDRC) feature when you are using IBM cartridge model 3490 or 3480 tape drives that have this feature.

If you specify the `-i` option, the tape subsystem activates or deactivates IDRC accordingly when the file specified on the `tpmnt` utility is opened and when new volumes are mounted if the file is a multivolume file. The tape subsystem does not change the IDRC setting when the tape is unloaded.

Status is one of the following:

`on` Activates IDRC. When data is read from the device, the control unit decompresses the data automatically.

`off` Deactivates IDRC. Data is not compressed.

If you omit the `-i` option, you do not have any control over IDRC usage.

If you specify the `-i` option with incompatible hardware, an error occurs when data is first written to the device.

`-I` Specifies that mount verification using the format ID should be bypassed. This option requires that the user have bypass label or tape manager permission. This option is valid only with ER90 volumes.

`-l` *label-type* Defaults to an installation-specified label type. The *label-type* may be one of the following:

`al` ANSI label

`blp` Bypass label processing

`nl` Not labeled

`sl` IBM standard label

`st` Single tape mark format (single tape mark terminates reel)

`ulp` User label processing. This argument causes the tape subsystem to ignore the label type (`unknown` is the operator mount message) and to rewind the tape to the beginning of the volume so the user can read the volume and header records. Normal access control is checked prior to allowing the user to read the tape labels. The tape is mounted with the write protect ring out to prevent any accidental writes at the beginning of the tape. This argument is used primarily by the `tplist`(1) utility to examine the contents of a tape, and it eliminates the need for `tplist`(1) to use `blp`.

Note: Special permission is needed for `blp` and `nl` labels. See your system administrator.

`-L` *record-length* Indicates the maximum record length in bytes (represented in decimal). If you specify this option, the specified *record-length* is used rather than the value in the HDR2, EOV2, and EOF2 labels of an existing tape file. If you are creating a new file, and you omit this option, an installation default value is used. If you do not specify this option when writing over an existing file, the value in the *record-length* field of the label is used. For nonlabeled tapes, this option is ignored.

`-m` *front-end id* Specifies a front-end ID. It identifies the servicing front end (32 character maximum).

`-M` Delays mount message until file is opened. A tape unit is assigned when you issue `tpmnt`.

`-n` Specifies that the file is a new file. You cannot use this option with the `-a` or `-o` option.

`-o` Specifies that the file is an old file. You cannot use this option with the `-a` or `-n` option.

`-O` *offset-of-first-volume-id*

Specifies an offset into the volume identifier list. The offset points to the volume that will be mounted first. The first volume identifier has an offset of 1, which is the default. The offset is set to 1 after the file is opened.

If you use the `-O` option to specify an offset to a volume identifier, the output of the `tpmql`(8) command is no longer accurate. The `tpmql`(8) command displays the entire list of volume identifiers. If an offset has been set, the list of identifiers from `tpmql`(8) may no longer correspond to the order in which the tapes have been accessed with the `tpmnt` utility.

`-p` *path-name* Specifies the path name of the tape file. This path name is used to create a file that will be associated with the device used by the tape file. If the file to which this path name points already exists, an error will be issued. You must specify a path name by using the `-p`, `-P`, or `-c` option; these options are mutually exclusive. Your program must use this path name for the open function. If the path name is not a full path name, `tpmnt` appends what you have specified to your working directory. You must not move or remove this file, because unpredictable results will occur. If you do not specify a *file-identifier* with the `-f` option, the file name portion of this path name is used as the *file-identifier*.

`-P` *path-name* Specifies the path name for this tape file. The file to which this path name points, if it exists, is removed before a new one is created. (No error will be issued.) You must specify a path name by using the `-p`, `-P`, or `-c` option. Your program must use this path name in the open function. If the path name is not a full path name, `tpmnt` appends what you have specified to your working directory. A file is created by using the path name. The file is associated with the device that the tape file will use. You must not move or remove this file, because unpredictable results will occur. If you do not specify a *file-identifier* with the `-f` option, the file name portion of the path name is used as the *file-identifier*.

`-q` *file-sequence-number*

Specifies the file sequence number of the file being processed. The default is 1, which indicates the first file on the tape. The following are valid values for *file-sequence-number*:

number Is used if *file-sequence-number* is specified as a number and the tape is either labeled or unlabeled.

If the tape is labeled, the file sequence number is compared to that recorded in the HDR1 label of the first file of an existing volume. the result determines how many files the tape subsystem must skip to position to the correct file.

If the tape is unlabeled, the tape subsystem skips *file-sequence-number* tape marks from the beginning of the volume to position to the correct file.

`u` Indicates that positioning should be based on a file name.

`n` Creates a new file at the end of the tape. The file sequence number is not validated until the file is opened.

`-Q` *file-sequence-number*

Skips to position `n-1` and verify the header and trailer label of the file before the desired file. This option is designed for users who are confident of the contents of single volume tapes. It reverts to `-q` processing if any of the following are true:

- CRL is in use.
- User tape marks are specified.
- Multivolume tape is being accessed.

`-r` *ring-option* Specifies whether the write protect ring should be in or out. The allowable value for *ring-option* is either `in` or `out`. If you omit `-r`, an installation-defined default is used.

`-R` *path-name* Lets you release a tape volume and request a new tape by using one command. Your can use `-R` in two ways. If `-R` is used to specify a path name specified by the `-p` or `-P` option of a previously executed `tpmnt` utility, a tape mounted on this path is remounted with a new volume. If `-R` is used in conjunction with the `-p` or `-P` option, the tape volume mounted on the path specified with `-R` is released, and the volume is mounted on the path specified with the `-p` or `-P` option.

`-s` Specifies that the file is to be protected on the servicing front end. The `-s` option is valid only for new files being cataloged on the servicing front end.

`-t` *retention-period* Specifies the number of days a labeled tape will be kept. The default retention period is 0. This option is ignored on nonlabeled tapes.

`-T` Lets you read or write tape marks that are embedded in data.

`-u` Disables tape unload at release time. Usually, a tape is unloaded automatically when a job terminates or releases the tape resources. This option is useful when a tape is used repeatedly, and it minimizes operator time spent mounting tapes. This option has the same effect as specifying the `-n` option on the `rls`(1) utility.

`-U` Specifies unbuffered I/O for transparent I/O. Data for an I/O operation should not be buffered. When `-U` is specified, the read size must be a multiple of 4096 bytes and must be greater than or equal to the maximum block size. If `-U` is specified on a write request, data is written to tape as a tape block. If `-U` is specified on a read request, a tape block is transferred into the user's read buffer.

`-v` *ivid*[=*evid*][=*fid*][/*part*]
[:*ivid*[=*evid*][=*fid*][/Vpart*]...*
Specifies the volume identifier of the tapes. This option specifies a list of volume identifiers. You can specify the volume identifiers in lowercase or uppercase letters; the system automatically converts the internal and external volume IDs to uppercase letters.
The *ivid* is the ID that appears on the volume label. The *evid* specifies the identifier that appears on the physical label of the tape reel; this differs from the volume ID on the volume label. If the external volume ID is not specified, the default is the internal ID. The *fid* specifies the volume identifier recorded on the tape during a volume format. If a format ID is not specified, the internal volume ID is used. The *fid* is used for mount verification (used for ER90 volumes only). The *part* specifies the partition number to which the volume should be positioned. The partition number must be in the range of 0 through 1023. The default is 0, the first partition on the volume (a partition number is valid only on ER90 volumes). The `-V` and `-v` options are mutually exclusive.

`-V` *volume-identifier-file*
Specifies the file that contains a list of volume identifiers and external volume identifiers that compose a multivolume tape file. The `-V` option is useful for tape files that have many volume identifiers. The `-V` and `-v` options are mutually exclusive.

`-w` Lets you suspend execution until the `tpmnt` request is completed. This lets you check status to determine whether the mount was successful.

`-x` *expiration-date* Specifies the Julian date (in *xyyddd* format) on which the tape file expires. The year (*yy*) in the date ranges from 00 to 99, and the days (*ddd*) range from 001 to 366. If *x* is a blank, the first two significant digits of the year are 19. If the first character is a 0, the first two significant digits of the year are 20.
If the first digit is @, using [`98`*xxx*] or [`99`*xxx*] overrides the special meaning that the years 1998 and 1999 have to the Catalog Management System (TMS) on an IBM Multiple Virtual Storage (MVS) Operating System. Entering the @ causes the real 1998 and 1999 expiration dates to be used.
For nonlabeled tapes, this option is ignored. If you use CRL, specify this option by using the CRL file expiration date syntax as that used on CRL file commands (`rlfedit`(1) and `rlfsubmit`(1)), as follows:

`Adays` The file expires if not accessed in a period longer than the number of days specified.

`Gnum` The file expires when the number of newer generations equals the number specified by `num`. Valid values for `num` are 0 through 99.

`I`	The file never expires.
`Rdays`	The file expires the specified number of days after it is created.
`S`	The file is always expired; scratch.
`Xdate`	The file expires on the specified date; the format of the date is *mm/dd/yy*.

`-z` ER90 volumes only. Specifies that the volume be left at its current position after a tape mount and before an unload. By default, the tape is positioned to the beginning of the tape. To use this option, users must have bypass label or tape manager permission.

CRL options:

`-h` *file-access-mode* Specifies CRL mode protection value for files. When creating or recataloging a tape file, use *file-access-mode* as the CRL mode protection value.

`-H` *volume-access-mode*
Specifies CRL mode protection value for volumes. When creating or recataloging a tape file, use *volume-access-mode* as the CRL mode protection value.

`-j` *CRL-volume-ID*[:*CRL-volume-ID*]
Specifies the use of *CRL-volume-ID* as the volume identifier. You can use this option instead of the `-v` or `-V` options to uniquely identify the list of volumes to use. When using CRL, you do have to specify the entire list of volume identifiers, because the list is retrieved in its entirety when the volume set has been identified by any option that identifies file name (`-f`), path name (`-p`), volume set name (`-S`), or volume identifier (`-v` or `-V`).

`-J` *CRL-volume-ID-file*
Specifies that this file contains a list of unique CRL volume IDs that constitute a multivolume tape file. You may use this option instead of the `-v` or `-V` option to uniquely identify the list of volumes to use. When using CRL, you do not have to specify the entire list of volume identifiers, because the list is retrieved in its entirety when the volume set has been identified by any option that identifies file name (`-f`), path name (`-p`), volume set name (`-S`), volume identifier (`-v` or `-V`), or first CRL volume identifier (`-j`).

`-k` *file-password* Specifies the password to be associated with a file that is being created, or the password to be compared with the password for an existing file.

`-K` *volume-password* Specifies the password to be associated with a volume that is being created, or the password to be compared with the password for an existing volume.

`-S` *volume-set-name* Specifies the CRL volume set name to be accessed. You can use this option in place of the `-v`, `-V`, `-j`, or `-J` option. All appropriate volume identifiers are retrieved from the CRL catalog for access.

`-W` *pool-name* Specifies the CRL pool name to be used when drafting volumes for use in writing new tape files. If you omit this option, CRL uses the default pool that has been assigned to the user.

`-X` Causes all file and volume activity to be processed by the CRL catalog server. If CRL is not enabled, specifying this option causes an error. If CRL is enabled and mandatory, this option is not required; if it is used, it has no effect.

`-y` *volume-set-expiration-date*

Specifies the volume set expiration date for newly created volume sets. The format of the expiration date is the same as that used on the CRL volume set commands (`rlvcreate`(1), `rlvedit`(1), and `rlvsubmit`(1)), as follows:

`Adays`	The volume expires if not accessed in a period longer than the number of days specified.
`Gnum`	The volume expires when the number of newer generations equals the number specified by `num`. Valid values for `num` are 0 through 99.
`I`	The volume never expires.
`L`	The volume expires when all files in the volume set have expired.
`Orot_name`	The volume set follows the `rot_name` rotation schedule.
`Rdays`	The volume expires the specified number of days after it is created.
`S`	The volume is always expired; scratch.
`Xdate`	The volume expires on the specified date; the format of the date is *mm/dd/yy*.
`Z`	Duplicate 98000 expiration date functionality. If you specify `Z` and the VSNs specified by the `-V` option are not found in the CRL catalog, CRL processing is bypassed and tape processing continues.

NOTES

The UNICOS system can enforce tape labeling. The tape daemon automatically sets the accessibility byte of the VOL1 and HDR1 labels, and it writes the security label in the HDR2 label for each classified file written to tape. ANSI and IBM labels are supported.

EXIT STATUS

If `tpmnt` completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

The following examples illustrate different uses of the `tpmnt` utility.

Example 1: The `tpmnt` utility requests that volume 000011 be mounted, and tape processing begins at partition 3. The format ID of the mounted tape must be ER011. If a file is created, it will be a blocked file that consists of blocks of size 1,199,832 bytes.

```
tpmnt -v 000011==ER011/3 -l sl -p x -n -g ER90 -B -b 1199832
```

Example 2: The `tpmnt` utility requests that volume 000011 be mounted, and tape processing begins at partition 0. Format ID verification will not be done. If a file is created, it will be a byte stream file. If another volume is needed, the tape will be positioned to partition 1 of the same volume without unloading volume 000011.

```
tpmnt -v 000011/0:000011/1:000011/2 -l sl -p x -n -g ER90 -I
```

Example 3: The `tpmnt` utility requests that volume 000012 be mounted, and the tape is left at its load position. After opening the `tapefile` file, the first request to the tape file must be a request to establish the logical position. This is done by issuing a `TPC_PABS ioctl` request.

```
tpmnt -v 000012 -l blp -p x -f tapefile -g ER90 -B -b 1048576 -z
```

FILES

`/usr/include/errno.h`	Error code header file

SEE ALSO

`rls`(1), `rsv`(1), `tprst`(1), `tpstat`(1)

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

Tape Subsystem Administration, Cray Research publication SG–2307

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

NAME

tpquery – Queries autoloaders for VSN information

SYNOPSIS

```
tpquery [-l] [-m loader_name] [-t trace_file] -v vsns
tpquery [-l] [-m loader_name] [-t trace_file] -V vsn_list_file
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpquery command queries an autoloader for information concerning one or more volume serial numbers (VSNs). The returned information contains text that indicates whether the specified VSNs are within the domain of the queried autoloader.

The tpquery command accepts the following options:

`-l` Provides the following information for each specified VSN:

loader name: vsn: text on cartridge state: possible cartridge location

If you omit the `-l` option, no information concerning the specified cartridge(s) is provided.

`-m` *loader_name* Specifies the autoloaders that the command will query. Enter one of the following for *loader_name*:

name Specifies the name of an autoloader in the tape configuration file. If you specify an autoloader name, tpquery directs its query to the specified autoloader after it has verified that the name appears in the tape configuration file, that the autoloader is connected to a network, and that the autoloader is in an UP or MANUAL state.

all or ALL Specifies all autoloaders listed in the tape configuration file that are connected to a network and that are in an UP or MANUAL state. If one of these autoloaders contains all the specified VSNs, the remaining autoloaders are not queried. The default is all.

`-t` *trace_file* Specifies the name of a file to which tpquery writes trace information to while the command is being executed. If you omit the `-t` option, tpquery executes without writing trace information.

`-v` *vsns* Specifies one or more cartridge VSNs in the format that is documented for the tpmnt(1) command. The `-v` and `-V` options are mutually exclusive; one is required.

`-V` *vsn_list_file* Specifies the name of a file that contains a list of one or more VSNs. The `-v` and `-V` options are mutually exclusive; one is required.

EXIT STATUS

If `tpquery` completes successfully and has found all specified VSNs in one autoloader, it returns a zero status. If the command executes successfully, but does not find all specified VSNs in one autoloader, it returns a status with value `122: ETQRY - ERRBASE`.

If `tpquery` fails, an exit status code in the range of 1 through 255 is returned; where possible, the number is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

Example 1: This example shows the `tpquery` command being executed for VSN `xxxxxx`. Because the `-m` option is omitted, `tpquery` interrogates all of the autoloaders. The `-l` option is specified; therefore, all output is displayed.

```
# tpquery -v xxxxxx -l
stksun: XXXXXX: volume_not_in_loader.
wolfy:  XXXXXX: volume_not_in_loader.
ibm:    XXXXXX: volume_not_in_loader.
# echo $?
122
```

This output shows that VSN `xxxxxx` is not located in any of the online autoloaders. The status code returned to the calling program is `ETQRY` (`122`). The output also shows that lowercase alphabetic characters in the VSN are converted to uppercase alphabetic characters.

Example 2: This example shows the `tpquery` command being executed for VSN `S66580`. Because the `-m` option is specified, `tpquery` interrogates only the autoloader, called `wolfy`. It displays all of the output because the `-l` option is specified;

```
# tpquery -m wolfy -v S66580 -l
wolfy:  S66580: volume_in_loader: home:         0,3,7,3.
# echo $?
0
```

This output shows that VSN `S66580` is located in `wolfy` and that the VSN is located in its home position, which has the address `0,3,7,3`. The status code returned to the calling program is 0.

Example 3: This example shows the `tpquery` command being executed for VSN `003600`. The command interrogates all of the autoloaders because the `-m` option is omitted and displays all output because the `-l` option is specified,

```
# tpquery -l -v 003600
stksun: 003600: volume_not_in_loader.
wolfy:  003600: volume_not_in_loader.
ibm:    003600: volume_in_loader: home.
# echo $?
0
```

This output shows that VSN 003600 is located in the ibm autoloader and that this VSN is located in its home position, which is not further substantiated for IBM and EMASS autoloaders. The status code returned to the calling program is 0.

Example 4: This example shows the tpquery command being executed for VSN 003600. Neither the -m or -l option is specified; tpquery interrogates all of the autoloaders and returns 0 to the calling program. This status code indicates that VSN 003600 was found in one of the online autoloaders.

