NAME

ctags – Creates a tags file

SYNOPSIS

```
ctags [-a] [-f tagsfile] [-t] [-u] [-w] [-B] [-F] pathname ...
ctags -v [-t] [-w] pathname ...
ctags -x [-t] [-w] pathname ...
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-t, -u, -v, -w, -B, and -F options)

DESCRIPTION

The ctags utility makes a tags file for ex(1) from the specified C, Pascal, Fortran, yacc, lex, and Lisp sources. A tags file provides the locations of specified objects (in this case, functions and typedefs) in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Functions are searched with a pattern, typedefs with a line number. Specifiers are entered in separate fields on the line, separated by <blank>s or <tab>s. Using the tags file, ex(1) can quickly find these object definitions.

Files that have names that end in .c or .h are assumed to be C source files and are searched for C routine and macro definitions. Files that have names that end in .y are assumed to be yacc source files. Files whose names end in .l are assumed to be either Lisp files (if their first non-<blank> character is ;, (, or [), or lex files (if the first non-<blank> character is anything else). Other files are first examined to see whether they contain any Pascal or Fortran routine definitions; if they do not, they will be processed again for C definitions.

The ctags utility accepts the following options:

-a Appends to tags file.

-f *tagsfile* Tag descriptions are placed in *tagsfile*. By default, they are placed in file tags.

-t Creates tags for typedefs.

-u Updates the specified files in *tagsfile*; that is, all references to them are deleted, and the new values are appended to the file.
Note: It is usually faster to rebuild the *tagsfile* file.

-v Produces an index on the standard output. This listing contains the function name, file name, and page number (assuming 64 line pages). Because the output will be sorted into lexicographic order, you may want to run the output through sort -f. A sample follows:

```
ctags -v files | sort -f > index
```

`-w` Suppresses warning diagnostics.

`-x` Produces a list of object names, the line number, and the file name on which each is defined, as well as the text of that line, and it prints this on standard output. This is a simple index that can be printed out as an offline-readable function index.

`-B` Uses backward searching patterns (`?...?`).

`-F` Uses forward searching patterns (`/.../`) (default).

The `main` tag is treated in a special way in C programs. The tag formed is created by prepending of `M` to the name of the file, with a trailing `.c`, if any, removed, and leading path name components also removed. This makes the use of `ctags` practical in directories that have more than one program.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

BUGS

Procedures cannot have the same name even if they are in different blocks.

`#ifdef`s are not recognized.

`ctags` relies on well-formed input to detect `typedef`s. The use of the `-tx` options shows only the last line of `typedef`s.

`ctags` does not recognize Pascal types.

FILES

`tags` Output tags file

SEE ALSO

`ex`(1), `vi`(1)

NAME

`cut` – Cuts out selected fields of each line of a file

SYNOPSIS

```
cut -b list [-n] [file...]
cut -c list [file...]
cut -f list [-d delim] [-s] [file...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `cut` utility cuts out columns from a table or fields from each line of a file; in database parlance, it implements the projection of a relation. The fields as specified by *list* can be of fixed length, that is, character positions as on a punched card (`-c` option), or the length can vary from line to line and be marked with a field delimiter character such as `<tab>` (`-f` option). The `cut` utility can be used as a filter; if no files are specified, the standard input is used. In addition, a file name of `-` explicitly refers to standard input.

The `cut` utility accepts the following options:

`-b` *list* — The *list* following `-b` specifies byte positions. For example, `-b 8-56` passes all bytes that start at byte 8 through and including byte 56 of each line.

`-c` *list* — The *list* following `-c` specifies character positions. For example, `-c 1-72` passes the first 72 characters of each line.

`-d` *delim* — The character following `-d` is the field delimiter (`-f` option only). Default is the `<tab>` character. You must enclose in quotation marks spaces or other characters that have special meaning to the shell.

`-f` *list* — The *list* following `-f` is a list of fields assumed to be separated in the file by a delimiter character (see `-d`). For example, `-f 1,7` only copies the first and seventh fields. Lines with no field delimiters are passed through intact (useful for table subheadings) unless you specify `-s`.

`-n` — When specified with the `-b` option, `-n` prevents characters which are made up of more than one byte from being split. (In UNICOS 8.0 release, multibyte characters are not supported.)

`-s` — Suppresses lines with no delimiter characters in the case of the `-f` option. Unless specified, lines with no delimiters are passed through untouched.

file — A path name of a file (any type) to be used by this command.

list A list, separated by commas or blank characters, of integer field numbers (in increasing order), with optional `-` to indicate ranges. For example, `1,4,7`; `1-3,8`; `-5,10` (short for `1-5,10`); or `3-` (short for third through last field.)

You must specify either the `-b`, `-c` or the `-f`.

Use `grep`(1) to make horizontal ''cuts'' (by context) through a file, or use `paste`(1) to put files together by columns (that is, horizontally). To reorder columns in a table, use `cut` and `paste`.

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

0 All input files were output successfully.

>0 An error occurred.

MESSAGES

`ERROR: line too long`
A line can have no more than 1022 characters or fields, or no newline character exists.

`ERROR: bad list for c/f option`
Missing `-c` or `-f` option or incorrectly specified *list*. If a line has fewer fields than the *list* indicates, no error occurs.

`ERROR: no fields`
The *list* is empty.

`ERROR: no delimiter`
Missing *delimiter* on `-d` option.

`ERROR: cannot handle multiple adjacent backspaces`
Adjacent backspaces cannot be processed correctly.

`WARNING: cannot open <filename>`
Either *filename* cannot be read or it does not exist. If multiple file names are present, processing continues.

EXAMPLES

Example 1: This example shows the mapping of user IDs to names:

```
cut -d: -f 1,5 /etc/passwd
```

Example 2: This example shows your login name:

```
name=‘who am i | cut -f 1 -d" "‘
```

SEE ALSO

`grep`(1), `paste`(1)

NAME

cxref – Generates C-language program cross-reference table

SYNOPSIS

cxref [-C] [-c] [-d] [-D *name* [*-def*]]... [-F] [-I *dir*] [-l] [-L *cols*] [-o *file*] [-s] [-t] [-U *dir*]... [-V] [-w *num*] [-W *name, file, function, line*] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

AT&T (-d, -l, -C, -F, -L, -V, -W)

DESCRIPTION

The cxref utility analyzes a collection of C files and builds a cross-reference table. cxref uses a special version of cc(1) to include #define'd information in its symbol table. It generates a list of all symbols (auto, static, and global) in each individual file, or, with the -c option, in combination. The table includes four fields: NAME, FILE, FUNCTION, and LINE. The line numbers appearing in the LINE field also show reference marks as appropriate. The reference marks include the following:

=	assignment
-	declaration
*	definition

If no reference marks appear, you can assume a general reference.

The cxref utility interprets the -D, -I, -U options in the same manner that cc(1) does. In addition, cxref interprets the following options:

-c Combines the source files into a single report. Without the -c option, cxref generates a separate report for each file on the command line.

-C Runs only the first pass of cxref, creating a .cx file that can later be passed to cxref. This is similar to the -c option of cc(1) or lint(1).

-d Disables printing declarations, making the report easier to read.

-F Prints the full path of the referenced file names.

-l Does not print local variables. Prints only global and file scope statistics.

-L *cols* Modifies the number of columns in the LINE field. If you do not specify a number, cxref defaults to five columns.

-o *file* Direct output to *file*.

`-s` Operates silently; does not print input file names.

`-t` Formats listing for 80-column width.

`-V` Prints version information on the standard error.

`-w` *num* Width option that formats output no wider than *num* (decimal) columns. This option will default to 80 if *num* is not specified or is less than 51.

`-W` *name, file, function, line*
Changes the default width of at least one field. The default widths are as follows:

Field	Characters
NAME	15
FILE	13
FUNCTION	15
LINE	20 (4 per column)

DIAGNOSTICS

Error messages usually mean you cannot compile the files.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

EXAMPLES

```
$ nl a.c
1       main()
2       {
3             int i;
4               extern char c;
5
6               i=65;
7               c=(char)i;
8       }
```

```
NAME    FILE          FUNCTION   LINE
c       a.c           ---        4-    7=
i       a.c           main       3*    6=   7
```

NAME	FILE	FUNCTION	LINE
main	a.c	---	2*
u3b2	Predefined	---	0*
unix	Predefined	---	0*

FILES

TMPDIR/tcx.*	Temporary files
TMPDIR/cx.*	Temporary files
LIBDIR/xref	Accessed by cxref
LIBDIR	/usr/lib by default
TMPDIR	Usually /var/tmp but can be redefined by setting the environment variable TMPDIR (see tempnam in tmpnam(3C)).

SEE ALSO

cc(1), lint(1)

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

NAME

`date` – Prints and sets the date

SYNOPSIS

```
date [-u] [+format]
date [-b] [-u] mmddhhmm[[cc]yy]
date -a [-]sss.fff
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (`-a` option)
CRI extension (`-b` option)

DESCRIPTION

If you omit arguments, or if the argument begins with `+`, the current date and time are printed; otherwise, the current date is set (only by super user).

The `date` utility accepts the following options:

`-a` [`-`] *sss.fff*
: Slowly adjusts the time by *sss.fff* seconds (*fff* represents fractions of a second). This adjustment can be positive or negative. The system's clock will be sped up (positive) or slowed down (negative) until it has drifted by the number of seconds specified.

`-b`
: Indicates that this is a system boot; therefore, the date change records written to `/etc/wtmp` should not be written. You should not call `date` with this option at any other time.

`-u`
: Displays (or sets) the date in Greenwich mean time (GMT-universal time), bypassing the normal conversion to (or from) local time.

  mm Month

  dd Day

  HH Hour (24-hour system)

  MM Minute

  cc Century minus one

  yy The last 2 digits of the year

  The month, day, year, and century may be omitted; the current values are supplied as defaults. The following example sets the date to October 8, 12:45 A.M.:

```
date 10080045
```

The current year is the default because no year is supplied. The system operates in GMT. `date` converts to and from local standard and daylight time. Only an appropriately authorized user may change the date. After successfully setting the date and time, `date` displays the new date according to the default format. The `date` utility uses `TZ` to determine the correct time zone information (see `environ`(7)).

+ *format* If the argument begins with +, the output of `date` is under the user's control. Each Field Descriptor (a description follows) is preceded by % and is replaced in the output by its corresponding value. One % is encoded by %%. All other characters are copied to the output without change. The string is always terminated with a <newline> character. If the argument contains embedded blanks, you must enclose it within single quotation marks (see the EXAMPLE section).

Specifications of native language translations of month and week day names are supported. The month and week day names used for a language are based on the locale specified by the `LC_TIME` and `LANG` environment variables (see `environ`(7)).

The month and week day names used for a language are taken from a file whose format is specified in `strftime`(3C). This file also defines country-specific date and time formats such as `%c`, which specifies the default date format. The following form is the default for `%c`:

```
%a %b %e %T %Z %Y
```

for example,

```
Fri Dec 23 10:10:42 EST 1988
```

Field Descriptors

Field descriptors must be preceded by a %:

a Abbreviated week day name

A Full week day name

b Abbreviated month name

B Full month name

c Country-specific date and time format

C Century (a year divided by 100 and truncated to an integer) as a decimal number – 00 through 99

d Day of month – 01 through 31

D Date as `%m/%d/%y`

e Day of month – 1 through 31 (single digits are preceded by a blank)

h Abbreviated month name (alias for `%b`)

H Hour – 00 through 23

- `I` Hour – 01 through 12
- `j` Day of year – 001 through 366
- `m` Month of year – 01 through 12
- `M` Minute – 00 through 59
- `n` Insert a `<newline>` character
- `p` String containing ante-meridiem or post-meridiem indicator (by default, AM or PM)
- `r` Time as `%I:%M:%S %p`
- `R` Time as `%H:%M`
- `S` Second – 00 through 61, allows for leap seconds
- `t` Insert a `<tab>` character
- `T` 24h clock time as `%H:%M:%S`
- `u` Week day as a decimal number – Monday = 1
- `U` Week number of year (Sunday as the first day of the week) – 00 through 53
- `V` Week of the year (Monday as the first day of the week) – 01 through 53
- `w` Day of week – Sunday = 0
- `W` Week number of year (Monday as the first day of the week) – 00 through 53
- `x` Country-specific date format
- `X` Country-specific time format
- `y` Year within century – 00 through 99
- `Y` Year as *ccyy* (4 digits)
- `Z` Time zone name

Modified Field Descriptors

Modified field descriptors must be preceded by a `%`. You can modify some field descriptors by the `E` and `O` modifier characters to indicate a different format or specification as specified in the `LC_TIME` locale description. If the corresponding keyword is not specified or not supported for the current locale, the unmodified field descriptor value is used. (In the UNICOS 9.0 release, alternate locale representations are not supported.)

- `Ec` Alternate appropriate date and time representation
- `EC` The name of the base year (period) in the alternate representation
- `Ex` Alternate date representation
- `Ey` Offset from `%EC` (year only) in the alternate representation
- `EY` Full alternate year representation

`Od` Day of the month using the alternate numeric symbols

`Oe` Day of the month using the alternate numeric symbols

`OH` Hour (24h clock) of the month using the alternate numeric symbols

`OI` Hour (12h clock) of the month using the alternate numeric symbols

`Om` Month using the alternate numeric symbols

`OM` Minutes using the alternate numeric symbols

`OS` Seconds using the alternate numeric symbols

`Ou` Week day as a number in the alternate representation – Monday = 1

`OU` Week number of the year (Sunday as the first day of the week) using the alternate numeric symbols

`OV` Week number of the year (Monday as the first day of the week) using the alternate numeric symbols

`Ow` Week day as a number in the alternate representation (Sunday = 0)

`OW` Week number of the year (Monday as the first day of the week) using the alternate numeric symbols

`Oy` Year (offset from `%C`) in alternate representation

NOTES

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

Active Category	Action
`system`, `secadm`	Allowed to set the date.
`sysadm`, `sysops`	Allowed to set the date. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to set the date.

If you try to set the current date to one of the dates that the standard and alternate time zones change (for example, the date that daylight time is starting or ending), and you try to set the time to a time in the interval between the end of standard time and the beginning of the alternate time (or the end of the alternate time and the beginning of standard time), the results are unpredictable.

EXIT STATUS

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

0 The date was written successfully.

>0 An error occurred.

MESSAGES

`No permission` You are not authorized to set the date.

`bad conversion` The date set is syntactically incorrect.

EXAMPLES

The following command:

```
date '+DATE: %m/%d/%y%nTIME: %H:%M:%S'
```

generates the following output:

```
DATE: 08/01/76
TIME: 14:45:05
```

SEE ALSO

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

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

`environ`(7) (available only online)

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

General UNICOS System Administration, Cray Research publication SG–2301

NAME

dd – Converts and copies a file to the specified output device

SYNOPSIS

dd [*operand...*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extension (iseek and oseek operands)

DESCRIPTION

The dd utility copies the specified input file to the specified output device with possible conversions. Standard input and output are used by default. You can specify the input and output block size to take advantage of raw physical I/O. *operand* takes the form option=*value*.

Operands are as follows:

bs=*n* Sets both input and output block size, superseding ibs and obs. When you do not specify a conversion, it is particularly efficient because an in-core copy does not have to be done.

cbs=*n* Specifies the conversion block size for block and unblock in bytes by *expr* (the default is 0). If cbs=*n* is omitted or is 0, using lock or unblock produces unspecified results. The cbs operand is used only when ascii or ebcdic conversion is specified. In the former case, cbs characters are placed in the conversion buffer and converted to ASCII, trailing blanks are trimmed, and <newline>s are added before the line is sent to output. In the latter case, ASCII characters are read into the conversion buffer and converted to EBCDIC, and blanks are added to make up an output block of size cbs.

conv=*value*[,*value...*]
Performs the specified conversion; *value* are comma-separated symbols from the following list:

ascii	Converts EBCDIC to ASCII.
block	Converts <newline>-terminated ASCII records to fixed length.
ebcdic	Converts ASCII to EBCDIC.
ibm	Maps ASCII to EBCDIC in a slightly different way.
lcase	Maps alphabetic characters to lowercase.
noerror	Does not stop processing on an error.

`notrunc`	Does not truncate the output file. Preserves blocks in the output file not explicitly written by this invocation of the `dd` utility.
`swab`	Swaps every pair of bytes.
`ucase`	Maps alphabetic characters to uppercase.
`unblock`	Converts fixed-length ASCII records to `<newline>`-terminated records.
`sync`	Pads every input block to `ibs`.

`count=`*n* Copies only *n* blocks.

`files=`*n* Copies and concatenates *n* input files before terminating. Using this operand makes sense only when input is a magnetic tape or similar device.

`ibs=`*n* Specifies input block size of *n* bytes. Default is 512.

`if=`*file* Specifies input file name. Standard input is default.

`iseek=`*n* Seeks *n* input blocks from beginning of input file before starting to copy.

`obs=`*n* Specifies output block size of *n* bytes. Default is 512.

`of=`*file* Specifies output file name. Standard output is default. The `dd` utility creates an explicit output file; therefore, the `seek` option is usually useless with an explicit output file except in special cases, such as when using disk files.

`seek=`*n*
`oseek=`*n* Seeks *n* output blocks from beginning of output file before copying. This option generally works only with raw disk files and does not work when the explicit output file was specified using the `of` option.

`skip=`*n* Skips *n* input blocks before starting copy. The skipped blocks are actually read; therefore, this can take a considerable amount of time.

When sizes are specified, a number of bytes is expected. A number may end with `b`, `s`, `k`, `B`, or `w` to specify multiplication by 512, 512, 1024, 4096, or 8, respectively; a pair of numbers may be separated by `x` to indicate a product.

After completion, `dd` reports the number of whole and partial input and output blocks.

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 copy any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
`sysadm`	Allowed to copy 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 copy any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXIT STATUS

The dd utility exits with one of the following values:

0 The input file was copied successfully.

>0 An error occurred.

MESSAGES

f+p records in(out) Numbers of full and partial records read (written)

EXAMPLES

Example 1: The following command reads an EBCDIC file that is blocked, ten 80-byte EBCDIC card images per block, into ASCII file x:

```
dd if=file of=x ibs=800 cbs=80 conv=ascii,lcase
```

Example 2: The dd utility is especially suited to I/O on raw physical devices because it permits reading and writing in arbitrary block sizes.

To use dd to copy the myfile file in your home directory out to a tape with the volume serial number of UA1234, enter the following:

```
rsv
tpmnt -l nl -p /tmp/$$u -v UA1234 -b 32768
dd if=$HOME/myfile of=/tmp/$$u bs=32768
rls -a
```

To read this file back in from tape to a disk file called newfile in your home directory, enter the following:

```
rsv
tpmnt -l nl -p /tmp/$$u -v UA1234 -b 32768
dd if=/tmp/$$u of=$HOME/newfile bs=32768
rls -a
```

The block size matches the maximum block size in the tpmnt(1) command.

SEE ALSO

cp(1), rls(1), rsv(1), tpmnt(1)

NAME

`define` – Displays definitions of Cray Research technical terms and terms added by a local site that match a specified search term

SYNOPSIS

```
define searchterm
define -a
define [-k] searchterm
define -l
define [-x] searchterm
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `define` utility displays definitions of Cray Research technical terms and terms added by a local site that match a specified search term (*searchterm*). If you enter `define` *searchterm*, `define` displays all matched terms and associated definitions. If `define` finds an exact match for *searchterm*, `define` displays the exact match and its definition first, followed by all other possible matches. If an exact match is not found, `define` displays any other possible matches and definitions.

If a term contains the specified *searchterm* string, `define` considers it a match. For example, entering `define uid` displays all terms (and their definitions) that contain the string `uid` (such as `setuid` and `subcooled liquid`).

Always specify the desired search term in lowercase characters (for example, enter `define ascii`, not `define ASCII`).

When using any of the following special or wildcard characters within a *searchterm*, protect the characters by using single quotation marks. The special or wildcard characters include:

```
$  *  [  ^  |  (  )  +  /
```

For example, enter `define 'sys$input'`, not `define sys$input`; or enter `define 'net*'`, not `define net*`.

The `define` utility accepts multiple word search terms (for example, you may specify `define character set` on the command line; `character set` is the search term).

The `define` utility accepts the following options and arguments:

`-a` Displays the entire glossary (all available terms and definitions) in alphabetical order.

`-k` When you use with *searchterm*, displays all matched terms (without the definitions). For example, entering `define -k check` retrieves an alphabetical list of terms that match the *searchterm* `check` (such as `check bits (cb)`, `check digit, checkbyte`, `checkpoint`, `checksum`, `cyclic redundancy check (crc)`, `debug (troubleshoot) (checkout)`, and `parity check`).

`-l` Displays in alphabetical order all available terms, synonyms, and acronyms (without the definitions). Synonyms and acronyms are displayed within parentheses next to the term. The list of terms produced by the `-l` option appears in all lowercase characters; however, when the `define` utility accesses a term, it will appear in the correct case.

`-x` When used with the *searchterm* argument, displays only a term (and its associated definition) that matches the *searchterm* exactly. For example, entering `define -x file` displays the definition for only the term `file`.

searchterm
Specifies the term for which to search in the online glossary.

If the output device is a terminal, `define` pipes its output through the pager specified by the `PAGER` environment variable. If you omit `PAGER`, `define` uses `more -s` as the default pager. If the output device is not a terminal, `define` sends output to the standard output device (`stdout`).

NOTES

Generally, the `define` utility's glossary does not contain definitions for UNICOS commands or standard UNIX terms. To obtain information on UNICOS commands, use the `man`(1) command. (For example, to find out the function of the `more` command, enter`man more`).

The default definitions directory that `define` uses is `/usr/lib/define`, unless you set the `DEFINEDIR` environment variable. The `define` utility reads files that have a `_k` suffix in the definitions directory as keyword files and corresponding files without the `_k` suffix as definition files. Each keyword file in the definitions directory is checked for validity. If a keyword file is not valid, its corresponding definition file will not be used. `define` searches each definition file in the directory in sequence.

For information on modifying the Cray Research definitions file or creating a local definitions file, see `builddefs`(1).

FILES

`CRAYdefs`	Cray formatted definition file
`CRAYdefs_i`	Cray formatted definitions with keywords embedded
`CRAYdefs_k`	Cray keyword file (non-ASCII)
`CRAYdefs_src`	Cray unformatted source definitions
`define.cat`	The `define` message catalog
`define.exp`	The `define` explain message catalog

`define.msg` The `define` message text file

SEE ALSO

`builddefs`(1), `man`(1)

General UNICOS System Administration, Cray Research publication SG–2301

NAME

delta – Makes a delta (change) to an SCCS file

SYNOPSIS

delta [-g *list*] [-m *mrlist*] [-n] [-p] [-r *SID*] [-s] [-y[*comment*]] *files*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The delta utility is used to permanently introduce into the named Source Code Control System (SCCS) file changes that were made to the file retrieved by get(1) (called the *g-file*, or generated file).

The delta utility makes a delta to each named SCCS file. If a directory is named, delta behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a dash (-) is given as a name, the standard input is read (see the WARNINGS section); each line of the standard input is taken to be the name of an SCCS file to be processed.

The delta utility may issue prompts on the standard output depending upon certain options specified and flags (see admin(1)) that may be present in the SCCS file (see -m and -y options).

Option arguments apply independently to each named file.

-g *list* Specifies a *list* (see get(1) for the definition of *list*) of deltas that are to be ignored when the file is accessed at the change level (SID) created by this delta.

-m *mrlist* If the SCCS file has the v flag set (see admin(1)), a modification request (MR) number *must* be supplied as the reason for creating the new delta.

-n Specifies retention of the edited *g-file* (normally removed at completion of delta processing).

-p Causes delta to print (on the standard output) the SCCS file differences before and after the delta is applied in a diff(1) format.

-r *SID* Uniquely identifies which delta is to be made to the SCCS file. The use of this option is necessary only if two or more outstanding gets for editing (get -e) on the same SCCS file were done by the same person (login name). The *SID* value specified with the -r option can be either the source identifier (SID) specified on the get command line or the SID to be made as reported by the get command (see get(1)). A message results if the specified SID is ambiguous, or if necessary and omitted on the command line.

`-s` Suppresses the issue, on the standard output, of the created delta's SID, as well as the number of lines inserted, deleted, and unchanged in the SCCS file.

If `-m` is not used and the standard input is a terminal, the prompt `MRs?` is issued on the standard output before the standard input is read; if the standard input is not a terminal, a prompt is not issued. The `MRs?` prompt always precedes the `comments?` prompt (see `-y` option).

MRs in a list are separated by blanks and/or tab characters. An unescaped new-line character terminates the MR list.

If the `v` flag has a value (see `admin`(1)), it is taken to be the name of a program (or shell procedure), that will validate the correctness of the MR numbers. If a nonzero exit status is returned from MR number validation program, `delta` terminates (it is assumed that the MR numbers were not all valid).

`-y`[*comment*] Arbitrary text used to describe the reason for making the delta. A null string is considered a valid *comment*.

If `-y` is not specified and the standard input is a terminal, the prompt `comments?` is issued on the standard output before the standard input is read; if the standard input is not a terminal, a prompt is not issued. An unescaped new-line character terminates the comment text.

files Specifies the SCCS files to change.

WARNINGS

Lines beginning with an `SOH` ASCII character (binary 001) cannot be placed in the SCCS file, unless the `SOH` is escaped. This character has special meaning to SCCS (see `sccsfile`(5)) and will cause an error.

A `get` of many SCCS files, followed by a `delta` of those files, should be avoided when the `get` generates a large amount of data. Instead, multiple `get/delta` sequences should be used.

If the standard input (`-`) is specified on the `delta` command line, the `-m` (if necessary) and `-y` options must also be present. If you omit these options, an error occurs.

Comments are limited to text strings of 512 characters.

MESSAGES

Use `help`(1) for explanations.

EXAMPLES

In the following example, changes found in the file `example.c` are applied to `s.example.c`. The file `example.c` is removed. User input is shown in bold type:

```
$ delta s.example.c
comments? Changed some message text.
1.2
1 inserted
1 deleted
4 unchanged
$
```

FILES

g-file	Existed before the execution of `delta`; removed after completion of `delta`.
p-file	Existed before the execution of `delta`; may exist after completion of `delta`.
q-file	Created during the execution of `delta`; removed after completion of `delta`.
x-file	Created during the execution of `delta`; renamed to SCCS file after completion of `delta`.
z-file	Created during the execution of `delta`; removed during the execution of `delta`.
d-file	Created during the execution of `delta`; removed after completion of `delta`.
`/usr/bin/bdiff`	Program to compute differences between the ‘‘gotten’’ file and the *g-file*.

SEE ALSO

`admin`(1), `bdiff`(1), `cdc`(1), `comb`(1), `diff`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

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

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

NAME

deplib – Manipulates dependent library names in bld library files

SYNOPSIS

deplib [-l] [-a *file...*] [-d *file...*] [-r *file...*] *bld_library*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The deplib utility lets users associate library names with bld(1) libraries. With deplib, users can list, add, delete, and replace dependent library names in the bld library file headers. If dependent library names are present within a bld library, segldr(1) searches those libraries immediately after searching the library in which they are specified. For example, if the user specifies libraries A, B, and C, and B declares dependent libraries D and E, the search order is A, B, D, E, C followed by the default libraries.

Dependent libraries are only recognized by segldr (ld(1) will ignore them). Also, deplib will operate only on bld libraries, not on ar(1) libraries.

The deplib utility accepts the following options:

-l Lists any dependent library names associated with *bld_library*.

-a *file* Adds the file or files to the dependent library list in *bld_library*. If more than one file is to be added, enclose the list in single quotation marks or separate the list with commas.