```
# tpquery -v 003600
# echo $?
0
```

SEE ALSO

tpmnt(1)

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

NAME

tprst – Displays reserved tape status for current job ID

SYNOPSIS

```
tprst
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tprst command displays the status of the tapes reserved for the current job ID. The following information is provided on standard output (stdout):

dev grp	Device group name.
w	User is waiting for a device. Indicated by an asterisk (*).
rsvd	Number of reserved devices.
used	Number of devices that have been used or are in use.
available	Number of devices available for reservation.

EXIT STATUS

If no error has occurred, tprst exits with a return code of 0. If an error has occurred, tprst exits with one of the following error codes (in decimal).

Symbol	Value	Description
ETSYS	23	Tape subsystem error
ETDCE	31	Cannot communicate with the tape subsystem

The symbols for these codes are located in the /usr/include/taperr.h file.

EXAMPLES

The following example shows output from tprst. The user has two TAPE devices and one CART device reserved and has used one of the two TAPE devices. The user is not waiting for any devices.

```
dev grp  w     rsvd     used available
TAPE             2        1    1
CART             1        0    1
```

SEE ALSO

`rls`(1), `rsv`(1), `tpmnt`(1), `tpstat`(1)

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

NAME

tpstat – Displays current tape status

SYNOPSIS

tpstat [-a] [-l]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tpstat utility returns the status of all tape devices.

The tpstat utility accepts the following options:

-a Outputs device status only for tape devices that have a status other than DOWN. If you omit this option, tpstat displays the device status for all tape devices.

-l Outputs the device status in long format. The long format has an added field length for the block count and outputs the format ID of mounted volumes.

The tpstat utility provides the following status:

userid Specifies user ID; blank if device is not assigned.

jobid Specifies job ID of the user if device is assigned.

dgn Specifies device group name.

a Specifies automatic volume recognition (AVR) for this device. If a plus sign (+) is displayed, AVR is turned on; if a minus sign (–) is displayed, AVR is turned off. If an at sign (@) is displayed, overcommitted mount requests are available; if an at sign (@) is not displayed, overcommitted mount requests are not available.

stat Specifies status of the device; status can be one of the following:

- assn Device is assigned to a user.
- cnfp Device is waiting for a configuration up command to complete.
- conn Device is temporarily not available for assignment.

  The conn status can result from the situations described in the following examples, but is not limited to these situations:

  Example 1: If a loader drive is available, but is not the most desirable drive, you see conn in the tpstat output for the drive currently owned by a tape daemon child process while the child process is selecting a more desirable drive.

Example 2: If a tape unit in an AVR system is up and a tpmnt(1) request is made for a tape volume currently not on a tape unit, the available tape unit is temporarily assigned to the requesting task and tpstat shows conn. When the tape is mounted on that tape unit, it is assigned for the requesting task to use and the tpstat output shows assn.

dldr Device is waiting for a configuration down command to complete.

down Device is unavailable.

free Device is being freed by an asynchronous process.

idle Device is available and not in use.

sdwn Device has been made unavailable by the system.

unav Device is unavailable.

wdwn Device is waiting to go to down state.

wsdn Device is waiting to go to sdwn state.

wsup Device is waiting to go to idle state.

If the device is configured as a no unload device (see tpconfig(8)), or if you requested no unload at release time by specifying the -u option of tpmnt(1), or by specifying the -n option of rls(1), a plus sign (+) is displayed next to the assn or idle status.

dvn Specifies device name.

bx Specifies minor device number of the tape device.

i or ion Specifies the following:

i Specifies hexadecimal cluster number.

ion Specifies the node.

rl
r = Ring status if assigned
i = Ring in
o = Ring out
l = Label status if assigned
a = ANSI label
b = Bypass label
n = Not labeled
s = IBM standard label

ivsn Specifies the internal volume identifier if assigned. If the tape is not yet mounted, an asterisk sign (*) appears before the vsn.

evsn Specifies the external volume identifier if assigned. This is the physical outer label of the tape volume. For more information, see the -v option of tpmnt(1).

`fvsn` Specifies the format identifier if assigned. The format identifier is the identifier that is recorded on the volume when the volume is formatted.

`blks` Specifies number of tape blocks read or written if assigned.

`NQSid` Specifies Network Queuing System (NQS) batch job ID if the assigned tape has been submitted through NQS. You may use this ID for `qstat`(1) or `qdel`(1).

EXIT STATUS

If `tpstat` completes successfully, 0 is returned; otherwise, a nonzero value is returned. Where possible, this exit status code is normalized to the last three digits. Exit status values are documented in the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

EXAMPLES

In the following example, devices `tape100` through `cart509` are configured as `down`. Devices `cart300` and `cart301` are not assigned to a job; therefore, they are `idle`. They have IBM-standard tapes `WE0022` and `WE0023` mounted on them.

User `hew` with job id `113` has `cart302` assigned as ring-in, ANSI-labeled, with an external volume identifier equal to `XXX` and an internal volume identifier equal to `ISCAL`. Nine blocks of data have been processed.

User `jas` with job ID `44` has been assigned `cart303`. It has been specified to be no unload (+), so that the tape will remain mounted after user `jas` issues an `rls`(1) utility. This tape has been assigned as ring-in, nonlabeled, volume identifier equal to `ISCNL`, and with 30 blocks of data processed. AVR is set to `on` for all tapes.

All tapes are attached to IOS 0.

```
    userid jobid dgn  a stat  dvn     bx i rl ivsn   evsn   blks NQSid
                 TAPE + down  tape100 02 0
                 TAPE + down  tape101 03 0
                 TAPE + down  tape102 04 0
                 TAPE + down  tape103 05 0
                 CART + down  cart508 06 0
                 CART + down  cart509 07 0
                 CART + idle  cart300 10 0 is WE0022 WE0022
                 CART + idle  cart301 09 0 is WE0023 WE0023
    hew    113   CART + assn  cart302 11 0 ia ISCAL  XXX    9
    jas    44    CART + assn+ cart303 08 0 in ISCNL  ISCNL  30
                 CART + down  cart400 12 0
                 CART + down  cart401 13 0
                 CART + down  cart402 14 0
                 CART + down  cart403 15 0
```

The following `tpstat` example is from a system with scalable I/O.

```
userid   jobid dgn     a stat dvn      bx ion  rl ivsn   evsn   blks  NQSid
               STK4890 - down s4890s0  02 0000
               STK4890 - down s4890s1  03 0000
               DLT4000 - down d4000s0  04 0000
               DLT4000 - down d4000s1  05 0000
               IBM3490E- idle 3490s0   06 0000
               IBM3490E- down 3490s1   07 0000
               DAT     + down h1533sX  08 0100
               STK9490 - down s9490s0  09 0100
               STK9490 - down s9490s1  10 0100
               STK9490 - down s9490s2  11 0100
               STK9490 - down s9490s3  12 0100
               IBM3590 - down 3590s0   13 0100
               IBM3590 - down 3590s1   14 0100
               STKSD3  - down ssd3_s0  15 0100
               STKSD3  - down ssd3_s1  16 0100
```

FILES

`/usr/include/tapereq.h`	Tape daemon interface definition file

SEE ALSO

`qdel`(1), `qstat`(1), `rls`(1), `rsv`(1), `tpmnt`(1), `tprst`(1)

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

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

NAME

tput – Initializes a terminal or query terminfo database

SYNOPSIS

```
tput [-T type] capname [parms ...]
tput [-T type] clear
tput [-T type] init
tput [-T type] longname
tput [-T type] reset
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (*capname* and longname operands)

DESCRIPTION

The tput utility uses the terminfo(5) database to make the values of terminal-dependent capabilities and information available to the shell (see sh(1)), to initialize or reset the terminal, or return the long name of the requested terminal type. tput outputs a string if the attribute (*cap*ability *name*) is of type string, or an integer if the attribute is of type integer. If the attribute is of type Boolean, tput simply sets the exit code (0 for true if the terminal has the capability; 1 for false if it does not) and produces no output. Before using a value returned on standard output, the user should test the exit code. (See the EXIT STATUS section.) For a complete list of capabilities and the *capname* associated with each, see terminfo(5).

The tput utility supports the following option and operands:

-T *type* Indicates the *type* of terminal. Normally this option is unnecessary, because the default is taken from the TERM environment variable. If -T is specified, the shell variables LINES and COLUMNS will not be referenced.

capname Indicates the attribute from the terminfo(5) database.

parms... If the attribute is a string that accepts parameters, the *parms* arguments will be instantiated into the string. An all-numeric argument is passed to the attribute as a number.

clear Clears the terminal screen.

longname If the terminfo(5) database is present and an entry for the user's terminal exists (see -T *type*), the long name of the terminal will be output. The long name is the last name in the first line of the terminal's description in the terminfo(5) database (see term(5)).

init If the terminfo(5) database is present and an entry for the user's terminal exists (see -T *type*) the following will occur:

1. Delays specified in the entry will be set into the tty driver.
2. If the hardware knows about tabs (as determined by the definition of the `ht`, `hts`, and `it` capabilities), the tty driver will not expand tabs.
3. If the `iprog` capability is not null, the specified program will be executed.
4. The `is1` string is sent if it exists, followed by the `is2` string, if it exists.
5. An attempt is made to initialize the tabstops of a terminal every 8 characters; if it does not work, you can fix it by using either steps 6 or 7.
6. If not null, the file specified by the *if* capability is copied to the terminal.
7. If not null, the `is3` string is sent.

`reset` Instead of putting out initialization strings, the terminal's reset strings `rs1`, `rs2`, `rs3`, and `rf`. will be output if present. If the reset strings are not present, but initialization strings are, the initialization strings will be output. Otherwise, `reset` acts the same as `init`.

EXIT STATUS

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

0 The requested string was written successfully.

1 Unspecified.

2 Usage error.

3 No information is available about the specified terminal type.

4 The specified operand is invalid.

>4 An error occurred.

EXAMPLES

Example 1: The following example initializes the terminal according to the type of terminal in the `TERM` environmental variable. This command should be included in everyone's `.profile` after the `TERM` environmental variable has been exported, as illustrated on the `profile`(5) man page:

```
tput init
```

Example 2: The following example resets an AT&T 5620 terminal, overriding the type of terminal in the `TERM` environment variable:

```
tput -T5620 reset
```

Example 3: The following command sends the sequence to move the cursor to row 0, column 0 (the upper left corner of the screen, usually known as the "home" cursor position):

```
tput cup 0 0
```

Example 4: The following command echoes the clear-screen sequence for the current terminal:

```
tput clear
```

Example 5: The following command prints the number of columns for the current terminal:

```
tput cols
```

Example 6: The following command prints the number of columns for the 450 terminal:

```
tput -T450 cols
```

Example 7: The following command sets the standard shell variables `bold` to begin stand-out mode sequence, and `offbold`, to end stand-out mode sequence, for the current terminal. This might be followed by a prompt:

```
bold=‘tput smso‘
offbold=‘tput rmso‘
echo "${bold}Please type in your name: ${offbold}\c"
```

FILES

`/usr/lib/terminfo/?/*`	Compiled terminal description database
`/usr/include/curses.h`	`curses`(3) header file
`/usr/include/term.h`	`terminfo`(5) header file

SEE ALSO

`sh`(1) to invoke the standard command interpreter and command-level language
`stty`(1) to set the options for a terminal
`tabs`(1)

`curses`(3) to optimize terminal screens (available only online)

`profile`(5) for information on format of shell start-up file
`term`(5)
`terminfo`(5) for information on terminal capability database
in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

NAME

`tr` – Translates characters

SYNOPSIS

`tr` [`-c`] [`-s`] *string1 string2*
`tr -s` [`-c`] *string1*
`tr -d` [`-c`] *string1*
`tr -d -s` [`-c`] *string1 string2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tr` utility copies standard input to standard output with the substitution or deletion of selected characters.

Input characters found in *string1* are mapped into the corresponding characters of *string2*.

The `tr` utility supports the following options:

`-c` Complements the set of characters specified by *string1*.

`-s` Replaces instances of repeated characters with a one character.

`-d` Deletes all occurrences of input characters specified by *string1*.

string1
string2 Translation control strings. Each string represents a set of characters to be converted into an array of characters used for the translation.

The *string1* and *string2* operands define two arrays of characters. You can use the constructs in the following list to specify characters or single-character collating elements:

character Any character not described by one of the following conventions.

octal Octal sequences can be used to represent characters that have specific coded values. An octal sequence consists of a backslash followed by the longer sequence of one-, two-, or three-octal digit characters (01234567). The sequence causes the character whose encoding is represented by the one-, two-, or three-digit octal integer to be placed into the array.

character The following backslash-escape sequences may be specified:

`\\` backslash
`\a` <alert>
`\b` <backspace>

```
\f   <form-feed>
\n   <newline>
\r   <carriage-return>
\t   <tab>
\v   <vertical-tab>
```

c-*c* Represents the range of collating elements between the range endpoints, inclusive, as defined by the current setting of the LC_COLLATE locale category. The starting endpoint precedes the second endpoint in the current collation order. The characters or collating elements in the range are placed in the array in ascending collation sequence.

[:*class*:] Represents all characters that belong to the defined character class, as defined by the current setting of the LC_CTYPE locale category. The following character class names are accepted when specified in *string1*: alnum, alpha, blank, cntrl, digit, graph, lower, print, punct, space, upper, and xdigit.

When you specify both the -d and -s options, any of the character class names are accepted in *string2*; otherwise, only character class names lower or upper are valid in *string2* and then only if the corresponding character class (upper and lower, respectively) is specified in the same relative position in *string1*. Such a specification is interpreted as a request for case conversion.

[=*equiv*=] Represents all characters or collating elements that belong to the same equivalence class as *equiv*, as defined by the current setting of the LC_COLLATE locale category. An equivalence class expression is allowed only in *string1*, or in *string2*, when it is being used by the combined -d and -s options.

[*x***n*] Represents *n* repeated occurrences of the character *x*. Because this expression is used to map multiple characters to one, it is valid only when it occurs in *string2*. If you omit *n* or it is 0, it is interpreted to be large enough to extend the *string2*-based sequence to the length of the *string1*-based sequence. If *n* has a leading 0, the value is interpreted as octal; otherwise, the value is interpreted as decimal.

NOTES

Currently, UNICOS supports only the POSIX and C locales. These locales support only the ASCII character set.

The tr utility returns an error for tr -d *string1 string2* to report superfluous *string2* arguments. In UNICOS releases prior to 8.0, *string2* was simply ignored and processing occurred only on *string1*.

EXIT STATUS

The tr utility exits with one of the following values:

0 All input was processed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following example creates a list of all words in *file1*, one word per line, in *file2*; a word is taken to be a maximal string of alphabetic characters. The strings are quoted to protect the special characters from interpretation by the shell; 012 is the ASCII code for a <newline> character.

```
tr -cs "[A-Z][a-z]" "[\012*]" < file1 > file2
```

Example 2: The following example performs the same function as example 1:

```
tr -cs "[:alpha:]" "[\n*]" < file1 > file2
```

Example 3: The following example creates a file2 in which all alphabetic characters are uppercase:

```
tr "[:lower:]" "[:upper:]" < file1 > file2
```

SEE ALSO

ed(1), sh(1)

NAME

troff – Typesets or formats documents

SYNOPSIS

troff [-a] [-f] [-i] [-z] [-F*dir*] [-m*name*] [-n*N*] [-o*list*] [-r*aN*] [-s*N*] [-T*dest*] [-u*N*] [*filename*] ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The troff utility formats text in the *filename* argument. Input to troff is expected to consist of text interspersed with formatting requests and macros. If no *filename* argument is present, troff reads standard input. A hyphen as a *filename* argument indicates that standard input is to be read at that point in the list of input files; troff reads the files named ahead of the hyphen in the arguments list, then text from the standard input, and then text from the files named after the hyphen.

The following options may appear in any order, but they all must appear before the first *filename*.

-a Sends a printable approximation of the formatted output to the standard output file.

-f Does not print a trailer after the final page of output or causes the postprocessor to relinquish control of the device.

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

-z Suppresses formatted output. Only diagnostic messages and messages output using the .tm request are output.

-F*dir* Searches the directory *dir* for font width tables instead of using the system-dependent default directory.

-m*name* Prepends the macro file /usr/lib/tmac/tmac.*name* to the input *filename*s. Note that most references to macro packages include the leading m as part of the name; for example, the man macro package resides in /usr/lib/tmac/tmac.an.

-n*N* Numbers first generated page *N*.

-o*list* Prints only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. 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.

-r*aN* Sets register *a* (1 character) to *N*.

-s*N* Stops the phototypesetter every *N* pages. On some devices, troff produces a trailer so you can change cassettes; resume by pressing the typesetter's start button.

-T*dest* Prepares output for typesetter *dest*. Only PostScript (-Tpsc) devices are currently supported.

`-u`*N* Sets the emboldening factor for the font mounted in position 3 to *N*. If *N* is missing, sets the emboldening factor to 0.

filename Specifies the file to be interpreted.

FILES

`/usr/lib/tmac/tmac.*`	Pointers to standard macro files
`/usr/lib/macros/*`	Standard macro files
`/usr/lib/font/devpsc`	Font width tables

SEE ALSO

`checknr`(1), `chmod`(1), `eqn`(1), `lpr`(1B), `nroff`(1), `tbl`(1)

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

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

NAME

`true`, `false` – Provides truth values

SYNOPSIS

```
true
false
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `true` utility does nothing; it is always successful. It returns 0 as an exit code. `false` does nothing; it is always unsuccessful. It returns a nonzero as an exit code.

EXIT STATUS

The `true` utility has exit status 0; `false` has nonzero.

EXAMPLES

The following standard shell script executes the loop indefinitely:

```
while true
do
    command
done
```

SEE ALSO

`sh`(1)

NAME

tset, reset – Terminal-dependent initialization

SYNOPSIS

/usr/ucb/tset [-A] [-e*c*] [-E*c*] [-I] [-k*c*] [-n] [-Q] [-r] [-s] [-S] [-m [*ident*] [*test baudrate*]:*type*] [*type*] [-]

/usr/ucb/reset [-A] [-e*c*] [-E*c*] [-I] [-k*c*] [-n] [-Q] [-r] [-s] [-S] [-m [*ident*] [*test baudrate*]:*type*] [*type*] [-]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The tset utility may be used to set up your terminal when you first log in to a UNICOS system. It performs terminal-dependent processing such as setting erase and kill characters, setting or resetting delays, and sending any sequences needed to properly initialize the terminal.

It first determines the *type* of terminal involved, and then performs necessary initializations and mode settings. Type names for terminals can be found in the terminfo(5) database. If a port is not wired permanently to a specific terminal (not hardwired), it is given an appropriate generic identifier such as dialup.

The tset utility accepts the following options:

-A Asks user for default terminal type again. Used interactively to change terminal types without logging out and in again, or rereading the .profile file (sh(1)) or the .login file (csh(1)). For example:

```
$ TERM=xterm
     ...
$ tset -A
$ TERM = (xterm)
```

This allows a user to rerun tset as if a new login has been executed.

-e*c* Sets the erase character to the named character *c* on all terminals. The default is the backspace character on the terminal, usually <CONTROL-h>. The character *c* can either be typed directly, or entered using the hat notation used here.

-E*c* Sets the erase character to *c* on all terminals except those that cannot backspace (such as a TTY 33). *c* defaults to <CONTROL-h>.

-I Suppresses transmitting terminal initialization strings.

-k*c* Sets the line kill character to the named character `c` on all terminals. *c* defaults to `<CONTROL-u>`. The kill character is left alone if `-k` is not specified. The hat notation can also be used for this option.

`-n` On systems with the Berkeley 4BSD tty driver, `-n` specifies that the new tty driver modes should be initialized for this terminal. For a CRT, the `CRTERASE` and `CRTKILL` modes are set only if the baud rate is 1200 or greater. See `tty`(4) for more details.

`-Q` Quiet. Suppresses printing the `Erase set to` and `Kill set to` messages.

`-r` Reports terminal type.

`-s` Outputs `setenv` commands for `TERM`. This can be used with the following:

```
'tset -s ...'
```

This is preferred to using the following:

```
setenv TERM 'tset - ...'
```

`-S` Similar to `-s`, but outputs two strings suitable for use in `csh .login` files as follows:

```
set noglob
set term=('tset -S .....')
setenv TERM $term[1]
setenv TERMCAP "$term[2]"
unset term
unset noglob
```

`-m` [*ident*] [*test baudrate*]:*type*]
Specifies terminal type on ports that are not hardwired.

`-` Specifies the name of the terminal finally decided upon to be output on the standard output. This is intended to be captured by the shell and placed in the `TERM` environment variable.

If you do not specify any arguments, `tset` simply reads the terminal type out of the `TERM` environment variable and reinitializes the terminal.

When used in a startup procedure (`.profile` for `sh`(1) or `.login` for `csh`(1) users) it is desirable to provide information about the type of terminal you usually use on ports that are not hardwired. These ports are identified as `dialup`, `plugboard`, or `arpanet`, and so on.

To specify what terminal type you usually use on these ports, the `-m` (map) option is followed by the appropriate port type identifier (*ident*), an optional baud rate specification (*baudrate*), and the terminal type (*type*). (The effect is to "map" from some conditions to a terminal type, that is, to tell `tset` "If I am on this kind of port, assume that I am on that kind of terminal.") If you specify more than one mapping, the first applicable mapping prevails. A missing port type identifier matches all identifiers. Any of the alternate generic names given in `terminfo` can be used for the identifier.

A *baudrate* is specified as with stty(1), and is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate *test* may be any combination of the >, @, <, and ! characters; the @ character means ''at'' and the ! character inverts the sense of the test. To avoid problems with metacharacters, it is best to place the entire argument to -m within \ characters; csh(1) users must also put a \ character before any ! character used here.

The following example maps baud rates and terminal types:

```
tset -m ´dialup>300:adm3a´ -m dialup:dw2 -m ´plugboard:?adm3a´
```

The preceding example, placed in the .profile file, causes the terminal type to be set to adm3a if the port in use is a dialup at a speed greater than 300 baud; to dw2 if the port is a dialup (that is, at 300 baud or less). If the *type* finally determined by tset begins with a question mark, you are asked if you really want that type. A null response means you want that type; otherwise, you can enter another type to be used. In this case, you will be queried on a plugboard port as to whether you are actually using an adm3a.

If no mapping applies and you specify a final *type* option (not preceded by a -m) on the command line, that type is used; otherwise, the identifier found in the environment is taken to be the terminal type. This should always be the case for hardwired ports.

It is usually desirable to return the terminal type, as finally determined by tset, and information about the terminal's capabilities to the shell environment. You can do this by using the - option.

When using the standard or Korn shell, put the following command in your .profile file:

```
eval ‘tset -s options...‘
```

When using the C shell, put the following command in your .login file:

```
eval ‘tset -s options...‘
```

Using the C shell, it is also convenient to create the following alias in your .cshrc file:

```
alias tset ’eval ‘tset -s \!*‘’
```

This alias allows the following command to be invoked at any time from your login csh.

```
tset 2621
```

Note to standard and Korn shell users: It is not possible to get this aliasing effect with a shell procedure because shell procedures cannot set the environment of their parent.

These commands cause tset to place the name of your terminal in the variable TERM in the environment.

When the terminal type is known, tset engages in terminal driver mode setting. This normally involves sending an initialization sequence to the terminal, setting the single-character erase (and optionally the line-kill (full-line erase)) characters, and setting special character delays. Tab and newline expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character (# on standard systems), the erase character is changed to backspace (<CONTROL-h>).

If `tset` is invoked as `reset`, it sets cooked and echo modes, turns off cbreak and raw modes, turns on newline translation, and restores special characters to a sensible state before any terminal-dependent processing is done. Any special character found to be NULL or –1 is reset to its default value.

This is most useful after a program terminates and leaves a terminal in an unusual state. You may have to type <LF>reset<LF> to get it to work because <CR> may not work in this state. Often none of this will echo.

EXAMPLES

The following examples are not interactive. They are lines to be included in the `.profile` file. These examples all assume the standard or Korn shell and use the `-` option. If you use `csh`(1), use one of the variations described above. A typical use of `tset` in a `.profile` or `.login` file will also use the `-e` and `-k` options, and often the `-n` or `-Q` options as well.

Example 1: Assume you are on a 2621. This is suitable for typing by hand but not for a `.profile`, unless you are always on a 2621.

```
export TERM; TERM=`tset - 2621`
```

Example 2: You have an h19 at home that you dial up on, but your office terminal is hardwired and known in `/etc/ttytype`.

```
export TERM; TERM=`tset - -m dialup:h19`
```

Example 3: You have a switch that connects everything to everything, making it nearly impossible to indicate on which port you are coming. You use a VT100 in your office at 9600 baud, and dial up to switch ports at 1200 baud from home on a 2621. Sometimes you use someone else's terminal at work, so you want it to ask you to make sure what terminal type you have at high speeds, but at 1200 baud you are always on a 2621. Note the placement of the question mark, and the quotes to protect the greater than and question mark from interpretation by the shell.

```
export TERM; TERM=`tset - -m 'switch>1200:?vt100' -m 'switch<=1200:2621''
```

Example 4: All of the above entries will fall back on the terminal type specified in the environment if none of the conditions hold. The following entry is appropriate if you always dial up at the same baud rate on different kinds of terminals. Your most common terminal is an adm3a. It always asks you what kind of terminal you are on, defaulting to adm3a.

```
export TERM; TERM=`tset - ?adm3a`
```

Example 5: If the environment is not properly initialized and you want to key entirely on the baud rate, the following can be used:

```
export TERM; TERM=`tset - -m '>1200:vt100' 2621`
```

Example 6: The following example illustrates the power of `tset`. You dial up at 1200 baud or less on a concept100, sometimes over switch ports and sometimes over regular dialups. You use various terminals at speeds higher than 1200 over switch ports, most often the terminal in your office, which is a VT100. However, sometimes you log in from the university over the ARPANET; in this case you are on an ALTO emulating a dm2500. You also often log in on various hardwired ports, such as the console, all of which are properly entered in the environment. You want your erase character set to <CONTROL-h>, your kill character set to <CONTROL-u>, and do not want `tset` to print the `Erase set to Backspace, Kill set to Control U`message. The following is to be typed as one continuous command line or enter the \ character to continue after pressing the <RETURN> key.

```
export TERM; TERM=`tset -e -k^U -Q - -m 'switch<=1200:concept100' -m\
'switch:?vt100' -m dialup:concept100 -m arpanet:dm2500`
```

FILES

`/usr/lib/terminfo`
Terminal capability database

SEE ALSO

`csh`(1), `sh`(1), `stty`(1)

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

NAME

`tsort` – Performs a topological sort

SYNOPSIS

`tsort` [*file*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `tsort` utility produces on the standard output a totally ordered list of items consistent with a partial ordering of items mentioned in the input *file*. If *file* is not specified, the standard input is used.

The input consists of pairs of items (nonempty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

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 sort using any input file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to sort using any input file subject to security label restrictions. Shell-redirected I/O is subject to security label restrictions.

Because `segldr`(1) and `ld`(1) perform multiple passes over relocatable object files, the use of `lorder`(1) and `tsort` is of little value.

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

MESSAGES

`cycle in data`	A topological sort is not possible. The pairs causing the difficulty are printed to `stderr`.
`odd data`	An odd number of fields is in the input file.

BUGS

The `tsort` utility uses a quadratic algorithm, which is not worth fixing for the typical use of ordering a library archive file.

SEE ALSO

`ar`(1), `ld`(1), `lorder`(1), `segldr`(1)

NAME

`tty` – Prints the pathname of the user's terminal

SYNOPSIS

`tty [-l] [-s]`

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-l` option)

DESCRIPTION

The `tty` utility prints the pathname of the user's terminal.

The `tty` utility accepts the following options:

`-l` Prints the synchronous line number to which the user's terminal is connected if it is on an active synchronous line.

`-s` Inhibits the printing of the terminal's pathname, allowing just the exit code to be tested.

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 `tty` utility exits with one of the following values:

0 Standard input is a terminal.

1 Standard input is not a terminal.

>1 An error occurred.

MESSAGES

`not a tty`
: The standard input is not a terminal and `-s` is not specified.

`not on an active synchronous line`
: The standard input is not a synchronous terminal and `-l` is specified.

NAME

`type` – Writes a description of a command type

SYNOPSIS

`type` *name...*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The `type` utility indicates how each argument would be interpreted if used as a command name.

The `type` utility accepts the following operands:

name Specifies a name to be interpreted.

The following environment variables affect the execution of `type`:

`LANG`	Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.
`LC_ALL`	If set to a nonempty string value, overrides the values of all the other internationalization variables.
`LC_CTYPE`	Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).
`LC_MESSAGES`	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
`NLSPATH`	Determines the location of message catalogs for the processing of `LC_MESSAGES`.
`PATH`	Determines the location of *name*, as described in the XBD specification, Chapter 6, Environment Variables.

The standard output of `type` contains information about each operand in an unspecified format. The information provided typically identifies the operand as a shell built-in, function, alias, or keyword, and where applicable, may display the operand's pathname.

Since `type` must be aware of the contents of the current shell execution environment (such as the lists of commands, functions, and built-ins processed by `hash`(1)), it is always provided as a shell regular built-in. If it is called in a separate utility execution environment, such as one of the following:

```
nohup type writer
find . -type f | xargs type
```

it might not produce accurate results.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

SEE ALSO

`command`(1)

NAME

udbsee – Writes the ASCII source of a user database

SYNOPSIS

udbsee [-g] [-l] [-q] [-s] [-u] [-v] [-o *filename*] [-p *udb_path*] [*id*]

udbsee -a [-l] [-q] [-s] [-u] [-v] [-o *filename*] [-p *udb_path*]

udbsee -d [-l] [-q] [-s] [-u] [-v] [-o *filename*] [-p *udb_path*]

udbsee -f *fields* [-l] [-q] [-u] [-e '*field op expression*'] [-m *print_mask*] [-o *filename*] [-p *udb_path*] [*id*]

udbsee -f *fields* -a [-l] [-q] [-u] [-e '*field op expression*'] [-m *print_mask*] [-o *filename*] [-p *udb_path*]

udbsee -f *fields* -d [-l] [-q] [-u] [-e '*field op expression*'] [-m *print_mask*] [-o *filename*] [-p *udb_path*]

udbsee -G [-l] [-o *filename*] [-p *udb_path*]

udbsee -h

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The udbsee utility converts information from the user database (UDB) into an ASCII file. This file presents to administrators user information in a readable form, provides a vehicle by which minor changes to user information can be made with an ordinary editor and major changes may be made with the existing UNICOS text processing filters, allows easy reconstruction of the UDB, and allows users to easily view their own control parameters. The current values for the global tape name map and the UDB defaults will be included in the output only if the -a, -d, or -g option is used.

The output of the first, second, and third forms of udbsee (see the SYNOPSIS section) is UDB source accepted by udbgen(8). The output of the fourth, fifth, and sixth forms (using *print_mask*) can be specified by users as described below.

The last form of udbsee is informational and writes the usage messages and a list of the UDB fields to the stderr file.

The general format of the udbsee display is field_name :field_value:. Generally the field names and values should be meaningful, but see udbgen(8) for definitions of names and possible values. If the # character appears anywhere except within a field value, all characters from the # to the end of line will be ignored by udbgen.

Only appropriately authorized users are shown sensitive information from the protected user database.

Options can appear in any order as long as they all appear before *filename* or *id*. Restrictions are shown in the SYNOPSIS section for the various forms of the utility.

The `udbsee` utility accepts the following options:

`-a` Displays all records in the database in ascending user identification (UID) order. This option is not allowed when `-d`, *filename*, or *id* is present. By default, only named records are displayed.

`-d` Displays all records in the database in disk order. This option is not allowed when `-a` or *id* is present. By default only named records are displayed. This option should be used instead of `-a` if the database is corrupted and you want to get as much of the source as possible for regeneration.

`-e '`*field op expression*`'`
For each entry in the UDB selected with `-a`, `-d`, or an *id* list, evaluates the *expression* argument. If the result is nonzero, the record will be selected to be printed according to the `-m` option. The *field* arguments are any of the field names listed with the `-f` option.

The *op* arguments are as follows:

`||` Binary OR. Nonzero if the left-hand side or the right-hand side evaluates to a nonzero value.

&& Binary AND. Nonzero if the left-hand side and the right-hand side are both nonzero.

== != Equal/not equal to. Nonzero if the left-hand side and the right-hand side are equal/not equal.

> < Greater/less than. Nonzero if the left-hand side is greater/less than the right-hand side.

>= <= Greater/less than or equal to. Nonzero if the left-hand side is greater/less than or equal to the right-hand side. For example, `uid >= 100 uid <= 200` would select all records with UID values in the range 100 through 200.

~ Regular expression matching. Nonzero if the string on the left-hand side matches the regular expression given by the string on the right-hand side. Regular expressions are given in the style of `ed`(1). For example, `name ~ "^[a-c].*"` would select all records with user names beginning with `a`, `b`, or `c`.

! Unary not. Nonzero if the right-hand side evaluates to zero.

".." A string of characters.

{..} A date. Date specifications are in the style `{[[[[[yy]mm]dd]hh]mm][.ss]}`. For example, `{01271200}` would be noon on the 27th of January in the current year.

(..) A subexpression.

Note that the expression may have to be quoted to stop the shell from interpreting symbols (for example, &) as symbols having special meaning.

`-f` *fields*
Field list. One or more UDB field names must be provided as comma-separated strings. If the `-m` option is not also specified, the first field name must be `create`, `delete`, or `update`. That string is first on the output line and is followed by the user name (login) within colons. Each remaining named field is output in the listed order, using the format *'name :value:'*. Each field name is separated from its predecessor by a single blank. A new line occurs at the end of the list. The special word `all` may be the terminal name in the list and causes all fields not yet written to appear in an internally specified order in the output file. (See the `-h` option for a way to view this order.) When the `-m` option is specified, no specific requirements other than the field names being recognized are imposed. In this case, the special word `all` is not recognized and field names, colons, and new-line characters are not automatically provided. Strings `create`, `delete`, `name`, and `update` always refer to the field known as `name` in the database.

`-g` Prints global tables. This option is not allowed with the `-a`, `-d`, or `-f` option. Global tables are always printed when `-a` or `-d` is used without the `-f` option.

`-G` Prints only the global tables. This option is used to print the global defaults independent of any user records. The only options allowed with the `-G` option are `-l`, `-p`, and `-o`. In addition, no *id*s can be specified on the command line.