-d *file* Deletes the file or files from the dependent library list in *bld_library*. If more than one file is to be deleted, enclose the list in single quotation marks or separate the list with commas.

-r *file* Replaces the current list of dependent library names with the list given on the command line. Note that this is a total replacement of filenames. If more than one file is in the replacement list, enclose the list in single quotation marks or separate the list with commas.

bld_library Specifies the name of the bld library.

Dependent library names can be either a full or relative path name or just a file name. If the string begins with a . or /, segldr assumes the string is a complete path name. If the string begins with a ., the current directory is used, which is not necessarily the directory in which the bld library resides. Otherwise, segldr assumes that the string is only a file name, with no path. In this case, segldr will look in each directory specified by -L (on the segldr command line) for the file and take the first one found.

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 manipulate any file. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to manipulate 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 manipulate any file. Shell-redirected I/O on behalf of the super user is not subject to file protections.

EXAMPLES

Example 1: The following example adds two library names to the bld library, mylib.a:

```
$ deplib -a 'mylibf.a mylibm.a' mylib.a
```

Example 2: The following example lists the current dependent libraries in mylib.a, replaces the list, and lists them again:

```
$ deplib -l mylib.a
mylibf.a
mylibm.a
$ deplib -r 'yourlibf.a yourlibm.a' mylib.a
$ deplib -l mylib.a
yourlibf.a
yourlibm.a
```

SEE ALSO

bld(1), segldr(1)

NAME

deroff – Removes nroff, troff, tbl, and eqn constructs

SYNOPSIS

deroff [-w] *filenames*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The deroff utility reads each file in sequence and removes all nroff(1) and troff(1) command lines, backslash constructions, macro definitions, eqn constructs (between .EQ and .EN lines or between delimiters), and table descriptions and writes the remainder on the standard output. deroff follows chains of included files (.so and .nx commands); if a file has already been included, a .so is ignored and a .nx terminates execution. If no input file is given, deroff reads from the standard input file.

The deroff utility accepts the following option and operand:

-w Generates a word list, one word per line. A *word* is a string of letters, digits, and apostrophes, beginning with a letter; apostrophes are removed. All other characters are ignored.

filenames Specifies the files on which to run deroff.

NOTES

The deroff utility is not a complete troff(1) interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

The deroff utility does not work well with files that use .so to source in the standard macro package files.

SEE ALSO

eqn(1), nroff(1), tbl(1), troff(1)

NAME

df – Reports free disk space

SYNOPSIS

```
df [-S] [-d] [-f] [-l] [-p] [-P | -t] [file ...]
df -B [-d] [-f] [-l] [-p] [-P | -t] [file ...]
df -k [-d] [-f] [-l] [-p] [-P | -t] [file ...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-B, -d, -f, -l, -p, -S, and -t options)

DESCRIPTION

The df utility reports the number and percent of free blocks available for file systems by examining the counts kept in the super blocks; you can specify *file* by device name (such as /dev/dsk/root), by mounted directory name (such as /usr), or by the pathname of a file that exists on a file system. If you do not specify the *file* operand, the free space on all of the mounted file systems is printed. The default output reports the following information in order: file name, device name, number of free blocks, the block size in bytes, free percentage, and, if appropriate, the number of free file items. The items may be *inodes* or *processes* (/proc). If no tracks remain on the file system, an asterisk (*) is displayed after the free percentage. The default block size when reporting space figures is 512-byte units. You can use the -B or -k options to override this default, showing instead the number of 4096-byte units and 1024-byte units, respectively.

The df utility accepts the following options and operands:

-B Reports number of blocks that in 4096-byte units, instead of the default 512-byte units.

-d Reports information on partitions which are flagged as DOWN, READ-ONLY, or NO-ALLOCATE. For mounted file systems, the default behavior of df is to not report usage information for partitions that are "offline." For unmounted file systems, this option has no effect. (For additional information on marking file systems as DOWN, READ-ONLY, or NO-ALLOCATE, see pddconf(8) for Cray Research systems having I/O model E subsystems (IOS-E).

-f Option provided for backward compatibility only; this option does nothing, and will be removed in the next release of UNICOS.

-k Reports number of blocks in 1024-byte units, instead of the default 512-byte units.

-l Reports only on local file systems. NFS file systems are omitted.

`-p` Reports the same information as the `-t` option (in sectors), thresholds for big file allocations, and allocation size for primary or secondary partitions, if greater than one block. It also reports the following information for each partition, in table form:

`part`	Relative partition number
`start`	Starting block number
`total`	Total block count of partition
`free`	Free block count of partition
`frags`	Count of discontiguous areas in the partition
`device`	ASCII name of the device on which the partition resides

For `NC1FS` file systems, the inode regions are also displayed.

`-P` Produces output in the format specified by POSIX 1003.2. The unit size used to report space figures depends on other options. By default, the figures are in 512-byte units. If you specify `-k`, the figures are in 1024-byte units; if you specify `-B`, the figures are in 4096-byte units. The output consists of a single header line followed by lines with the following fields:

file system name	Device name.
total space	Total size of file system in specified units.
space used	Total amount of space, in specified units, allocated to existing files in the file system.
space free	Total amount of space, in specified units, available within the file system.
percentage used	Percentage of the normally available space that is currently allocated to all files on the file system.
file system root	Directory below which the file system hierarchy appears.

`-S` Causes the output to be in a format easily modified by utilities, such as `sed`(1) or `cut`(1). The `-B` and `-k` options are ignored when this option is specified. The following information is displayed, delimited by the `<tab>` character:

file name	File system name.
device name	Device(s) on which the file system resides.
free blocks	Number of free blocks on the file system.
block size	The size of a block (in bytes).
free percentage	The ratio in percentage of free blocks to total blocks.
free item count	Count of free items. If items are not applicable to the file system, total items will be 0, and item type is `undef`.
total blocks	Total possible blocks on the file system.
total items	Total possible items.

item type A description of the item. Either `Inodes` for local file systems, `procs` for the `/proc` file system, or `undef` for file systems in which items are unavailable, or do not apply.

`-t` Includes in the output the total blocks and file items. Cannot be specified with the `-P` option.

file Name of the disk device.

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:

If the `PRIV_SU` configuration option is enabled, users are allowed to specify any file.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following is an example of information displayed by the `-t` and `-B` options:

```
file (/dev/dsk/scr05):   16580 4K blocks ( 33.6%) 1505 inodes
                 total:   49392 4K blocks          6600 inodes
```

Example 2: The following is an example of information displayed by the `-p` option only. This example shows an `NC1FS` file system, `/dev/dsk/qtest1`, that was constructed with the following command lines:

```
$ mkfs -q -P 4 -S 16 -p 1
$ df -p /dev/dsk/qtest1
```

```
qtest1     (/dev/dsk/qtest1  ):  23780 sectors       0 trks    6076 Inodes
                          total:  24192 sectors      (0 trks)   6080 Inodes

       Big file threshold:            32768 bytes
       Big file allocation minimum:      24 blocks

       Primary partitions allocation unit:         16K blocks
       Secondary partitions allocation unit:       64K blocks

       part   start     total         free (%)          frags (%)      device
       ----  --------  --------  -----------------  ----------------  --------
          0p        0      8064      7652 ( 94.9%)      0 (  0.000%)  50-A2-22
          1s     8064      8064      8064 (100.0%)      0 (  0.000%)  50-A1-22
          2s    16128      8064      8064 (100.0%)      0 (  0.000%)  50-A2-22

                              part: 0
          Inode region: 0                  Inode region: 1
        start    total  free    (%)      start    total  free    (%)
       --------  -----  -----  -----    --------  -----  -----  -----
              0   6080   6076  99.93
```

Example 3: The following is an example of the POSIX 1003.2 output format, using the `-P` and `-k` options.

```
$ df -P -k /dev/dsk/qtest

Filesystem        1024-blocks  Used       Available   Capacity  Mounted on
/dev/dsk/qtest    1096640      673776     422864       61%      qtest
```

FILES

`/dev/dsk/*`	Disk devices

SEE ALSO

`du`(1) to summarize disk usage

`statfs`(2) to get file system information

`dsk`(4) for information on disk drive, buffer memory, and SSD interface
`fs`(5) for file system partition format
in the *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

`pddconf`(8) to control the state of an IOS model E disk drive
in the *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022

General UNICOS System Administration, Cray Research publication SG–2301

NAME

diff – Differential file comparator

SYNOPSIS

diff [-bitw] [-c | -e | -f | -h | -n] *filename1 filename2*
diff [-bitw] [-c | -e | -f | -h | -n] [-l] [-r] [-s] [-S *name*] *directory1 directory2*
diff [-bitw] [-C *number*] *filename1 filename2*
diff [-bitw] [-C *number*] [-l] [-r] [-s] [-S *name*] *directory1 directory2*
diff [-bitw] [-D *string*] *filename1 filename2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-D, -i, -f, -h, -l, -n, -s, -S, -t, and -w options)

DESCRIPTION

The diff utility tells the lines that you must change in two files to bring them into agreement. If *filename1* (*filename2*) is -, the standard input is used. If *filename1* (*filename2*) is a directory, a file in that directory with the name *filename2* (*filename1*) is used. The typical output contains lines of these forms:

n1 a *n3,n4*
n1,n2 d *n3*
n1,n2 c *n3,n4*

These lines resemble ed(1) commands to convert *filename1* into *filename2*. The numbers after the letters pertain to *filename2*. In fact, by exchanging a for d and reading backward, you may ascertain equally how to convert *filename2* into *filename1*. As in ed(1), identical pairs, in which *n1 = n2* or *n3 = n4*, are abbreviated as one number.

Following each of these lines come all of the lines that are affected in the first file flagged by <, then all of the lines that are affected in the second file flagged by >.

The diff utility accepts the following options and operands:

-b Ignores trailing blanks (<space> and <tab> characters) and treats other strings of <blank>s as equivalent.

-i Ignores the case of letters (for example, 'A' will compare equal to 'a').

-t Expands <tab> characters in output lines. Typical or -c output adds character(s) to the front of each line that may adversely affect the indentation of the original source lines and make the output lines difficult to interpret. This option will preserve the original source's indentation.

`-w` Ignores all <blank>s (<space> and <tab> characters) and treats all other strings of <blank>s as equivalent. For example, '`if ( a == b )`' will compare equal to '`if(a==b)`'.

The following options are mutually exclusive:

`-c` Produces a listing of differences that has three lines of context. With this option, output format is modified slightly: output begins with identification of the files involved and their creation dates; then each change is separated by a line with a dozen `*`'s. The lines removed from *filename1* are marked with `-`; those added to *filename2* are marked `+`. Lines that are changed from one file to the other are marked with `!` in both files.

`-C` *number* Produces a listing of differences identical to that produced by `-c` with *number* lines of context.

`-e` Produces a script of *a*, *c*, and *d* commands for the editor `ed`(1), which will re-create *filename2* from *filename1*. In connection with `-e`, the following shell program may help maintain multiple versions of a file. Only an ancestral file (`$1`) and a chain of version-to-version `ed`(1) scripts (`$2`,`$3`,...) made by `diff` must be on hand. A ''latest version'' appears on the standard output.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

Except in rare circumstances, `diff` finds a smallest sufficient set of file differences.

`-f` Produces a similar script, not useful with `ed`(1), in the opposite order.

`-h` Does a fast, half-hearted job. It works only when changed stretches are short and well separated, but it does work on files of unlimited length. Options `-e` and `-f` are unavailable with `-h`.

`-n` Produces a script similar to `-e`, but in the opposite order and with a count of changed lines on each insert or delete command.

`-D` *string* Creates a merged version of *filename1* and *filename2* with C preprocessor controls included so that a compilation of the result without defining *string* is equivalent to compiling *filename1*, while defining *string* yields *filename2*.

The following options are used for comparing directories:

`-l` Produces output in long format. Before the `diff`, each text file is piped through `pr`(1) to paginate it. Other differences are remembered and summarized after all text file differences are reported.

`-r` Applies `diff` recursively to common subdirectories encountered.

`-s` Reports files that are identical; these would not otherwise be mentioned.

`-S` *name* Starts a directory `diff` in the middle, beginning with the file *name*.

filename1
filename2 Path names of files to be compared.

directory1
directory2 Path names of directories to be compared.

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.

Editing scripts produced under the `-e` or `-f` option are naive about creating lines that consist of a one (`.`).

EXIT STATUS

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

0 No differences were found.

1 Differences were found.

>1 An error occurred.

MESSAGES

`Missing newline at end of file` *X*
This message indicates that the last line of file *X* did not have a `<newline>` character. If the lines differ, they will be flagged and output; however, the output will seem to indicate that they are the same.

FILES

`TMPDIR/d`*tempfile*	Temporary working file
`/usr/lib/diffh`	Fast `diff` program used with the `-h` option
`/bin/pr`	`pr`(1) utility

SEE ALSO

`bdiff`(1), `cmp`(1), `comm`(1), `diff3`(1), `dircmp`(1), `ed`(1), `pr`(1),

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

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

NAME

diff3 – Makes a three-way differential file comparison

SYNOPSIS

```
diff3 [-e] file1 file2 file3
diff3 -x file1 file2 file3
diff3 -3 file1 file2 file3
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The diff3 utility compares three versions of a file, and it publishes disagreeing ranges of text flagged with these codes:

====	All three files differ.
====1	*file1* is different.
====2	*file2* is different.
====3	*file3* is different.

The type of change produced in the conversion of a given range of a given file to some other is indicated in one of the following ways:

f : *n1* a	Text is to be appended after line number *n1* in file *f*, *f* = 1, 2, or 3.
f : *n1* , *n2* c	Text is to be changed in the range line *n1* to line *n2*. If *n1* = *n2*, the range may be abbreviated to *n1*.

The original contents of the range follow immediately after a c indication. When the contents of two files are identical, the contents of the lower-numbered file are suppressed.

The diff3 utility accepts the following options:

-e Publishes for the ed(1) editor a script that incorporates into *file1* all changes between *file2* and *file3*, that is, the changes that normally would be flagged ==== and ====3.

-x Produces a script to incorporate only changes flagged ====.

-3 Produces a script to incorporate only changes flagged ====3.

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 compare any three files. In a privileged administrator shell environment, shell-redirected I/O is not subject to file protections.
sysadm	Allowed to compare any three files 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 compare any three files. Shell-redirected I/O on behalf of the super user is not subject to file protections.

BUGS

Text lines that consist of a single . defeat the -e option.

There is an arbitrary limit of 200 on the total number of disagreeing ranges of text.

EXAMPLES

The following command applies the resulting script to *file1*:

```
(cat script; echo '1,$p') | ed - file1
```

FILES

TMPDIR/d3*	Temporary working file
/usr/lib/diff3prog	Executable file

SEE ALSO

diff(1)

NAME

diffmk – Marks differences between versions of a troff(1) input file

SYNOPSIS

diffmk *oldfile newfile markedfile*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The diffmk command compares two versions of a file and creates a third version that includes change mark (.mc) commands for nroff(1) and troff(1). *oldfile* and *newfile* are the old and new versions of the file. diffmk generates *markedfile*, which contains the text from *newfile* with troff(1) change mark commands (.mc) inserted where *newfile* differs from *oldfile*. When *markedfile* is formatted, changed or inserted text is shown by a | symbol at the right margin of each line. The position of deleted text is shown by a single asterisk (*).

The diffmk command can also be used in conjunction with the proper troff(1) requests to produce program listings with marked changes, as shown in the following command line:

```
diffmk old.c new.c marked.c ; nroff reqs marked.c | pr
```

The file reqs contains the following troff(1) requests:

```
.pl 1
.ll 77
.nf
.eo
.nh
```

These troff(1) requests eliminate page breaks, adjust the line length, set no-fill mode, ignore escape characters, and turn off hyphenation, respectively.

If the characters | and * are inappropriate, you can run *markedfile* through the sed(1) command to globally change them.

NOTES

Aesthetic considerations may dictate manual adjustment of some output. File differences involving only formatting requests may produce undesirable output, that is, replacing .sp by .sp 2 produces a change mark on the preceding or following line of output.

SEE ALSO

nroff(1), sed(1), troff(1)

NAME

`dircmp` – Compares directories

SYNOPSIS

`dircmp` [`-d`] [`-s`] [`-w` *n*] *dir1 dir2*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `dircmp` utility examines *dir1* and *dir2* and generates various tabulated information about the contents of the directories.

Listings of files unique to each directory are generated for all options. If you do not specify an option, a list is output, indicating whether the files common to both directories have the same contents.

The `dircmp` utility accepts the following options:

`-d`	Compares the contents of files with the same name in both directories and outputs a list indicating what must be changed in the two files to bring them into agreement. The list format is described in `diff`(1).
`-s`	Suppresses messages about identical files.
`-w` *n*	Changes the width of the initial multicolumn output to *n* characters. The default width is 72.
dir1 dir2	Directories to be compared.

SEE ALSO

`cmp`(1), `diff`(1)

NAME

`domainname` – Sets or displays name of current network information service (NIS) domain

SYNOPSIS

`domainname` [*nameofdomain*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

Without an argument, `domainname` displays the name of the current domain. Only the super user can set the domain name by giving an argument; this is usually done in the `/etc/rc` start-up script. Currently, domains are used only by the NIS to refer collectively to a group of hosts.

SEE ALSO

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

NAME

du – Summarizes disk usage

SYNOPSIS

```
du [-a] [-B | -k] [-mrRx] [file ...]
du -A [-B | -k] [-mrRx] [file ...]
du -s [-B | -k] [-mrRx] [file ...]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
CRI extensions (-A, -B, -m, and -R options)

DESCRIPTION

The du utility, by default, gives the size (in 512-byte units) of each *file* operand and (recursively) each subdirectory of the specified *file* operand. To change the default unit size, use the -B or -k options. If you omit *file* operands, the current directory is used.

The du utility accepts the following options:

-a In addition to the default output, reports the size of each nondirectory in the file hierarchy. Regardless of the presence of the -a option, nondirectories given as *file* operands are always reported.

-A Generates an entry for each file but not for directories.

-B Expresses all block counts in terms of 4096-byte units, rather than the default 512-byte units.

-k Expresses all block counts in terms of 1024-byte units, rather than the default 512-byte units.

-m Reports block counts for migrated (offline) files and nonmigrated (online) files. Valid only with Cray Research file systems. See the EXAMPLES section for the format of this output.

-r Generates messages about directories that cannot be read and files that cannot be opened. If one of these errors occurs, the exit status of the du utility will be nonzero. (This option is on by default.)

-R Suppresses messages about directories that cannot be read and files that cannot be opened, and exit status is not affected if one of these errors occurs.

-s Reports only the grand total for each of the specified *file* operands.

-x Reports only the size of files that have the same device (st_dev field in the stat structure) as the file specified by the *file* operand.

NOTES

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

Active Category	Action
`system`, `secadm`	Allowed access to any file.
	(All other users are allowed access to 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 access to any file.

The default unit size for reporting the number of blocks was changed to 512 bytes to accommodate the POSIX 1003.2 standard. When file space is computed, the block size for the file system (`f_bsize` field in the `statfs` structure) is used to compute the number of 512-byte blocks. This means that NFS file systems will report numbers consistent with the actual size of the file.

Regardless of the presence of `-a` or `-A` options, nondirectories given as *file* operands are always listed.

A file with two or more links is counted only once.

If more than 1000 distinct linked files exist, the `du` utility counts the excess files more than once.

EXIT STATUS

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

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: The following example lists disk usage by directory for directories found in `/b`. The list is sorted by descending usage:

```
$ du -s /b/* | sort -k 0nr
```

Example 2: The following example lists disk usage for online and offline files in the current directory:

```
$ du -am
4          online   0          offline file1
4          online   0          offline .
$
```

SEE ALSO

`df`(1) for information about free disk space on file systems

`stat`(2),
`statfs`(2) for information for file system statistics
in the *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012

General UNICOS System Administration, Cray Research publication SG–2301

NAME

echo – Echoes arguments

SYNOPSIS

echo [*args*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The echo utility writes its arguments, separated by <blank>s and terminated by a <newline>, on standard output. It also recognizes C-like escape conventions; beware of conflicts with the shell's use of \ as in the following:

\a Writes an <alert> character.

\b Writes a <backspace> character.

\c Prints line without a <newline> (terminates the echo string).

\f Writes a <form-feed> character.

\n Writes a <newline> character.

\r Writes a <carriage-return> character.

\t Writes a <tab> character.

\v Writes a <vertical-tab> character.

\\ Writes a backslash character.

\0*n* Writes an 8-bit value that is the zero-, one-, two- or three-digit octal number *n*.

The echo utility is useful for producing diagnostics in command files and for sending known data into a pipe.

NOTES

When representing an 8-bit character by using the escape convention \0*n*, the *n* must always be preceded by the digit 0. For example, typing the following: echo 'WARNING:\07' prints the phrase WARNING: and sounds the ''bell'' on your terminal. The use of single (or double) quotation marks (or two backslashes) is required to protect the ''\'' that precedes the 07.

Following the \0, up to three digits are used in constructing the octal output character. If, following the \0*n*, you want to echo additional digits that are not part of the octal representation, you must use the full 3-digit *n*. For example, if you want to echo ESC 7, you must use the three digits 033 rather than just the two digits 33 after the \0.

For the octal equivalents of each character, see the ascii(7) man page.

csh(1) has a built-in echo utility with slightly different characteristics. See the csh(1) man page.

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 echo utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

EXAMPLES

Example 1: This example could be in a standard shell script prompting a user for the name:

```
echo "Enter name: \c"
read name
```

Example 2: This example prints the error message shown in double quotation marks to file test:

```
$ echo "Usage: command [-a] [-b] file" >test
```

SEE ALSO

csh(1), sh(1)

ascii(7) (available only online)

NAME

ed, red – Text editor

SYNOPSIS

ed [-p *string*] [-s] [-x] [-C] [*file*]
red [-p *string*] [-s] [-x] [-C] [*file*]

Obsolescent version; may not be supported in future releases:

ed [-p *string*] [-x] [-C] [-] [*file*]
red [-p *string*] [-x] [-C] [-] [*file*]

IMPLEMENTATION

Cray PVP systems
CRAY T3D systems

STANDARDS

POSIX, XPG4
AT&T extensions (-x and -C options)

DESCRIPTION

The ed utility is the standard text editor. If you specify file, ed simulates an e command (see the following text) on the specified file; that is, the file is read into the ed buffer so that you can edit it.

The ed utility accepts the following options:

-p *string* Lets users specify a prompt string.

-s Suppresses the printing of character counts by e, E, r, and w commands, of diagnostics from e and q commands, and of the ! prompt after a !shell command. See the WARNINGS section.

-x Encryption option; when used, ed simulates an X command and prompts the user for a key, which is used to encrypt and decrypt text using the algorithm of crypt(1). The X command makes an educated guess to determine whether or not text read in is encrypted. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the -x option. See crypt(1) and the WARNINGS section.

-C Encryption option; the same as the -x option, except that ed simulates a C command. The C command is like the X command, except that all text read in is assumed to have been encrypted. See the WARNINGS section.

- (Obsolescent) Equivalent to -s.

The `ed` utility operates on a copy of the file it is editing; changes made to the copy have no effect on the file until you specify a `w` (write). The copy of the text being edited resides in a temporary file called the *buffer*. Only one buffer exists.

The `red` utility is a restricted version of `ed`. It allows the editing of files only in the current directory. It prohibits the execution of shell commands through `!shell` *command*. If you try to bypass these restrictions, an error message (`restricted shell`) results.

Commands to `ed` have a simple and regular structure: zero, one, or two addresses followed by a single-character command, possibly followed by arguments to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses; therefore, you can usually omit the addresses.

Generally, only one command can appear on a line. Certain commands allow the input of text, which is placed in the appropriate place in the buffer. When `ed` is accepting text, it is in input mode. In this mode, commands are not recognized; all input is merely collected. To leave input mode, type a single period (`.`) at the beginning of a line, and immediately press `<RETURN>`.

The `ed` utility supports a limited form of regular expression notation; regular expressions are used in addresses to specify lines and, in some commands (for example, `s`), to specify portions of a line that will be substituted. A regular expression (RE) specifies a set of character strings. A member of this set of strings is said to be *matched* by the RE. The REs allowed by `ed` are constructed as follows:

The following 1-character REs match a single character:

1. An ordinary character (not one of those discussed in item 2) is a 1-character RE that matches itself.
2. A backslash (`\`) followed by any special character is a 1-character RE that matches the special character itself. The special characters are as follows:
   a. A period (`.`), asterisk (`*`), left bracket, (`[`), or backslash (`\`), which are always special, except when they appear within brackets (`[]`; see item 4).
   b. A caret or circumflex (`^`), which is special at the beginning of an entire RE (see items 11 and 13), or when it immediately follows the left of a pair of brackets (`[]`) (see item 4).
   c. A dollar sign (`$`), which is special at the end of an entire RE (see item 12).
   d. The character used to bound (such as, delimit) an entire RE, which is special for that RE (for example, see how slash (`/`) is used in the *g* command).
3. A period (`.`) is a 1-character RE that matches any character except a `<newline>`.

4. A nonempty string of characters enclosed in brackets (`[]`) is a 1-character RE that matches any one character in that string. If, however, the first character of the string is a circumflex (`^`), the 1-character RE matches any character except `<newline>` and the remaining characters in the string. The `^` has this special meaning only if it occurs first in the string. You can use the minus sign (`-`) to indicate a range of consecutive ASCII characters (for example, `[0-9]` is equivalent to `[0123456789]`). The `-` loses this special meaning if it occurs first (after an initial `^`, if any) or last in the string. The right bracket (`]`) does not terminate such a string when it is the first character within it (after an initial `^`, if any) (for example, `[]a-f]` matches either a right bracket (`]`) or one of the letters `a` through `f`, inclusive). The four characters listed in item 2 stand for themselves within such a string of characters.

You can use the following rules to construct REs from 1-character REs:

5. A 1-character RE is a RE that matches whatever the 1-character RE matches.
6. A 1-character RE followed by an asterisk (`*`) is a RE that matches zero or more occurrences of the 1-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.
7. A 1-character RE followed by `\{`*m*`\}`, `\{`*m*`,\}`, or `\{`*m*`,`*n*`\}` is a RE that matches a range of occurrences of the 1-character RE. The values of *m* and *n* must be nonnegative integers less than 256; `\{`*m*`\}` matches exactly *m* occurrences; `\{`*m*`,\}` matches at least *m* occurrences; `\{`*m*`,`*n*`\}` matches any number of occurrences between *m* and *n*, inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.
8. The concatenation of REs results in an RE that matches the concatenation of the strings matched by each component of the RE.
9. A RE enclosed between the character sequences `\(` and `\)` is a RE that matches whatever the unadorned RE matches.
10. The expression `\`*n* matches the same string of characters as was matched by an expression enclosed between `\(` and `\)` earlier in the same RE. Here, *n* is a digit; the subexpression specified is that beginning with the *n*th occurrence of `\(` counting from the left. For example, the expression `^\(.*\)\1$` matches a line that consists of two repeated appearances of the same string.

Finally, an entire RE may be constrained to match only an initial or final segment of a line (or both).

11. A circumflex (`^`) at the beginning of an entire RE constrains that RE to match an initial segment of a line.
12. A dollar sign (`$`) at the end of an entire RE constrains that RE to match a final segment of a line.
13. The construction `^`*entire RE*`$` constrains the entire RE to match the entire line.

The null RE (such as `//`) is equivalent to the last RE encountered.

To understand addressing in `ed`, you must know that at any time there is a current line. Generally, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command. Addresses are constructed as follows:

1. The character `.` addresses the current line.
2. The character `$` addresses the last line of the buffer.

3. A decimal number *n* addresses the *n*th line of the buffer.
4. ′*x* addresses the line marked with the mark name character *x*, which must be an ASCII lowercase letter (`a` – `z`). Lines are marked with the `k` command described later in this man page.
5. A RE enclosed by slashes (`/`) addresses the first line found by searching forward from the line following the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched.
6. An RE enclosed in question marks (`?`) addresses the first line found by searching backward from the line preceding the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line.
7. An address followed by a plus sign (`+`) or a minus sign (`-`) followed by a decimal number specifies that address plus or minus the indicated number of lines. You can omit the plus sign.
8. If an address begins with `+` or `-`, the addition or subtraction is taken with respect to the current line (for example, `-5` means `.-5`).
9. If an address ends with `+` or `-`, 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of rule 8, in this list, the address `-` refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the `^` character in addresses is entirely equivalent to `-`.) Moreover, trailing `+` and `-` characters have a cumulative effect, so `- -` refers to the current line less 2.
10. For convenience, a comma (`,`) stands for the address pair `1,$`, and a semicolon (`;`) stands for the pair `.,$`.

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (`,`). They may also be separated by a semicolon (`;`). In the latter case, the current line (`.`) is set to the first address, and only then is the second address calculated. This feature can be used to determine the starting line for forward and backward searches (see rules 5 and 6, in the preceding list). The second address of any two-address sequence must correspond to a line that follows, in the buffer, the line corresponding to the first address.

In the following list of `ed` commands, the default addresses are shown in parentheses. The parentheses are not part of the address; they show that the specified addresses are the default.

It is generally illegal for more than one command to appear on a line. However, any command (except `e`, `f`, `r`, or `w`) may be suffixed by `l`, `n`, or `p`; in which case, the current line is listed, numbered, or printed, respectively, as discussed under the `l`, `n`, and `p` commands in the following list:

`(.)a`
<*text*>

`.` The command reads the text entered and appends it after the addressed line; `.` is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the appended text to be placed at the beginning of the buffer. The maximum number of characters that may be entered from a terminal is 256 per line (including the `<newline>` character).

`(.)b`*a* The browse command displays the addressed line and *a* lines below it; *a* is a number. The initial default for *a* is 22 lines; when you give a different value, that value becomes the default. `.` is left at the last line displayed. You may append the `b` command to any other command except `e`, `f`, `r`, or `w`.

`(.)c`
<text>
`.` The change command deletes the addressed lines, then accepts input text that replaces these lines; `.` is left at the last line input, or, if there were none, at the first line that was not deleted.

`(.,.)d` The delete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

`e` *file* The edit command deletes the entire contents of the buffer, and then reads in the specified file; `.` is set to the last line of the buffer. If no file name is given, the currently remembered file name, if any, is used (see the `f` command). The number of characters read is typed; *file* is remembered for possible use as a default file name in subsequent `e`, `r`, and `w` commands. If *file* is replaced by `!`, the rest of the line is taken to be a shell (`sh`(1)) command whose output is to be read. Such a shell command is not remembered as the current file name. See also the MESSAGES section.

`E` *file* The edit command is like `e`, except that the editor does not check to see whether any changes were made to the buffer since the last `w` command.

`f` *file* If you specify *file*, the file-name command changes the currently remembered file name to *file*; otherwise, it prints the currently remembered file name.

`(1,$)g/`*RE*`/`*command list*
In the global command, the first step is to mark every line that matches the specified RE. Then, for every such line, the given *command list* is executed with `.` initially set to that line. A single command or the first of a list of commands appears on the same line as the global command. You must end all lines of a multiline list, except the last line with a `\`; `a`, `i`, and `c` commands and associated input are permitted. You can omit the `.` terminating input mode if it would be the last line of the *command list*. An empty *command list* is equivalent to the `p` command. The `g`, `G`, `v`, and `V` commands are not permitted in the *command list*. See also the BUGS section.

`(1,$)G/`*RE*`/` In the interactive global command, the first step is to mark every line that matches the given RE. Then, for every such line, that line is printed, `.` is changed to that line, and any one command (other than one of the `a`, `c`, `i`, `g`, `G`, `v`, and `V` commands) may be input and is executed. After the execution of that command, the next marked line is printed, and so on; a `<newline>` acts as a null command; `&` causes the reexecution of the most recent command executed within the current invocation of `G`. The commands input as part of the execution of the `G` command may address and affect any lines in the buffer. You can terminate the `G` command with an interrupt signal (ASCII DEL or BREAK).

`h` The `h`elp command gives a short error message that explains the reason for the most recent `?` diagnostic message.

`H` The `H`elp command causes `ed` to enter a mode in which error messages are printed for all subsequent `?` diagnostic messages. It also explains the previous `?` if one exists. The `H` command alternately turns this mode on and off; it is initially off.

`help` The `help` command gives a one-page synopsis of `ed` commands.

`(.)i`
<text>
`.` The `i` command inserts the specified text before the addressed line; `.` is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the `a` command only in the placement of the input text. Address 0 is not legal for this command. The maximum number of characters that may be entered from a terminal is 256 per line (including the `<newline>` character).

`(.,.+1)j` The join command joins contiguous lines by removing the appropriate `<newline>` characters. If exactly one address is given, this command does nothing.

`(.)k` *x* The mark command marks the addressed line with name *x*, which must be a lowercase letter. The address *'x* then addresses this line; `.` is unchanged.

`(.,.)l` The list command prints the addressed lines in an unambiguous way: a few nonprinting characters (for example, `<tab>` and `<backspace>`) are represented by visually-mnemonic overstrikes. All other nonprinting characters are printed in octal, and long lines are folded. You may append an `l` command to any command other than `e`, `f`, `r`, or `w`.

`(.,.)m` *a* The move command repositions the addressed line(s) after the line addressed by `a`. Address 0 is legal for `a` and causes the addressed line(s) to be moved to the beginning of the file. Address `a` must not fall within the range of moved lines; `.` is left at the last line moved.

`(.,.)n` The number command prints the addressed lines, preceding each line by its line number and a tab character; `.` is left at the last line printed. You can append the `n` command to any other command other than `e`, `f`, `r`, or `w`.

`N` The `N` command toggles the functions of the `p` and `n` commands. Entering `N` causes `n` to function as `p` and vice versa. Another `N` toggles them back.

`(.)o`*a* The `o` command displays the addressed line and *a* lines above and below it; *a* is a number. The initial default for *a* is 11 lines; when you specify a different value, that value becomes the default. `.` is left at the line addressed by `o`. You can append the `o` command to any other command except `e`, `f`, `r`, or `w`.

`(.,.)p` The print command prints the addressed lines; `.` is left at the last line printed. You can append the `p` command to any other command other than `e`, `f`, `r`, or `w`. For example, `dp` deletes the current line and prints the new current line.

`P` The editor prompts with a `*` for all subsequent commands. The `P` command alternately turns this mode on and off; it is initially off.

`q` The quit command causes `ed` to exit. No automatic write of a file is done. See also the MESSAGES section.

`Q` The editor exits without checking whether changes have been made in the buffer since the last `w` command.

`($)r` *file* The read command reads in the given file after the addressed line. If you omit *file*, the currently remembered file name, if any, is used (see commands `e` and `f`). The currently-remembered file name is not changed unless *file* is the very first file name mentioned since `ed` was invoked. Address 0 is legal for `r` and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed; `.` is set to the last line read in. If *file* is replaced by `!`, the rest of the line is taken to be a shell (`sh`(1)) command whose output will be read. For example, `$r !ls` appends the file names in the current directory to the end of the file being edited. Such a shell command is not remembered as the current file name.

`(.,.)s/`*RE*`/`*replacement*`/` or
`(.,.)s/`*RE*`/`*replacement*`/g` or

(.,.)s/*RE/replacement*/n *n* = 1–512

The substitute command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (nonoverlapped) matched strings are replaced by the *replacement* if global replacement indicator g appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number *n* appears after the command, only the *n*th occurrence of the matched string on each addressed line is replaced. It is an error for the substitution to fail on all addressed lines. Any character other than <space> or <newline> may be used instead of / to delimit the RE and the *replacement*; . is left at the last line on which a substitution occurred.

An ampersand (&) that appears in the *replacement* is replaced by the string matching the RE on the current line. To suppress the special meaning of & in this context, precede it by \. As a more general feature, the characters *n*, where *n* is a digit, are replaced by the text matched by the nth regular subexpression of the specified RE enclosed between \(and \). When nested parenthesized subexpressions are present, *n* is determined by the number of occurrences of \(starting from the left. When the character % is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The % loses its special meaning when it is in a replacement string of more than one character or is preceded by a \.

You may split a line by substituting a <newline> character into it. The <newline> in the *replacement* must be escaped by preceding it with \. Such substitution cannot be done as part of a g or v command list.

(.,.)t*a* This command acts just like the m command, except that a copy of the addressed lines is placed after address a (which may be 0); . is left at the last line of the copy.

u The undo command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent a, c, d, g, i, j, m, r, s, t, v, G, or V command.

(1,$)v/*RE/command list*

This command is the same as the global command g except that the *command list* is executed with . initially set to every line that does not match the RE.

(1,$)V/*RE/* This command is the same as the interactive global command G except that the lines that are marked during the first step are those that do not match the RE.

`(1,$)w` *file* The write command writes the addressed lines into the specified file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless your `umask` setting (see `umask`(1)) dictates otherwise. The currently remembered file name is not changed unless *file* is the very first file name mentioned since `ed` was invoked. If you omit *file*, the currently remembered file name, if any, is used (see commands `e` and `f`); `.` is unchanged. If the command is successful, the number of characters written is typed. If *file* is replaced by `!`, the rest of the line is taken to be a shell (`sh`(1)) command whose standard input is the addressed lines. Such a shell command is not remembered as the current file name.

`X` An encryption key is requested from the standard input. Subsequent `e`, `r`, and `w` commands use this key to encrypt or decrypt the text (see `crypt`(1)). An explicitly empty key turns off encryption. Also, see the `-x` option of `ed`.

`z` The `z` command is a UNICOS extension that writes the file under the original name and then exits `ed`. It is functionally equivalent to entering the `w` and `q` commands.

`($)=` The line number of the addressed line is typed; `.` is unchanged by this command.

`!`*shell command* The remainder of the line after the `!` is sent to the UNIX system shell (`sh`(1)) to be interpreted as a command. Within the text of that command, unescaped character `%` is replaced with the remembered file name; if a `!` appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, `!!` repeats the last shell command. If any expansion is performed, the expanded line is echoed; `.` is unchanged.

`(.+1) <newline>`
An address alone on a line causes the addressed line to be printed. A `<newline>` alone is equivalent to `.+1p`; it is useful for stepping forward through the buffer.

If an interrupt signal (ASCII DEL or BREAK) is sent, `ed` prints `?` and returns to its command level.

Some size limitations: 512 characters per line, 256 characters per global command list, and 64 characters per file name. The limit on the number of lines depends on the amount of user memory: each line takes 1 word.

When reading a file, `ed` discards ASCII null characters.

If a file is not terminated by a `<newline>` character, `ed` adds one and outputs a message that explains what it did.

If the closing delimiter of a RE or of a replacement string (such as, `/`) would be the last character before a new line, that delimiter may be omitted, in which case the addressed line is printed. The following pairs of commands are equivalent:

```
s/s1/s2   s/s1/s2/p
g/s1      g/s1/p
?s1       ?s1?
```

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.

The `-` option, although supported in this release for upward compatibility, will no longer be supported in the next major release of the system. Convert shell scripts that use the `-` option so that they use the `-s` option instead.

Reasonable editing sessions should be kept under 10 Mbytes. Lines are limited to 4096 characters.

When reading a file, `ed` discards ASCII null characters and all characters after the last `<newline>`. The `ed` utility cannot edit files (such as `a.out`) that contain characters that are not in the ASCII set (bit 8 on).

Size limitations: Large files generate larger editor temporary files and cost many processor cycles on entry to `ed`.

Files encrypted on Cray Research systems using a UNICOS release prior to 5.0 must be reencrypted using `crypt`(1) before `ed -x` can read them.

EXIT STATUS

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

0 Successful completion without any file or command errors.

>0 An error occurred.

MESSAGES

? For command errors.

?*file* For an inaccessible file (use the `help` and `Help` commands for detailed explanations).

When changes have been made in the buffer since the last `w` command that wrote the entire buffer, `ed` warns the user if an attempt is made to destroy the `ed` buffer by using the `e` or `q` commands. It prints `?` and allows editing to continue. A second `e` or `q` command at this point takes effect. The `-s` command-line option inhibits this feature.

BUGS

The `!` command cannot be subject to a `g` or a `v` command.

You cannot use the `!` command and the `!` escape from the `e`, `r`, and `w` commands if you invoke the editor from a restricted shell (see `sh`(1)).

The sequence `\n` in a RE does not match a `<newline>` character.

If the editor input is coming from a command file (for example, `ed file < ed-cmd-file`), the editor will exit at the first failure.

FILES

`/usr/tmp`	Default directory for temporary work file.
`TMPDIR`	If this environmental variable is not null, its value is used in place of `/usr/tmp` as the directory name for the temporary work file.
`ed.hup`	Work is saved in this file if the terminal is hung up.

SEE ALSO

`chown`(1), `crypt`(1), `edit`(1), `ex`(1), `grep`(1), `sed`(1), `sh`(1), `stty`(1), `umask`(1), `vi`(1)

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

NAME

edit – Text editor (variant of ex(1))

SYNOPSIS

edit [-r] [-x] *name*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The edit editor is a variant of the text editor ex(1) that is recommended for new or casual users who want to use a command-oriented editor.

The edit command accepts the following options:

-r Recovers file after an editor or system crash.

-x Encryption option; when this option is used, the file is encrypted as it is being written and requires an encryption key to be read (see crypt(1)). Also, see the WARNINGS section of this man page.

name Specifies the name of a file to edit.

The following introduction will help you get started with edit. If you are using a CRT terminal you may want to learn about the display editor vi(1).

To edit the contents of an existing file, enter the command edit *name* to the shell. edit makes a copy of the file, which you can then edit, and tells you how many lines and characters are in the file. To create a new file, make up a name for the file and try to run edit on it; this will create an error message.

The edit command prompts for commands by using the colon (:) character, which you see after starting the editor. If you are editing an existing file, you will have some lines in the edit buffer (its name for the copy of the file you are editing). Most commands to edit use the current line unless you specify the line to be used. If you specify print (p), and press the carriage return (as you should after all edit commands), this current line will be printed. If you delete (d) the current line, edit will print the new current line. When you start editing, edit makes the last line of the file the current line. If you delete the current line, the next line in the file becomes the current line. (Deleting the last line is a special case.)

If you start with an empty file or want to add new lines, the append (a) command can be used. After you enter this command, edit reads lines from your terminal, placing these lines after the current line, until you enter a line consisting of just a period (.). The last line you typed then becomes the current line. The command insert is like append, but places the lines you enter before, rather than after, the current line.

The edit command numbers the lines in the buffer, with the first line having number 1. If you enter the command 1, edit will type the first line. If you then enter the command delete, edit will delete the first line, line 2 will become line 1, and edit will print the current line (the new line 1). In general, the current line is always the last line affected by a command.

You can make a change to text within the current line by using the `substitute` (`s`) command. Enter `s/`*old*`/`*new*`/`, where *old* is where you insert the old characters you want replaced and *new* is where you insert the new characters to replace the old characters.

The command `file` (`f`) tells you how many lines are in the buffer you are editing, and indicates `[Modified]` if you have changed it. After modifying a file, you can put the buffer text back to replace the file by issuing a `write` (`w`) command. You can then leave the editor by issuing a `quit` (`q`) command. If you run `edit` on a file but do not change the file, it is not necessary to `write` the file back. If you try to `quit` from `edit` after modifying the buffer without writing it out, you will be warned that there has been `No write since last change` and `edit` will wait for another command. If you wish not to `write` the buffer out, then you can issue another `quit` command. The buffer is then irretrievably discarded, and you return to the shell.

By using the `delete` and `append` commands and specifying line numbers to see lines in the file, you can make any changes. You should learn at least a few more things, however, if you are going to use `edit` very often.

The `change` (`c`) command changes the current line to a sequence of lines you supply (as in `append`, you supply lines up to a line consisting of only a period (`.`). You can use `change` to change more than one line by giving the line numbers of the lines you want to change, for example, `3,5change`. You can print lines this way too. The command `1,23p` prints the first 23 lines of the file.

The `undo` (`u`) command reverses the effect of the last command that changed the buffer. If you issue a `substitute` command that does not do what you want, you can enter `undo` to restore the old contents of the line. You can also undo an `undo` command. `edit` gives you a warning message when commands you issue affect more than one line of the buffer. If the amount of change seems unreasonable, you should consider issuing an `undo` and looking to see what happened. If you decide that the change was correct, you can `undo` again to get it back. Commands such as `write` and `quit` cannot be undone.

To look at the next line in the buffer, press the carriage return key. To look at a number of lines, press `<CONTROL-d>` (hold down the `CONTROL` key and the `d` key at the same time) rather than carriage return. This shows you a half-screen of lines on a CRT or 12 lines on a hardcopy terminal. You can look at the text surrounding the line on which you are working by issuing the command `z`. The current line is then the last line printed; you can get back to the line where you were before the `z` command by entering ´´. The `z` command can also be given other following characters: `z-` prints a screen of text (or 24 lines) ending where you are; `z+` prints the next screenful. If you want less than a screenful of lines, type in `z.12` to get 12 lines total. You can delete five lines, starting with the current line, with the command `delete 5` .

To find things in the file, you can use line numbers. Note that the line numbers change when you insert and delete lines. You can search backward and forward in the file for strings by giving commands of the form `/text/` to search forward, for *text* or `?text?` to search backward. If a search reaches the end of the file without finding the text, it wraps around, and continues to search until it reaches the line from which you entered the command. A useful feature here is a search of the form `/^text/`, which searches for text at the beginning of a line. Similarly `/text$/` searches for text at the end of a line. You can omit the trailing / or ? in these commands.

The current line has a symbolic name, a period (`.`). This is most useful in a range of lines as in the `.,$print` command, which prints the rest of the lines in the file. To get to the last line in the file, you can refer to it by its symbolic name, `$`. The command `$ delete` or `$d` deletes the last line in the file, no matter which was the current line. Arithmetic with line references is also possible. The command `$-5` takes you to the fifth line before the last line, and `.+20` moves to 20 lines after the present line.

You can determine the current line by entering `.=`. This is useful if you want to move or copy a section of text within a file or between files. Find out the first and last line numbers you want to copy or move (for example, 10 to 20). To move lines, you can enter `10,20delete a`, which deletes these lines from the file and places them in a buffer named `a`. The `edit` command has 26 buffers named a through z. You can get these lines back by entering `put a`, which will put the contents of buffer `a` after the current line. If you want to move or copy these lines between files, copy the lines, and enter an `edit` (`e`) command, following it with the name of the other file you want to edit, for example, `edit chapter2`. You can also use the `yank` command instead of `delete` to copy or move lines. If the text you want to copy or move is within one file, you can enter `10,20move $`. It is not necessary to use named buffers in this case.

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.

SEE ALSO

`crypt`(1), `ed`(1), `ex`(1), `vi`(1)

NAME

emacs – GNU project Emacs

SYNOPSIS

emacs [*file* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

FSF

DESCRIPTION

GNU Emacs is written by Richard Stallman, the author of the original (PDP-10) Emacs. The primary documentation of GNU Emacs is in the *GNU Emacs Manual*, which you can read online using Info, a subsystem of Emacs. Please look there for complete and up-to-date documentation.

The user functionality of GNU Emacs encompasses everything other Emacs editors do, and it is easily extensible because its editing commands are written in Lisp.

emacs has an extensive interactive help facility, but the facility assumes that you know how to manipulate Emacs windows and buffers. <CONTROL-h> (backspace or <CONTROL-h>) enters the help facility. Help Tutorial (<CONTROL-h t>) requests an interactive tutorial. This tutorial can teach beginners the fundamentals of emacs in a few minutes. Help Apropos (<CONTROL-h a>) helps you find a command given its functionality. Help Character (<CONTROL-h c>) describes a given character's effect. Help Function (<CONTROL-h f>) describes a given Lisp function specified by name.

The emacs Undo command can undo several steps of modification to your buffers; therefore you can easily recover from editing mistakes.

The GNU Emacs contains many special packages that handle mail reading (RMail) and sending (Mail), outline editing (Outline), compiling (Compile), running subshells within Emacs windows (Shell), and running a Lisp read-eval-print loop (Lisp-Interaction-Mode).

An extensive reference manual exists, but users of other versions of Emacs should have little trouble adapting even without a copy. Users new to Emacs use basic features fairly rapidly by studying the tutorial and using the self-documentation features.

emacs accepts the following options:

file Edits *file*.

+*number* Goes to the line specified by *number* (does not insert a space between the "+" sign and the number).

`-d` *displayname*
Creates the Emacs window on the display specified by *displayname.* This must be the first argument listed in the command line.

`-q` Does not load an init file.

`-u` *user* Load *user's* init file.

`-t` *file* Use specified *file* as the terminal, rather than using `stdin`/`stdout`. This must be the first argument specified in the command line.

The following options are Lisp-oriented (these options are processed in the order encountered):

`-f` *function* Execute the Lisp function *function*.

`-l` *file* Load the Lisp code in file *file*.

The following options are useful when running `emacs` as a batch editor:

`-batch` *commandfile*
Edit in batch mode by using the commands found in *commandfile*. The editor sends messages to `stdout`. This option must be the first one in the argument list.

`-kill` Exit `emacs` while in batch mode.

Using `emacs` with X Windows

`emacs` has been tailored to work well with the X Window System environment. If you run `emacs` using the X Window System, it will create its own display window. You should start the editor as a background process so that you can continue using your original window. `emacs` can be started with the following X Window System switches:

`-r` Displays the Emacs window in reverse video.

`-i` Uses the "gnu" bit map icon when iconifying the Emacs window.

`-font` *font* Sets the Emacs window's font to that specified by *font*. Emacs accepts only fixed-width fonts. Under the X11 Release 4 font-naming conventions, any font with the value `m` or `c` in the eleventh field of the font name is a fixed-width font. Furthermore, fonts whose name are of the form 'width x height' are generally fixed-width, as is the font `fixed`.

When you specify a font, do not include the *.onx* extension; be sure to put a space between the `-font` switch and the font specification argument.

`-b` *pixels* Sets the Emacs window's border width to the number of pixels specified by *pixels*. The defaults is one pixel on each side of the window.

`-ib` *pixels* Sets the Emacs window's internal border width to the number of pixels specified by *pixels*. The defaults is one pixel of padding on each side of the window.

`-geometry` *geometry*
Sets the Emacs window's width, height, and position as specified. The geometry specification is in the standard X format. The width and height are specified in characters; the default is 80 by 24.

`-fg` *color* On color displays, sets the color of the text to *color*.

`-bg` *color* On color displays, sets the color of the window's background to *color*.

`-bd` *color* On color displays, sets the color of the window's border to *color*.

`-cr` *color* On color displays, sets the color of the window's text cursor to *color*.

`-ms` *color* On color displays, sets the color of the window's mouse cursor.

`-d` *displayname*
Creates the Emacs window on the display specified by *displayname*. This option must be the first one specified in the command line.

`-nw` Disables the Emacs special interface to the X Window System environment. If you use this switch when invoking Emacs from an *xterm* window, display is done in the *xterm* window. This option must be the first one specified in the command line.

You can set *X* default values for your Emacs windows in your *.Xdefaults* file. Use the following format:

`emacs.`*keyword*`:`*value*

where *value* specifies the default value of *keyword*.

Emacs lets you set default values for the following keywords:

`BodyFont` Sets the window's text font.

`ReverseVideo` If the value of `ReverseVideo` is set to `on`, the window will be displayed in inverse video.

`BitMapIcon` If the value of `BitMapIcon` is set to `on`, the window will iconify using the "gnu" icon.

`BorderWidth` Sets the window's border width in pixels.

`Foreground` For color displays, sets the window's text color.

`Background` For color displays, sets the window's background color.

`Border` For color displays, sets the color of the window's border.

`Cursor` For color displays, sets the color of the window's text cursor.

`Mouse` For color displays, sets the color of the window's mouse cursor.

If you try to set color values when using a black and white display, the window's characteristics will default as follows: the foreground color will be set to black, the background color will be set to white, the border color will be set to gray, and the text and mouse cursors will be set to black.

Using the Mouse

The following table lists the key bindings for the mouse cursor when used in an Emacs window.

	Left Button	Middle Button	Right Button
Unshifted	Select window set point	Paste text	Cut text
Shift		Cut text	Paste text
Control		Cut & wipe text	Select & split into two wndows
Control + Shift	X buffer menu †	X help menu ‡	Keep one window

† X buffer menu: hold the buttons and keys down, wait for menu to appear, select buffer, and release. Move mouse out of menu and release to cancel.

‡ X help menu: pop up index card menu for Emacs help.

Manuals

You can order printed copies of the *GNU Emacs Manual* from the Free Software Foundation, which develops GNU software. See the file `ORDERS` for ordering information.

NOTES

Emacs is free; anyone may redistribute copies of Emacs to anyone under the terms stated in the Emacs General Public License, a copy of which accompanies each copy of Emacs and which also appears in the reference manual.

Copies of Emacs may sometimes be received packaged with distributions of UNIX systems, but it is never included in the scope of any license covering those systems. Such inclusion violates the terms on which distribution is permitted. In fact, the primary purpose of the General Public License is to prohibit anyone from attaching any other restrictions to redistribution of Emacs.

Richard Stallman encourages you to improve and extend Emacs. You contribute your extensions to the GNU library. Eventually GNU (GNU's Not UNIX) will be a complete replacement for Berkeley UNIX. Everyone will be able to use the GNU system for free.

AUTHORS

Emacs was written by Richard Stallman and the Free Software Foundation. Joachim Martillo and Robert Krawitz added the X Window System features.

FILES

`/usr/src/prod/emacs/src`
: C source files and object files.

`/usr/lib/emacs/lisp`
: Lisp source files and compiled files that define most editing commands. Some are preloaded; others are autoloaded from this directory when used.

`/usr/src/prod/emacs/man`
Sources for the Emacs Reference Manual.

`/usr/lib/emacs/etc`
Various programs that are used with GNU Emacs, and some files of information.

`/usr/lib/emacs/etc/DOC.*`
Contains the documentation strings for the Lisp primitives and preloaded Lisp functions of GNU Emacs. They are stored here to reduce the size of Emacs proper.

`/usr/lib/emacs/etc/DIFF`
Discusses GNU Emacs versus Twenex Emacs.

`/usr/lib/emacs/etc/CCADIFF`
Discusses GNU Emacs versus CCA Emacs.

`/usr/lib/emacs/etc/GOSDIFF`
Discusses GNU Emacs versus Gosling Emacs.

`/usr/lib/emacs/info`
Identifies files to which the Info documentation browser (a subsystem of Emacs) may refer. Currently not much of UNIX is documented here, but the complete text of the Emacs reference manual is included in a convenient tree-structured form.

BUGS

The shell will not work with programs running in raw mode on some version of UNIX software.

There is a mailing list, `bug-gnu-emacs@prep.ai.mit.edu` on the internet (`ucbvax!prep.ai.mit.edu!bug-gnu-emacs` on UUCPnet), for reporting Emacs bugs and fixes. But before reporting something as a bug, try to be sure that it really is a bug, and not a misunderstanding or a deliberate feature. Read the section ''Reporting Emacs Bugs'' near the end of the Emacs Reference Manual (or Info system) for hints on how and when to report bugs. Also, include the version number of the Emacs you are running in *every* bug report that you submit.

Do not expect a personal answer to a bug report. Reporting bugs gets them fixed for everyone in the next release, if possible.

Do not send anything but bug reports to this mailing list. Send requests to be added to mailing lists to the special list `info-gnu-emacs-request@prep.ai.mit.edu` (or the corresponding UUCP address). For more information about Emacs mailing lists, see the file `/usr/lib/emacs/etc/MAILINGLISTS`. Bugs tend to be fixed if they can be isolated, so it is in your interest to report them in such a way that they can be reproduced easily.

NAME

env – Sets environment for command execution

SYNOPSIS

env [-i] [*env-vars*]... [*utility* [*args*]]

Obsolescent version; may not be supported in future releases:

env [-] [*env-vars*]... [*utility* [*args*]]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The env utility obtains the current environment, modifies it according to its arguments, and then executes the *utility* and its *args*, if any, with the modified environment.

You can specify zero or more *env-vars* arguments that must be of the form *name=value*. These arguments are merged into the inherited environment before *utility* is executed.

The env utility accepts the following options:

-

-i Causes the inherited environment to be ignored completely, so that *utility* is executed with the environment specified by the arguments.

If you omit *utility*, the resulting environment is printed, one name-value pair per line.

CAUTIONS

The env utility allows a maximum of 255 environment variable names.

STATUS EXIT

If the *utility* utility is invoked, the exit status of env is the exit status of *utility*; otherwise, the env utility exits with one of the following values:

0 The env utility successfully completed.

1–125 An error occurred in the env utility.

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

127 The utility specified by *utility* could not be found.

EXAMPLES

The following example performs a profile run of `mycode.f` source file and alters the size of the statistical buckets only for the duration of the run. The file is compiled by using `f90`(1). `segldr`(1) then loads with the `prof` library. Finally, `env` sets the value of the `PROF_WPB` environment variable to 1 only for the duration of this
execution.

```
f90 -g mycode.f
segldr -l prof mycode.o
env PROF_WPB=1 a.out
```

If the `PROF_WPB` environment variable was set previously to some value in your global environment variables, the `env` execution will temporarily override the value.

SEE ALSO

`sh`(1),

`exec`(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

eqn, neqn, checkeq – Typesets mathematics

SYNOPSIS

```
eqn [-dxy] [-fn] [-pn] [-sn] [-Tdev] [filename] ...
neqn [filename] ...
checkeq [filename] ...
```

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The eqn and neqn commands are language processors to assist in describing equations. eqn is a preprocessor for troff(1) and is intended for devices that can print output from troff. The neqn command is a preprocessor for nroff(1) and is intended for use with terminals.

The checkeq command reports missing or unbalanced delimiters and .EQ/.EN pairs.

If no *filename*s are specified, eqn and neqn read from the standard input. A line beginning with .EQ marks the start of an equation; the end of an equation is marked by a line beginning with .EN. Neither of these lines is altered, so they may be defined in macro packages to get centering, numbering, and so on. It is also possible to set 2 characters as delimiters; subsequent text between delimiters is also treated as eqn input.

The following options are available for eqn and neqn:

-d*xy* Sets equation delimiters to characters *x* and *y* with the command-line argument. The more common way to do this is with delim*xy* between .EQ and .EN. The left and right delimiters may be identical. Delimiters are turned off by delim off appearing in the text. All text that is neither between delimiters nor between .EQ and .EN is passed through untouched.

-f*n* Changes font to *n* globally in the document. The font can also be changed globally in the body of the document by using the gfont directive.

-p*n* Reduces subscripts and superscripts by *n* point sizes from the previous size. In the absence of the -p option, subscripts and superscripts are reduced by 3 point sizes from the previous size.

-s*n* Sets equations in point size *n* globally in the document. The point size can also be changed globally in the body of the document by using the gsize directive.

-T*dev* Prepares output for device *dev*. If no -T option is present, eqn looks at the environment variable TYPESETTER to see what the intended output device is. If no such variable is found in the environment, a system-dependent default device is assumed. Not available using neqn.

filename Specifies the file to be typeset.

eqn Language

Tokens within `eqn` are separated by braces, double quotation marks, tildes, circumflexes, space, tab, or newline characters. Braces `{ }` are used for grouping; generally speaking, anywhere a single character like *x* could appear, a complicated construction enclosed in braces may be used instead. Tilde (`~`) represents a full space in the output, circumflex (`^`) half as much.

Subscripts and superscripts are produced with the keywords `sub` and `sup`. Thus, `x sub i` makes x_i, `a sub i sup 2` produces a_i^2 and `e sup {x sup 2 + y sup 2}` gives $e^{x^2+y^2}$.

Fractions are made with the keyword `over`: `a over b` yields $\frac{a}{b}$.

Square roots are made with the keyword `sqrt`: `1 over down 10 sqrt {ax sup 2 +bx+c}` results in

$$\frac{1}{\sqrt{ax^2+bx+c}}.$$

Although `eqn` tries to get most things at the right place on the paper, occasionally you will need to tune the output to make it just right. In the previous example, a local motion, *down 10*, was used to get more space between the square root and the line above it.

The keywords `from` and `to` introduce lower and upper limits on arbitrary things: `lim from {n-> inf } sum from 0 to n x sub i` produces $\lim_{n\to\infty}\sum_{0}^{n} x_i$.

Left and right brackets, braces, and so on, of the right height made with the keywords `left` and `right`: `left [ x sup 2 + y sup 2 over alpha right ] ~=~1` produces

$$\left[x^2+\frac{y^2}{\alpha} \right] = 1.$$

The `right` clause is optional. Legal characters after `left` and `right` are braces, brackets, bars, `c` and `f` for ceiling and floor, and `""` for nothing at all (useful for a right-side-only bracket).

Vertical piles of things are made with the keywords `pile`, `lpile`, `cpile`, and `rpile`: `pile {a above b above c}` produces $\begin{matrix} a \\ b \\ c \end{matrix}$.

There can be an arbitrary number of elements in a pile. `lpile` left justifies, `pile` and `cpile` center, with different vertical spacing, and `rpile` right justifies.

Matrices are made with the keyword `matrix`: `matrix { lcol { x sub i above y sub 2 } ccol { 1 above 2 } }` produces

$$\begin{matrix} x_i & 1 \\ y_2 & 2 \end{matrix}.$$

In addition, there is the `rcol` keyword for a right-justified column.

Diacritical marks are made with the keywords `dot`, `dotdot`, `hat`, `tilde`, `bar`, `vec`, `dyad`, and `under`: `x dot = f(t) bar` produces $\dot{x} = \overline{f(t)}$, `y dotdot bar ~=~ n under` produces $\ddot{\bar{y}} = \underline{n}$, and `x vec ~=~ y dyad` produces $\vec{x} = \vec{\vec{y}}$.

Sizes and font can be changed with `size` *n* or `size` ±*n*, `roman`, `italic`, `bold`, and `font` *n*. Size and fonts can be changed globally in a document by `gsize` *n* and `gfont` *n*, or by the command-line arguments `-s`*n* and `-f`*n*.

Successive display arguments can be lined up. Place `mark` before the desired lineup point in the first equation; place `lineup` at the place that is to line up vertically in subsequent equations.

Shorthands may be defined or existing keywords redefined with `define`:

`define` *thing* `%` *replacement* `%`

This example shows a new token called *thing* that will be replaced by *replacement* whenever it appears thereafter. The `%` may be any character that does not occur in *replacement*.

Keywords like `sum` (∑), `int` (∫), `inf` (∞), and shorthands like `>=` (≥), `->` (→), and `!=` (≠) are recognized. Greek letters are spelled out in the desired case, as in `alpha` or `GAMMA`. Mathematical words like sin, cos, and log are made Roman automatically. `troff`(1) four-character escapes, like \(bu (•), can be used anywhere. Strings enclosed in double quotation marks `"`...`"` are passed through untouched; this permits keywords to be entered as text, and can be used to communicate with `troff`(1) when all else fails.

NOTES

To make digits, parentheses, and so on, appear in bold, enclose them in quotation marks, as in `bold "12.3"`.

EXAMPLES

```
eqn filename ... | troff

neqn filename ... | nroff
```

SEE ALSO

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

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

NAME

etags – Generates tag file for Emacs

SYNOPSIS

etags [-CDSTVHadt] [-i *file*] [-o *outfile*] [-defines] [-no-defines] [-c++] [-typedefs] [-typedefs-and-c++] [-ignore-indentation] [-help] [-version] [-include=*file*] [-output=*outfile*] [-append] *file* ...

IMPLEMENTATION

All Cray Research Systems

DESCRIPTION

The etags program is used to create a tag table file, in a format understood by emacs(1). This program understands the syntax of C, Fortran, LaTeX, Scheme and Emacs Lisp/Common Lisp. It reads the files specified on the command line, and writes a tag table (default is TAGS) in the current working directory. The programs recognize the language used in an input file based on its file name and contents; there are no switches for specifying the language.

etags accepts the following options and unambiguous abbreviations for long option names.

-d, -defines
: Creates tag entries for C preprocessor definitions. This is the default.

-D, -no-defines
: Does not create tag entries for C preprocessor definitions.

-C, -c++
: Treats files with .c and .h extensions as C++ code, not C code. Files with .C, .H, .cxx, .hxx, or .cc extensions are always assumed to be C++ code.

-t, -typedefs
: Records typedefs in C code as tags.

-T, -typedefs-and-c++
: Generates tag entries for typedefs, struct, enum, and union tags, and C++ member functions.

-S, -ignore-indentation
: Does not assume that a closing brace in the first column is the final brace of a function or structure definition.

-H, -help
: Prints usage information.

-V, -version
: Prints the current version of the program.

-i *file*, -include=*file*
: Includes a note in tag file indicating that, when searching for a tag, one should also consult the tags file *file* after checking the current file.

-o *outfile*, -output=*outfile*
: Explicit name of file for tag table (ignored with -v or -x).

-a, -append
: Appends to existing tag file.

file Name of file to be used to generate tag file.

SEE ALSO

`cxref`(1), `emacs`(1), `vgrind`(1)

NAME

ex – Text editor

SYNOPSIS

ex [-C] [-r] [-R] [-l] [-L] [-V] [-s] [-c *command*] [-t *tag*] [-w *n*] [-x] [*files*]
ex [-C] [-r] [-R] [-l] [-L] [-V] [-v] [-c *command*] [-t *tag*] [-w *n*] [-x] [*files*]

Obsolescent version; may not be supported in future releases:

ex [-C] [-r] [-R] [-l] [-L] [-V] [-] [+ *command*] [-t *tag*] [-w *n*] [-x] [*files*]
ex [-C] [-r] [-R] [-l] [-L] [-V] [-v] [+ *command*] [-t *tag*] [-w *n*] [-x] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX
AT&T extensions (-C, -l, -L, -V and -x options)

DESCRIPTION

The ex editor is the root of a family of editors: ex and vi(1). ex is a superset of ed(1), with the most notable extension being a display editing facility. Display-based editing is the focus of vi.

If you have a CRT terminal, you may want to use a display-based editor; in this case, see vi(1), which is a utility that focuses on the display editing portion of ex.

For ed Users

If you have used ed(1), you will find that ex has several new features useful on CRT terminals. Generally, the ex editor uses far more of the capabilities of terminals than ed(1) does and uses the terminal capability database (see the terminfo(5) man page) and the type of the terminal you are using from the TERM environment variable in the environment to determine how to drive your terminal efficiently. The editor makes use of features such as insert and delete character and line in its visual command (which can be abbreviated vi) and which is the central mode of editing when using vi(1).

The ex editor contains several new features to easily view the text of the file. The z command gives easy access to windows of text. Press <CONTROL-d> to scroll a half-window of text. This is more useful for stepping quickly through a file than just pressing <RETURN>. Of course, the screen-oriented visual mode gives constant access to editing context.

The ex editor gives you more help when you make mistakes. The undo (u) command lets you reverse any single change that has undesired results. ex gives you a lot of feedback, normally printing changed lines, and indicates when more than a few lines are affected by a command so that it is easy to detect when a command has affected more lines than it should.

The ex editor also normally prevents you from overwriting existing files. If the system (or editor) crashes, or if you accidentally hang up the telephone, you can use the recover command to retrieve your work. This command returns you back to within a few lines of where you stopped.

The ex editor has several features for dealing with more than one file at a time. You can specify a list of files on the command line and use the next (n) command to deal with each file in turn. You can also give the next command a list of file names or a pattern, as used by the shell, to specify a new set of files to be edited. In general, file names in the editor may be formed with full shell metasyntax. Metacharacter % is also available in forming file names and is replaced by the name of the current file.

For moving text between files and within a file, the editor has a group of buffers, named a through z. You can place text in these named buffers and carry it over when you edit another file.

The & command in ex repeats the last substitute command. In addition, the confirmed substitute command exists. You specify a range of substitutions to be made, and the editor interactively asks whether each substitution is desired.

You can ignore the case of letters in searches and substitutions. ex also allows you to construct regular expressions that match words to be constructed. This is convenient, for example, in searching for the word edit if your document also contains the word editor.

The ex editor has a set of options that you can set to tailor it to your liking. One very useful option is the autoindent option, which allows the editor to supply leading white space automatically to align text. You can then use the <CONTROL-d> key as a backtab and space and tab forward to align new code easily.

Miscellaneous new useful features include an intelligent join (j) command, which supplies white space between joined lines automatically; commands < and >, which shift groups of lines; and the ability to filter parts of the buffer through commands such as sort.

The ex editor accepts the following options and operands:

-c *command*
+ *command* (Obsolescent)
Begins editing by executing the specified editor search or positioning *command*.

-C Same as -x but assumes *file* began in encrypted form.

-l Sets lisp mode.

-L Prints a list of all recoverable files. Same as -r with no *file* specified.

-V Sets verbose mode.

-r Recovers *file* after an editor or system crash. If no *file* is given, a list of all recoverable files available to the user are written.

-R Sets readonly mode; prevents accidental overwriting of a file.

-s
- (Obsolescent)
Suppresses all interactive-user feedback. This is useful in processing editor scripts.

`-t` *tag*	Edits the file that contains the *tag* and positions the editor at its definition.
`-v`	Invokes `vi`.
`-w` *n*	Sets the value of the window editor option to *n*.
`-x`	Sets encryption option. When this option is used, the file is encrypted as it is written and will require an encryption key to be read (see `crypt`(1)). Also, see the WARNINGS section.
files	Specifies one or more path names of files to be edited.

ex States

Command	Normal and initial state. A colon (`:`) prompts for input. Your kill character cancels partial command.
Insert	Entered by `a`, `i`, or `c`. You may enter arbitrary text. Insert is terminated normally by a line having only `.` on it, or abnormally with an interrupt.
Visual	Entered by `vi`, terminates with `Q` or ^\.

ex Command Names and Abbreviations

abbrev	ab	next	n	undo	u
append	a	number	nu	unmap	unm
args	ar	preserve	pre	version	
change	c	print	p	visual	vi
copy	co	put	pu	write	w
delete	d	quit	q	xit	x
edit	e	read	re	yank	ya
file	f	recover	rec	window	z
global	g	rewind	rew	escape	!
insert	i	set	se	lshift	<
join	j	shell	sh	print next	<return>
list	l	source	so	resubst	&
map	map	stop	st	rshift	>
mark	ma	substitute	s	scroll	^D
move	m	unabbrev	una		

ex Command Addresses

n	Line *n*	/*pat*	Next with *pat*
.	Current	?*pat*	Previous with *pat*
$	Last	*x*-*n*	*n* before *x*
+	Next	*x*,*y*	*x* through *y*
-	Previous	´*x*	Marked with *x*
+*n*	*n* forward	´´	Previous context
%	1,$		

Initializing Options

`EXINIT`	Places `set`'s here in environment variable
`$HOME/.exrc`	Editor initialization file
`./.exrc`	Editor initialization file
`set` *x*	Enable option
`set no`*x*	Disable option
`set` *x*=*val*	Gives value *val*
`set`	Shows changed options
`set all`	Shows all options
`set` *x*?	Shows value of option *x*

Most Useful Options

`autoindent`	`ai`	Supplies indent
`autoprint`	`ap`	Prints current line after buffer changed
`autowrite`	`aw`	Writes before changing files
`beautify`	`bf`	Discards all control characters other than tab, newline, and form-feed
`directory`	`dir`	Specifies the directory
`edcompatible`	`ed`	Causes suffixes to be remembered
`flash`		Flashes screen on errors
`hardtabs`		Value of any hardware tab settings
`ignorecase`	`ic`	Ignore case
`lisp`		Modifies commands for lisp code
`list`		Prints `^I` for tab, `$` at end
`magic`		`. [ *` special in patterns
`mesg`		Allows non-`vi` messages to appear on screen during `vi`
`number`	`nu`	Numbers lines
`paragraphs`	`para`	Macro names that start ...
`prompt`		When set, prompted with a colon (:)
`readonly`		Disallows writing buffer contents to current file name
`redraw`		Simulates smart terminal
`remap`		Macro translation
`report`		Indicates number of lines that must be changed
`scroll`		Command mode lines
`sections`	`sect`	Macro names ...
`shell`		Shell executed when doing :! or !
`shiftwidth`	`sw`	For `< >`, and input `^D`
`showmatch`	`sm`	To `)` and `}` as 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		Shortens error messages
timeout		Must enter a macro name in less than 1 second
warn		Gives warning when :! is issued
window		Visual mode lines
wrapscan	ws	Around end of buffer?
wrapmargin	wm	Splits line automatically
writeany	wa	Allows a write to any file

Scanning Pattern Formation

^	Beginning-of-line
$	End-of-line
.	Any character
\<	Beginning-of-word
\>	End-of-word
[*str*]	Any character in *str*
[^*str*]	... not in *str*
[*x*-*y*]	... between *x* and *y*
*	Any number of preceding

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.

Files that were encrypted on UNICOS systems prior to release 5.0 must be re-encrypted using crypt(1) before ex -x can read them.

EXIT STATUS

The ex utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

BUGS

The undo command causes all marks to be lost on lines changed and then restored if the marked lines were changed.

The undo command never clears the buffer modified condition.

The z command prints a number of logical lines rather than physical lines. More than a screenful of output may result if long lines are present.

File input/output errors do not print a name if the command-line - option is used.

There is no easy way to do a single scan that ignores case.

The editor does not warn you when text is placed in named buffers and not used before exiting the editor.

Null characters are discarded in input files and cannot appear in resultant files.

FILES

`/usr/lib/exstrings`	Error messages
`/usr/lib/exrecover`	Recovers command
`/usr/lib/expreserve`	Preserves command
`/usr/lib/terminfo`	Describes capabilities of terminals
`HOME/.exrc`	Editor start-up file
`./.exrc`	Editor start-up file
`TMPDIR/Ex`*nnnnn*	Editor temporary
`TMPDIR/Rx`*nnnnn*	Named buffer temporary
`/usr/preserve/`*login*	Preservation directory (where *login* is the user's login)

SEE ALSO

`awk`(1), `crypt`(1), `ed`(1), `edit`(1), `emacs`(1), `grep`(1), `sed`(1), `vi`(1)

`curses`(3) (available only online)

`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

exdf – Transfers files to or from the IOS partition on the system console (expander disk for the CRAY EL series)

SYNOPSIS

```
exdf -i directory/file [-r] [-s]
exdf -o directory/file [-r] [-s]
```

IMPLEMENTATION

CRAY J90 series
CRAY EL series

DESCRIPTION

The exdf command can read or write a file on the IOS disk. The program determines from the command line whether it is reading or writing a file.

The exdf command accepts the following options:

-i Indicates input from a file.

-o Indicates output to a file.

-r Indicates that a file being output can replace an existing file. The default is to abort a transfer file if the file already exists. The option is ignored for files being read.

-s Turns off warning messages (the silent switch).

directory/file
Specifies the directory/file name entry as known to the IOS. For all CRAY EL systems, IOS directory file name combinations are automatically forced to uppercase letters. The CRAY J90 IOS is case-sensitive. Standard input or output is used for the UNICOS file. UNICOS directory and file names can be specified through the normal redirect methods.

The exdf command can easily be used as a link in a pipe to move files onto or off of the IOS disk.

WARNINGS

For the CRAY EL IOS, directory and file names must be specified in uppercase. For both the CRAY EL IOS and CRAY J90 IOS, file names must be separated by commas.

BUGS

The IOS and Cray Research systems force full words to be transferred, resulting in null characters added to the end of files. This is a problem with text files that you want to compile. The C preprocessor blows up when it runs into the null characters. exdf cannot filter the data because the last bytes in a file are not always accompanied by the end-of-file status.

Character files on the IOS do not have carriage return and line-feed characters as end-of-lines, but you cannot edit text files created on the IOS.

If `exdf` cannot write a file to the IOS disk because the disk is write-protected, no error messages are displayed.

NAME

expand, unexpand – Expands tabs to spaces and vice versa

SYNOPSIS

expand [-t *tablist*] *files*

unexpand [-a] [*files*]
unexpand [-t *tablist*] [*files*]

Obsolescent version; may not be supported in future releases:

expand [-*tabstop*] [-*tab1,tab2,...,tabn*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The expand utility writes files or the standard input to the standard output with <tab> characters replaced with one or more <space> characters that are needed to pad to the next tab stop. expand preserves backspace characters into the output and decrements the column count for tab calculations. expand is useful for preprocessing character files (for example, before sorting and looking at specific columns) that contain <tab>s.

The expand utility accepts the following options:

-t *tablist*
-*tabstop* (Obsolescent)
-*tab1,tab2,...,tabn* (Obsolescent)

Specifies the tab stops. If the *tablist* argument is a single decimal integer, tabs are set *tablist* column positions apart, rather than of the default 8. If you specify multiple numbers, the tabs are set at those specific column positions. Multiple numbers are specified as one argument, each number separated with a comma or <blank>s.

In the obsolescent version, the single number is specified as *tabstop* with a leading minus; multiple tab stops are specified after a leading minus as *tab1*, *tab2*, and so on.

The unexpand utility copies files or standard input to standard output, converting <blank>s at the beginning of each line into the maximum number of <tab>s followed by the minimum number of <space>s needed to fill the same column positions originally filled by the translated <blank>s. By default, tabstops are set at every eighth column position.

The unexpand utility accepts the following options:

`-a` In addition to translating <blank>s at the beginning of each line, translates all sequences of two or more <blank>s immediately preceding a tab stop to the maximum number of <tab>s, followed by the minimum number of <space>s that are needed to fill the same column positions originally filled by the translated <blank>s.

`-t` *tablist* Specifies the tab stops. If the *tablist* argument is a single decimal integer, tabs are set *tablist* column positions apart instead of the default 8. If you specifiy multiple numbers, the tabs are set at those specific column positions. Multiple numbers are specified as one argument, each number separated with a comma or <blank>s.

No <space>-to-<tab> conversions occur for characters at positions beyond the last of those specified in a multiple tab-stop list.

When you specify `-t`, any `-a` option is ignored.

files Names of input files you specify.

EXIT STATUS

The expand and unexpand utilities exit with one of the following values:

0 Successful completion.

>0 An error occurred.

NAME

`explain` – Displays the explanation for an error message

SYNOPSIS

`explain` *msgid*

IMPLEMENTATION

UNICOS systems

IRIX systems

DESCRIPTION

The `explain` utility retrieves and outputs a message explanation from an online explanation catalog. If the output device is a terminal, the output of `explain` is piped through the pager specified in the `PAGER` variable. If `PAGER` is not specified, the default pager `more -s` is used. If the output device is not a terminal, the output of `explain` is sent to the standard output device (`stdout`).

The `explain` utility requires the following argument:

msgid Specifies the message ID string associated with a message that appears when an error message is output. This string consists of the product group code and the message number. The product group code (*group*) is a string that identifies the product issuing the message. The message number (*msg#*) specifies which message within the product you have received. Enter the message ID as an argument to `explain` in the form *groupmsg#* or in the form *group-msg#*. If the group code ends in one or more digits (for example, `cf90`), you must use the form that includes the dash (`-`).

Recognized Group Codes

The following tables show the products and group codes that have message and explanation catalogs on UNICOS systems and on IRIX systems. The first column lists the group code (needed to look up the explanations for messages); the second column gives the complete name of the software product or products associated with the group code; the third column lists the publication number in which explanations for messages in that group code can be found.

UNICOS group codes

Group Code	Software Product	Publication
`apprentice`	MPP Apprentice tool	Online only
`as`	CAL assembler	SR–3108
`asm`	Cray Assembler for MPP (CAM)	Online only
`builddefs`	`builddefs`(1) command of the UNICOS glossary	Online only
`cc`	Cray Standard C compiler	Online only
`cf90`	CF90 Fortran language	Online only
`cld`	Loader for CRAY T3E systems	Online only

UNICOS group codes (continued)

Group Code	Software Product	Publication
cmd	UNICOS utilities that are compliant with XPG4	Online only
define	define(1) command of the UNICOS glossary	Online only
deplib	deplib(1) command	Online only
df	df(1) utility	Online only
dm	Data Migration Facility	SG–2135
du	du(1) utility	Online only
ex	ex(1) and vi(1) utilities	Online only
fsquota	File system quota feature	Online only
hpm	hpm(1) performance tool	Online only
hpmall	hpmall(8) performance tool	Online only
hpmflop	hpmflop(8) performance tool	Online only
ldr	segldr(1) and ld(1) loader commands	SR–0066
lib	Fortran and I/O libraries	Online only
libm	Mathematical libraries	SG–2138
libp	Pascal library	Online only
lprof	libprof library	Online only
ltrace	Jumptrace library	Online only
mppldr	Loader for CRAY T3D systems	Online only
mtdump	mtdump(1) performance tool	Online only
nqs	Network Queuing System	Online only
perf	libperf library	Online only
proc	procstat(1) performance tool	Online only
ps	ps(1) utility	Online only
sh	Standard shell	Online only
sys	UNICOS system error list (syserrlist[])	SR–2012
syslog	System message logger (newsys(8) and syslogd(8))	Online only
talk	talk(1B) utility	Online only
urm	Unified Resource Manager	Online only
usm	UNICOS Source Manager (USM)	SG–2097

IRIX group codes

Group Code	Software Product	Publication
cf90	f90 version 7.2	Online only
lib	Fortran 90 library version 2.0, and libffio	Online only
msgsys	explain(1) and caterr(1) utilities	Online only

Location of Explanation Catalogs

To find the explanation for the message, `explain` searches the path specified in the `NLSPATH` environment variable for the *group*`.exp` file. The value of the `NLSPATH` variable depends on the `LANG` environment variable or the `LC_MESSAGES` category. If `explain` cannot access an existing explanation catalog, check the value of the `NLSPATH` environment variable and either the `LANG` environment variable or the `LC_MESSAGES` category to ensure that the path name of the catalog is in the message system search path.

On UNICOS systems, you can use the `whichcat`(1) utility to determine which catalog is being accessed by `explain`. For more information, see `whichcat`(1).

On UNICOS systems, see the `catopen`(3C) man page for a description of the `NLSPATH`, `LANG`, and `LC_MESSAGES` components.

On IRIX systems, see `environ`(5) for a description of the `NLSPATH`, `LANG`, and `LC_MESSAGES` components.

Message Format Variables

If *msgid* does not appear with the error message or if you want to change the format of the messages you receive, you can modify your `MSG_FORMAT` and `CMDMSG_FORMAT` environment variables. Both variables can be set to define the fields and the order of the fields to be included in message output.

The `MSG_FORMAT` variable controls the format in which you receive error messages from programs that use the message formatting function `catmsgfmt`(3C). On UNICOS systems, these are most messages in group codes other than `cmd`. On IRIX systems, these are most messages in group codes other than `msgsys`.

On UNICOS systems, the `CMDMSG_FORMAT` variable controls the format in which you receive error messages issued by system utilities that are included in the XPG4 standard. These are messages in the `cmd` group code. A separate environment variable is provided to control message output from these utilities, because it is often desirable to capture and process this output through a script or program; limiting the output to one line makes this easier. The default format (shown below) for non-utility messages outputs at least two lines. The `CMDMSG_FORMAT` variable exists to provide a one-line default format for utility messages.

On IRIX systems, the `explain` and `caterr` utilities use either `CMDMSG_FORMAT` or `MSG_FORMAT` variables; however, `CMDMSG_FORMAT` is used before `MSG_FORMAT`.

Message Format Syntax

Valid fields for `MSG_FORMAT` and `CMDMSG_FORMAT` are as follows:

`%G` Group code

`%N` Message number

`%C` Command name

`%S` Severity level

`%P` Position of the error

`%M` Message text

`%D` Debugging information

`%T` Time stamp

If one of the % fields is not present in the contents of `MSG_FORMAT` or `CMDMSG_FORMAT`, the corresponding message field is not printed.

The default message format is produced by the following assumed `MSG_FORMAT` contents:

```
%G-%N  %C: %S %P\n  %M\n
```

For messages issued by UNICOS utilities, and by the IRIX utilities `explain`(1) and `caterr`(1), the default message format is produced by the following assumed `CMDMSG_FORMAT` contents:

```
%G-%N: %C %M\n
```

Messages issued from the `cmd` group code determine their format according to the following precedence:

1. Content of `CMDMSG_FORMAT` variable, if it exists
2. Content of `MSG_FORMAT` variable, if it exists
3. Default message format for utilities

The format of the time stamp (`%T`) is equivalent to that produced by the `cftime`(3C) function and can be overridden by the `CFTIME` environment variable. For details about time stamp formats, see the `strftime`(3C) man page, which documents the `cftime` function.

The Standard C `printf` escape sequences also can be embedded in the contents of `MSG_FORMAT`. The following table lists special characters that can be embedded.

Description	Symbol	Sequence
New-line character	NL (LF)	`\n`
Horizontal tab	HT	`\t`
Vertical tab	VT	`\v`
Backspace	BS	`\b`
Carriage return	CR	`\r`
Form feed	FF	`\f`
Audible alert	BEL	`\a`
Backslash	\	`\\`
Question mark	?	`\?`
Single quotation mark	’	`\’`
Double quotation mark	"	`\"`
Octal number	*ooo*	`\`*ooo*
Hexadecimal number	*hh*	`\x`*hh*

The escape *ooo* consists of the backslash followed by 1, 2, or 3 octal digits, which are taken to specify the value of the desired character. A common example of this construction is `\0`, which specifies the null character. The escape `\x`*hh* consists of the backslash, followed by `x`, followed by hexadecimal digits, which are taken to specify the value of the desired character. There is no limit on the number of digits, but the behavior is undefined if the resulting character value exceeds that of the largest character.

Any characters other than those listed in this table are passed through without the backslash, (for example, `\q` produces `q`).

NOTES

On UNICOS systems, 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

UNICOS systems

`caterr`(1), `catxt`(1), `gencat`(1), `whichcat`(1)

`catgetmsg`(3C), `catgets`(3C), `catmsgfmt`(3C), `catopen`(3C), `strftime`(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

IRIX systems

`caterr`(1), `gencat`(1)

`catgetmsg`(3C), `catmsgfmt`(3C)

`msg`(7D)

Cray Research publications

Cray Message System Programmer's Guide, Cray Research publication SG–2121

Segment Loader (SEGLDR) and ld Reference Manual, Cray Research publication SR–0066

UNICOS System Calls Reference Manual, Cray Research publication SR–2012

UNICOS Source Manager (USM) User's Guide, Cray Research publication SG–2097

Cray Data Migration Facility (DMF) Administrator's Guide, Cray Research publication SG–2135

Intrinsic Procedures Reference Manual, Cray Research publication SR–2138

Cray Assembly Language (CAL) for Cray PVP Systems Reference Manual, Cray Research publication SR–3108

NAME

`expr` – Evaluates arguments as an expression

SYNOPSIS

`expr` *arguments*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `expr` utility evaluates an expression and writes the result to standard output. All *arguments* are taken as expressions. After evaluation, the result is written on standard output. You must express the terms of the expression with blanks and escape characters special to the shell. Note that 0 is returned to indicate a value of 0, rather than the null string. Strings containing blanks or other special characters should be enclosed in quotation marks. Integer-valued arguments can be preceded by a unary minus sign. Internally, integers are treated as 64-bit, twos-complement numbers. The length of the expression is limited to 512 characters.

The following is a list of operators and keywords. Characters that must be escaped are preceded by `\`. The list is in order of increasing precedence, with equal-precedence operators grouped between `{}` symbols.

Grouping may be accomplished for the purpose of precedence control by use of the `\(` and `\)` operators. `y = (a+b)/c` becomes `expr \($a+$b\)/$c`.

`expr | expr`	Returns the first `expr` when it is neither null nor 0; otherwise, it returns the second `expr`.
`expr \& expr`	Returns the first `expr` when `expr` is neither null nor 0; otherwise, it returns 0.
`expr` { `=`, `\>`, `\>=`, `\<`, `\<=`, `!=` } `expr`	Returns the result of an integer comparison when both arguments are integers; otherwise, it returns the result of a lexical comparison.
`expr` { `+`, `-` } `expr`	Addition or subtraction of integer-valued arguments.
`expr` { `\*`, `/`, `%` } `expr`	Multiplication, division, or remainder of the integer-valued arguments.
`expr` : `expr`	Matching operator `:` compares the first argument with the second argument, which must be a regular expression. Regular expression syntax is the same as that of `ed`(1), except that all patterns are ‘‘anchored’’ to the beginning of the line (that is, begin with `^`). Therefore, `^` is not a special character in that context. Usually, the matching operator returns the number of characters matched (0 on failure). Alternatively, you can use the `\(...\)` pattern symbols to return part of the first argument.

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

As a side effect of expression evaluation, `expr` returns the following exit values:

0 Expression is neither null nor 0, or expression is false.

1 Expression is null or 0, or expression is true.

2 Expressions are not valid.

MESSAGES

`syntax error`	Operator or operand errors.
`nonnumeric argument`	Arithmetic is tried on a noninteger string.

BUGS

After argument processing by the shell, `expr` cannot tell the difference between an operator and an operand except by the value. If `$a` is `=`, the command:

```
expr $a = '='
```

looks like the following, because the arguments are passed to `expr` (and they will all be taken as the `=` operator):

```
expr = = =
```

The following command works:

```
expr X$a = X=
```

EXAMPLES

Example 1: The following example adds 1 to standard shell variable `a`.

```
a=‘expr $a + 1‘
```

Example 2: The following example adds 1 to C shell variable a.

```
set a=‘expr $a + 1‘
```

Example 3: The following example returns the last segment of a path name (that is, *file*). Watch out for / alone as an argument; expr takes it as the division operator (see the BUGS section).

```
# ’For $a equal to either "/usr/abc/file" or just "file"’
expr $a : ’.*/\(.*\)’ \| $a
```

The addition of the // characters eliminates any ambiguity about the division operator and simplifies the whole expression.

```
#  A better representation of the previous example.
expr //$a : ’.*/\(.*\)’
```

Example 4: The following example returns the number of characters in $VAR.

```
expr $VAR : ’.*’
```

SEE ALSO

csh(1), ed(1), sh(1)

regexp.h(3C) in the *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

NAME

`factor` – Factors a number

SYNOPSIS

`factor` [*number*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

When `factor` is invoked without an argument, it waits for a number to be typed in. If you type in a positive number less than or equal to 1.0×10^{14} it will factor the number and print its prime factors; each one is printed the proper number of times. Then it waits for another number. If it encounters a 0 or any nonnumeric character, it exits.

If `factor` is invoked with an argument, it factors the number as above and then exits.

Maximum time to factor is proportional to $\sqrt{n}$ and occurs when n is prime or the square of a prime.

The `factor` utility requires the following option:

number The input number you specify.

MESSAGES

`Ouch!` Input out of range or garbage input

BUGS

Garbage input is not always diagnosed (for example, `factor hello`).

NAME

fc – Displays process command history list

SYNOPSIS

```
fc [-r] [-e editor] [first [last] ]
fc -l [-n] [-r] [first [last] ]
fc -s [old=new] [first]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The fc utility lists, or edits and reexecutes, the command previously entered to an interactive shell (see sh(1)).

The command history list references commands by number. The first number in the list is selected arbitrarily. The relationship of a number to its command does not change except when the user logs in and no other process is accessing the list, at which time the system may reset the numbering to start the oldest retained command at another number, usually 1. When the number reaches an upper limit, the shell wraps the numbers, starting the next command with the number fc retains the time-ordering sequence of the commands.

When commands are edited (when you omit the -l option), the line(s) that result are entered at the end of the history list and then reexecuted by the shell. The fc command that caused the editing is not entered into the history list. If the editor returns a nonzero exit status, the entry is not placed into the history list and the command reexecution. Any command-line variable assignments or redirection operators used with fc affect both the fc command itself and the command that results. For example, the following command line reinvokes the previous command, suppressing standard error for both fc and the previous command:

```
fc -s -- -1 2>/dev/null
```

The fc utility accepts following options and operands:

-e *editor* Specifies the *editor* to edit the commands. The *editor* string is a utility name, subject to search by using the PATH environment variable. The value in the FCEDIT variable is used as a default when you omit the -e option. If FCEDIT is null or unset, ed(1) is used as the editor.

-l Lists the commands rather than invoking an editor on them. The commands are written in the sequence indicated by the *first* and *last* operands, as affected by -r, with each command preceded by the command number.

-n Suppresses command numbers when listing with -l.

`-r` Reverses the order of the commands listed (with `-l`) or edited (with neither `-l` or `-s`).

`-s` Reexecutes the command without invoking an editor.

first
last Specifies the commands to list or edit. The value of the `HISTSIZE` environment variable determines the number of previous commands. The value of *first* or *last* or both are one of the following:

[+]*number* A positive number that represents a command number; to display command numbers, use the `-l` option.

`-`*number* A negative decimal number that represents the command that was executed *number* of commands previously. (For example, the `-l` is the immediately previous command).

string A string that indicates the most recently entered command that begins with that string. If you do not also specify the *old=new* operand with `-s`, the string form of the *first* operand cannot contain an embedded equal sign.

When the synopsis form with `-s` is used:

- If you omit *first*, the previous command is used.

For the synopsis forms without `-s`:

- If you omit *last*, *last* defaults to the previous command when you specify `-l`; otherwise, it defaults to *first*.
- If you omit *first* and *last*, the previous sixteen commands are listed or the previous one command is edited (based on the `-l` option).
- If *first* and *last* are both present, all of the commands from *first* to *last* are edited (without `-l`) or listed (with `-l`). To edit multiple commands, present to the editor all of the commands at one time, each command starting on a new line. If *first* represents a newer command than *last*, the commands are listed or edited in reverse sequence, equivalent to using `-r`.
- When you use a range of commands, it is not an error to specify *first* or *last* values that are not in the history list; `fc` substitutes the value that represents the oldest or newest command in the list, as appropriate.

old=new Replaces the first occurrence of string *old* in the commands to be reexecuted by the string *new*.

NOTES

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

ENVIRONMENT VARIABLES

FCEDIT — Interprets the variable, when expanded by the shell, as the default value for the `-e` *editor* argument. If `FCEDIT` is null or unset, `ed`(1) is used as the editor.

HISTFILE — Interprets this variable as a path name specifying a command history file. If the `HISTFILE` is not set, the shell tries to access or create a file `.sh_history` in the user's home directory.

HISTSIZE — Interprets this variable as a decimal number that represents the limit to the number of previous commands that are accessible. If this variable is unset, the default is 128.

EXIT STATUS

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

0 Successful completion of the listing.

>0 An error occurred.

Otherwise, the exit status is that of the commands executed by `fc`.

SEE ALSO

`sh`(1)

NAME

fck – Provides file information from file system internals

SYNOPSIS

fck [-b] [-i] [-l] [-p] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fck utility provides information concerning the named files that are gathered from reading the inode and the address blocks from the block special device in the /dev/dsk directory. Information is printed only on files for which you have read permission.

The fck utility accepts the following options:

-b Displays block-oriented logical addressing information. This includes logical device name, slice, block offsets, and lengths for each data and address extent.

-i Displays information returned by the stat(2) system call. This includes file ownership, permissions, length, access and creation times, and so on. This is the default if no other options are specified.

-l Displays addressing information about file addressing and information blocks, as present in the physical disk inode. This includes inode information that is not reported by the stat(2) system call.

-p Displays physical addressing information. This includes physical device name, cylinder, track, and sectors for each data and address extent. On CRAY PVP systems, users who are not super-users are not shown cylinder and track value information. These values are represented by ***.

files Specifies files for which to gather information.

The program uses the access(2) system call to assure that the user has read permission to the file in question.

SEE ALSO

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

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

NAME

fg – Runs jobs in the foreground

SYNOPSIS

fg [*job_id* ...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The fg utility moves a background job from the current shell execution environment (see sh(1)) into the foreground.

Using fg to place a job into the foreground removes its process ID from the list of those "known in the current shell execution environment."

The fg utility supports the following operand:

job_id Specifies the job to be run as a foreground job. If you omit the *job_id*, the *job_id* for the job that was most recently suspended, placed in background, or run as a background job is used. The Jobs subsection in the sh(1) man page describes the the format of *job_id*.

NOTES

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

EXIT STATUS

The fg utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

bg(1), jobs(1), sh(1)

NAME

`file` – Determines file type

SYNOPSIS

```
file [-f ffile] [-h] [-m mfile] [-o] file ...
file [-h] [-m mfile] [-o] -f ffile
file -c [-m mfile]
```

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-c`, `-h`, `-f`, and `-m` options)
CRI extensions (`-o` option)

DESCRIPTION

The `file` utility performs a series of tests on each *file* and, optionally, on each file supplied in *ffile*, in an attempt to classify them. If a *file* argument seems to be a text file, `file` examines the first 512 bytes and tries to guess its language. If a *file* argument is a symbolic link, by default the link is followed, and `file` tests the file that the symbolic link references.

The `file` utility uses the `/etc/magic` file to identify files that have some sort of magic number; that is, any file contains a numeric or string constant that indicates its type. Commentary at the beginning of `/etc/magic` explains its format.

The `file` utility accepts the following options:

`-c` Causes the `file` utility to check the *magic* file for format errors, and prints on standard output the *magic* file in a readable format. If the `-c` option is specified, all options except [`-m` *mfile*] are ignored. For reasons of efficiency, this validation is not usually performed.

`-f` *ffile* Indicates that the next argument is a file that contains the names of files to be examined.

`-h` Instructs the `file` utility to not follow symbolic links. If you specify the `-h` option and *file* is a symbolic link, `file` prints the following message:

```
symbolic link to filename
```

The operand *filename* refers to the contents of the symbolic link.

`-m` *mfile* Instructs the `file` utility to use an alternate *magic* file.

`-o` Causes major and minor device numbers, and device-specific parameters of character and block special devices to be printed in octal. The device-specific parameters are also printed in octal.

file Name of file to be examined.

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

0 Successful completion.

>0 An error occurred.

BUGS

The `file` utility may mistakenly classify a text file because a bit map in the text file coincides with a bit map in the `/etc/magic` file.

EXAMPLES

The following example determines the type of files `example.c` and `a.out`:

```
$ file example.c a.out
example.c:  c program text
a.out:      executable not stripped
$
```

FILES

`/etc/magic`	File that contains magic numbers for different file types

SEE ALSO

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

NAME

`find` – Finds files

SYNOPSIS

`find` *pathnames expression*

IMPLEMENTATION

Cray PVP systems

STANDARDS

POSIX, XPG4
AT&T extensions (`-follow`, `-fstype`, `-inum`, `-local`, and `-mount` primaries)
CRI extensions (`-acid` and `-dev` primaries)

DESCRIPTION

The `find` utility locates files and directories that are located under the path names you specify and that meet the characteristics you list in the *expression.*

You can use metacharacters in many of the primaries if the special characters are *quoted*, that is, preceded with a backslash or surrounded by single quotation marks. (For more information on quoting, see `sh`(1).) The `find` utility searches for files that cause the Boolean *expression*, composed of any of the following primaries, to be true. When you specify a value for *n* in these primaries, you can prefix that number with either `+` or `-`. The meanings are as follows:

`+`*n* More than *n*

n Exactly *n*

`-`*n* Less than *n*

The primaries fall into two categories: those that select files and those that perform actions on the selected files.

Primaries That Select Files

CAUTION: For POSIX compliance, UNICOS 9.0 included a change in the functionality of the `-atime`, `-ctime`, and `-mtime` primaries. The meaning of the numeric argument to those primaries changed in two ways:

1. From day units (0 equals the 24 hours since the last midnight) to 24-hour blocks starting with the current time (0 equals the 24-hour block starting now).
2. `+` and `-` now refer to the same starting point in time, instead of `-` being time after the end of the 24-hour block and `+` being time before the start of the 24-hour block.

The primaries that select files are as follows:

pathnames Specifies the path names under which `find` will search. A path name can be either full, such as `/abc/xyz`, or relative, such as . to indicate the current directory.

`-acid` *acctid* Locates files with the specified account ID. You can find the account ID of a file by using the `chacid`(1) utility. If the account has the specified account ID, the expression is true.

`-atime` *n* Locates files that have been accessed in *n* days. The `find` utility itself changes the access time of directories in *pathnames*. If the file has been accessed in *n* days, the expression is true.

`-ctime` *n* Locates files that have been modified or whose attributes have been changed. A file's attributes include, for example, its user ID or group ID or the number of links to true. If the file has been changed in the last *n* days, the expression is true.

`-depth` Descends the directory hierarchy. All entries in a directory are acted on before the directory itself. This can be useful when `find` is used with `cpio`(1) to transfer files contained in directories without write permission. Always true.

`-dev` *minor* Locates files that reside on the device with the specified number. If the file resides on the device with minor device number *minor*, it is true.

`-fstype` *type* True if the file system to which the file belongs is of type *type*. Valid *type*s include: `DFS`, `NC1FS`, `NFS`, `PROC`, and `SFS`.

`-group` *gname* Locates files belonging to the specified group. If *gname* is numeric and is not one of the groups listed in the file `/etc/group`, it is taken as a group ID. True if the file belongs to the group *gname*.

`-inum` *ino* Locates files that have only the specified inode number. If the file being processed has an inode number of *ino*, it is true.

`-links` *n* Locates files that have *n* links. If the file being processed has *n* links, it is true.

`-local` Locates files that reside only on the local system. If the file being processed physically resides on the local system, it is true.

`-mount`
`-xdev` Restricts the search to the file system that contains the directory specified. Always true.

`-mtime` *n* Locates files that have been modified in the past *n* days. If the file being processed has been modified in the past *n* days, it is true.

`-name` *file* Locates files that have the specified basename of the file name. You can use metacharacters in the file name if you precede them by \ or surround them with single quotation marks. (Be careful when using `[`, `?`, and `*`.) If *file* matches the file name being processed, it is true.

`-newer` *file* Locates files that have been modified more recently than the specified file. You can use metacharacters in the file name if you precede them by \ or surround them with single quotation marks. If the file being processed has been modified more recently than *file*, it is true.

`-nogroup` True if the file belongs to a group not in the `/etc/group` file.

`-nouser` True if the file belongs to a user not in the `/etc/passwd` file.

`-perm` [`-`]*mode* Locates file that have access permission indicated by the symbolic mode *mode*. The *mode* argument is used to represent file mode bits. It is identical in format to the `chmod` *symbolic mode* operand described in `chmod`(1) and is interpreted as follows. To start, a template is assumed with all file mode bits cleared. An *op* symbol of `+` sets the appropriate mode bits in the template; `-` clears the appropriate bits; `=` sets the appropriate mode bits, without regard to the contents of the file mode creation mask of the process. The *op* symbol of `-` cannot be the first character of *mode*.

If the hyphen is omitted, `-perm` evaluates as true when the file permission bits exactly match the value of the resulting template.

Otherwise, if *mode* is prefixed by a hyphen, the `-perm` evaluates as true if at least all the bits in the resulting template are set in the file permission bits.

`-perm` [`-`]*onum* Locates files that have the access permission indicated by octal number *onum*. (For information about specifying access permissions, see `chmod`(1).) You can prefix *onum* with a hyphen to make more flag bits (07777, see `stat`(2)) significant. True if the file permission flags match octal number *onum*.

`-prune` Always yields true. Do not examine any directories or files in the directory structure below the *pattern* just matched.

`-size` *n*[`c`] Locates files of the specified block length. To specify size in characters, follow the *n* specification with a `c`. If the file being processed is *n* blocks long (512 bytes per block), the expression is true.

`-type` *filetype* Locates files of the specified file type. The file type is specified by one of the following letters:

`b` Block special file
`c` Character special file
`d` Directory
`f` Regular file
`l` Symbolic link
`m` Migrated file (type IFOFL)
`M` Migrated file (has DMF handle)
`p` FIFO or named pipe
`s` Sockets

If *filetype* is `b`, `c`, `d`, `f`, `l`, `m`, `M`, `p`, or `s`, the file type is true.

`-user` *uname* Locates files belonging to the specified user. If *uname* is numeric and is not one of the login names in the file `/etc/udb`, it will be taken as a user ID. True if the file being processed belongs to user *uname*.

Primaries That Act on Files

The primaries that act on files are as follows:

`-exec` *utility_name* [*argument...*] `;`
Executes the specified utility with the given arguments. Follow the command or last argument with a `<space>` and an escaped semicolon (`\;`), as in the following example:

```
find pathnames -exec ls \;
```

`find` replaces a utility argument `{}` with the name of the *pathname* file that is currently being processed. (For more information on parameter substitution with `{}`, see `sh`(1).) If the executed utility *utility_name* returns a value of 0 as the exit status, this expression is true.

If the last argument to `-exec` is `{}`, and you specify '`+`' rather than '`;`', the command will be invoked fewer times with `{}` replaced by groups of pathnames.

`-follow` Follows symbolic links. Always true. When following symbolic links, `find` keeps track of the directories visited to that it can detect infinite loops; for example, such a loop would occur if a symbolic link pointed to an ancestor. This expression should not be used with the `-type l` expression.

`-ok` *utility_name* [*argument...*] `;`
Prompts for execution of the utility with the given arguments. This primary executes *utility_name* as `-exec` does, except that the generated command line is printed with a question mark prompt first, and it is executed only if you respond by typing `y`. If the executed utility returns a value of 0 as the exit status, the expression is true.

`-print` Writes the names of files selected from *pathnames* to `stdout`. Always true.

`(` *expression* `)` If the parenthesized expression is true, the expression is true. Each element in *expression* is a separate argument; you must separate each argument from each other by space characters. A space character must appear on both sides of each parenthesis, exclamation point, or other element (such as a primary or an argument to a primary). When you use a backslash to quote a special character, the space characters go on either side of the pair of characters (for example, " `\[` "). *Expression* is evaluated left to right.

You can combine the primaries by using the following operators (in order of decreasing precedence):

1. The negation of a primary (`!` is the unary NOT operator).
2. Concatenation of primaries (the AND operation is implied by the juxtaposition of two primaries). Any primaries that appear after a primary evaluated as false are false.
3. Alternation of primaries (`-o` is the OR operator).

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

0 All *path* operands were traversed successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following command is used to find a file named `data1` under the current directory. The dot character specifies the current directory, the `-name` primary specifies the file `data1` as the file to find, and the `-print` primary causes the output to be displayed on standard output (`stdout`). If `find` does not find the file, no message appears. If `data1` is in the current directory or one of its subdirectories, `find` will display the path to it from the current directory.

```
find . -name data1 -print
```

Example 2: The following command finds files that begin with the letter `b` under the directory `/usr/man`. The output is displayed on `stdout`.

```
find /usr/man -name b'*' -print
```

Example 3: The following example redirects standard error to the null file while finding all files and directories that begin with the letter `b` in the current directory and its subdirectories. This redirection can be a convenient way to separate error messages (for example, those directories to which you may not have access) from the output in which you are interested. This type of redirection of standard error is appropriate for the standard shell.

```
find . -name b'*' -print 2>/dev/null
```

Example 4: The following example removes files named `a.out` and files that end in `.o` that have not been accessed for a week. `-atime +7` indicates a time of 8 or more days, and the final characters instruct `find` to execute the `rm`(1) utility on the selected files.

```
find / \( -name a.out -o -name '*.o'\) -atime +7 -exec rm {} \;
```

FILES

`/etc/group`	Group file that contains group names and group IDs
`/etc/udb`	User validation file that contains user control limits
`/usr/include/sys/fsid.h`	File that contains file system names

SEE ALSO

`chacid`(1), `chmod`(1), `cpio`(1), `rm`(1), `sh`(1), `test`(1)

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

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

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

NAME

`finger`, `f` – Provides user information

SYNOPSIS

`/usr/ucb/finger` [`-b`] [`-f`] [`-h`] [`-i`] [`-l`] [`-m`] [`-p`] [`-q`] [`-s`] [`-w`] *names*
`/usr/ucb/finger` [*name*]`@`*host*

`/usr/ucb/f` [`-b`] [`-f`] [`-h`] [`-i`] [`-l`] [`-m`] [`-p`] [`-q`] [`-s`] [`-w`] *names*
`/usr/ucb/f` [*name*]`@`*host*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

By default, the `finger` and `f` commands (they are functionally equivalent) list the login name, full name, terminal name, write status (as an asterisk (`*`) before the terminal name if write permission is denied), idle time, login time, office location, and phone number (if they are known) for each current user. Idle time is represented in minutes if it is a single integer, hours and minutes if a colon (`:`) is present, or days and hours if a `d` is present in the date field.

These commands have the following options:

`-b` Provides brief list of users.

`-f` Suppresses heading in short and quick output format.

`-h` Suppresses printing of `.project` file.

`-i` Same as quick list but includes idle time.

`-l` Forces long output format.

`-m` Matches arguments only on user name.

`-p` Suppresses printing of `.plan` file.

`-q` Provides quick list with only login name, terminal name, and login time.

`-s` Provides short list of users.

`-w` Suppresses printing of full name in short-list format.

name Name of user you specify.

A longer list format also exists; `finger` and `f` use it whenever a list of names is given. (First and last names of users are accepted.) This is a multiline format, and it includes all information previously described, as well as the user's home directory and login shell, any plan that the person has placed in the `.plan` file in their home directory, and the project on which they are working from the `.project` file, also in the home directory (text must begin on line 1 of the file; if it does not, the command will not read it).

FILES

`/etc/utmp`	File that holds user information
`/etc/udb`	User validation file that contains user control limits
`$HOME/.plan`	Plan file
`$HOME/.project`	Project file

NAME

flodump – Recovers Flowtrace data from a dump or running process

SYNOPSIS

flodump [-e] [-V] [-f *corefile* | -p *procid*]

IMPLEMENTATION

Cray PVP systems

DESCRIPTION

The flodump utility produces Flowtrace output from an abnormally terminated program, compiled using the Cray Research CF90 or Standard C compilers, and with the Flowtrace feature enabled. Programs that make calls to FLOWMARK(3C) are also candidates for this recovery process.

In addition, flodump can capture the Flowtrace data from a running program, if it is owned by the caller to flodump and the running program was compiled with the Flowtrace feature enabled.

The data written by this utility to stdout is raw Flowtrace data, suitable for postprocessing by flowview(1) or by other tools, such as awk(1).

When it is recovering data from a core dump, flodump automatically recovers the data that would have been generated by a nonfatal termination of the program. It reads the file core by default, whose name can be changed by the -f option.