`-h` Causes all other options to be ignored and writes to the `stderr` file a usage message showing the options and a table of the internal field names and their default conversion specifications. The order of the names in the table is the natural order in which the fields are written when the `all` field name is used. This is the same as a usage error, except that the program exits with a value of 0.

`-l` Locks the database while reading. This option is useful when a full source file is needed, but updates by other users could be going on at the same time.

`-m` *print_mask*
Prints mask (requires the `-f` option). This provides a way to control the format of the output. The print mask is presented as a format to `fprintf`(3C) when the output is written.

`-o` *filename*
Writes output to *filename*. When this option is not present, `stdout` is used.

`-p` *udb_path*
Sets the path name for access to the UDB files; the default path name is /`etc`. This option allows private or test versions of the UDB files to be created and maintained. The path name must end with a directory name; if it does not, it will be declared incorrect. For example, to specify that the UDB files are to be in `/c/abc/udbtest`, enter `-p /c/abc/udbtest` on the command line. When `-p` is specified, read access privilege to the protected database file is required.

`-q` Quiet. Suppresses informative messages and eliminates the comment lines in the normal output format.

`-s` Suppresses sensitive fields in the output. The administrator can use this to create a public version of the source. An ordinary user does not have access to sensitive fields in environments that are established as suggested. When the `-f` option is specified, this option is illegal.

`-u` Displays database access statistics.

`-v` Shows all UDB fields. If this option is not present, only fields that contain non-zero values will be shown in the output. This option presents a predictable sequence of output to simplify automated processing. On a secure system, for users who do not have security administrator privileges, security fields are printed showing zeros (meaning having no privileges), regardless of their actual authorized attributes. This is due to accessing of the public version of the UDB, which does not contain security information. When the `-f` option is specified, this option is not valid.

id Displays the UDB record belonging to *id*. *id* can be either a user name or a user ID (UID). This argument is illegal with the `-a` or `-d` option. If no record is found for the user name and the name is numeric, an attempt will be made to find a record with a UID equal to that number. Mixed user names and UIDs are allowed in *id* lists because `udbsee` evaluates the type of each ID as it is encountered in the option list. Any number of names or UIDs may be present, up to the limit of the command-line length; each name or UID is taken in order. If this option is not present, the record belonging to the login of the running process will be accessed.

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
`seeany`	Allowed to retrieve sensitive information from the protected user database.

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 retrieve sensitive information from the protected user database.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to retrieve sensitive information from the protected user database.

BUGS

If the file produced by use of the `-s` option or by a user not privileged to access the private UDB files (see the FILES section) is used to create a UDB, all sensitive fields will be set to their default values.

EXAMPLES

In the following example, assume that user `fil` is defined in the user database (user input is shown in bold):

```
$ udbsee fil
create  :fil:   uid     :64:
        comment :F. I. Ling:
        passwd  :TI..ekLKRkiZI:
        pwage   :-force, -super user,
                 4w, 1d, 703461565: #Max, min age, changed 04/16/92 16:59:25
        gids    :102,   64,     1,      2,      3,
                 14,    15,     16:
        acids   :61,    16,     14,     13,     12,
                 1:
        dir     :/w/fil:
        resgrp          :0: # uid
        jproclim[b]     :98:
        jcpulim[b]      :unlimited:
        jmemlim[b]      :unlimited:
        jfilelim[b]     :unlimited:
        pcpulim[b]      :unlimited:
        pmemlim[b]      :unlimited:
        pfilelim[b]     :unlimited:
        jproclim[i]     :98:
        jcpulim[i]      :unlimited:
        jmemlim[i]      :unlimited:
        jfilelim[i]     :unlimited:
        pcpulim[i]      :unlimited:
        pmemlim[i]      :unlimited:
        pfilelim[i]     :unlimited:
```

For definitions of the field names, and possible values for each, see udbgen(8).

To write the ASCII source of the entire database in its long form to a file named dbsource, enter the following command:

```
$ udbsee -av -o dbsource
```

The following command displays output for the UDB entry for user fil, shown previously:

```
$ udbsee -f update,dir,uid fil
update :fil: dir :/w/fil: uid :64:
```

The -e option lets you match an element in a list with an exact string. For example, if you want to select all users with a group ID (GID) of 4, specify the following:

```
$ udbsee -a -f update,gids -e'gids~"^([0-9]+,)*4(,[0-9]+)*$"'
```

FILES

`/etc/udb`	User validation file containing user control limits
`/etc/udb.public`	Public version of the user database file
`/etc/udb_2/udb.index`	Public extension file index
`/etc/udb_2/udb.priva`	Private field extension file
`/etc/udb_2/udb.pubva`	Public field extension file

SEE ALSO

`ed`(1), `privtext`(1)

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

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

`ulimit` – Sets or reports file size limit

SYNOPSIS

`ulimit` [`-f`] [*blocks*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The `ulimit` utility sets or reports the file-size writing limit imposed on files written by the shell and its child processes (files of any size may be read). Only a process with appropriate privileges can increase the limit.

The `ulimit` utility accepts the following options and operands:

`-f` Sets (or reports, if no *blocks* operand is present) the file size limit in blocks. The `-f` option is also the default case.

blocks The number of 512-blocks to use as the new file size limit.

The standard output is used when no *blocks* operand is present. If the current number of blocks is limited, the number of blocks in the current limit is written in the following format:

`"%d\n", <`*number of 512-byte blocks*`>`

If there is no current limit on the number of blocks, in the POSIX locale the following format is used:

```
"unlimited\n"
```

Since `ulimit` affects the current shell execution environment, it is always provided as a shell regular built-in. If it is called in separate utility execution environment, such as one of the following:

```
nohup ulimit -f 10000
env ulimit 10000
```

it will not affect the file size limit of the caller’s environment.

Once a limit has been decreased by a process, it cannot be increased (unless appropriate privileges are involved), even back to the original system limit.

The following environment variables affect the execution of `ulimit`:

`LANG`	Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.
`LC_ALL`	If set to a non-empty string value, override the values of all the other internationalization variables.
`LC_CTYPE`	Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).
`LC_MESSAGES`	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
`NLSPATH`	Determines the location of message catalogues for the processing of `LC_MESSAGES`.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

`sh`(1)

NAME

umask – Sets file-creation mode mask

SYNOPSIS

umask [-S] [*mask*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The user file-creation mode mask of the current shell execution environment is set to the value specified by the *mask* operand.

If you omit *mask*, the current value of the mask is printed.

The umask utility accepts the following option and operand:

-S Product symbolic output, which is in the form:

u=<*owner permissions*>, g=<*group permissions*>, o=<*other permissions*>

mask A string that specifies the new file mode creation mask. The string is treated in the same way as the *mode* operand described in chmod(2); *mask* may be represented as three octal digits or as a symbolic mode.

The octal digits refer to read, write, and execute permissions for *owner*, *group*, and *others*, respectively (see chmod(2) and umask(2)). The value of each specified octal digit is first logically complemented and then logically AND'ed with the corresponding "digit" specified by the system for the creation of a file (see creat(2)).

For a symbolic mode value, the new value of the file mode creation mask is the logical complement of the file permission bits portion of the file mode specified by the symbolic mode string.

In a symbolic mode value, the permissions *op* characters + and - are interpreted relative to the current file mode creation mask. A + character clears the bits for the indicated permissions in the mask. A - character sets the bits for the indicated permissions in the mask.

As in chmod, application use of the octal number form for the *mode* values is obsolescent.

You can include umask in your .profile file (see profile(5)) and it can be invoked at login to set automatically your permissions on files or directories created. See the NOTES section.

NOTES

The umask utility is a built-in utility to the standard shell sh(1). An executable version of this utility is available in /usr/bin/umask.

The default setting for umask is 077. You will be affected by this setting in the following ways if you do not have a umask specified in your $HOME/.profile or $HOME/.login files:

- If you assume that new files automatically have group and/or world access, you must manually set the file access permission bits to ensure the appropriate type of access is available.
- If you want a umask other than 077, you must place the umask command in your $HOME/.profile or $HOME/.login file.

EXIT STATUS

The umask utility exits with one of the following values:

0 The file mode creation mask was successfully changed, or no *mask* operand was supplied.

>0 An error occurred.

EXAMPLES

Example 1: The following command removes write permission for *group* and *others* (files usually created with mode 777 become mode 755, and files created with mode 666 become mode 644):

```
umask 022
```

SEE ALSO

chmod(1), sh(1),

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

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

NAME

unalias – Removes alias definitions

SYNOPSIS

```
unalias alias-name ...
unalias -a
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The unalias utility removes the definition for each alias name specified (see sh(1)). The aliases are removed from the current shell execution environment.

The unalias utility accepts the following option and operand:

alias-name Denotes the name of an alias to be removed.

-a Remove all alias definitions from the current shell execution environment.

NOTES

The unalias utility is a built-in utility to the standard shell (sh(1)). An executable version of this utility is available in /usr/bin/unalias.

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

EXIT STATUS

The unalias utility exits with one of the following values:

0 Successful completion.

>0 One of the *alias-name* operands specified did not represent a valid alias definition, or an error occurred.

SEE ALSO

alias(1), csh(1), sh(1)

NAME

uname – Prints name of current system

SYNOPSIS

uname [-a] [-m] [-n] [-r] [-s] [-v]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The uname utility prints the current system name on the standard output file. It is useful mainly to determine the name of the system that you are using.

The uname utility accepts the following options, which print selected information returned by uname.

-a Prints information in the following format:

system node release version hardware

-m Prints the machine's hardware name.

-n Prints the node name (the node name may be a name by which a system is known to a communications network).

-r Prints the operating system release number.

-s Prints the system name (default).

-v Prints the operating system version number.

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 uname utility exits with one of the following values:

0 The requested information was written successfully.

>0 An error occurred.

EXAMPLES

The following example prints all uname information about the system:

```
$ uname -a
snq1 snq1 UNICOS aj CRAY-YMP
```

SEE ALSO

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

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

NAME

uncompress – Expands expanded files

SYNOPSIS

```
uncompress [-c] [-f] [-v] [filename]
uncompress [-f] [-v] [filename ...]
uncompress -P
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T

DESCRIPTION

The uncompress utility will restore files to their original state after they have been compressed using the compress(1) utility. If no files are specified, the standard input wil be uncompressed to the standard output. If the invoking process has appropriate privileges, the ownership, modes, access time, and modification time of the original file are preserved.

This utility supports the uncompressing of any files produced by the compress(1) utility on the same implementation. For files produced by compress(1) on any other systems, uncompress supports 9- to 14-bit compression (see the -b option of compress(1)); it is implementation-dependent whether values of -b greater than 14 are supported.

The uncompress utility accepts the following options:

-c Writes to the standard output; no files are changed. The nondestructive behavior of zcat(1) is identical to that of specifying uncompress -c.

-f Does not prompt for overwriting files. Except when run in the background, if -f is not given, the user will be prompted as to whether an existing file should be overwritten. If the standard input is not a terminal and -f is not given, uncompress will write a diagnostic message to standard error and exit with a status greater than zero.

-v Verbose. Displays the percentage reduction for each file compressed.

-P (Only affects uncompress of input from standard input.) If -P is not specified and the input is not compressed, a message is output and the command terminates with an exit status of 1. With -P, the message is suppressed, the input is copied to the standard output, and the command terminates with an exit status of 0.

NOTES

Although compressed files are compatible between machines with large memory, `-b 12` should be used for file transfer to architectures with a small process data space (64 Kbytes or less).

The `compress` utility should be more flexible about the existence of the `.Z` suffix.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

MESSAGES

filename`: already exists; do you wish to overwrite (y or n)?`
Respond `y` if you want the output file to be replaced; `n` if not.

`uncompress: corrupt input`
A `SIGSEGV` violation was detected, which usually means that the input file is corrupted.

SEE ALSO

`compress`(1), `pack`(1), `sh`(1), `zcat`(1)

''A Technique for High Performance Data Compression,'' Terry A. Welch, *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8–19.

NAME

unget – Undoes a previous get of an SCCS file

SYNOPSIS

unget [-n] [-r *SID*] [-s] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The unget utility, used before creating the intended new delta, undoes the effect of a get -e. If a directory is named, unget behaves as though each file in the directory were specified as a named file, except that non-SCCS files and unreadable files are silently ignored. If a name of - is specified, the standard input is read with each line being taken as the name of a Source Code Control System (SCCS) file to be processed.

Options apply independently to each named file. The valid options are as follows:

-n Causes the retention of the selected file, which would normally be removed from the current directory.

-r *SID* Uniquely identifies which delta is no longer intended. (This would have been specified by get(1) as the "new delta"). The use of this option is necessary only if two or more outstanding get(1) commands for editing on the same SCCS file were done by the same person (login name). A diagnostic results if the specified *SID* is ambiguous, or if it is necessary and omitted on the command line.

-s Suppresses the printout, on the standard output, of the intended delta's *SID*.

files Specifies the file to be restored.

EXIT STATUS

The unget utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

MESSAGES

Error messages from SCCS are printed. Use help(1) for explanations.

SEE ALSO

admin(1), cdc(1), comb(1), delta(1), get(1), help(1), prs(1), rmdel(1), sact(1), sccsdiff(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

unifdef – Resolves and removes lines surrounded by #ifdef preprocessor statements

SYNOPSIS

/usr/ucb/unifdef [-c] [-l] [-t] [-D*name*] [-U*name*] [-iD*name*] [-iU*name*] [*file*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The unifdef utility removes lines surrounded by #ifdef preprocessor statements from *file*; it does not affect *file* in any other way. It deals appropriately with nested #ifdef statements, comments, and single and double quotation marks of C syntax, but it does not include or interpret macros. In addition, it does not strip out comments, although it recognizes and ignores them. You can specify which symbols you want defined by using the -D options, and which you want undefined by using the -U options. Lines within #ifdef preprocessor statements are either copied to the output or removed, as appropriate. Any #ifdef, ifndef, else, and endif statements associated with *name* are also removed.

#ifdef preprocessor statements involving symbols you did not specify remain and are copied along with their associated #ifdef, else, and endif statements.

When an #ifdef X statement is nested inside another #ifdef X statement, the inside #ifdef statement is treated as if it were an unrecognized symbol. If the same symbol appears in more than one argument, only the first occurrence will be significant.

The unifdef utility copies its output to standard output and takes its input from standard input if a file argument is not specified.

The unifdef utility accepts the following options:

-c Complements the normal operation. Lines that would have been removed or blanked are retained, and vice versa.

-l Replaces removed lines with blank lines.

-t Specifies plain text option. unifdef does not recognize comments, or single or double quotation marks.

Of the following options, one or more of each may be used, but at least one must be specified:

-D*name* Defines symbol *name*. (see previous discussion).

-U*name* Undefines symbol *name*. (see previous discussion).

-iD*name* Ignores but prints lines associated with defined symbol *name*. If you use #ifdefs to delimit non-C lines, such as comments or code under construction, you must specify which symbols are used for that purpose so that unifdef does not try to parse for quotation marks and comments within them.

`-iU`*name* Ignores but prints lines associated with undefined symbol *name*.

file Specifies the name of file to be processed.

CAUTIONS

Premature EOF signifies an inappropriate `else` or `endif` statement.

EXIT STATUS

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

0 Output is exact copy of input.

1 Output is different from input.

2 Trouble.

BUGS

The `unifdef` utility does not know how to handle `cpp`(1) constructs such as the following:

```
#if defined(X) || defined(Y)
```

EXAMPLES

The following shows the contents of `file`:

```
hello() {
#ifdef _CRAY2
        printf("Hello cray20);
#else
        printf("Hello cray10);
#endif
}
```

The following command removes the `#ifdef` preprocessor statements from `file`.

```
$ unifdef -D_CRAY2 file
```

The following shows the contents of `file` after using the `unifdef` command.

```
hello() {
       printf("Hello cray20);
}
```

SEE ALSO

`cpp`(1), `diff`(1)

NAME

uniq – Reports repeated lines in a file

SYNOPSIS

uniq [-c] [-f *fields*] [-s *chars*] [*input_file* [*output_file*]]
uniq -d [-f *fields*] [-s *chars*] [*input_file* [*output_file*]]
uniq -u [-f *fields*] [-s *chars*] [*input_file* [*output_file*]]

Obsolescent version; may not be supported in future releases:

uniq [-c] [-*n*] [+*m*] [*input_file* [*output_file*]]
uniq -d [-*n*] [+*m*] [*input_file* [*output_file*]]
uniq -u [-*n*] [+*m*] [*input_file* [*output_file*]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The uniq utility reads the input and compares adjacent lines. By default, uniq removes the second and succeeding copies of repeated lines, and the remainder is written on the output file. When you omit *input_file*, or specify *input_file* as -, the standard input is used. When you omit *output_file*, output is written to standard output. To be found, repeated lines must be adjacent (see sort(1)).

The typical mode output is the union of the -u and -d mode output.

The uniq utility accepts the following options and operands:

-c Precedes each line with the number of times it occurred.

-f *fields* Ignores the first *fields* on each input line when doing comparisons; *fields* is a decimal integer. A field is the maximal string matched by the basic regular expression:

```
[[:blank:]]*[^[:blank:]]*
```

If the *fields* argument specifies more fields than appear on an input line, a null string is used for comparison.

-s *chars* Ignores the first *chars* characters when doing comparisons. *chars* is a decimal integer. Fields are skipped before characters. If you specify in conjunction with option the -f option, the first *chars* characters after the first *fields* are ignored. If the *chars* argument specifies more characters than remain on an input line, a null string is used for comparison.

-d Suppresses the writing of lines that are not repeated in the input.