When it is capturing the Flowtrace data from a running process, flodump takes a "snapshot" of the current program state.

Flowtrace is discussed on the man page flowtrace(7); flowview(1) is discussed on its own man page. Both commands are also described in the *Guide to Parallel Vector Applications*, Cray Research publication SG–2182.

The flodump utility writes the captured or recovered data only under the following conditions:

- If your program was compiled with the Flowtrace compiler option turned on, and/or the program executed calls to FLOWMARK
- If the version of Flowtrace loaded with your program was the same as that generated with flodump

The data is less accurate than if the program terminated normally. flodump assumes that the program state is at the last subprogram entry or exit it processed. The distortion introduced by this assumption can occasionally produce abnormal results, such as apparently negative run times.

The flodump utility accepts the following options:

-e Encodes error messages into special raw-data records if errors occur during the data recovery or capture process. Otherwise the default is for flodump to issue error messages and terminate.

`-V` Prints the version and a copyright statement to the `stderr` file.

`-f` *corefile* Specifies a different file name from which to recover the Flowtrace data. By default, if `flodump` is reading a core file, the file name read will be `core`. This option may not be specified at the same time as the `-p` option.

`-p` *procid* Specifies the process number (decimal) of the running program from which to capture the Flowtrace statistics. The program executing in this process must have been compiled with the Flowtrace feature enabled. If this option is not specified, the default is for `flodump` to attempt to recover Flowtrace data from a core dump of an abnormally terminated program. This option cannot be specified at the same time as the `-f` option.

FILES

`core` Program memory dump

SEE ALSO

`flowview`(1)

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

`f90`(1) in *CF90 Commands and Directives Reference Manual*, Cray Research publication SR–3901

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

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

`flowtrace`(7), `performance`(7) (available only online)

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

NAME

fmgen – Fortran makefile generator

SYNOPSIS

fmgen [-c *compiler*] [-f *flags*] [-m *makefile*] [-o *command_name*] *files*

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fmgen utility splits out all the Fortran subroutines from *files* using fsplit(1) and produces a makefile using make(1) to compile and load the program.

The fmgen utility accepts the following options:

-c *compiler*	The Fortran compiling system used to convert the .f files to .o files and linking them to form an executable.
-f *flags*	Options to the Fortran compiling system. Be sure to enclose the argument specifying the option(s) in single quotation marks.
-m *makefile*	The name of the makefile produced by the fmgen utility (makefile by default).
-o *command_name*	Resultant executable (a.out by default).
names	Names of input files that you specify.

The makefile created has the following targets:

all	Creates the executable file *command_name* (default rule)
command_name	Compiles and loads the program
command_name.prof	Compiles and loads the program with profiling
clean	Removes all the .o files
clobber	Executes clean and then removes the executable files
void	Removes everything created by the fmgen utility

FILES

makefile	Default make file created
a.out	Executable binary file
a.out.prof	Executable binary file with profiling information
file.f	Input Fortran source code file
file.o	Relocatable object code file

SEE ALSO

f90(1), fsplit(1), make(1), prof(1)

NAME

`fold` – Folds long lines of files for finite-width output device

SYNOPSIS

`fold` [`-b`] [`-s`] [`-w` *width*] [*files*]

Obsolescent version:

`fold` [`-`*width*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The `fold` utility acts as a filter that splits lines (inserts a `<newline>`) of the specified *files*, or the standard input (if you do not specify *files*), to have maximum width *width*.

The following options are recognized:

`-b` Count *width* in bytes rather than column positions.

`-s` If segment of a line contains a `<blank>` within the first *width* column positions (or bytes), break the line after the last such `<blank>` that meets the width constraints. If no `<blank>` meets the requirements, the `-s` option has no effect for that output segment of the input line.

`-`*width* (obsolescent)
`-w` *width* Specifies the maximum line length (in column positions) (or bytes if `-b` is specified). The default *width* is 80. The output goes to `stdout`. If `<tab>`s are present, or if they must be expanded (using `expand`(1) before using `fold`), the *width* should be a multiple of 8.

files Specifies the path name of a file to be used by this utility.

NOTES

Because multibyte characters are not supported in the UNICOS 8.0 release, the `-b` option assumes 1 byte per character.

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

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

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

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

EXIT STATUS

The fold utility exits with one of the following values:

0 All input files were processed successfully.

>0 An error occurred.

BUGS

If underlining is present, it may not work correctly with the fold utility.

EXAMPLES

The following command expands all <tab> characters in the file xyz, then creates a xyz.folded file in which the longest line in the file is no longer than 80 characters.

```
expand xyz | fold > xyz.folded
```

SEE ALSO

expand(1) to expand tabs to spaces and vice versa

NAME

`from` – Prints mail header lines to show you from whom your mail originated

SYNOPSIS

`/usr/ucb/from` [`-s` *sender*] [*user*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `from` utility prints out the mail header lines in your mailbox file to show you from whom your mail originated.

The `from` utility accepts the following option and operand:

`-s` *sender* Prints only headers for mail sent by *sender*.

user Examines *user*'s mailbox instead of your own. To examine another user's mailbox, you must have the appropriate read privileges.

FILES

`/usr/mail/*` Post office directory

SEE ALSO

`mail`(1), `mailx`(1)

NAME

`fsplit` – Splits Fortran files

SYNOPSIS

`fsplit` [`-s`] [*files*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `fsplit` utility splits the named Fortran *files* into separate files, with one routine per file, and lists them. If *files* is not specified, data is read from standard input. A procedure includes the following program segments: `blockdata`, `function`, `main program`, and `subroutine`.

The following naming conventions apply:

- The `program` segment is put in file `MAIN.f`.
- The segment *X* is put in file *X*`.f`.
- Duplicate segments *X* are put into file *XN*`.f`.
- Unnamed `blockdata` segments are put into files `blockdata`*N*`.f`.
- Unnamed segments or segments whose name is unable to be determined are placed in file `zzzzzz`*N*`.f`.

N is a unique integer value for each duplicate file which exists.

The `fsplit` utility accepts the following option and operand:

`-s` Strips trailing blanks.

files Names of files to be split.

NOTES

Expand tabs before splitting files.

As documented on this page, the `.f` suffix is used to name the files generated by `fsplit`. If a # character appears in column one of an input file, the suffix used to name the output file will be `.F`, indicating that the preprocessor must be executed on the output file.

BUGS

The `fsplit` utility is a simple program that can be used effectively. However, the full rules of Fortran input make it difficult to parse for anything less than the full input phase of the compiler. The program has been enhanced to permit comments to precede the `subroutine` statement, to permit spaces inside the `end` keyword, to permit `!` -comments following the `end` keyword, and to permit line numbers beginning in column 73 of the `subroutine` statement. However, a Fortran continue statement with the `subroutine` keyword appearing on a different line than the subroutine name causes the file to be named `zzzzzz`*N*`.f`.

The operation of `fsplit` may be verified using `cat`(1) to send the output files to `wc`(1) and comparing the result to the size of the input file to `fsplit`.

FILES

file`*.[fF]`	Fortran source code files of a named program segment
`zzzzzz*.[fF]`	Source code files containing an unnamed program segment
`blockdata*.[fF]`	Source code files of a blockdata segment

SEE ALSO

`cat`(1), `csplit`(1), `expand`(1), `split`(1), `wc`(1)

NAME

ftp – Transfers files to and from a remote network site

SYNOPSIS

/usr/ucb/ftp [-c *copybufsize*] [-d] [-g] [-i] [-n] [-s *sockbufsize*] [-t] [-v] [-Sc *tos*] [-Sd *tos*] [*host* [*port*]]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The ftp utility is the user interface to the Internet standard file transfer protocol (FTP). The program allows users to transfer files to and from a remote network site.

The client host with which ftp will communicate can be specified on the command line. If this is done, ftp immediately attempts to establish a connection to an FTP server on that host; otherwise, ftp enters its command interpreter and waits for instructions from the user.
When ftp is waiting for commands from the user, the prompt ftp> is provided for the user. If the user supplies insufficient command arguments, ftp prompts for them.

The port number of the ftpd daemon may be specified if the host name is specified. If you do not specify a port number, the default ftp port from the /etc/services file is used.

You can interrupt the ftp utility by pressing the terminal interrupt key, usually <CONTROL-c>. If this is done while ftp is attempting to carry out a command, ftp reverts to the command interpreter and displays the ftp> prompt; otherwise, ftp is terminated.

If your site has installed Kerberos version 4, ftp will attempt to perform Kerberos authentication. If authentication negotiation is successful, an encoded authentication string is exchanged between ftp and ftpd. This version of ftp protects authentication data using base 64 encoding. When authentication is successful, all commands sent over the control channel are protected with cryptographic checksum or encryption. Data sent over the data channel may be protected with cryptographic checksum or encryption. Encryption is available only in the United States and Canada.

The ftp utility accepts the following options:

-c *copybufsize*
: Sets the copy buffer size. A numeric argument sets the buffer to that size. The letter K or k can follow a numeric buffer size to specify a multiple of 1024.

-d
: Enables debugging.

-g
: Disables file name globbing (see the glob option).

-i
: Turns off interactive prompting during multiple file transfers.

`-n` Restrains `ftp` from attempting autologin during initial connection. If autologin is enabled, `ftp` checks the `.netrc` file in the user's home directory for an entry that describes an account on the remote machine. If no entry exists, `ftp` uses the login name on the local machine as the user identity on the remote machine, and prompts for a password and, optionally, an account with which to log in.

`-s` *sockbufsize*
Sets the socket buffer size. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024.

`-t` Enables tracing.

`-v` Forces `ftp` to show all responses from the remote server, and reports data transfer statistics (verbose option).

`-Sc` *tos* Sets the Internet Protocol (IP) type of service option for the FTP control connection to the value *tos*, which may be a numeric Type-of-Service (TOS) value or a symbolic TOS name that is found in the `/etc/iptos` file.

`-Sd` *tos* Sets the IP type of service option for the FTP data connection to the value *tos*, which may be a numeric TOS value or a symbolic TOS name that is found in the `/etc/iptos` file.

host [*port*] Specifies the host or port with which `ftp` will communicate.

The `ftp` utility recognizes the following commands. They may be given aliases but they must remain unique.

`!` [*command* [*args*]]
Invokes an interactive shell on the local machine. If arguments exist, the first is considered a command to execute directly, and the others are considered arguments.

`$` *macro-name* [*args*]
Executes the macro *macro-name* that is defined by the `macdef` command. Arguments are passed to the macro unglobbed.

`account` [*passwd*]
Supplies a supplemental password that is required by a remote system to access resources after a login is completed successfully. If no argument is included, the user is prompted for an account password through a nonechoing input mode.

`allo` [*arg*]
Specifies whether the `FTP ALLO` and `FTP SIZE` commands will be sent before `put` and `get` commands are executed. The `FTP ALLO` command informs the server of the size of the file that will be sent by using the `put` command so that it can preallocate the disk space if necessary. The `FTP SIZE` command inquires about the size of a file that will be retrieved by using the `get`(1) command, so that the disk space can be preallocated before the file is fetched.

If *arg* is `on`, the `FTP ALLO` or `FTP SIZE` command is sent automatically before a `put` or `get` is executed; if *arg* is `off`, they are not sent. The default is `on`.

`append` *local-file* [*remote-file*]
: Appends *local-file* to a file on the remote machine. If *remote-file* is unspecified, the name of *local-file* is used to name *remote-file*. File transfer uses the current settings for `type`, `format`, `mode`, and `structure`.

`ascii` Sets the file transfer type to network ASCII. This is the default type.

`bell` Sounds a bell after each file transfer command is completed.

`binary` Sets the file transfer type to support binary image transfer.

`bye` Terminates the FTP session with the remote server and exits `ftp`.

`case` Toggles remote computer file name case mapping during `mget` commands. When `case` is `on` (default is `off`), remote computer file names that leave all letters in uppercase are written in the local directory with the letters mapped to lowercase.

`cd` *remote-directory*
: Changes the working directory on the remote machine to *remote-directory*.

`cdup` Changes the remote machine working directory to the parent of the current remote machine working directory.

`chmod` *remote-file*
: Changes file permissions of a remote file.

`clear` Sets the file protection level to clear text for file transfer. All commands sent over the control channel are protected by cryptographic checksum until a different protection level is specified.

`close` Terminates the FTP session with the remote server and returns to the command-line interpreter. Erases any defined macros.

`copybuf` [*arg*]
: Sets the copy buffer size. This buffer is used for reading and writing between the data socket and the source or destination file. FTP automatically chooses a copy buffer size; however, you can alter the selection. `copybuf` takes an optional argument. An argument of `off` turns off copy buffer sizing and the buffer defaults to the size specified in the `tcp_config.h` file by the `COPYBUFSIZE` variable. An argument of `auto` returns buffer sizing to automatic. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. Without an argument, `copybuf` shows the current copy buffering. The default setting is `auto`.

`cr` Toggles carriage-return stripping during ASCII type file retrieval. Denotes records by a carriage return or linefeed sequence during ASCII type file transfer. When `cr` is `on` (the default), carriage returns are stripped from this sequence to conform with the UNIX single-line-feed record delimiter. Records on remote systems other than UNIX can contain single linefeeds; when an ASCII type transfer is made, you can distinguish these linefeeds from a record delimiter only when `cr` is `off`.

`delete` *remote-file*
: Deletes the file *remote-file* on the remote machine.

`debug` [*debug-value*]
: Toggles debugging mode. If an optional *debug-value* is specified, it is used to set the debugging level. When debugging is on, `ftp` prints each command that is sent to the remote machine when it is preceded by the string `-->`.

`dir` [*remote-directory*] [*local-file*]
: Prints a listing of the contents of *remote-directory* and, optionally, places the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If *local-file* is not specified, or *local-file* is `-`, output goes to the terminal. The `dir` command is synonymous with `ls`.

`disconnect`
: Indicates a synonym for `close`.

`form` *format*
: Sets the file transfer format to *format*. The default format is nonprint. Currently, only the default is supported.

`fullbuf` Toggles the use of full buffers on write-to-disk functions. During a get operation, FTP fills the copy buffer with reads from the data socket before writing the contents of the buffer to the destination file. You can use the `fullbuf` command to disable this feature.

`get` *remote-file* [*local-file*]
: Retrieves *remote-file* and stores it on the local machine. If the name of *local-file* is not specified, it is given the same name that it has on the remote machine; however, it can be altered by the current `case`, `ntrans`, and `nmap` settings. The current settings for `type`, `form`, `mode`, and `structure` are used when the file is transferred.

`glob` Toggles file name expansion for `mdelete`, `mget`, and `mput`. If globbing is turned off with the `glob` command, the file name arguments are taken literally and not expanded. Globbing for `mput` is done the same as for `csh`(1). For `mdelete` and `mget`, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is probably different from expansion of the name of an ordinary file. The exact result depends on the foreign operating system and `ftp` server, and it can be previewed by using `mls` *remote-files* `-`. The `mget` and `mput` commands are not used for transferring entire directory subtrees of files. To do that, transfer a `tar`(1) archive of the subtree (in binary mode).

`hash` Toggles the printing of a hash sign (#) on each data block that is transferred. The size of a data block is 4096 bytes.

`help` [*command*]
: Prints an informative message about the meaning of *command*. If no argument is given, `ftp` prints a list of the known commands.

`idle` [*seconds*]
: Gets or sets idle timer on the remote side.

`image` Indicates a synonym for binary.

`lcd` [*directory*]
Changes the working directory on the local machine. If *directory* is not specified, the user's home directory is used.

`ls` [*remote-directory*] [*local-file*]
Prints an abbreviated listing of the contents of a directory on the remote machine. If *remote-directory* is unspecified, the current working directory is used. If *local-file* is not specified, or if *local-file* is `-`, the output is sent to the terminal.

`macdef` *macro-name*
Defines a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates the macro input mode. All defined macros leave a limit of 16 macros and 4096 total characters. Macros remain defined until a `close` command is executed. The macro processor interprets `$` and `\` as special characters. A `$` followed by a number is replaced by the corresponding argument on the macro invocation command line. A `$` followed by an `i` signals the macro processor that the executing macro must be looped. On the first pass, `$i` is replaced by the first argument on the macro invocation command line; on the second pass, it is replaced by the second argument, and so on. A `\` followed by any character is replaced by that character. Use the `\` to prevent special treatment of the `$`.

`mdelete` *remote-files*
Deletes the specified files on the remote machine. If globbing is enabled, you can use `ls` to expand the specification of *remote-files* first.

`mdir` *remote-files local-file*
Provides a function similar to `dir`, except that multiple remote files can be specified. If interactive prompting is on, `ftp` prompts the user to verify that the last argument is the target local file that receives the `mdir` output.

`mget` *remote-files*
Expands the *remote-files* on the remote machine and performs a `get` process for each file name that is produced. See `glob` for details of the file name expansion. The resulting file names are then processed according to `case`, `ntrans`, and `nmap` settings. Files are transferred into the local working directory, which can be changed by entering `lcd` *directory*; new local directories can be created by entering `! mkdir` *directory*.

`mkdir` *directory-name*
Makes a directory on the remote machine.

`mls` *remote-files local-file*
Provides a function similar to `ls`, except multiple remote files can be specified. If interactive prompting is on, `ftp` prompts the user to verify that the last argument is the target local file that receives the `mls` output.

`mode` [*mode-name*]
Sets the file transfer mode to *mode-name.* The default mode is stream mode. (Currently, only the default is supported.)

`modtime` *remote-file*
Shows last modification time of a remote file.

`mput` *local-files*
Expands wildcards on the list of local files that are given as arguments and performs a `put` operation for each file in the resulting list. See `glob` for details of file name expansion. Resulting file names are then processed according to `ntrans` and `nmap` settings.

`newer` *remote-file* [*local-file*]
Gets file if the remote file is newer than the local file.

`nlist` [*remote-directory* [*local-file*]]
Lists the contents of the remote directory.

`nmap` [*inpattern outpattern*]
Sets or unsets the file name-mapping mechanism. If no arguments are specified, the file name-mapping mechanism is unset. If arguments are specified, remote file names are mapped during the execution of `mput` commands and `put` commands that are issued without a specified remote target file name; local file names are mapped during the execution of `mget` commands and `get` commands that are issued without a specified local target file name.

This command is useful when you are connecting to a remote system other than UNIX that uses different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*; *inpattern* is a template for incoming file names (which can be already processed according to the `ntrans` and `case` settings).

To template variables, include the sequences `$1`, `$2`, ..., `$9` in *inpattern*. Use `\` to prevent this special treatment of the `$` character. All other characters are treated literally, and they are used to determine the `nmap` *inpattern* variable values (for example, given *inpattern* `$1.$2` and the remote file name `mydata.data`, `$1` has the value `mydata`, and `$2` has the value `data`). The *outpattern* determines the resulting mapped file name. The sequences `$1`, `$2`, ..., `$9` are replaced by any value that results from the *inpattern* template. The sequence `$0` is replaced by the original file name. If *seq1* is not a null string, the sequence [*seq1*,*seq2*] is replaced by *seq1*; otherwise, it is replaced by *seq2* (for example, `nmap $1.$2.$3 [$1,$2].[$2,file]` yields the output file name `myfile.data` for input file names `myfile.data` and `myfile.data.old`, and it yields `myfile.file` for the input file name `myfile`, and yields `myfile.myfile` for the input file name `.myfile`). You can include spaces in *outpattern* (for example, `nmap $1.$2.$3 [$1,$2] [$2,file]` yields the same results as the previous example, except that the output file names will have a space in place of the period (.)). Use the `\` character to prevent special treatment of the `$`, `[`, `]`, and `,` characters.

`ntrans` [*inchars* [*outchars*]]

Sets or unsets the file name, character-translation mechanism. If no arguments are specified, the file name, character-translation mechanism is unset. If arguments are specified, characters in remote file names are translated during the execution of `mput` commands and `put` commands that are issued without a specified remote target file name; characters in local file names are translated during execution of `mget` commands and `get` commands that are issued without a specified local target file name. This command is useful when you are connecting to a remote system other than UNIX that uses different file naming conventions or practices. A character in a file name that matches a character in *inchars* is replaced by the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

`open` *host* [*port*]

Establishes a connection to the specified *host* FTP server. If an optional *port* number is supplied, `ftp` attempts to contact an FTP server at that port. If autologin is enabled (default), `ftp` also attempts to log in the user to the FTP server automatically.

`private` Sets the file protection level to `private` for the control channel and file transfer. This command is not available outside of the United States and Canada. All messages sent on the control and data channels are protected by encryption.

`protect` [`clear` | `safe` | `private`]

Sets the protection level for file transfer. The `private` option is not available outside of the United States and Canada.

`prompt` Toggles interactive prompting. Interactive prompting occurs during multiple file transfers to allow users to retrieve or store files selectively. If prompting is turned off, an `mget` or `mput` command transfers all files. The default for `prompt` is `on`. If started with the `-i` option, `prompt` starts set at `off`.

`proxy` *ftp-command*

Executes an `ftp` utility on a secondary control connection. This command allows simultaneous connection for transferring files between servers to two remote FTP servers. The first `proxy` command must be open to establish the secondary control connection. Enter the `proxy ?` command to see other `ftp` utilities that are executable on the secondary connection. The following commands act as follows when prefaced with `proxy`:

- `open` does not define new macros during the autologin process.
- `close` does not erase existing macro definitions.
- `get` and `mget` transfer files from the host on the primary control connection to the host on the secondary control connection.

- `put`, `mput`, and `append` transfer files from the host on the secondary control connection to the host on the primary control connection.

  Third-party file transfers depend on support of the `ftp` protocol `PASV` command by the server on the secondary control connection.

  Proxy file transfers are not supported for safe and private protection modes.

`put` *local-file* [*remote-file*]
: Stores *local-file* on the remote machine. If *remote-file* is unspecified, the name of *local-file* is used after processing according to any `ntrans` and `nmap` setting in naming *remote-file*. File transfer uses the current settings for `type`, `format`, `mode`, and `structure`.

`pwd`
: Prints the name of the current working directory on the remote machine.

`quit`
: Indicates a synonym for `bye`.

`quote` *arguments*
: The arguments specified are sent to the remote FTP server exactly as specified. In return, one FTP reply code is expected.

`rawbuf`
: Toggles the use of raw I/O on write-to-disk and read-from-disk functions. Usually, the system overhead is lower when raw I/O is used. The default is that raw I/O is enabled.

`recv` *remote-file* [*local-file*]
: Indicates a synonym for `get`.

`rename` [*from*] [*to*]
: Renames the file *from* on the remote machine to the file *to*.

`reget` *remote-file* [*local-file*]
: Gets file restarting at the end of the local file.

`reset`
: Clears the reply queue. This command resynchronizes the command and reply sequencing by using the remote FTP server. Resynchronization may be necessary when the remote server violates the FTP protocol.

`restart` *bytecount*
: Restarts the file transfer at *bytecount*.

`rhelp` *command-name*
: Requests help from the remote `ftp` server. If *command-name* is specified, it is also supplied to the server.

`rmdir` *directory-name*
: Deletes a directory on the remote machine.

`rstatus`
: Shows the status of the remote machine.

`runique` Toggles storing of files that have unique file names on the local system. If a file already exists with a name equal to the target local file name for a `get` or `mget` command, a `.1` is appended to the name. If the resulting name matches another existing file, a `.2` is appended to the original name. If this process continues up to `.99`, an error message is printed, and the transfer does not occur. The generated unique file name is reported. `runique` does not affect local files that are generated from a shell command. The default value is `off`.

`safe` Sets the file protection level to `safe` for the control channel and file transfer. All messages sent on the control and data channels are protected by cryptographic checksum.

`send` *local-file* [*remote-file*]
Indicates a synonym for `put`.

`sendport`
Toggles the use of `PORT` commands. By default, `ftp` attempts to use a `PORT` command when establishing a connection for each data transfer. The use of `PORT` commands can prevent delays when performing multiple file transfers. If the `PORT` command fails, `ftp` uses the default data port. When the use of `PORT` commands is disabled, no attempt is made to use `PORT` commands for each data transfer. This is useful for certain FTP implementations that ignore `PORT` commands, but incorrectly indicate that they are accepted.

`showbuf` Toggles display of copy buffer and socket buffer sizing information during a transfer. Default is no display of information.

`site` *arguments*
The specified arguments are sent to the remote FTP server as arguments to an FTP `SITE` command. In return, one FTP reply code is expected.

`size` *remote-file*
Shows the size of the remote file.

`sockbuf` [*arg*]
Sets the socket buffer size. This kernel buffer is used for data transfer on the data socket. FTP automatically chooses a socket buffer size; however, you can alter the selection. `sockbuf` takes an optional argument. An argument of `off` turns off socket buffer sizing and the buffer defaults to the kernel's default socket buffer size. An argument of `auto` returns buffer sizing to automatic. A numeric argument sets the buffer to that size. The letter `K` or `k` can follow a numeric buffer size to specify a multiple of 1024. Without an argument, `sockbuf` shows the current socket buffering. The default setting is `auto`.

`status` Shows the current status of `ftp`.

`struct` [*struct-name*]
Sets the file transfer structure to *struct-name*. By default, file structure is used. Currently, only the default is supported.

`sunique` Toggles storing of files that have unique file names on a remote machine. The remote FTP server must support the FTP protocol `STOU` command. The remote server reports unique names. The default value is `off`.

`system` Shows the remote system type.

`tenex` Sets the file transfer *type* to that which is needed to talk to TENEX machines.

`trace` Toggles packet tracing.

`type` [*type-name*]
Sets the file transfer `type` to *type-name*. If *type-name* is not specified, the current type is printed. The three valid types are `tenex`, `binary`, and `ascii`, which is the default.

`umask` *mask*
Gets and sets `umask` on the remote side.

`user` *user-name* [*password*] [*account*]
Identifies you to the remote FTP server. If *password* is not specified and the server requires it, `ftp` prompts you for it (after disabling the local echo). If *account* is not specified, and the FTP server requires it, you are prompted for it. If an account field is specified, an account command is relayed to the remote server after the login sequence is completed if the remote server does not require it to log on. Unless you invoke `ftp` with the autologin disabled, this process is completed automatically during the initial connection to the FTP server.

`verbose` Toggles verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. If `verbose` is on or when a file transfer completes, statistics are reported about the efficiency of the transfer. If `ftp` is started with the `-v` option or if input is a terminal, the default is `on`.

`winshift` [*value*]
Gets or sets the value for the TCP window shift option to be used on data connections. If no argument is specified, this option reports the current value. If *value* is `off`, the option is disabled; if *value* is `on`, the option is enabled with a value of 4; if *value* is an integer value between 0 and 14, the TCP window shift option is enabled with that value. Because the client side of an FTP always performs a passive open operation, you do not have to disable the sending of the TCP window shift option (if the incoming SYN packet does not contain the option, none will be sent in the SYN,ACK packet, regardless of whether the application has enabled the option).

`?` [*command*]
Indicates a synonym for help.

Command arguments that have embedded spaces may be enclosed with quotation marks.

Aborting a File Transfer

To abort a file transfer, use the terminal interrupt key (`<CONTROL-c>`). Sending transfers is halted immediately. Receiving transfers is halted by sending a FTP protocol `ABOR` command to the remote server or by discarding any further data that is received. The speed at which this is accomplished depends on the remote server's support for `ABOR` processing. If the remote server does not support the `ABOR` command, an `ftp>` prompt does not appear until the remote server completes sending the requested file.

The terminal interrupt key sequence is ignored when ftp completes any local processing and is waiting for a reply from the remote server. A long delay in this mode can result from the ABOR processing described in the preceding paragraph, or from the unexpected remote server behavior, including violations of the ftp protocol. If the delay is caused by unexpected remote server behavior, the local ftp program must be killed by hand.