`-u`	Suppresses the writing of lines that are repeated in the input.
`-`*n*	(Obsolescent) Equivalent to `-f` *fields* with *fields* set to *n*.
`+`*m*	(Obsolescent) Equivalent to `-s` *chars* with *chars* set to *m*.
input_file	File about which to be reported.
output_file	File that contains the results.

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 *input_file* and *output_file* operands should always be different files.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

SEE ALSO

`comm`(1), `sort`(1)

NAME

units – Unit conversion program

SYNOPSIS

units

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The units utility converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

```
$ units
  You have:  inch
  You want:  cm
          *2.540000e+00
          /3.937008e-01
```

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign:

```
$ units
  You have:  15 lbs force/in2
  You want:  atm
          *1.020689e+00
          /9.797299e-01
```

The units utility does only multiplicative scale changes (for example, it can convert Kelvin to Rankine, but not Celsius to Fahrenheit). Most familiar units, abbreviations, and metric prefixes are recognized, along with the following:

pi	Ratio of circumference to diameter
c	Speed of light (meters/second)
e	Charge on an electron (coulombs)
g	Acceleration of gravity (meters/second^2)
force	Same as g
mole	Avogadro's number
water	Pressure head per unit height of water (meters/kilograms/second^2)
au	Astronomical unit (meters)

`lb` is recognized as a unit of mass; `pound` is not. Compound names are run together (for example, `lightyear`). British units that differ from their U.S. counterparts are shown with `br` as a prefix (for example, `brgallon`). For a complete list of units, enter the following:

```
cat /usr/lib/unittab
```

Press `<CONTROL-d>` to exit the program.

MESSAGES

`Conformability`	You specified mutually incompatible scales.
`Cannot recognize` *string*	Unknown keyword.

FILES

`/usr/lib/unittab`	File containing conversion table

NAME

ustat – Displays Unified Resource Manager (URM) session information

SYNOPSIS

ustat [-a] [-c *rate*] [-H *host*] [-h] [-l] [-m] [-O *category*] [-P] [-r] [-S *state*] [-s *service*] [-T *type*] [-v] [-W *reason*] [-w] [*login* [*login* ...]]

ustat [-h]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The ustat utility displays session (job) information from the Unified Resource Manager (URM), including jobs queued by the Network Queuing System (NQS).

By default, you can display information for your jobs only; only the super user can see all jobs. However, the rmgr(1) command can be used to set up a group of users as *authorized users*; these users are allowed to see each others' job information. To allow all users to see all jobs on the system, the system administrator can set the rmgr(1) configuration object /ustat/showall to 1. For more information on setting up authorized users or changing the /ustat/showall configuration object, see the rmgr(1) man page.

When invoked with no options, ustat shows a short display of job information.

The ustat utility accepts the following options and operands:

-a Displays all users. Only authorized users and the super user can see other users' jobs; see the rmgr(1) man page for more information.

-c *rate* Runs a continuous display with the specified refresh rate (in seconds). Specify *rate* as 10 or greater; any value between 1 and 9 defaults to 10 seconds.

-H *host* Specifies the host machine (local by default).

-h Displays a usage statement and online help. It contains the site-specific names defined with the user exits, if any.

-l Specifies long format for the Why field; that is, expands the brief description of the reason that the job is in the current state.

-m Displays minimum rank of a job (called *interqueue priority* in NQS). Nonbatch jobs always display the default value of 0.

-O *category* Sorts output by one of the following categories:

Category	Description
name	Sorts output by user name; each user's jobs are also sorted by time of initiation.
rank	Sorts output by job priority (URM rank).

`resource` Sorts output by resources used. The abbreviation `res` can also be used.

`type` Sorts output by session initiator type. See `-T` option for a list of valid session initiators.

`-P` Displays the priority (URM rank) of jobs.

`-r` Displays requested resources for tape, secondary data segments (SDS), and massively parallel processing (MPP), along with the requested amount, in the rightmost column of output. An asterisk next to the resource name indicates that the request is unsatisfied.

`-S` *state* Limits display to sessions (jobs) in the specified state. Valid session states include:

State	Notes
`incomplete`	
`invalid`	
`none`	Job state is displayed as `no status`
`pend_delete`	Job state is displayed as `pending delete`
`ranked`	
`registered`	
`rejected`	
`restarted`	
`running`	
`selected`	
`svchold`	Job state is displayed as `service hold`
`urm_hold`	

`-s` *service* Specifies a URM socket (`urm` by default). This option is useful only for internal testing purposes; it is not needed in normal use.

`-T` *type* Limits display to sessions (jobs) matching the specified session initiator type. The following session initiators are valid: `batch`, `ftp`, `login`, `Unknown`, `rsh`, `rexec`, `cron`, `site1`, `site2`, and `site3`.

Note: Session initiators `site1`, `site2`, and `site3` are reserved for use with the user exit capability.

`-v` Displays information on current system memory. This information appears before the list of sessions (jobs) in the following format:

```
Memory Target (Megawords):   486.36
Memory  Load  (Megawords):   534.63
```

`-W` *reason* Limits display to sessions (jobs) matching the specified reason (the brief reason displayed in the `Why` field). To display the reason in a long format, use the `-l` option. Valid reasons include the following:

Reason	Description
`availres`	The requested resource is unavailable.

`bb_max`	The job exceeds the system-wide MPP barrier resource target.
`bb_tar`	The job exceeds the per-user MPP barrier resource limit.
`chkpnt`	A checkpoint operation has been recommended.
`cpu_max`	The job exceeds the CPU resource limit.
`fifo`	The job has received an intermediate rank (fifo); this is an internal state that is rarely displayed by `ustat`.
`invalid`	The reason code is invalid; this is an internal state that is rarely displayed by `ustat`.
`job_max`	The job exceeds the maximum number of jobs allowed on the system.
`job_tar`	The target job count has been reached.
`jstate`	The job is being held while the service provider is disabled; this reason is valid for the batch resource (and site-specific resources, if enabled).
`mem_max`	The job exceeds the maximum amount of memory for the system.
`mem_tar`	The target memory usage value has been reached.
`merit`	The job has been selected for recommendation.
`none`	No reason is available for this job. Specify `-W none` to display all jobs with no associated reason code.
`pe_max`	The job exceeds the system-wide MPP processing element (PE) resource maximum limit.
`pe_tar`	The target count for the MPP PE resource has been reached.
`petime_max`	The job exceeds the MPP PE time maximum.
`preempt`	Preemption has been recommended for this job; this status results in a checkpoint or suspension operation.
`poolres`	The job cannot be started in any MPP pool.
`recstart`	A start operation has been recommended.
`restore`	A restore operation has been recommended.
`rstart`	A restart operation has been recommended.
`sds_max`	The job exceeds the SDS maximum.
`sds_tar`	The target SDS usage has been reached.

`site1_tar`
`site2_tar`
`site3_tar`
	These reasons specify site-specific status, if available. Sites can change these reasons using the user exit capability.

`start_max`	The start limit (the number of jobs initiated during one URM cycle) has been reached for this service provider.
`svc_max`	A resource request exceeds the service limit.
`tape_max`	The job's tape request exceeds the maximum amount.
`tape_tar`	The target tape usage has been reached.
`when`	A *when condition* has not occurred (deferred implementation).

`-w` Specifies wide display; shows the memory and CPU resources that have been requested and used.

login Displays information for specified user (valid only for super user unless all users have been authorized)

EXAMPLES

Example 1: You can display information about your own jobs in short format by running `ustat` with no options. Note that if `rmgr`(1) is configured to allow all users to see all jobs, this utility displays all jobs on the system. (In this case, you can use the command `ustat` *my_login* to limit the display to your own jobs.)

```
$ ustat

User      Sid   Type    Age(mn) State       Why
----      ---   ----    ------- -----       ---
crs       345   login   7.8     running
```

The fields of `ustat` output have the following meaning:

Field	Description
`User`	Displays the user ID
`Sid`	Displays the session ID assigned by the system
`Type`	Displays the type of session initiator that started the job (see the `-T` option)
`Age`	Displays the time, in minutes, that the job has been registered with URM
`State`	Displays the current state of the job (see the `-S` option)
`Why`	Displays the reason for the state of the job (see the `-W` option)

Example 2: Running `ustat` with the `-a` option displays all information that you are authorized to see. By default, users that are not the super user see only their own jobs. However, if URM is configured to allow users to see all jobs, users will see all jobs on the system, as in the following example:

```
$ ustat -a

User      Sid   Type    Age(mn) State       Why
----      ---   ----    ------- -----       ---
rkb       15    login   28.0    running
darason   166   login   16.1    running     preempt
augs      169   login   15.9    running
darason   172   login   15.7    running
n3875     106   NQS     171.3   running
tzioufas  0     NQS     7.1     urm hold    mem_tar
root      0     NQS     4.0     urm hold    mem_tar
root      0     NQS     1.2     urm hold    mem_tar
```

`ustat` displays a reason for job status in the `Why` field, if a reason is available. This includes jobs that are held before being allowed to execute, waiting for the total system memory load to drop. In example 2, the reason for the job with session ID 166 is `preempt`, which signifies that the job is being preempted (suspended).

Example 3: Authorized users can specify the following command to display verbose information for all batch jobs on the system:

```
$ ustat -a -l -T batch

User      Sid   Type    Age(mn) State       Why
----      ---   ----    ------- -----       ---
n3875     106   NQS     171.6   running
elpoole   111   NQS     59.3    running
n4074     107   NQS     1743.2  running
n3875     113   NQS     454.3   running
n3825     0     NQS     159.2   urm hold    target memory usage reached
tzioufas  0     NQS     7.5     urm hold    target memory usage reached
root      0     NQS     4.3     urm hold    target memory usage reached
root      0     NQS     1.5     urm hold    target memory usage reached
```

Example 4: Authorized users can use the following command to display information for all batch jobs (`-a` and `-T`) on the system, sorted by resource use (`-O`), with an expanded reason in the `Why` field (`-l`), including columns for memory and CPU resources (`-w`), and preceded by a summary of current system memory (`-v`).

```
$ ustat -alwv -T batch -O res

Looking on hot using service port urm at users:
        all users...

          Memory Target (Megawords):      486.36
          Memory  Load  (Megawords):      534.63

User     Sid Type Age(mn) State    ReqMem UseMem ReqCpu UseCpu Why
----     --- ---- ------- -----    ------ ------ ------ ------ ---
n4074    107 NQS  1743.5  running  215.00 211.85    360      0
n3875    113 NQS  454.6   running  128.00 121.05    240      0
n3875    106 NQS  171.9   running   64.00  42.90     60     20
orapat   105 NQS  22.1    running    0.00  40.01      0     20
orapat   109 NQS  240.1   running  128.00  39.98     60     20
elpoole  111 NQS  59.6    running   22.20  20.39     60     19
slow     108 NQS  75.3    running   20.03   6.28    500     20
n2811    110 NQS  182.5   running    8.00   2.19   1440     20
hertr    0   NQS  17.2    urm hold  16.00   0.00    360      0 target memory usage reached
tzioufas 0   NQS  7.8     urm hold  20.20   0.00    240      0 target memory usage reached
n3825    0   NQS  159.6   urm hold 100.00   0.00    403      0 target memory usage reached
root     0   NQS  4.6     urm hold   0.49   0.00      0      0 target memory usage reached
hertr    0   NQS  5.9     urm hold  16.00   0.00    360      0 target memory usage reached
```

The `-w` option adds the following columns to the `ustat` output:

Column	Description
`ReqMem`	Displays the amount of requested memory. This information is supplied by NQS (`qsub`(1)) for batch jobs; system defaults are used for other jobs.
`UseMem`	Displays the actual amount of memory used (smoothed by URM, if smoothing is enabled).
`ReqCpu`	Displays the amount of CPU time requested, in minutes.
`UseCpu`	Displays the actual amount of CPU time used, in minutes.

SEE ALSO

`qsub`(1), `rmgr`(1)

UNICOS Resource Administration, Cray Research publication SG–2302

NAME

uucp – System-to-system copy

SYNOPSIS

uucp [-cCdfjmr] [-n *user*] *source-file...* *destination-file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The uucp utility copies files named by the *source-file* arguments to the *destination-file* argument. The files named can be on local or remote systems.

The uucp utility supports the following options and operands:

-c Does not copy local files to the spool directory for transfer to the remote machine (default).

-C Forces the copy of local files to the spool directory for transfer.

-d Makes all necessary directories for the file copy (default).

-f Does not make intermediate directories for the file copy.

-j Writes the job identification string to standard output. This job identification can be used by uustat(1) to obtain the status or terminate a job.

-m Sends mail to the requester when the copy is completed.

-n *user* Notifies *user* on the remote system that a file was sent.

-r Does not start the file transfer; just queues the job.

destination-file
source-file A path name of a file to be copied to, or from, respectively. Either name can be a path name on the local machine, or can have the form:

system-name!*pathname*

where *system-name* is taken from a list of system names that uucp knows about. The destination *system-name* can also be a list of names such as:

system-name!*system-name*!...!*system-name*!*pathname*

in which case, an attempt is made to send the file via the specified route to the destination. Care should be taken to ensure that intermediate nodes in the route are willing to forward information.

The shell pattern matching notation characters `?`, `*`, and `[. . .]` appearing in *pathname* will be expanded on the appropriate system.

Path names can be one of the following:

- An obsolete path name.
- A path name preceded by *~user* where *user* is a login name on the specified system and is replaced by that user's login directory. Note that if an invalid login is specified, the default is to the public directory (called `PUBDIR`; the actual location of `PUBDIR` is implementation-specific).
- A path name preceded by *~/destination* where *destination* is appended to `PUBDIR`.

  Note: This destination will be treated as a file name unless more than one file is being transferred by this request or the destination is already a directory. To ensure that it is a directory, follow the destination with a `/`. For example, `~/dan/` as the destination will make the directory `PUBDIR/dan` if it does not exist and put the requested files in that directory.
- Anything else is prefixed by the current directory.

If the result is an erroneous path name for the remote system, the copy will fail. If the *destination-file* is a directory, the last part of the source-file name is used.

The read, write, and execute permissions given by `uucp` are implementation-dependent.

The following environment variables affect the execution of `uucp`:

`LANG` Provides a default value for the internationalization variables that are unset or null. If `LANG` is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

`LC_ALL` If set to a nonempty string value, overrides the values of all the other internationalization variables.

`LC_COLLATE` Determines the locale for the behavior of ranges, equivalence classes and multicharacter collating elements within bracketed file name patterns.

`LC_CTYPE` Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments and input files) and the behavior of character classes within bracketed file name patterns (for example, `'[[:lower:]]*'`).

`LC_MESSAGES` Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error, and informative messages written to standard output.

LC_TIME	Determines the format of date and time strings output by uucp.
NLSPATH	Determines the location of message catalogs for the processing of LC_MESSAGES.
TZ	Determines the timezone used with date and time strings.

NOTES

The domain of remotely accessible files can (and for obvious security reasons usually should) be severely restricted.

Note that the ! character in addresses has to be escaped when using csh(1) as a command interpreter because of its history substitution syntax. For ksh and sh, the escape is not necessary, but may be used.

As noted above, shell metacharacters appearing in path names are expanded on the appropriate system. On an internationalized system, this is done under the control of local setting of LC_COLLATE and LC_CTYPE. Thus, care should be taken when using bracketed file name patterns, as collation and typing rules may vary from one system to another. Also be aware that certain types of expression (that is, equivalence classes, character classes and collating symbols) need not be supported on non-internationalized systems.

The uucp utility cannot guarantee support for all character encodings in all circumstances. For example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and file names need not be portable to non-internationalized systems, and so forth. Under these circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used and that only characters defined in the Portable Filename Character Set be used for naming files.

The uucp utility exists for compliance with the XPG4 standard; therefore, it is a minimal implementation limited to execution only on the current host system.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

SEE ALSO

mailx(1), uuencode(1), uustat(1), uux(1)

NAME

uuencode, uudecode – Encodes or decodes a binary file

SYNOPSIS

uuencode [*file*] *name*

uudecode [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The uuencode and uudecode utilities transmit binary files over transmission mediums that support only ASCII data.

The uuencode utility reads *file* (or by default, the standard input) and writes an encoded version to standard output. The encoding uses only printable ASCII characters and includes the mode of the file and the *name* operand for use by uudecode.

The uudecode utility transforms encoded files (or by default, the standard input) into the original form. The file that results is named *name*; it will have the mode of the original file except that *setuid* and execute bits are not retained. The uudecode utility ignores any leading and trailing lines.

The uuencode and uudecode utilities accept the following operands:

file Specifies the name of file to be encoded or decoded.

name Specifies the name of the file that results from decoding.

EXIT STATUS

The uuencode and uudecode utilities exit with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

The encoded form of the file is expanded by 35% (3 bytes become 4, plus control information).

EXAMPLES

The following example packages up a source tree, compresses it, encodes it, and mails it to a user on another system. When `uudecode` is run on the target system, the `src_tree.tar.Z` file will be created, which may then be uncompressed and extracted into the original tree.

```
tar cf – src_tree | compress | uuencode src_tree.tar.Z | mail joe
```

SEE ALSO

`compress`(1), `mail`(1)

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

NAME

uustat – uucp status inquiry and job control

SYNOPSIS

uustat [-q]

uustat [-k *jobid*]

uustat [-r *jobid*]

uustat [-s *system*] [-u *user*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The uustat utility displays the status of, or cancels, previously specified uucp requests, or provides general status on uucp connections to other systems. When no options are given, uustat writes to standard output the status of all uucp requests issued by the current user.

The uustat utility supports the following options:

-q Writes the jobs queued for each machine.