File Naming Conventions

Files specified as arguments to ftp utilities are processed according to the following rules:

- If the file name is specified as a dash (-), stdin (for reading) or stdout (for writing) is used.
- If the first character of the file name is |, the remainder of the argument is interpreted as a shell command. ftp then forks a shell with the argument supplied and reads (or writes) from stdout (or stdin). If the shell command includes spaces, the argument must be enclosed in quotation marks (for example, "| ls -lt"). A particularly useful example of this mechanism is dir *directory_name* "|pg".
- Failing the preceding checks, if globbing is enabled, local file names are expanded according to the rules used in csh(1); see the glob command. If the ftp utility expects a single local file (for example, put), only the first file name that glob operation generates is used.
- When using mget and get commands that have unspecified local file names, the local file name is the remote file name, which can be altered by a case, ntrans, or nmap setting. When runique is on, the resulting file name may then be altered.
- When using mput and put commands that have unspecified remote file names, the remote file name is the local file name, which may be altered by an ntrans or nmap setting. When sunique is on, the remote server can alter the resulting file name.

File Transfer Parameters

The ftp specification specifies many parameters that can affect a file transfer. The type can be one of ASCII, image (binary), EBCDIC, and local byte size (for PDP-10s and PDP-20s). The ftp utility supports the ASCII and image types of file transfer, and also local byte size 8 for TENEX mode transfers.

The ftp utility supports only the default values for the remaining file transfer parameters: mode, form, and struct.

NOTES

The SHELL environment variable defines the shell that is used in shell-escapes. If SHELL is undefined, the default shell /bin/sh is used. The HOME environment variable is used to locate the user's home directory when it reads the .netrc file.

BUGS

Several FTP server implementations do not support operations such as `pwd`.

You must use the `mget` and `mdelete` commands with caution. If you specify a directory that expects a plain file name, unexpected results can be produced.

EXAMPLES

The following example transmits an executable binary file from the system `host` to the system `cray`. Typewriter bold font indicates user input:

```
host% ftp cray
Cray FTP server ready.
Name (cray:dep): dep
Password (cray:dep):            Enter your password
Password required for dep.
User dep logged in.
ftp> binary
Type set to I.
ftp> put a.out
PORT command successful.
Opening data connection for a.out (port number).
Transfer complete.
N bytes sent in n seconds
ftp> quit
Goodbye.
```

FILES

`/etc/hosts`	File that contains the database of all locally known hosts on the TCP/IP network
`$HOME/.netrc`	File that contains login information required for `ftp` access to a remote machine

SEE ALSO

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

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

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

Software Overview for Users, Cray Research publication SG–2052

NAME

fts – Performs file transfer server function for bftp(1B)

SYNOPSIS

/usr/ucb/fts

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The fts utility is the transfer request server for the background file transfer program (BFTP). You can use bftp(1B) to submit a request to have a file transferred at some time in the future by the use of the standard Internet file transfer protocol (FTP), which is described in RFC 959.

The fts utility is not a user command and is used only in association with bftp(1B).

For information on BFTP, see RFC 1068.

SEE ALSO

at(1), bftp(1B), crontab(1), ftp(1B)

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

RFC 959, Postel, J.B., Reynolds, J.K. File Transfer Protocol. October 1985: 69 pages.
RFC 1068, DeSchon, A.L., Braden, R.T. Background File Transfer Program (BFTP). August 1988; 27 pages.

NAME

gencat – Generates a message catalog

SYNOPSIS

gencat *catfile msgfiles*

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

This utility was formerly called catgen(8). It has been renamed gencat(1) and moved to /usr/bin for UNICOS 8.3, UNICOS 9.0, and subsequent releases to conform with the X/Open XPG4 specification. The utility that was called gencat(8) in former releases is now called mfsck(8). The functionality of neither utility has changed, only the names.

The gencat utility generates a formatted message catalog from the output of the caterr(1) utility. The files output from caterr are input to gencat *msgfile*.

The gencat utility accepts the following arguments:

catfile Specifies the name to be given to the catalog file output from gencat. If *catfile* exists, its messages are included in the updated *catfile*. Messages and explanations from the input files replace messages and explanations of identical set and message numbers in the existing *catfile*. If *catfile* does not exist, gencat creates it.

msgfiles Specifies the name of the file or files input to gencat. The input to gencat must be a file output from caterr(1). The caterr utility processes the message file; the gencat utility outputs the message or explanation catalog.

NOTES

When the -c option is specified on the caterr command line, gencat is called from caterr. It is recommended that you call gencat in this fashion. The gencat utility exists as a separate utility to maintain compatibility with industry standards for message processing. No advantage exists in calling gencat directly.

Message catalogs that gencat produces are binary encoded. This means that their portability cannot be guaranteed among various types of machines. Therefore, just as C programs must be recompiled for each type of machine, message catalogs must be re-created using gencat.

EXIT STATUS

An exit value of 0 is returned if the command completed successfully. An exit value greater than 0 is returned if an error occurred that prevented successful completion.

EXAMPLES

The following example updates the existing catalog `dbg.cat` with the information in the `dbg.in` file:

```
gencat dbg.cat dbg.in
```

SEE ALSO

`caterr`(1), `catxt`(1), `explain`(1), `whichcat`(1)

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

`nl_types`(5) 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

get – Gets a version of an SCCS file

SYNOPSIS

get [-a "*seq-no*"] [-b] [-c *cutoff*] [-e] [-g] [-i *list*] [-k] [-l] [-L] [-m] [-n] [-p] [-r *SID*] [-s] [-t] [-w *string*] [-x *list*] *file*

Obsolescent version; may not be supported in future releases:

get [-a "*seq-no*"] [-b] [-c *cutoff*] [-e] [-g] [-i *list*] [-k] [-l[p]] [-m] [-n] [-p] [-r *SID*] [-s] [-t] [-w *string*] [-x *list*] *file*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The get utility generates an ASCII text file from each named Source Code Control System (SCCS) file according to the specifications given by its options, which begin with a dash (-). The options may be specified in any order, but all options apply to all named SCCS files. If a directory is named, get behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with s.) and unreadable files are silently ignored. If a name of - is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

The generated text is usually written into a file called the *g-file* whose name is derived from the SCCS file name by simply removing the leading s. (See the FILES section.)

Each of the options is explained below as though only one SCCS file is to be processed, but the effects of any option applies independently to each named file.

-a"*seq-no*" The delta sequence number of the SCCS file delta (version) to be retrieved (see sccsfile(5)). This option is used by the comb(1) command; it is not a generally useful option, and users should not use it. If both the -r and -a options are specified, the -a option is used. Care should be taken when using the -a option in conjunction with the -e option, because the source identifier (SID) of the delta to be created may not be what you expect. The -r option can be used with the -a and -e options to control the naming of the SID for the delta to be created.

`-b` Used with the `-e` option to indicate that the new delta should have an SID in a new branch as shown in the table below. This option is ignored if the `b` flag is not present in the file (see `admin`(1)) or if the retrieved *delta* is not a leaf *delta*. (A leaf *delta* is one that has no successors on the SCCS file tree.)

Note: A branch *delta* may always be created from a nonleaf *delta*.

`-c`*cutoff* Indicates the *cutoff* date-time. This appears in the form:

YY[*MM*[*DD*[*HH*[*MM*[*SS*]]]]]

No changes (deltas) to the SCCS file that were created after the specified *cutoff* date-time are included in the generated ASCII text file. Units omitted from the date-time default to their maximum possible values; that is, `-c7502` is equivalent to `-c750228235959`. Any number of nonnumeric characters may separate the various 2-digit pieces of the *cutoff* date-time. This feature lets you specify a *cutoff* date-time in the form `-c77/2/2 9:22:25`. This implies that you may use the `%E%` and `%U%` identification keywords (see below) for nested `gets`:

```
~!get "-c%E% %U%" s.file
```

`-e` Indicates that the `get` is for the purpose of editing or making a change (delta) to the SCCS file through a subsequent use of `delta`(1). The `-e` option used in a `get` for a particular version (SID) of the SCCS file prevents further `get`s for editing on the same SID until *delta* is executed or the `j` (joint edit) flag is set in the SCCS file (see `admin`(1)). Concurrent use of `get -e` for different SIDs is always allowed.

If the *g-file* generated by `get` with an `-e` option is accidentally damaged in the process of editing it, it may be regenerated by re-executing the `get` command with the `-k` option in place of the `-e` option.

SCCS file protection specified by using the ceiling, floor, and authorized user list stored in the SCCS file (see `admin`(1)) are enforced when the `-e` option is used.

`-g` Suppresses the actual retrieval of text from the SCCS file. It is primarily used to generate an *l-file*, or to verify the existence of a particular SID.

`-i`*list* A *list* of deltas to be included (forced to be applied) in the creation of the generated file. The *list* has the following syntax:

<list> ::= <range> | <list> , <range>
<range> ::= SID | SID - SID

SID, the SCCS identification of a delta, may be in any form shown in the "SID specified" column of the table below. Partial SIDs are interpreted as shown in the "SID retrieved" column of the table below.

`-k` Suppresses replacement of identification keywords in the retrieved text by their value. The `-k` option is implied by the `-e` option.

`-l` Causes a delta summary to be written into an *l-file*. See the FILES section for the format of the *l-file*.

`-L` Causes a delta summary to be written to standard output.

`-lp` Equivalent to `-L`.

`-m` Causes each text line retrieved from the SCCS file to be preceded by the SID of the delta that inserted the text line in the SCCS file. The format is: SID, followed by a horizontal tab, followed by the text line.

`-n` Causes each generated text line to be preceded with the `%M%` identification keyword value. The format is: `%M%` value, followed by a horizontal tab, followed by the text line. When both the `-m` and `-n` options are used, the format is: `%M%` value, followed by a horizontal tab, followed by the `-m` option generated format.

`-p` Causes the text retrieved from the SCCS file to be written on the standard output. No *g-file* is created. All output that normally goes to the standard output goes to file descriptor 2 instead, unless the `-s` option is used, in which case it disappears.

`-r` *SID* The *S*CCS *ID*entification string (SID) of the version (delta) of an SCCS file to be retrieved. Table 1 shows, for the most useful cases, what version of an SCCS file is retrieved (as well as the SID of the version to be created eventually by `delta`(1) if the `-e` option is also used), as a function of the SID specified.

`-s` Suppresses all output usually written on the standard output. However, fatal error messages (which always go to file descriptor 2) remain unaffected.

`-t` Used to access the most recently created (‘‘top’’) delta in a given release (such as, `-r1`), or release and level (such as, `-r1.2`).

`-w` *string* Substitutes *string* for all occurrences of "`% w%`" when using the `get` command on a file.

`-x` *list* A *list* of deltas to be excluded (forced not to be applied) in the creation of the generated file. See the `-i` option for the *list* format.

file Specifies the SCCS file to get.

For each file processed, `get` responds (on the standard output) with the SID being accessed and with the number of lines retrieved from the SCCS file.

If the `-e` option is used, the SID of the delta to be made appears after the SID accessed and before the number of lines generated. If there is more than one named file or if a directory or standard input is named, each file name is printed (preceded by a new line) before it is processed. If the `-i` option is used, included deltas are listed following the notation `Included`; if the `-x` option is used, excluded deltas are listed following the notation `Excluded`.

SID† specified	-b keyletter used††	Other conditions	SID retrieved	SID of delta to be created
None†††	No	R defaults to mR	mR.mL	mR.(mL+1)
None†††	Yes	R defaults to mR	mR.mL	mR.mL.(mB+1).1
R	No	R > mR	mR.mL	R.1††††
R	No	R = mR	mR.mL	mR.(mL+1)
R	Yes	R > mR	mR.mL	mR.mL.(mB+1).1
R	Yes	R = mR	mR.mL	mR.mL.(mB+1).1
R	–	R < mR and R does *not* exist	hR.mL†††††	hR.mL.(mB+1).1
R	–	Trunk successor in release > R and R exists	R.mL	R.mL.(mB+1).1
R.L	No	No trunk successor	R.L	R.(L+1)
R.L	Yes	No trunk successor	R.L	R.L.(mB+1).1
R.L	–	Trunk successor in release ≥ R	R.L	R.L.(mB+1).1
R.L.B	No	No branch successor	R.L.B.mS	R.L.B.(mS+1)
R.L.B	Yes	No branch successor	R.L.B.mS	R.L.(mB+1).1
R.L.B.S	No	No branch successor	R.L.B.S	R.L.B.(S+1)
R.L.B.S	Yes	No branch successor	R.L.B.S	R.L.(mB+1).1
R.L.B.S	–	Branch successor	R.L.B.S	R.L.(mB+1).1

† R, L, B, and S are the release, level, branch, and sequence components of the SID, respectively; m means maximum. Thus, for example, R.mL means the "maximum level number within release R"; R.L.(mB+1).1 means "the first sequence number on the new branch (that is, maximum branch number plus one) of level L within release R." Note that if the SID specified is of the form R.L, R.L.B, or R.L.B.S, each of the specified components must exist.

†† hR is the highest existing release that is lower than the specified, nonexistent, release R.

††† This is used to force creation of the first delta in a new release.

†††† The -b option is effective only if the b flag (see admin(1)) is present in the file. An entry of - means ‘‘irrelevant.’’

††††† This case applies if the d (default SID) flag is not present in the file. If the d flag *is* present in the file, the SID obtained from the d flag is interpreted as if it had been specified on the command line. Thus, one of the other cases in this table applies.

Identification Keywords

Identifying information is inserted into the text retrieved from the SCCS file by replacing identification keywords with their value wherever they occur. The following keywords may be used in the text stored in an SCCS file:

Keyword	Value
%M%	Module name: either the value of the m flag in the file (see admin(1)), or if absent, the name of the SCCS file with the leading s. removed.
%I%	SCCS identification (SID) (%R%.%L%.%B%.%S%) of the retrieved text
%R%	Release.
%L%	Level.
%B%	Branch.
%S%	Sequence.
%D%	Current date (*Y*/*MM*/*DD*).
%H%	Current date (*MM*/*DD*/*YY*).
%T%	Current time (*HH*:*MM*:*SS*).
%E%	Date newest applied delta was created (*YY/MM/DD).*
%G%	Date newest applied delta was created (*MM/DD/YY).*
%U%	Time newest applied delta was created (*HH:MM:SS).*
%Y%	Module type: value of the t flag in the SCCS file (see admin(1)).
%F%	SCCS file name.
%P%	Fully qualified SCCS file name.
%Q%	The value of the q flag in the file (see admin(1)).
%C%	Current line number. This keyword identifies messages output by the program such as ‘‘this should not have happened’’ type errors. It is not intended to be used on every line to provide sequence numbers.
%Z%	The 4-character string @(#) recognizable by what(1).
%W%	A shorthand notation for constructing what(1) strings for UNICOS system program files. %W%~=~%Z%%M%<horizontal-tab>%I%
%A%	Another shorthand notation for constructing what(1) strings for non-UNICOS system program files. %A%~=~%Z%%Y%~%M%~%I%%Z%

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

MESSAGES

Use help(1) for explanations.

BUGS

If the effective user has write permission (either explicitly or implicitly) in the directory containing the SCCS files, but the real user does not, only one file may be named when the `-e` option is used.

EXAMPLES

The following example retrieves the latest delta for editing. The file `example.c` is created. User input is indicated with bold type:

```
$ get -e s.example.c
1.1
new delta 1.2
5 lines
$
```

FILES

Several auxiliary files may be created by `get`. These files are known generically as the *g-file*, *l-file*, *p-file*, and *z-file*. The letter before the hyphen is called the tag. An auxiliary file name is formed from the SCCS file name: the last component of all SCCS file names must be of the form `s.`*module-name*, the auxiliary files are named by replacing the leading `s` with the tag. The *g-file* is an exception to this scheme: the *g-file* is named by removing the `s.` prefix. For example, `s.xyz.c`, the auxiliary file names would be `xyz.c`, `l.xyz.c`, `p.xyz.c`, and `z.xyz.c`, respectively.

The *g-file*, which contains the generated text, is created in the current directory (unless the `-p` option is used). A *g-file* is created in all cases, whether or not any lines of text were generated by `get`. It is owned by the real user. If the `-k` option is used or implied, its mode is 644; otherwise, its mode is 444. Only the real user must have write permission in the current directory.

The *l-file* contains a table showing which deltas were applied in generating the retrieved text. The *l-file* is created in the current directory if the `-l` option is used; its mode is 444 and it is owned by the real user. Only the real user needs to have write permission in the current directory.

Lines in the *l-file* have the following format:

- A blank character if the delta was applied; otherwise, `*`.
- A blank character if the delta was applied or was not applied and ignored; `*` if the delta was not applied and was not ignored.
- A code indicating a ‘‘special’’ reason why the delta was or was not applied:

  `I`: Included
  `X`: Excluded
  `C`: Cut off (by a `-c` option)

- Blank
- SCCS identification (SID)

- Tab character
- Date and time (in the form *YY*/*MM*/*DD*˜*HH*:*MM*:*SS*) of creation
- Blank
- Login name of person who created *delta*

The comments and memory resident (MR) data follow on subsequent lines, indented one horizontal tab character. A blank line terminates each entry.

The *p-file* is used to pass information resulting from a `get` with an `-e` option along to *delta*. Its contents are also used to prevent a subsequent execution of `get` with an `-e` option for the same SID until *delta* is executed or the joint edit flag, `j`, (see `admin`(1)) is set in the SCCS file. The *p-file* is created in the directory containing the SCCS file and the effective user must have write permission in that directory. Its mode is 644 and it is owned by the effective user. The format of the *p-file* is: the retrieved SID, followed by a blank, followed by the SID that the new delta will have when it is made, followed by a blank, followed by the login name of the real user, followed by a blank, followed by the date-time the `get` was executed, followed by a blank and the `-i` option if it was present, followed by a blank and the `-x` option if it was present, followed by a new line. There can be an arbitrary number of lines in the *p-file* at any time; no two lines can have the same new delta SID.

The *z-file* serves as a *lock-out* mechanism against simultaneous updates. Its contents are the binary process ID of the command (that is, `get`) that created it. The *z-file* is created in the directory containing the SCCS file for the duration of `get`. The same protection restrictions as those for the *p-file* apply for the *z-file*. The *z-file* is created mode 444.

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

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

NAME

getconf – Gets configuration values

SYNOPSIS

getconf *system_var*
getconf *path_var pathname*

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

In the first synopsis form, the getconf utility writes to the standard output the value of the variable specified by the *system_var* operand.

In the second synopsis form, the getconf utility writes to the standard output the value of the variable specified by the *path_var* operand for the path specified by the *pathname* operand.

The following operands are supported:

system_var A name of a configuration variable that has a value available from sysconf(2) and confstr(3C). The values in the following table are supported:

ARG_MAX	CRAY_SYSMEM	POSIX2_VERSION
BC_BASE_MAX	CRAY_USRMEM	POSIX_ARG_MAX
BC_DIM_MAX	CS_PATH	POSIX_CHILD_MAX
BC_SCALE_MAX	EXPR_NEST_MAX	POSIX_JOB_CONTROL
BC_STRING_MAX	INT_MAX	POSIX_LINK_MAX
CHARCLASS_NAME_MAX	INT_MIN	POSIX_MAX_CANON
CHAR_BIT	LINE_MAX	POSIX_MAX_INPUT
CHAR_MAX	LONG_BIT	POSIX_NAME_MAX
CHAR_MIN	LONG_MAX	POSIX_NGROUPS_MAX
CHILD_MAX	LONG_MIN	POSIX_OPEN_MAX
CLK_TCK	MB_LEN_MAX	POSIX_PATH_MAX
COLL_WEIGHTS_MAX	MN_NMAX	POSIX_PIPE_BUF
CRAY_AVL	NGROUPS_MAX	POSIX_SAVED_IDS
CRAY_BDM	NL_ARGMAX	POSIX_SSIZE_MAX
CRAY_CHIPSZ	NL_LANGMAX	POSIX_STREAM_MAX
CRAY_CPCYCLE	NL_MSGMAX	POSIX_TZNAME_MAX
CRAY_EMA	NL_SET_MAX	POSIX_VERSION
CRAY_HPM	NL_TEXT_MAX	RE_DUP_MAX

CRAY_IOS	NZERO	SCHAR_MAX
CRAY_MFSUBTYPE	OPEN_MAX	SCHAR_MIN
CRAY_MFTYPE	POSIX2_BC_BASE_MAX	SHRT_MAX
CRAY_NBANKS	POSIX2_BC_DIM_MAX	SHRT_MIN
CRAY_NBUF	POSIX2_BC_SCALE_MAX	SSIZE_MAX
CRAY_NCPU	POSIX2_BC_STRING_MAX	STREAM_MAX
CRAY_NDISK	POSIX2_CHAR_TERM	TMP_MAX
CRAY_NMOUNT	POSIX2_COLL_WEIGHTS_MAX	TZNAME_MAX
CRAY_NPTY	POSIX2_C_BIND	UCHAR_MAX
CRAY_NUSERS	POSIX2_C_DEV	UINT_MAX
CRAY_NVHISP	POSIX2_C_VERSION	ULONG_MAX
CRAY_OS_HZ	POSIX2_EXPR_NEST_MAX	USHRT_MAX
CRAY_RELEASE	POSIX2_FORT_DEV	WORD_BIT
CRAY_SCTRACE	POSIX2_FORT_RUN	XOPEN_VERSION
CRAY_SDS	POSIX2_LINE_MAX	XOPEN_XCU_VERSION
CRAY_SECURE_MAC	POSIX2_LOCALEDEF	XOPEN_XPG2
CRAY_SECURE_SYS	POSIX2_RE_DUP_MAX	XOPEN_XPG3
CRAY_SERIAL	POSIX2_SW_DEV	XOPEN_XPG4
CRAY_SSD	POSIX2_UPE	

The symbol PATH also is recognized, yielding the same value as the confstr(3C) name value CS_PATH.

path_var A name of a configuration variable that has a value available from pathconf(2). The values in the following table are supported:

LINK_MAX	NAME_MAX	POSIX_CHOWN_RESTRICTED
MAX_CANON	PATH_MAX	POSIX_NO_TRUNC
MAX_INPUT	PIPE_BUF	POSIX_VDISABLE

pathname A path name for which the variable specified by *path_var* will be determined.

If the specified variable is valid, but is undefined on the system, getconf writes undefined to stdout. If the variable name is not valid or an error occurs, getconf writes nothing to stdout.

EXIT STATUS

The getconf utility exits with one of the following values:

0 The specified variable is valid, and information about its current state was written successfully.

>0 An error occurred.

EXAMPLES

Example 1: The following command prints out the multigroup size:

```
getconf NGROUPS_MAX
```

Example 2: The following command prints out the value of {NAME_MAX} for a specific directory:

```
getconf NAME_MAX /usr
```

Example 3: The following shell script fragment example shows how to deal more carefully with results that might be unspecified:

```
if value=$(getconf PATH_MAX /usr); then
     if [ "$value" = "undefined" ]; then
          echo PATH_MAX in /usr/ is infinite.
     else
          echo PATH_MAX in /usr is $value.
     fi
else
     echo Error in getconf.
fi
```

SEE ALSO

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

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

NAME

`getopt` – Parses command options

SYNOPSIS

`getopt` *optstring* [*arguments*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `getopt` utility breaks up options in command lines for easy parsing by shell procedures and checks for legal options.

optstring is a string of recognized option letters (see `getopt`(3C)). When a letter is followed by a colon, the option is expected to have an argument that may or may not be separated from it by white space.

arguments is a list of 0 or more blank-separated words. These words are usually options, option-arguments, and operands (nonoption words, such as file names).

The following output from `getopt` is displayed on one line in the order given:

- Each option, optionally followed by a separate word, the option-argument. Each option is preceded by a minus sign.
- The characters `--`. If these characters are not specified, they will be generated automatically by `getopt` to indicate the end of the options list.
- Operands (nonoption words such as file names).

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.

`getopts`(1) is the preferred method for parsing command lines of standard shell procedures.

MESSAGES

When `getopt` encounters an option letter not included in *optstring*, it prints an error message on the standard error.

EXAMPLES

The following standard shell script fragment shows how you might process the arguments for a command that can take the `-a` or `-b` options, as well as the `-o` option, which requires an argument:

```
oarg=
flag=
USAGE="Usage: $0 [-ab] [-o oarg] file1 file2"
opts=‘getopt abo: $*‘
err=$?
set -- $opts
#
if [ $err -eq 0 ]
then
      while [ "$1" != "--" ]
      do
            case $1 in
            -a | -b)    flag=$1;;
            -o)         oarg=$2; shift;;
            esac
            shift
      done
      shift
fi
#
if [ $err -ne 0 -o $# -ne 2 ]
then
      echo $USAGE >&2
      exit 1
.
.
.
```

The following are some of the command lines that both procedures accept as equivalent:

cmd `-abo` *arg file file*
cmd `-ab -o` *arg file file*
cmd `-b -a -o`*arg file file*
cmd `-bo` *arg* `-a` *file file*
cmd `-o` *arg* `-ba --` *file file*
cmd `-a -o` *arg* `-b --` *file file*

Because `-o` takes an argument, the following command line is *not* equivalent to those in the previous list:

cmd `-oab` *arg file file*

SEE ALSO

`getopts`(1), `sh`(1)

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

NAME

getopts – Parses utility options

SYNOPSIS

getopts *optstring name* [*args*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The getopts utility is used by standard shell procedures to parse positional parameters and to check for legal options. It supports all applicable rules of the utility argument syntax standard.

The *optstring* argument must contain the option letters recognized by the command using getopts; however, when a letter is followed by a colon, the option is expected to have an argument, or group of arguments, which may or may not be separated from it by white space. You should use white space to separate options and their arguments.

Each time it is invoked, getopts places the next option in *name* shell variable and the index of the next argument to be processed in the OPTIND shell variable. Whenever the shell or a shell procedure is invoked, OPTIND is initialized to 1.

When an option requires an option-argument, getopts places it in shell environment variable OPTARG.

If an illegal option is encountered, ? will be placed in *name*.

When the end of options is encountered, getopts exits with a nonzero exit status. You can use the special option ‘‘--’’ to delimit the end of the options.

By default, getopts parses the positional parameters. If you specify extra arguments (*args*) on the getopts command line, getopts will parse them instead.

To ensure that all new shell procedures adhere to the utility argument syntax standard, they should use getopts.

The shell variable specified by the *name* operand, OPTIND, and OPTARG affect the current shell execution environment.

CAUTIONS

Modifying the shell environment variable `OPTIND` to a value other than 1 or parsing different sets of arguments may lead to unexpected results.

NOTES

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

MESSAGES

When it encounters an option letter not included in *optstring*, `getopts` prints an error message on standard error. No output to standard error is written in this case if the first character in *optstring* is a colon (`:`).

EXAMPLES

Example 1: The following fragment of a shell program shows how you might process the arguments for a command that can take option `a` or `b`, as well as option `o`, which requires an argument:

```
flag=
oarg=
errflg=0
USAGE="Usage:  util_name [-ab] [-o oarg] file"
#
while getopts abo: option
do
        case $option in
        a | b)          flag=$option;;
        o)              oarg=$OPTARG;;
        \?)            errflg=1;;
        esac
done
shift `expr $OPTIND - 1`
if [ $errflg -ne 0 -o $# -ne 1 ]
then
        echo $USAGE >&2
        exit 2
fi
.
.
.
```

This code accepts any of the following as equivalent:

```
util_name -a -b -o "xxx z yy" file
util_name -a -b -o "xxx z yy" -- file
util_name -ab -o xxx,z,yy file
util_name -ab -o "xxx z yy" file
util_name -o xxx,z,yy -b -a file
```

SEE ALSO

getopt(1), sh(1)

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

NAME

grep, egrep, fgrep – Searches a file for a pattern

SYNOPSIS

grep [-E | -F] [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] -e *pattern_list*...
[-f *pattern_file*]... [*file*...]

grep [-E | -F] [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] [-e *pattern_list*]...
-f *pattern_file*... [*file*...]

grep [-E | -F] [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] *pattern_list* [*file*...]