-k *jobid* Kills the uucp request whose job identification is *jobid*. The killed uucp request must belong to the person invoking uustat unless that user has appropriate privileges.

-r *jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that the modification time is et to the current time. This prevents the cleanup program from deleting the job until the jobs modification time reaches the limit imposed by the program.

-s *system* Writes the status of all uucp requests for system *system*.

-u *user* Writes the status of all uucp requests issued by *user*.

The following environment variables affect the execution of uustat:

LANG Provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

LC_ALL If set to a nonempty string value, overrides the values of all the other internationalization variables.

LC_CTYPE	Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).
LC_MESSAGES	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error, and informative messages written to standard output.
LC_TIME	Determines the format of date and time strings output by uustat.
NLSPATH	Determines the location of message catalogs for the processing of LC_MESSAGES.
TZ	Determines the timezone used with date and time strings.

NOTES

The uustat utility exists for compliance with the XPG4 standard; therefore, it is a minimal implementation limited to execution only on the current host system.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

uucp(1)

NAME

uux – System-to-system copy

SYNOPSIS

uux [-np] *command-string*

uux [-jnpf7] *command-string*

uux [-] [-jn] *command-string*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The uux utility will gather zero or more files from various systems, execute a shell pipeline on a specified system, and then send the standard output of the command to a file on a specified system. Only the first command of a pipeline can have a system-name! prefix. All other commands in the pipeline are executed on the system of the first command.

The following restrictions are applicable to the shell pipeline processed by uux:

- In gathering files from different systems, path name expansion is not performed by uux. Thus, a request such as:

  ```
  uux "c89 remsys!~/*.c"
  ```

  would attempt to copy the file named literally *.c to the local system.
- The redirection operators >>, <<, >|, and >& cannot be used.
- The reserved word ! cannot be used at the head of the pipeline to modify the exit status.
- Alias substitution is not performed.

A file name can be specified as for uucp(1); it can be obsolete path name, a path name preceded by ~*name* (which is replaced by the corresponding login directory), a path name specified as ~/*dest* (*dest* is prefixed by the public directory called PUBDIR; the actual location of PUBDIR is implementation-specific), or a simple file name (which is prefixed by uux with the current directory). See uucp(1) for the details.

The execution of commands on remote systems takes place in an execution directory known to the uucp system. All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the nonlocal file names (without path or machine reference) must be unique within the uux request.

The uux utility will attempt to get all files to the execution system. For files that are output files, the file name must be escaped using parentheses.

The remote system will notify the user by mail if the requested command on the remote system was disallowed or the files were not accessible. This notification can be turned off by the -n option.

The uux utility supports the following options and operands:

-p
- Makes the standard input to uux the standard input to the *command-string*.

-j Writes the job identification string to standard output. This job identification can be used by uustat(1) to obtain the status or terminate a job.

-n Does not notify the user if the command fails.

command-string
A string made up of one or more arguments that are similar to normal command arguments, except that the command and any file names can be prefixed by *system-name!*. A null *system-name* is interpreted as the local system.

The following environment variables affect the execution of uux:

LANG Provides a default value for the internationalization variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.

LC_ALL If set to a nonempty string value, overrides the values of all the other internationalization variables.

LC_CTYPE Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multibyte characters in arguments).

LC_MESSAGES Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

NLSPATH Determines the location of message catalogs for the processing of LC_MESSAGES.

The standard output is not used unless the -j option is specified; in that case, the job identification string is written to standard output in the following format:

"%s\n", <*jobid*>

Output files are created or written, or both, according to the contents of *command-string*.

If the -n is not used, mail files will be modified following any command or file-access failures on the remote system.

NOTES

For security reasons, many installations limit the list of commands executable on behalf of an incoming request from `uux`. Many sites permit little more than the receipt of mail via `uux`.

Any characters special to the command interpreter should be quoted either by quoting the entire *command-string* or quoting the special characters as individual arguments.

Typical implementations of this utility require a communications line configured to use the XBD specification, Chapter 9, General Terminal Interface, but other communications means may be used. On systems where there are no available communications means (either temporarily or permanently), this utility will write an error message describing the problem and exit with a nonzero exit status.

As noted in `uucp`(1), shell pattern matching notation characters appearing in path names are expanded on the appropriate local system. This is done under the control of local settings of `LC_COLLATE` and `LC_CTYPE`. Thus, care should be taken when using bracketed file name patterns, as collation and typing rules may vary from one system to another. Also be aware that certain types of expression (that is, equivalence classes, character classes and collating symbols) need not be supported on non-internationalized systems.

The `uux` utility cannot guarantee support for all character encodings in all circumstances. For example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and file names need not be portable to non-internationalized systems, and so forth. Under these circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used and that only characters defined in the Portable Filename Character Set be used for naming files.

The `uux` utility exists for compliance with the XPG4 standard; therefore, it is a minimal implementation limited to execution only on the current host system.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following command gets *file1* from system `a` and *file2* file from system `b`, executes `diff` on the local system, and puts the results in `file.diff` in the local `PUBDIR` directory. (`PUBDIR` is the `uucp` public directory on the local system.)

```
uus "!diff a!/usr/file1 b!/a4/file2 >!~/file.diff"
```

Example 2: The following command will fail because `uux` places all files copied to a system in the same working directory. Although the files `xyz` are from two different systems, their file names are the same and will conflict.

```
uux "!diff a!/usr1/xyz b!/user2/xyz >!~/xyz.diff"
```

Example 3: The following command will succeed (assuming `diff` is permitted on system `a`) because the file local to system `a` is not copied to the working directory, and hence does not conflict with the file from system `c`:

```
uux "a!diff a!/usr/xyz c!/usr/xyz >!~/xyz.diff"
```

SEE ALSO

uucp(1), uuencode(1), uustat(1)

NAME

`val` – Validates SCCS file

SYNOPSIS

```
val -
val [-m name] [-r SID] [-s] [-y type] files
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The `val` utility determines whether the specified *files* are Source Code Control System (SCCS) files meeting the characteristics specified by the option list. Options to `val` may appear in any order. The options consist of keyletter arguments, which begin with a `-` symbol, and named files.

The `-` argument causes reading of the standard input until an end-of-file condition is detected. Each line read is independently processed, as if it were a command line argument list.

The `val` utility generates diagnostic messages on the standard output for each command line and file processed, and it also returns a single 8-bit code upon exit.

The options are defined as follows. The effects of any option apply independently to each named file on the command line.

`-m` *name* The argument value *name* is compared with the SCCS `man/man1/val.1` keyword in *files*.

`-r` *SID* An SCCS delta number. A check is made to determine whether the source identifier (SID) is ambiguous (such as `r1 1` is ambiguous because it physically does not exist, but it implies `1.1`, `1.2`, and so on, which may exist) or invalid (such as, `r1.0` or `r1.1.0` are invalid because neither case can exist as a valid delta number). If the *SID* is valid and not ambiguous, a check is made to determine if it actually exists.

`-s` Silences the diagnostic message normally generated on the standard output for any error that is detected while processing each named file on a given command line.

`-y` *type* The argument value *type* is compared with the SCCS `%Y%` keyword in *files*.

files Specifies the files to be validated.

EXIT STATUS

The 8-bit code returned by `val` is a disjunction of the possible errors; that is, it can be interpreted as a bit string with set bits interpreted as follows (moving from left to right):

Bit 0 = Missing file argument
Bit 1 = Unknown or duplicate keyletter argument
Bit 2 = Corrupted SCCS file
Bit 3 = Cannot open file or file is not SCCS
Bit 4 = *SID* is invalid or ambiguous
Bit 5 = *SID* does not exist
Bit 6 = `%Y%`, `-y` mismatch
Bit 7 = `%M%`, `-m` mismatch

Note that `val` can process two or more files on a given command line, and in turn can process multiple command lines (when reading the standard input). In these cases, an aggregate code is returned - a logical `OR` of the codes generated for each command line and file processed.

MESSAGES

Messages from the `val` utility do not contain the SCCS help codes. The messages are meant to be self-explanatory.

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `vc`(1), `what`(1)

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

NAME

vc – SCCS version control

SYNOPSIS

vc [-a] [-c *char*] [-s] [-t] [*args*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The vc utility copies lines from the standard input to the standard output under control of its arguments and control statements encountered in the standard input. In the process of performing the copy operation, user-declared *keywords* may be replaced by their string *value* when they appear in plain text and/or control statements.

The copying of lines from the standard input to the standard output is conditional, based on tests (in control statements) of keyword values specified in control statements or as vc utility arguments.

A control statement is a single line beginning with a control character, except as modified by the -t option. The default control character is colon (:), except as modified by the -c option. Input lines beginning with a backslash (\), followed by a control character are not control lines and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a noncontrol character are copied in their entirety.

A keyword is composed of 9 or fewer alphanumeric characters; the first character must be alphabetic. A value is any ASCII string that can be created with ed(1); a numeric value is an unsigned string of digits. Keyword values may not contain blanks or tabs.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The -a option forces replacement of keywords in all lines of text. An uninterpreted control character may be included in a value by preceding it with a \ symbol. If a literal \ is desired, it too must be preceded by a \ symbol.

The vc utility accepts the following options:

-a Forces replacement of keywords surrounded by control characters with their assigned value in all text lines and not just in vc statements.

-c *char* Specifies a control character to be used in place of the colon (:).

-s Silences warning messages (not error messages) that are usually printed on the diagnostic output.

-t Ignores all characters from the beginning of a line up to and including the first tab character for the purpose of detecting a control statement. If one is found, discards all characters up to and including the tab.

args *args* takes the form [*keyword=value* . . . *keyword=value*].

Version Control Statements

`:dcl` *keyword*[, . . ., *keyword*]
Declares keywords. All keywords must be declared.

`:asg` *keyword=value*
Assigns values to keywords. An `asg` statement overrides the assignment for the corresponding keyword on the `vc` command line and all previous `asg` statements for that keyword. Keywords declared, but not assigned values, have null values.

`:if` *condition*
.
.
.
`:end`
Skips lines of the standard input. If *condition* is true, all lines between the `if` statement and the matching `end` statement are copied to the standard output. If *condition* is false, all intervening lines are discarded, including control statements. Intervening `if` statements and matching `end` statements are recognized solely for the purpose of maintaining the proper `if-end` matching.

The syntax of *condition* is as follows:

```
<cond>      ::= [ "not" ] <or>
<or>        ::= <and> | <and> "|" <or>
<and>       ::= <exp> | <exp> "&" <and>
<exp>       ::= "(" <or> ")" | <value> <op> <value>
<op>        ::= "=" | "!=" | "<" | ">"
<value>     ::= <arbitrary ASCII string> | <numeric string>
```

The available operators and their meanings are as follows:

`=`	Equal
`!=`	Not equal
`&`	And
`|`	Or
`>`	Greater than
`<`	Less than
`( )`	Used for logical groupings
`not`	May only occur immediately after the *if*, and, when present, inverts the value of the entire condition

The `>` and `<` operate on only unsigned integer values (such as, `: 012 > 12` is false). All other operators take strings as arguments (such as, `: 012 != 12` is true). The precedence of the operators (from highest to lowest) is as follows:

`= != > <`	all of equal precedence
`&`	
`|`	

Parentheses may be used to alter the order of precedence.

Values must be separated from operators or parentheses by at least one blank or tab.

`::text` Used for keyword replacement on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the `-a` option.

`:on`
`:off` Turns on or off keyword replacement on all lines.

`:ctl` *char* Changes the control character to *char*.

`:msg` *message* Prints the given message on the diagnostic output.

`:err` *message* Prints the given message followed by the following on the diagnostic output:

```
ERROR: err statement on line ... (915)
```

The `vc` utility halts execution, and returns an exit code of 1.

EXIT STATUS

The `vc` utility returns an exit code of 0 on normal termination and returns a 1 if any error occurs.

MESSAGES

Use `help`(1) for explanations.

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `ed`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `what`(1)

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

NAME

vi, view, vedit – Invokes the screen-oriented (visual) display editor

SYNOPSIS

vi [-C] [-r] [-R] [-l] [-L] [-t *tag*] [-V] [-w *n*] [-x] [-c *command*] [*file* ...]

Obsolescent version:

vi [-C] [-r] [-R] [-l] [-L] [-t *tag*] [-V] [-w *n*] [-x] [+*command*] [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-C, -L, -V, and -x options)

DESCRIPTION

The vi utilty may be involved as view or vedit. The vi (visual) editor is a display-oriented text editor based on an underlying line editor (ex(1)). You can use the command mode of ex(1) from within vi and vice versa.

When using vi, changes you make to the file are reflected on your terminal screen. The position of the cursor on the screen indicates the position within the file.

The vi editor accepts the following options:

-c *command*
+*command* (Obsolescent)
: Interprets the specified ex(1) command-mode *command* before editing begins. When used with the -t option, the command is executed after moving to the tag.

-C : Same as -x but assume *file* began in encrypted form.

-l : Set lisp mode.

-L : List filenames that were saved due to an editor or system crash.

-r : Recovers *file* after an editor or system crash. If you omit *file*, a list of all saved files is printed.

-R : Read-only mode; the readonly flag is set, preventing accidental overwriting of the file.

-t *tag* : Edits the file that contains the *tag* and positions the editor at its definition.

-V : Verbose. Any non-tty input will be echoed on standard error.

-w *n* : Sets the default window size to *n*. This is useful when using the editor over a slow-speed line.

`-x` Encrypts the file as it is written and requires an encryption key that allows it to be read (see `crypt`(1)). See the WARNINGS section for more information.

The `view` utility is the same as `vi`, except that the `readonly` flag is set.

The `vedit` utility is intended for beginners. The `report` flag is set to 1, and the `showmode` and `novice` flags are set. These defaults make it easier to get started learning the editor.

vi Modes

At any one time, you are in one of the following modes in `vi`:

Command — Normal and initial mode. Other modes return to command mode on completion. To cancel a partial command, press the <ESCAPE> key.

Input — Entered by typing `a i A I o O c C s S R`. Text can then be entered. Input mode is normally terminated with the <ESCAPE> key or abnormally with an interrupt.

Last line — Reading input for `: / ?` or `!`; terminate by pressing the <RETURN> key to execute, and interrupt to cancel.

Command Summary

The following is a summary of commands `vi` accepts.

Sample Commands

← ↓ ↑ →

`h j k l`	Same as arrow keys
`i`*text*<ESCAPE>	Inserts *text*
`cw`*new*<ESCAPE>	Changes word to *new*
`ea`*s*<ESCAPE>	Changes word to plural form
`x`	Deletes a character
`dw`	Deletes a word
`dd`	Deletes a line
`3dd`	Deletes three lines
`u`	Undoes previous change to buffer
`ZZ`	Exits `vi`, saving changes
`:q!`<RETURN>	Quits, discarding changes
`/`*text*<RETURN>	Searches for *text*
<CONTROL-u>	Scrolls up
<CONTROL-d>	Scrolls down
`:`*ex cmd*<CR>	Enters any `ex`(1) or `ed`(1) command

Numbers Before vi Commands

You can type numbers as a prefix to some commands. They are interpreted as follows:

Column/line number	`z G`
Scroll amount	`^D ^U`
Repeat effect	Most commands accept numbers to indicate how many times to repeat

Interrupting and Canceling

<ESCAPE>	Ends insert or incomplete command
DELETE	(Delete or rubout) interrupts
^L	Reprints screen if DEL scrambles it
^R	Reprints screen if <CONTROL-l> is → key

File Manipulation

:w<CR>	Writes changes
:q<CR>	Quits
:q!<CR>	Quits, discard changes
:e *name*<CR>	Edits file *name*
:e!<CR>	Reedits, discarding changes
:e + *name*<CR>	Edits, starting at end
:e +*n*<CR>	Edits starting at line *n*
:e #<CR>	Edits alternate file
^6	Synonym for :e #
:w *name*<CR>	Writes file *name*
:w! *name*<CR>	Overwrites file *name*
:sh<CR>	Runs shell; uses exit to return
:!*cmd*<CR>	Runs *cmd*, then returns
:n<CR>	Edits next file in arglist
:n *args*<CR>	Specifies new arglist
^G	Shows current file and line
:ta *tag*<CR>	Positions cursor at tag and saves previous position on the tag stack (see ctags(1))
^]	:ta, following word is *tag*
:pop<CR>	Returns to previous saved position on the tag stack
^T	Same as :pop command

In general, any ex(1) or ed(1) command (such as substitute or global) may be typed, preceded by a colon (:) and followed by a carriage return (<CR>).

Positioning Within File

^F	Forward screen
^B	Backward screen
^D	Scrolls down half screen
^U	Scrolls up half screen
G	Goes to specified line (default last line)
/*pat*	Next line matching *pat*
?*pat*	Previous line matching *pat*
n	Repeats last / or ?
N	Reverses last / or ?
/*pat*/+*n*	*n*th line after *pat*
?*pat*?-*n*	*n*th line before *pat*
]]	Next section/function
[[	Previous section/function

(	Beginning of sentence
)	End of sentence
{	Beginning of paragraph
}	End of paragraph
%	Finds matching ( ) { or }

Adjusting the Screen

^L	Clears and redraws screen
^R	Retypes, eliminating @ lines
z<CR>	Redraws, current line at window top
z-<CR>	Redraws, current line at bottom of window
z .<CR>	Redraws, current line at center of window
/*pat*/z-<CR>	Redraws *pat* line at bottom
z*n* .<CR>	Uses *n*-line window
^E	Scrolls window down a line
^Y	Scrolls window up a line

Marking and Returning

``	Moves cursor to previous context
´´	Moves cursor to first nonwhite in line
m*x*	Marks current position with letter *x*
`*x*	Moves cursor to mark *x*
´*x*	Moves cursor to first nonwhite in line marked with *x*

Line Positioning

H	Goes to top line on screen
L	Goes to last line on screen
M	Goes to middle line on screen
+	Goes to next line, at first nonwhite
-	Goes to previous line, at first nonwhite
<CR>	Same as +
↓ or j	Goes to next line, same column
↑ or k	Goes to previous line, same column

Character Positioning

^	Goes to first nonwhite character
0	Goes to beginning of line
$	Goes to end of line
h or →	Goes backward
l or ←	Goes forward
^H	Same as ←
space	Same as →
f*x*	Finds *x* forward in line
F*x*	Finds f backward in line
t*x*	Up to *x* forward in line

T*x*	Back up to *x* in line
;	Repeats last f F t or T
,	Inverse of ;
|	Goes to specified column
%	Finds matching ({) or }

Words, Sentences, and Paragraphs

w	Word forward
b	Word backward
e	End of word
)	To next sentence
}	To next paragraph
(	Back sentence
{	Back paragraph
W	Blank delimited word
B	Back W
E	To end of W

Corrections during Insert

^H	Erases last character
^W	Erases last word
erase	Your erase, same as ^H
kill	Your kill, erase input this line
\	Quotes ^H, your erase and kill
<ESC>	Ends insertion, goes back to command
DEL	Interrupts, terminates insertion
^D	Uses backtab over *autoindent*
↑^D	Kills *autoindent*, saves for next
0^D	Same as ↑^D, but at margin next also
^V	Quotes nonprinting character
^T	Inserts shiftwidth spaces

Insert and Replace

a	Appends after cursor
i	Inserts before cursor
A	Appends at end of line
I	Inserts before first non blank character
o	Opens line below current line
O	Opens line above current line
r*x*	Replaces single character with *x*
R*text*<ESC>	Replaces characters

Operators

Operators are followed by a cursor motion and affect all text that would have been moved over. For example, because `w` moves over a word, `dw` deletes the word that would be moved over. Double the operator (for example, `dd`) to affect whole lines.

`d`	Deletes
`c`	Changes
`y`	Yanks lines to buffer
`<`	Left shifts
`>`	Right shifts
`!`	Filters through command
`=`	Indents for LISP

Miscellaneous Operations

`C`	Changes rest of line (`c$`)
`D`	Deletes rest of line (`d$`)
`s`	Substitutes characters (`cl`)
`S`	Substitutes lines (`cc`)
`J`	Joins lines
`x`	Deletes characters (`dl`)
`X`	Deletes characters before cursor (`dh`)
`Y`	Yanks lines (`yy`)

Yank and Put

The put operator inserts the text most recently deleted or yanked; however, if a buffer is specified, the text in that buffer is inserted instead.

`p`	Puts text after cursor
`P`	Puts text before cursor
`"`*x*`p`	Puts text from buffer *x*
`"`*x*`y`	Yanks text to buffer *x*
`"`*x*`d`	Puts text into buffer *x*

Undo, Redo, and Retrieve

`u`	Undoes last change to buffer
`U`	Restores current line
`.`	Repeats last change
`"`*d*`p`	Retrieves *d*'th last delete

Regular Expressions

The `vi` editor recognizes the following regular expressions:

[*cccc*]	Matches any of the specified characters (*cccc*)
[^*cccc*]	Matches any character (*cccc*) except those specified
[*c1*-*c2*]	Matches any character in the specified range (*c1* through *c2*)
^	Matches the beginning of a line

^*cccc*	Matches lines that begin with characters *cccc*
$	Matches the end of a line
cccc$	Matches lines that end with characters *cccc*
.	Matches any one character
*	Matches zero or more occurrences of the preceding character
.*	Matches any number of characters
\{#\}	Matches # occurrences of preceding search string
\{#,\}	Matches at least # occurrences of preceding search string
\{*#1*,*#2*\}	Matches between *#1* and *#2* occurrences of preceding search string
\(*string*\)\#	Matches *string* followed by # occurrences of *string*

To match special characters ($. * [ ] ^ \), precede the special character with a backslash (\).

Set Command and .exrc File

You can set defaults for a variety of editing characteristics while editing with vi. To set a characteristic for a particular editing session, enter the set command as follows:

:set [*option* [=*value*]]

(Specific set options follow.)

If you want to set new defaults for all editing sessions, insert the appropriate set command (without the leading :) into a file called .exrc. The editor searches for .exrc in your home directory first and executes the commands in that file. Then it searches for the .exrc file in your current directory and executes the commands in it; the .exrc file is skipped if you do not own the file. (See exrc(5).)

You can also use the EXINIT environment variable to set these defaults; if this environment variable is set, the $HOME/.exrc file will not be processed, but the .exrc file in your current directory will be processed.

To display set options you have changed, enter the set command without options. :set all shows the state of all set options. :set *x* enables the *x* option. :set no*x* disables the *x* option. :set *x*=*value* gives the *x* option the value of *value*. :set *x*? shows the value of the *x* option.

The set options are as follows (the alias, if one exists, is in parentheses):

autoindent (ai)	Continues previous indentation
autoprint (ap)	Prints current line after buffer changed
autowrite (aw)	Writes before changing files
beautify (bf)	Discards all control characters other than tab, new line, and form feed
directory (dir)	Specifies the directory
edcompatible (ed)	Causes suffixes to be remembered

errorbells (eb)	Sounds bell when error occurs
exrc (ex)	Allows execution of .exrc files that reside outside usr's home directory
flash	Flashes screen on errors
hardtabs	Value of any hardware tab settings
ignorecase (ic)	Ignores case when scanning
lisp	() {} are s-expressions
list	Displays ^I for tab, $ at end of line
magic	Turns on the normal metacharacter meaning of ., [, *
mesg	Allows non-vi messages to appear on screen during vi
modelines (ml)	Prevents accidental interpretation of ex: and vi: in files
novice	Makes it easier to learn to use vi
number (nu)	Numbers lines
optimize (opt)	Abolishes carriage returns at the end of lines when printing multiple lines (speeds output on dumb terminals when printing lines with leading white space)
paragraphs (para)	Macro names that start...
posix92 (px92)	Editing behavior is that which conforms to the P1003.2 standard
prompt	When set, prompted with a ":"
readonly	Disallows writing buffer contents to current file name
redraw	Redraws the screen
remap	Macro translation
report	Sets the threshold for the number of lines modified
scroll	Command mode lines
sections (sect)	Macro names...
shell	Shell executed when doing :! or !
shiftwidth (sw)	Gives the width of a software tab stop used in reverse tabbing
showmatch (sm)	Shows the match to) and { when typed
showmode (smd)	Shows insert mode in vi
slowopen (slow)	Stops updates during insert
tabstop (ts)	Software tab stops
taglength	Tag names (labels) only significant to this many characters

`tags`	List of files to be searched by the `:tag` command
`term`	Type of terminal you are using
`terse`	Error messages are shorter
`timeout`	Must enter a macroname in less than one second
`ttytype` (`tty`)	Sets terminal type (same as `set term`)
`warn`	Warning given when `:!` is issued
`window`	Visual mode lines
`wrapscan` (`ws`)	Searches that use regular expressions will wrap around past EOF
`wrapmargin` (`wm`)	Automatically splits line *n* characters from right (breaks at white space)
`writeany` (`wa`)	Allows a write to any file

CAUTIONS

If your terminal definition is not supported through `TERMINFO=/usr/lib/terminfo`, you may need to define your terminal.

WARNINGS

Inclusion of the Data Encryption Standard (DES) encryption code requires a special license for sites outside the United States and Canada. If these encryption functions are not available on your system, check with your system administrator or site analyst.

Tampering with entries in `/usr/lib/terminfo/?/*` (for example, changing or removing an entry) can affect programs such as `vi` that expect the entry to be present and correct. In particular, removing the dumb terminal can cause unexpected problems.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

BUGS

Software tabs in insert mode using `^T` work only immediately after the `autoindent`.

On intelligent terminals, left and right shifts do not make use of insert and delete character operations in the terminal.

Minor display glitches can occur under certain circumstances, such as screen wrapping while overwriting characters with the `R` command.

The `ye` sequence yanks a word into the buffer, but it often misses the last character.

FILES

`/usr/lib/terminfo/?/*` Default terminal information database

SEE ALSO

`awk`(1) `crypt`(1), `ctags`(1), `ed`(1), `edit`(1), `ex`(1), `grep`(1), `sed`(1)

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

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

Learning the vi Editor, Linda Lamb, O'Reilly & Associates, Inc., 1990

NAME

`wait` – Waits for completion of a process

SYNOPSIS

`wait` [*pid*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `wait` utility instructs the system to wait for your background process that has a process ID of *n* and to report its termination status.

If you omit *n*, the system waits for all of the invoking shell's currently active background processes, and exists with an exit status of 0.

The *pid* operand is one of the following:

- An unsigned decimal integer process ID of a command for which `wait` is to wait for the termination.
- A job control job ID that identifies a background process group to be waited for. This notation applies only to invocations of `wait` in the current shell execution environment.

The shell itself executes `wait`, without creating a new process.

The known process IDs apply only for invocations of `wait` in the current shell execution environment.

NOTES

The `wait` utility is a built-in utility to the standard shell (`sh`(1)). An executable version of this utility is available in `/usr/bin/wait`.

The `csh`(1) utility has a built-in `wait` utility with slightly different characteristics. See `csh`(1).

EXIT STATUS

If one or more operands were specified, all of them have terminated or were not known by the invoking shell, and the status of the last operand specified is known, then the exit status of `wait` is the exit status information of the command indicated by the last operand specified. If the process terminated abnormally due to the receipt of a signal, the exit status is greater than 128 and is distinct from the exit status generated by the other signals. Otherwise, the `wait` utility exits with one of the following values:

0 The `wait` utility was invoked with no operands and all process IDs known by the invoking shell have terminated.

1-126 The `wait` utility detected an error.

127 The command identified by the last *pid* operand specified is unknown.

SEE ALSO

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

NAME

wc – Counts bytes, characters, lines, and words in a file

SYNOPSIS

```
wc [-c] [-l] [-w] [files]
wc -m [-l] [-w] [files]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The wc utility counts bytes, characters, lines, and words in the specified files or in the standard input if you omit *files*. It also keeps a total count for all specified files. *word* is a maximal string of characters delimited by <space>s, <tab>s, or <newline> characters.

The wc utility accepts the following options:

-c Writes to the standard output the number of bytes in each input file.

-l Writes to the standard output the number of <newline> characters in each input file.

-m Writes to the standard output the number of characters in each input file.

-w Writes to the standard output the number of words in each input file.

files Specifies the files whose content are to be counted.

You can use the -l and -w options in any combination with the -c or -m option to specify that a subset of lines and words will be reported along with bytes or characters, respectively. By default bytes, words, and lines are printed.

When you specify *files* on the command line, they are printed along with the counts.

In addition to being useful for determining the size of a text file, wc can also be useful for determining the number of items displayed by other UNICOS commands.

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.

Because multibyte characters are not supported in the current UNICOS release, the `-m` option assumes 1 byte per character.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: In this example, the output of `who`(1) is piped to `wc`. `who` shows who is logged in to the system, displaying one line per user. By counting the number of lines, `wc` gives a count of how many people are logged into the system. The first number displayed by `wc` is line count, which corresponds to the number of users logged in.

```
who | wc
```

Example 2: A more sophisticated example follows:

```
who | wc | awk '{print "Number of Users:",$1}'
```

NAME

what – Identifies SCCS files and UNICOS Source Manager (USM) files

SYNOPSIS

what [-s] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The what utility searches the specified files for all occurrences of the pattern that get(1) substitutes for %Z% (this is @(#)) and prints out what follows until the first ~, >, new-line, \, or null character. If no *files* are specified, what searches standard input.

The what utility is intended to be used in conjunction with the Source Code Control System (SCCS) utility get(1), which automatically inserts identifying information, but it can also be used where the information is inserted manually.

The what utility accepts the following option:

-s Quits after finding the first occurrence of pattern in each file.

files Specifies the files to be identified.

EXIT STATUS

The what utility exits with one of the following values:

0 Matches were found.

>0 An error occurred.

BUGS

It is possible that an unintended occurrence of the pattern @(#) could be found, but this usually causes no harm.

EXAMPLES

If the C program in file `f.c` contains

```
char ident[] = "@(#)identification information";
```

and `f.c` is compiled to yield `f.o` and `a.out`, the command

```
$ what f.c f.o a.out
```

will write

```
f.c:
        identification information
        ...

f.o:
        identification information
        ...

a.out:
        identification information
        ...
```

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1)

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

NAME

whatis – Displays a one-line summary about a command

SYNOPSIS

whatis *command* ...

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whatis utility displays the header line of *command* from the manual section. You can then run the man(1) command to get more information. If the line starts with *name*(*section*) ..., you can enter man *section name* to display the documentation for it.

The function of the whatis utility is identical to the -f option of the man(1) command.

The whatis utility displays information from the /usr/man/whatis file.

FILES

/usr/man/whatis Database

SEE ALSO

apropos(1), man(1)

NAME

whereis – Locates source, binary, and/or manual for program

SYNOPSIS

/usr/ucb/whereis [-s] [-b] [-m] [-u] [[-S *dirs*] [-B *dirs*] [-M *dirs*] -f] *names*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whereis utility locates source, binary, and/or manual sections for specified files. The supplied names are first stripped of leading path name components and any (single) trailing extension of the form *file*.*ext* (for example .c). Prefixes of s. that result from use in source code control are also dealt with. The whereis utility then attempts to locate the desired program in a list of standard places.

The whereis utility accepts the following options:

-s Searches only for sources.

-b Searches only for binaries.

-m Searches only for manual sections.

-u Searches for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus, whereis -m -u * requests those files in the current directory that have no documentation.

-S *dirs* Changes or otherwise limits the places where whereis searches for sources to *dirs*.

-B *dirs* Changes or otherwise limits the places where whereis searches for binaries to *dirs*.

-M *dirs* Changes or otherwise limits the places where whereis searches for manual sections to *dirs*.

-f Terminates the last such directory list and signals the start of file names.

names Specifies file names to be located.

BUGS

Because the program uses chdir(2) to run faster, path names specified with -M, -S, and -B must be full path names; that is, they must begin with a /.

EXAMPLES

The following commands find all the files in /usr/bin that are not documented in /usr/man/cat1 with source in /usr/src/cmd:

```
cd /usr/bin
whereis -u -m -M /usr/man/cat1 -S /usr/src/cmd -f *
```

FILES

`/usr/src/*`	Source code for commands
`/usr/man/*`	Text files for manuals
`/lib`, `/etc`, `/bin`, `/usr/bin`, `/usr/ucb`, `/usr/lbin`	Executable binaries of commands

SEE ALSO

`man`(1)

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

NAME

whichcat – Returns the name of the message system catalog being accessed

SYNOPSIS

whichcat [-l] *group* [*groups*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whichcat utility returns the path name that contains the message system catalog for the group code(s) *group*. This command verifies that the expected catalog is being referenced or helps determine why no catalog is found.

If no options are specified, only the path name where the catalog is found is returned.

The whichcat utility accepts the following options and arguments:

-l Lists the paths that whichcat searched in looking for the catalogs for *group*.

group Group code of the message system catalog whose path is returned by whichcat. At least one group must be specified. Multiple groups can be specified.

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.

SEE ALSO

caterr(1), catxt(1), explain(1), gencat(1)

catgetmsg(3C), catgets(3C), catmsgfmt(3C), catopen(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

nl_types(5), msg(7D) in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

Cray Message System Programmer's Guide, Cray Research publication SG–2121

NAME

who – Reports who is on the system

SYNOPSIS

who -s [-A] [-b] [-h] [-H] [-l] [-m] [-p] [-r] [-t] [-T] [-u] *file*

who [-a] [-A] [-b] [-d] [-h] [-H] [-l] [-m] [-p] [-r] [-t] [-T] [-u] *file*

who [-q] [-n *number*] *file*

who am i

who am I

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-A, -d, and -n options)

DESCRIPTION

The who utility lists the user name, terminal line, login time, elapsed time since activity occurred on the line, and the process ID of the command interpreter (shell) for each current UNICOS user. It examines the /etc/utmp file to obtain its information. If you omit *file*, that file (which must be in utmp(5) format) is examined instead. Usually, *file* will be /etc/wtmp, which contains a history of all the logins since the file was last created.

The who utility with the -m or the am i operands identifies the invoking user.

Except for the default -s option, the general format for output entries is as follows:

name [*state*] *line time activity pid* [*comment*] [*exit*]

The *name* is the user's login name. The *state* describes whether someone else can write to that terminal. If the terminal can be written by anyone, a + appears. A - appears if it is not. root can write to all lines having a + or a - in the *state* field. If a bad line is encountered, a ? is printed. The *line* is the name of the line as found in the directory /dev. The *time* is the time that the user logged in. The *activity* is the number of hours and minutes since activity last occurred on that particular line. A dot (.) indicates that the terminal has seen activity in the last minute and is therefore ''current''. If more than 24 hours have elapsed or the line has not been used since boot time, the entry is marked old. This field is useful when trying to determine whether a person is working at the terminal. The *pid* is the process ID of the user's shell. The *comment* is the comment field associated with this line as found in /etc/inittab (see inittab(5)). This can contain commentary information. If no comment exists in /etc/inittab, the network host identifier of each user is displayed in parentheses under the comment field; otherwise, it appears after the comment field.

With options, who can list logins, logoffs, reboots, and changes to the system clock, as well as other processes spawned by the init process.

The who utility accepts the following options and operands:

-a Processes /etc/utmp or the specified *file* with all options turned on.

-A Displays utmp entries marked as accounting information.

-b Indicates the time and date of the last reboot of the system under the *line* and *time* headings, respectively.

-d Displays all processes that have expired and not been respawned by init. The *exit* field appears for dead processes and contains the termination and exit values (as returned by wait(2)) of the dead process. This can be useful in determining why a process terminated.

-h Displays network host identifiers.

-H Displays a header.

-l Lists only those lines at which the system is waiting for someone to log in. The *name* field is LOGIN in such cases. Other fields are the same as for user entries, except that the *state* field does not exist.

-m Lists information only about the current terminal.

-n *number*
Specifies the *number* of users per line for -q. The -q option spaces out the users names by using a format similar to that used by the ls(1) utility.

-p Lists any other process that is currently active and has been previously spawned by init. The *name* field is the name of the program executed by init as found in /etc/inittab. The *state, line,* and *activity* fields have no meaning. The *comment* field shows the *id* field of the line from /etc/inittab that spawned this process. See inittab(5).

-q Displays only the names and the number of users currently logged on; this is a quick who. When you use this option, all other options except -n are ignored.

-r Indicates the current *run-level* of the init process. In addition, this option produces the process termination status, process ID, and process exit status under the *idle, pid,* and *comment* headings, respectively.

-s Is the default and lists only the *name, line,* and *time* fields.

-t Indicates the last change to the system clock (through the date(1) utility) by root. See su(1).

-T Displays the *state* of the terminal line.

-u Lists information about users currently logged on the system.

file Specifies the file to contain login information.

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 `who` utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

FILES

`/etc/inittab`

`/etc/utmp`

`/etc/wtmp`

SEE ALSO

`date`(1), `login`(1), `ls`(1), `mesg`(1), `su`(1)

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

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

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

NAME

whoami – Displays the effective current user name

SYNOPSIS

```
/usr/ucb/whoami
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The whoami utility displays the login name corresponding to the current effective user ID. If you have used su(1) to temporarily adopt another user, whoami will report the login name associated with that user ID. whoami gets its information from the geteuid(2) system call and the getpwuid(3C) library routine.

FILES

/etc/passwd	User name data base

SEE ALSO

logname(1), su(1), who(1)

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

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

NAME

`write` – Lets you write to another user

SYNOPSIS

`write` *user_name* [*terminal*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `write` utility copies lines from your terminal to that of another user. When first called, it sends the following message to the other person:

```
Message from
yourname
(tty##)
[date]...
```

When it has completed the connection successfully, it sends two alert sequences to your own terminal, as well as to the recipient's terminal, to indicate what you are typing is being sent.

The message recipient should write back at this point. Communication continues until an end of file is read from the terminal or an interrupt is sent. At that point, the `write` utility writes `<EOT>` on the other terminal and exits.

The `write` utility accepts the following operands:

user_name The login name of person to whom the message will be written.

terminal Indicates terminal to which to send.

If you want to write to a user who is logged in more than once, use the *terminal* argument to indicate which terminal to send (such as, `ttyp001`); otherwise, the first writable instance of *user_name* found in `/etc/utmp` is assumed and the following message is posted:

```
user_name is logged on more than one place.
You are connected to "terminal".
Other locations are:
terminal
```

To deny or grant permission to write to another user's terminal, use the `mesg`(1) utility. By default, writing to others is usually allowed. Certain utilities, in particular `pr`(1), inhibit messages to prevent interference with their output. However, if the user has super-user permissions, messages can be forced onto a write-inhibited terminal.

If the character `!` is found at the beginning of a line, the `write` utility calls the shell to execute the rest of the line as a command.

The `write` utility detects nonprintable characters before sending them to the recipient's terminal. Control characters appear as a `^` character followed by the appropriate ASCII character. Characters with the high-order bit set will appear in meta-notation (for example, `'\0372'` is displayed as `'M-z'`).

The following protocol is suggested for using `write`: when you first `write` to another user, wait for them to `write` back before starting to send a line. Each person should end a message with a distinctive signal (that is, `(o)` for ''over'') so that the other person knows when to reply. The signal `(oo)` (for ''over and out'') is suggested when conversation is to be terminated.

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 write to any user.
`sysadm`	Allowed to write to any user, subject to security label restrictions on the user's terminal 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 write to any user.

EXIT STATUS

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

0 Successful completion.

>0 The addressed user is not logged on, or the addressed user denies permission.

MESSAGES

`User is not logged on.`
The person to whom you are trying to `write` is not logged on.

`Permission denied.`
The person to whom you are trying to `write` denies that permission (with `mesg`(1)).

`Warning: You have your terminal set to "mesg n". No reply possible.`
Your terminal is set to `mesg n` and the recipient cannot respond to you.

`Can no longer write to user.`
The recipient has denied permission (`mesg n`) after you had started writing.

`User is not at terminal.`
The recipient is not logged on at the specific terminal.

FILES

`/etc/utmp` To find *user_name*

SEE ALSO

`mail`(1), `mesg`(1), `pr`(1), `talk`(1B), `who`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

xargs – Constructs argument lists and executes a utility

SYNOPSIS

xargs [-p] [-t] [-e[*eofstr*] | -E *eofstr*] [-i[*replstr*] | -I *replstr*] [-l[*number*] | -L *number*] [-s *size*] [-x] [*utility* [*arguments*]]

xargs [-p] [-t] [-e[*eofstr*] | -E *eofstr*] [-i[*replstr*] | -I *replstr*] [-n *nargs*] [-s *size*] [-x] [*utility* [*arguments*]]

IMPLEMENTATION

Cray PVP systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The xargs utility combines the fixed *arguments* with arguments read from standard input to execute *utility* one or more times. The options specified determine the number of arguments read for each *utility* invocation and the manner in which they are combined.

The *utility* argument, which may be a shell file, is searched for using the user's $PATH. If you omit *utility*, /bin/echo is used.

Arguments read in from standard input are defined to be contiguous strings of characters delimited by one or more <blank>, <tab>, or <newline> characters; empty lines are always deleted. You can embed <blank>s and <tab> characters as part of an argument if escaped or quoted; characters enclosed in quotation marks (single or double) are taken literally, and the delimiting quotation marks are removed. Outside of quoted strings, a backslash (\) escapes the next character.

Each argument list is constructed starting with the *arguments*, followed by some number of arguments read from standard input (exception: see -I flag). Options -I, -i, -L, -l, and -n determine how arguments are selected for each command invocation. When none of these options are specified, the *arguments* are followed by arguments read continuously from standard input until an internal buffer is full, and then *utility* is executed with the accumulated arguments. This process is repeated until all arguments have been read. When option conflicts exist (such as -l with -n, or -e with -E), the last flag has precedence.

The xargs utility accepts the following options:

-E *eofstr* Use *eofstr* as the logical EOF string. Underscore (_) is assumed for the logical EOF string if neither -e nor -E is specified. The xargs utility reads standard input until either end-of-file or the logical EOF string is encountered.

-e[*eofstr*] This option is equivalent to -E *eofstr*. If the option-argument is not specified, the logical EOF string capability is disabled and underscores are taken literally.

`-I` *replstr* Insert mode. The *utility* will be executed for each line from standard input, taking the entire line as a single argument, inserting it in *arguments* for each occurrence of *replstr*. A maximum of five arguments in *arguments* can each contain one or more instances of *replstr*. Any `<blank>` or a `<tab>` characters at the beginning of each line are ignored. Constructed arguments may not be larger than {`NAME_MAX`} characters. Option `-x` is forced when this option is used.

`-i`[*replstr*] This option is equivalent to `-I` *replstr*. If the option-argument is not specified, the string `{}` is assumed for *replstr*.

`-L` *number* The *utility* will be executed for each nonempty *number* lines of arguments from standard input. The last invocation of *utility* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first `<newline>`, unless the last character of the line is a `<blank>` or a `<tab>`; a trailing `<blank>` or `<tab>` signals continuation through the next nonempty line. Option `-x` is forced when this option is used.

`-l`[*number*] This option is equivalent to `-L` *number*. If the option-argument is not specified, 1 is assumed.

`-n` *nargs* Executes *utility* by using as many standard input arguments as possible, up to *nargs* arguments maximum. Fewer arguments will be used if their total size is greater than *size* characters, and for the last invocation if there are fewer than *nargs* arguments remaining. If you also specify the `-x` option, each *nargs* arguments must fit in the *size* limitation; otherwise, `xargs` terminates execution.

`-p` Prompt mode. The user is asked whether to execute *utility* each invocation. Trace mode (`-t`) is turned on to print the command instance to be executed, followed by a `?...` prompt. A reply of `y` (optionally followed by anything) executes the command; anything else, including just a `<carriage-return>`, skips that particular invocation of *utility*.

`-s` *size* The maximum total size of each argument list is set to *size* characters. *size* must be a positive integer less than or equal to 3996. If you omit `-s`, 3996 is the default. The character count for *size* includes one extra character for each argument and the count of characters in the command name.

`-t` Trace mode. The *utility* and each constructed argument list are `echo`ed to standard error just prior to their execution.

`-x` Causes `xargs` to terminate whether any argument list is greater than *size* characters. The `-x` option is forced by the `-I`, `-i`, `-L`, and `-l` options. When you do not specify `-I`, `-i`, `-L`, `-l`, or `-n` options, the total length of all arguments must be within the *size* limit.

The `xargs` utility terminates if *utility* terminates because of a signal, a system call failure while trying to execute *utility*, or if it cannot execute *utility*. When *utility* is a shell program, it should explicitly `exit` (see `sh`(1)) with an appropriate value to avoid accidentally returning with -1.

EXIT STATUS

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

0 All invocations of *utility* returned exit status 0.

1–125 A command line that meets the specified requirements could not be assembled, one or more of the invocations of *utility* returned a nonzero exit status, or some other error occurred.

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

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

EXAMPLES

Example 1: The following command line moves all files from directory `$1` to directory `$2`, and `echo` each `mv`(1) utility just before doing it:

```
ls $1 | xargs -i{} -t mv $1/{} $2/{}
```

Example 2: The following combines the output of the parenthesized commands onto one line, which is then echoed to the end of file `log`:

```
(logname; date; echo $0 $*) | xargs >>log
```

Example 3: The user is asked which files in the current directory will be archived and archives them into file `arch` one at a time, or archives them into `arch` many files at a time.

```
ls | xargs -p -l1 ar r arch
ls | xargs -p -l1 | xargs ar r arch
```

Example 4: The following command line executes `diff`(1) with successive pairs of arguments originally typed as shell arguments:

```
echo $* | xargs -n 2 diff
```

SEE ALSO

`echo`(1), `find`(1), `sh`(1)

NAME

yacc – Yet another compiler compiler

SYNOPSIS

yacc [-b *file_prefix*] [-d] [-l] [-p *sym_prefix*] [-s *scale*] [-t] [-v] *file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The yacc utility converts a context-free grammar found in *file* into a set of tables for a simple automaton that executes an LR(1) parsing algorithm (see NOTES section for a definition of LR(1)). The grammar may be ambiguous; specified precedence rules are used to break ambiguities. The yacc utility produces the y.tab.c file as output.

The output file, y.tab.c, must be compiled by the C compiler to produce the yyparse program. This program must be loaded with the lexical analyzer program, yylex, as well as main and yyerror (an error-handling routine). These routines must be supplied by the user; lex(1) is useful for creating lexical analyzers usable by yacc.

Run-time debugging code is always generated in y.tab.c under conditional compilation control. By default, this code is not included when y.tab.c is compiled.

The yacc utility accepts the following options:

-b *file_prefix* Uses *file_prefix* instead of y as the prefix for all output filenames. The code file y.tab.c, the header file y.tab.h (created when -d is specified), and the description file y.output (created when -v is specified), are changed to *file_prefix*.tab.c, *file_prefix*.tab.h, and *file_prefix*.output, respectively.

-d Generates a y.tab.h file with the #define statements. that associate the token codes assigned by yacc with the user-declared token names. This allows source files other than y.tab.c to access the token codes.

-l Prevents the use of #line constructs in y.tab.c. This should be used only after the grammar and the associated actions are fully debugged.

-p *sym_prefix* Uses *sym_prefix* instead of yy as the prefix for all external names produced by yacc. The names affected include the functions yyparse(), yylex(), and yyerror(), and the variables yylval, yychar, and yydebug.

`-s` *scale*	Changes the default sizes of arrays that `yacc` uses to compile the grammar (productions, states, nonterminals, and so on) up or down according to scale. The scale must be a positive real number. To increase the array size, use a number greater than 1; to decrease the array size use a number less than 1.
`-t`	Changes the default to include run-time debugging code when `y.tab.c` is compiled. Whether or not the `-t` option was used, the run-time debugging code is under the control of `YYDEBUG`, a preprocessor symbol. When `YYDEBUG` has a nonzero value, the debugging code is included. When its value is 0, the code is not included. The size and execution time of a program produced without the run-time debugging code is smaller and slightly faster.
`-v`	Prepares the `y.output` file. This contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

NOTES

Because file names are fixed, only one `yacc` process can be active in a directory.

`LR`(1) is a type of Deterministic Context-Free Language (DCFL). `LR`(1) stands for "left-to-right scan of the input producing a rightmost derivation and using 1 symbol of lookahead on the input." `LR`(1) grammars have great importance for compiler design because they are broad enough to have efficient parsers that are essentially Deterministic Pushdown Automata (DPDA).

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

MESSAGES

The number of reduce-reduce and shift-reduce conflicts is reported on the standard error output; a more detailed report is found in the `y.output` file. Similarly, if some rules cannot be reached from the start symbol, this will also be reported.

FILES

file_prefix`.output`	Temporary files
file_prefix`.tab.c`	Temporary files
file_prefix`.tab.h`	Defines token names
`yacc.tmp`	Temporary files
`yacc.debug`	Temporary files
`yacc.acts`	Temporary files
`/usr/lib/yaccpar`	Parser prototype for C programs

SEE ALSO

`lex`(1)

lex & yacc, Doug Brown and Tony Mason, O'Reilly & Associates, Inc., 1992.

The UNIX Programming Environment, Brian W. Kernighan and Rob Pike, Prentice-Hall, Inc., 1984.

NAME

ypcat – Prints values in a network information service (NIS) database

SYNOPSIS

```
/usr/bin/ypcat [-k] [-t] [-d domainname] mname
/etc/yp/ypcat -x
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ypcat utility prints out values in a network information service (NIS) map. Because ypcat uses the NIS, no NIS server is specified.

The ypcat utility accepts the following options:

-k Displays the keys for those maps in which the values are null or in which the key is not part of the value. (None of the maps derived from files that have an ASCII version in /etc fall into this class.)

-t Inhibits translation of *mname* to map name. For example, ypcat -t passwd fails because no map is named passwd, whereas ypcat passwd is translated to ypcat passwd.byname.

-d *domainname*
Specifies a domain other than the default domain. *domainname* returns the default domain.

-x Displays the map nickname table. It lists the nicknames the utility knows, and it indicates the map name associated with each nickname.

mname Specifies a map name or a map nickname.

To look at the network-wide password database, passwd.byname (with the nickname *passwd*), enter the following:

```
ypcat passwd
```

See ypfiles(5) and ypserv(8) for an overview of the NIS.

SEE ALSO

domainname(1), ypmatch(1)

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

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

NAME

ypmatch – Prints the value of one or more keys from a network information service (NIS) map

SYNOPSIS

```
/usr/bin/ypmatch [-d domain] [-k] [-t] keys... mname
/etc/yp/ypmatch -x
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ypmatch utility prints the values associated with one or more keys from an NIS map (database) specified by *mname*, which can be either a map name or a map nickname.

You can specify multiple *keys*; the same map is searched for all. The capitalization and length of keys must be exact. No pattern matching is available. If a key is not matched, a diagnostic message is produced.

The ypmatch utility accepts the following options:

-d *domain*	Specifies a domain other than the default domain.
-k	Prints the key itself, followed by a colon (:), before printing the value of a key. This is useful only if the keys are not duplicated in the values, or you have specified so many keys that the output could be confusing.
-t	Inhibits translation of nickname to map name. (For example, ypmatch -t zippy passwd fails because there is no map named passwd, while ypmatch zippy passwd is translated to ypmatch zippy passwd.byname.)
keys...	Specifies the name of the key or keys for which ymatch prints a value.
mname	Specifies the name or nickname of the NIS map that contains the key.
-x	Displays the map nickname table. It lists the nicknames the utility knows, and it indicates the map name associated with each nickname.

SEE ALSO

ypcat(1)

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

NAME

yppasswd – Changes login password in network information service (NIS)

SYNOPSIS

/usr/bin/yppasswd [*name*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The yppasswd command changes (or installs) a password associated with the user *name* (your own name by default) in the NIS. The NIS password may be different than the one on your own machine.

The yppasswd command prompts for the old NIS password and then for the new one. The caller must supply both. The new password must be typed twice, to prevent mistakes.

New passwords must consist of at least 4 characters if they use a sufficiently rich alphabet and at least 6 characters if monocase.

Only the owner of the name or the super user can change a password; in either case, you must prove you know the old password.

BUGS

The update protocol passes all the information to the server in one RPC call, without ever looking at it. Thus, if you type in your old password incorrectly, you are not notified until after you have entered your new password.

SEE ALSO

passwd(1)

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

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