Obsolescent version; may not be supported in future releases:

egrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] -e *pattern_list*...
[-f *pattern_file*]... [*file*...]

egrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] [-e *pattern_list*]...
-f *pattern_file*... [*file*...]

egrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] *pattern_list* [*file*...]

fgrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] -e *pattern_list*...
[-f *pattern_file*]... [*file*...]

fgrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] [-e *pattern_list*]...
-f *pattern_file*... [*file*...]

fgrep [-c | -l | -q] [-b] [-h] [-i] [-n] [-s] [-v] [-x] [-y] *pattern_list* [*file*...]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4
AT&T extensions (-b, -h, and -y options)

DESCRIPTION

The grep utility searches files for a pattern and prints all lines that contain that pattern. It uses basic regular expressions (BRE), expressions that have string values that use a subset of the possible alphanumeric and special characters, such as those used with ed(1) to match the patterns. A null BRE matches every line.

When the -E option is specified (or the egrep utility is used), grep uses extended regular expressions (ERE) to match the patterns.

`egrep` accepts EREs as in `ed`(1), except for `\(` and `\)`, with the addition of the following:

1. An ERE followed by `+` that matches one or more occurrences of the ERE.
2. An ERE followed by `?` that matches zero or one occurrences of the ERE.
3. An ERE separated by `|` or by a `<newline>` character that match strings that are matched by any of the expressions.
4. An ERE that may be enclosed in parentheses `()` for grouping.

The order of precedence of operators is `[]`, then `* ? +`, then concatenation, then `|` and `<newline>`.

When the `-F` option is specified (or the `fgrep` utility is used), `grep` searches for a string, rather than searching for a pattern that matches an expression.

Because the `$`, `*`, `[`, `^`, `|`, `(`, `)`, and `\` characters in the regular expressions are also meaningful to the shell, it is safest to enclose the entire expression in single quotation marks (`' ... '`).

If files are not specified, `grep` assumes standard input. Usually, each line found is copied to standard output. If more than one input file exists, the file name is printed before each line found.

The `grep` utility accepts the following options:

`-E` Matches using extended regular expressions (ERE). Treat each pattern specified as an ERE. A null ERE matches every line.

`-F` Matches using fixed strings. Treat each pattern specified as a string, rather than a RE. A null string matches every line.

`-b` Precedes each line by the block number on which it was found. Blocks are 512-bytes in size. This can be useful in locating block numbers by context (first block is 0).

`-c` Prints only a count of the lines that contain the pattern.

`-e` *pattern_list*
Specifies one or more patterns to be used during the search for input. Patterns in *pattern_list* are separated by a `<newline>`. To specify a null pattern, use two adjacent `<newline>`s in *pattern_list*. Unless the `-E` or `-F` option is also specified, each pattern is treated as a basic regular expression. If multiple `-e` or `-f` options are specified, all of the specified patterns will be used when matching lines, but the order of evaluation is unspecified.

`-f` *pattern_file*
Reads one or more patterns from the file specified by the path name *pattern_file*. Patterns in *pattern_file* are separated by a `<newline>`. A null pattern can be specified by an empty line in *pattern_file*. Unless the `-E` or `-F` option is also specified, each pattern is treated as a basic regular expression.

`-h` Prevents the name of the file that contains the matching line from being appended to that line. Used when searching multiple files.

`-i`
`-y` Ignores uppercase and lowercase distinction during comparisons.

`-l` Prints the names of files that have matching lines once, separated by `<newline>` characters. Does not repeat the names of files when the pattern is found more than once.

`-n` Precedes each line by its line number in the file (first line is 1).

`-q` Quiet. Nothing is written to the standard output, regardless of matching lines. `grep` exits with zero status if any matches are found, even if an error was detected.

`-s` Suppresses error messages about nonexistent or unreadable files.

`-v` Prints all lines except those that contain the pattern.

`-x` Considers only input lines that use all characters in the line to match an entire fixed string or regular expression to be matching lines.

files The path names of files to be checked.

NOTES

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

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

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

EXIT STATUS

Exit status is 0 if any matches are found, 1 if none are found, or 2 if an error occurred, even if matches were found (see `-q` option).

EXAMPLES

Example 1: The following command prints on standard output all lines that contain the pattern `Tom Jones` in all files in the current directory:

```
grep 'Tom Jones' *
```

Example 2: The following command prints on standard output all lines within `manuals` file that do not contain the string `User Commands`:

```
grep -v 'User Commands' manuals
```

Example 3: The following command searches for the pattern `User Commands` in the `manuals` file and sends the output to the `users` file:

```
grep 'User Commands' manuals > users
```

SEE ALSO

`ed`(1), `sed`(1), `sh`(1)

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

NAME

`groups` – Shows group memberships

SYNOPSIS

`/usr/ucb/groups` [*user*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `groups` command shows the groups to which you or the optionally specified user belong. Each user belongs to the groups specified in `/etc/udb`. If you do not own a file but belong to the group by which it is owned, you are granted group access to the file.

When a new file is created, it is given the same group membership as that of the directory in which it is contained.

The `groups` command accepts the following option:

user Name of user that you specify.

NOTES

If this command is installed with the default privilege assignment list (PAL), a user with an active `secadm` category may override mandatory access control (MAC) and discretionary access control (DAC) protections in a privileged administrator shell environment on any file to which input or from which output is being redirected. A user with an active `sysadm` category may override DAC protections in a privileged administrator shell environment on any file to which input or from which output is being redirected, but is constrained by the MAC policy.

Use `id -Gn` to produce the same results as invoking `groups`.

The `groups` command will not be supported after the UNICOS 9.0 release.

EXAMPLES

The following example lists all groups in which you are a member. User input is shown in bold type.

```
$ groups
resrch dev acct pubs
```

FILES

`/etc/udb`	Password file containing login information
`/etc/group`	File containing group names and group IDs

SEE ALSO

`id`(1), `ls`(1)

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

NAME

guest – Performs UNICOS Guest administrative functions

SYNOPSIS

```
guest [-v]
guest -a [-v] [guest_directory]
guest -s [-v] [-T] [-K] [-w] [-n node_name] [-m memsize] [guest_directory]
guest -x [-v] [-o] [-n node_name] [guest_directory]
guest -r [-v] [-n node_name] [guest_directory]
guest -q [-v] [-o] [-n node_name] [guest_directory]
guest -f|-t [-v] [-n node_name] [guest_directory]
guest -d [-v] [-p path] [-n node_name] [guest_directory]
guest -D [-v] [-p path] [guest_directory]
guest -M [-v] [guest_directory]
guest -U [-v] [guest_directory]
guest -c [-v] [-K] [-T] [-n node_name] [-O]
guest -P [-v] name:percent[,name:percent[,...]]
guest -e routine (Can be specified on all guest invocations)
guest -E (Can be specified on all guest invocations)
```

IMPLEMENTATION

Cray PVP systems (except CRAY J90 series and CRAY EL series)

DESCRIPTION

The guest command provides functions to start, status, stop, dump, and change certain attributes of the operating systems running as guests under UNICOS. When specified with no parameters, it will return a text string indicating the *status* of the system on which it is currently running:

HOST Host system with active guests

host Stand-alone system with no guests active

guest Guest system

Except for status functions, all operations (including many of the uppercase options) require an appropriately authorized user.

Nearly all functions operate on a directory containing one or more of the following files or directories:

`guest.rc` A configuration file for the `guest` command. Required if the current directory does not contain both a UNICOS binary file (`unicos`) and a parameter file (`param`). Some command-line options override default settings in this file. The following directories will be searched (in order) for the `guest.rc` file:

guest_directory
`/usr/guest/$LOGNAME/`
`$HOME/`

For information on possible contents, see the FILES section.

`unicos` A UNICOS kernel binary file. Required only if the `quest.rc` file does not specify a binary file.

`param` A UNICOS param file. Required only if the `guest.rc` file does not specify a parameter file.

`crash` A crash binary associated with the UNICOS binary file. Not required.

`kcompress` A program to decompress the UNICOS binary (if necessary). Not required, but if not present, the start function will attempt to use the compression program in `/etc/kcompress`, which may not be compatible with the guest system binary.

`dump` A directory in which dumps will be placed. The directory will be created if it does not exist and if an alternate location is not specified in the `guest.rc` file.

The following command line options are available with most invocations:

`guest_directory`
A directory containing one or more of the files previously mentioned. It can be specified with most of the `guest` command invocations and defaults to the directory search list previously noted.

Status and debug options:

`-v` Lists a verbose `guest` status or status of the command invocation to `stdout`.

`-e` *routine* Provides extra debug output (not recommended).

`-E` Provides extra debug output (not recommended).

Verifying a guest directory:

`-a` This invocation will verify that the current (or specified) guest directory contains enough information to attempt to start a guest system. The system node name is not checked for uniqueness.

Starting a guest system:

`-s` [`-T`] [`-K`] [`-w`] [`-n` *node_name*] [`-m` *memsize*]

This invocation will start a guest system with information provided in the current (or specified) guest directory.

The `-T` option tells the host kernel to produce additional guest-related trace messages in the host and guest kernel.

The `-K` option informs the host system that a panic in the starting guest system should cause a subsequent panic in the host. (The `-T` and `-K` options require the UDB `guestadm` permission and are only useful for guest feature debugging.)

The `-w` option instructs the host to not allow CPUs to enter the guest system until a `guestctl`(2) (`TL RESUME`) call is made by a user program. This option is for internal debugging and is not generally supported.

The `-n node_name` option can be used to change the name of the kernel being started. Both options can be specified in the `guest.rc` file (`NODE_NAME`). The command will exit with a nonzero status if the system could not be started.

Stopping a guest system:

`-x` [`-o`] [`-n` *node_name*]
`-q` [`-o`] [`-n` *node_name*]

You must be the owner of the system that you are trying to stop. (For more information, see the Changing a guest system subsection.) If you do not release the guest memory, you may reload/restart a guest in the same memory, provided that the requested memory does not exceed the current allocation. The `-n` *node_name* option can be used to specify the guest system to be stopped. If a name is not specified on the command line, nor found in the user's `guest.rc` file, the user will be prompted with the names of systems currently owned by them. If the system status indicates that the guest is still in multiuser mode, the command will exit unless the `-o` (oblige) flag is set. The `-q` invocation is equivalent to executing both the stop (`-x`) and release (`-r`) invocations. The command will exit with a nonzero status if no guest system matches the specified criteria.

Releasing a guest system's memory:

`-r` [`-o`] [`-n` *node_name*]

This invocation will release mainframe memory held by the guest. If the guest's status is not *stopped*, the command will exit unless the `-o` (oblige) flag is set. The command will error exit if no guest memory is assigned to the user.

Freezing/Thawing a guest system:

`-f|-t` [`-n` *node_name*]
: In a guest system the `-f` (freeze) flag will only allow CPUs to enter the kernel (no exchanges will be done to a user process in the specified guest.) The `-t` (thaw) flag will reverse this constraint. The dump option uses the freeze/thaw concept internally.

Dumping a guest system:

`-d` [`-p` *path*] [`-n` *node_name*]
: This invocation will attempt to dump the associated guest system memory (and associated host memory) to a file. The dump output file can either be specified with the `-p` *path* option or with the `guest.rc` dump file option. The user will always be prompted for a dump reason to be included in the dump header. The command will exit with a nonzero status if no guest systems are in memory. Dumping a guest will not cause it to stop. Guest user processes will resume executing following the dump. (For more information, see the Freezing/Thawing a guest system subsection.)

`-D` [`-p` *path*]
: Dump all systems. This allows a guest administrator (or `root`) to dump all active systems. If no guest systems are active, only the host is dumped. The invocation requires guest administrator permissions.

Mounting/Unmounting filesystems:

`-M|-U`
: When a guest system is started (`-s`), a set of specified logical devices (see `guest.rc` file description) can be automatically unmounted. Associated `ldcache` is also released at this time. When the associated guest memory is released (`-r`), that same set of logical devices is checked with `fsck`(8) and then remounted. An attempt is made to restore `ldcache` (if previously allocated). The `-M` and `-U` invocations provide a means to mount and unmount (respectively) the file systems in the user's list, separate from starting a guest.

Changing a guest system:

`-c` [`-T`] [`-K`] [`-n` *node_name*] [`-O`]
: This invocation allows administrators to change control information for a running guest system. The `-T` and `-K` options are used for guest debugging. The `-T` option toggles the global guest-related trace flag. The `-K` option will toggle the *kill host on guest panic* flag. The `-n` *node_name* can be used to specify a guest system on which to act. The `-O` (owner) flag will change the owner of the guest system to the current user.

Changing CPU percentages:

`-P` *name:percent*[,*name:percent*[,...]]

The percentage of system CPU resources allocated to each system is, by default, equal to the percentage of mainframe memory occupied by each system or is based on an administrator defaults scheme. This allocation is important only when CPU resources are over-committed. The `-P` option will allow reallocation of the CPU resources across guest systems. A `name:percent` tuple must be included for each active guest and the host. The sum of the percentages must equate to 100. This option is only available to administrative users. Note that the next time a guest system is stopped or started, with the `guest` command, CPU resources will automatically be reallocated based on occupied memory or administrator defaults.

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 usurp guest system from the original owner. Locks or unlocks guest communication structures from the command line. Changes CPU percentages of active guests. Sets the guest kernel tracing or the *kill host on guest panic* flag.
`sysadm`	Allowed to use this command to perform all controlling functions except for those functions that require either an active `system` or `secadm` category.

If the `PRIV_SU` configuration option is enabled, the super user or a user with one of the following permission bits set are allowed to perform the following functions:

Permission	Action
`GUESTADMIN`	Allowed to usurp guest system from the original owner. Changes CPU percentages of active guests. Sets the guest kernel tracing or the *kill host on guest panic* flag.
`GUEST`	Allowed to use this command to perform all controlling functions except for those functions that require that the `GUESTADMIN` permission bit be set.

Read and write access to the `/usr/guest` directory can be confined to users in a particular UNICOS group (for example, `guest`) at the discretion of each site.

EXAMPLES

Example 1: Start a guest system from `/usr/guest/joeuser` directory. The directory contains the following files:

```
guest.rc        unicos  param   crash kcompress
% cd /usr/guest/joeuser
% guest -s
gst-45 guest: Info
   The guest kernel binary appears to be compressed.
   Decompressing it with:
        ./kcompress_70

gst-46 guest: Info
   The guest kernel binary decompression was successfully
   completed.

gst-43 guest: Info
   Changing the guest system name from:
        enterprise
   to:
        warpspeed1
/gst70/usr/src is mounted.
        Attempting to unmount...done.

/gst70/usr is mounted.
        Attempting to unmount...done.

/gst70 is mounted.
        Attempting to unmount...done.

gst-53 guest: Info
   A 8 MW Guest system (warpspeed1) has been successfully
   started for user:
        joeuser

gst-55 guest: Info
   The guest (warpspeed1) system console is:
        tty42
   on the host (enterprise) system's OWS.
```

Example 2: Dump a guest system.

```
% guest -d

Enter a dump comment (up to 80 characters):
process hung in guest
gst-23 guest: Info
   Guest dump files will be located below the following directory:
        ./dumps/04050534

Dump segment address:    030412000   cumulative words:    012110000
gst-25 guest: Info
   Successfully completed a guest dump of the system named: warpspeed1

Dump segment address:    077610700   cumulative words:    022703000
gst-25 guest: Info
   Successfully completed a guest dump of the system named: warpspeed1
```

Example 3: Stop a guest system and release its memory.

```
% guest -q
gst-15 guest: Info
   The guest options file:
        guest.rc
   was found in the following directory:
        /usr/guest/joeuser/

gst-16 guest: Info
   Changing the current working directory to:
        /usr/guest/joeuser/

gst-61 guest: Info
   The guest system (warpspeed1) has been successfully stopped.


gst-317 guest: Warning
   There is outstanding I/O on:
        ios: 0
        iop: 0
        chan: 034
        unit: 0
   Waiting for I/O completion.

sys-16 guest: Info
   Device busy
```

```
gst-37 guest: Info
   The guest memory (8 MW) has been returned to host.

/etc/mfsck: Starting pass 1

/etc/mfsck: Starting pass 2

/usr_j: file system opened
/usr_j: super block fname usr_j, fpack sn228
/src_j: file system opened
/src_j: super block fname src_j, fpack sn228
/root_j: file system opened
/root_j: super block fname root_j, fpack sn228
/usr_j: Phase 1 - Check Blocks and Sizes
/root_j: Phase 1 - Check Blocks and Sizes
/src_j: Phase 1 - Check Blocks and Sizes
/root_j: Phase 2 - Visit Directories
/usr_j: Phase 2 - Visit Directories
/root_j: Phase 3 - Checking Directories
/root_j: Phase 4 - Checking Non-Directories and Link Counts
/root_j: Phase 5 - Verify Dynamic Information - (Ignored)
/root_j: Phase 6 - Rebuilding Dynamic Information
/root_j: file system summary
/root_j:        32768 total i-nodes (29073 free i-nodes)
/root_j:        137088 total blocks  (65395 free blocks)
/root_j:  ***** FILE SYSTEM WAS MODIFIED *****
/usr_j: Phase 3 - Checking Directories
/usr_j: Phase 4 - Checking Non-Directories and Link Counts
/usr_j: Phase 5 - Verify Dynamic Information - (Ignored)
/usr_j: Phase 6 - Rebuilding Dynamic Information
/usr_j: file system summary
/usr_j:         32768 total i-nodes (25704 free i-nodes)
/usr_j:         137088 total blocks  (39475 free blocks)
/usr_j:  ***** FILE SYSTEM WAS MODIFIED *****
/src_j: Phase 2 - Visit Directories
/src_j: Phase 3 - Checking Directories
/src_j: Phase 4 - Checking Non-Directories and Link Counts
/src_j: Phase 5 - Verify Dynamic Information - (Ignored)
/src_j: Phase 6 - Rebuilding Dynamic Information
/src_j: file system summary
/src_j:         85712 total i-nodes (45087 free i-nodes)
/src_j:         365040 total blocks  (102446 free blocks)
/src_j:  ***** FILE SYSTEM WAS MODIFIED *****
/etc/mfsck: complete (9 secs.)
```

```
/gst70 is not mounted.
        Attempting to mount /dev/dsk/root_j on /gst70...done.

/gst70/usr is not mounted.
        Attempting to mount /dev/dsk/usr_j on /gst70/usr...done.

/gst70/usr/src is not mounted.
        Attempting to mount /dev/dsk/src_j on /gst70/usr/src...done.
```

Example 4: Example shell script (determines system types).

```
#
# Insure that we are running on the host
# before allowing the application to continue.
#
GUEST=`/etc/guest`
if [ "${GUEST}" = "guest" ]
then
    echo "Running as a guest; do not start"
    echo "production applications"
else
    # your code here
fi
```

FILES

`/usr/guest/Defaults`
: An optional file containing administrator-specified default values for guest systems. Updated by the installation tool.

`/guest`
: A directory containing `crash` and `unicos` binaries for active guests. Do not remove or move binaries placed here by the `guest` command.

`/guest/.diskinfo`
: A directory containing information about file systems unmounted by the `guest` command. Do not remove, move, or change files placed here by the `guest` command.

`/usr/guest/Users`
: An optional file listing individual users and their administrator-specific constraints. Updated by the installation tool.

`guest.rc`
: An optional file containing the default actions and/or the paths to required files such as `unicos` and `param`. Options include the following:

`CRASH`	Location of crash binary
`DUMP_DIRECTORY`	Directory below which system dumps are written (`./dump`)

KCOMPRESS	Location of kernel (de)compression binary (/etc/kcompress)
KERNEL	Location of kernel binary (./unicos)
LOGICAL_DEVICES	List of logical devices to unmount before startup (no default)
MEMSIZE	Requested memory size (value of MAX_GUEST_MEMORY from /usr/guest/Users)
MIN_MEMSIZE	Minimum memory size that you will accept (value of MIN_GUEST_MEMORY from /usr/guest/Default)
NODE_NAME	Node name (system name from the kernel binary)
PARAM	Location of system parameter file (./param)
PARAM_CHECKER	Location of system parameter file checker (/etc/econfig (E)) or (/etc/bconfig (D)) PARAM_CHECKER Location of system parameter file checker (/etc/econfig (E)) or (/etc/bconfig (D))
TTY_CONNECTIONS	Number of tty connections requested (1)

(For more information on your validation, see the /usr/guest/Defaults and /usr/guest/Users files.)

SEE ALSO

udbsee(1)

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

NAME

hash – Remembers or reports utility locations

SYNOPSIS

hash [*utility*...]

hash -r

IMPLEMENTATION

All Cray Research systems

STANDARDS

XPG4

DESCRIPTION

The hash utility affects the way the current shell environment remembers the locations of utilities found. Depending on the arguments specified, it adds utility locations to its list of remembered locations or it purges the contents of the list. When no arguments are specified, it reports on the contents of the list.

Utilities provided as built-ins to the shell are not reported by hash.

The hash utility accepts the following options:

-r Forgets all previously remembered utility locations.

The hash utility accepts the following operands:

utility The name of a utility to be searched for and added to the list of remembered locations. If *utility* contains one or more slashes, the results are unspecified.

The following environment variables affect the execution of hash:

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, 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 catalogs for the processing of LC_MESSAGES.

PATH Determines the location of *utility*, as described in the XBD specification.

The standard output of hash is used when no arguments are specified. Its format is unspecified, but includes the path name of each utility in the list of remembered locations for the current shell environment. This list consists of those utilities named in previous hash invocations, and may contain those invoked and found through the normal command search process.

Since hash affects the current shell execution environment, 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 hash -r
find . type f | xargs hash
```

it will not affect the command search process of the caller's environment.

The hash utility may be implemented as an alias, for example, alias -t -, in which case utilities found through normal command search will not be listed by the hash utility.

The effects of hash -r can also be achieved portably by resetting the value of PATH; in the simplest form, this can be:

```
PATH="$PATH"
```

The use of hash with *utility* names is unnecessary for most applications, but may provide a performance improvement of a few implementations; normally, the hashing process is included by default.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

sh(1)

NAME

head – Displays the first few lines of a file

SYNOPSIS

head [-n *number*] [*files*]

Obsolescent version; may not be supported in future releases:

head [-*number*] [*files*]

IMPLEMENTATION

All Cray Research systems

STANDARDS

POSIX, XPG4

DESCRIPTION

The head utility copies its input files to the standard output, ending the output for each file at a designated point. Copying ends at the point in each input file indicated by the -n *number* option (or the -*number* argument). The argument *number* is counted in units of lines.

The head utility accepts the following options and operands:

-*number*
-n *number* The first *number* lines of each input file are copied to the standard output.

files A path name of a file be displayed. If no *file* operands are specified, the standard input is used.

If no options are specified, -n defaults to 10.

If no *files* are specified, head copies lines from the standard input.

When more than one file is specified, the start of each file looks like the following:

==>*filename*<==

EXIT STATUS

The head utility exits with one of the following values:

0 Successful completion.

>0 An error occurred.

SEE ALSO

cat(1), more(1), pg(1), tail(1)

NAME

help – Provides explanation of SCCS messages and commands

SYNOPSIS

help [*args*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The help command is part of the Source Code Control System (SCCS) command set. It finds information to explain a message from a command or explain the use of a command. Zero or more arguments may be supplied.

The help command accepts the following operand:

args The arguments may be either message numbers (which usually appear in parentheses following messages) or command names, of one of the following types:

- Type 1 Begins with nonnumerics, ends in numerics. The nonnumeric prefix is usually an abbreviation for the program or set of routines that produced the message (for example, ge4, for message 4 from the get(1) command).
- Type 2 Does not contain numerics (as a command, such as get(1)).
- Type 3 Is all numeric (for example, 26).

The response of the program will be the explanatory information related to the argument, if there is any.

If help cannot find anything relevant, but the current argument matches the extended regular expression [A-Za-z]+-[0-9]+, it uses the exec command to execute /usr/bin/explain, assuming that the argument is a Cray Research message identifier. Otherwise, it executes man(1) with the argument.

help works across the arguments from left to right, until it finds one that does not appear to be associated with SCCS. At that time, it executes either explain or man, which effectively makes the current argument the last argument processed.

When all else fails, try help stuck.

EXAMPLES

Additional information is obtained from help for messages from SCCS commands as depicted in the following example:

```
$ get -e s.example.c
ERROR [s.example.c]: writable `example.c' exists (ge4)
$ help ge4

ge4:
"writable `...' exists"
For safety's sake, SCCS won't overwrite an existing g-file if it's writable.
If you don't need the g-file, remove it and rerun the get command.
$
```

FILES

`/usr/lib/help`	Directory containing files of message text

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `man`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `vc`(1), `what`(1)

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

NAME

host – Looks up host names by using domain server

SYNOPSIS

/usr/ucb/host [-a] [-c *class*] [-d] [-l] [-r] [-t *querytype*] [-v] [-w] *host* [*server*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The host utility looks for information about Internet hosts. It gets this information from a set of interconnected servers that are spread across the country. By default, it maps between host names and Internet addresses. However, with the -t or -a option, it can find all of the information about the specified host that is maintained by the domain server.

The host utility accepts the following options:

-a Operates equivalent to -v -t any (for all).

-c *class* Allows you to specify an address class for the request. Currently, supported types are in, hs, any, and specific numeric values. The default class is in, which represents the Internet address class.

-d Turns on debugging. Network transactions are shown in detail.

-l Produces a listing of a complete domain.

-r Specifies that recursion in the request will be turned off. This means that the name server returns only data it has in its own database; it does not ask other servers for more information.

-t *querytype*
Allows you to specify a particular type of information to be looked up. The arguments are defined in the man page for named(8). Currently supported types are a, ns, md, mf, cname, soa, mb, mg, mr, null, wks, ptr, hinfo, minfo, mx, uinfo, uid, gid, unspec, and the wildcard, which may be written as either any or *. You must specify types in lowercase. The default is to look first for a, and then mx, except that if the verbose option is turned on, the default is only a.

-v Specifies that the printout will be in a verbose format. This is the official domain master file format, which is documented in the man page for named(8). Without this option, output still follows this format in general terms, but an attempt is made to make it more intelligible to typical users. Without -v, a, mx, and cname, records are written as has address, mail is handled by, and is a nickname for, respectively, and time-to-live (TTL) and class fields are not shown.

-w Directs the host to wait for a response. It usually times out after approximately 1 minute.

host Specifies the name or number of the host to be looked up. The program first tries to interpret *host* as a host number. If this fails, it treats it as a host name.

server Specifies a particular server to query. If you do not specify this argument, the default server (usually the local machine) is used.

Specifying Hosts

If you specify *host* as a number, an *inverse query* is done; that is, the domain system looks in a separate set of databases that are used to convert numbers to names. A host number consists of four decimal numbers separated by dots (for example, `128.6.4.194`).

A host name consists of names separated by dots (for example, `topaz.rutgers.edu`). Unless the name ends in a dot, the local domain is appended automatically. Thus, you can specify `host topaz`, and host looks up the address of the machine named `topaz`. If this fails, the name (in this case, `topaz`) is tried unchanged. This same convention is used for mail and other network utilities. They obtain the actual suffix to append by looking at the results of a `hostname` call, and using everything, starting at the first dot. Following is a description of how to customize the host name lookup.

If you specify a name rather than a host number, you can see three different types of output. The following is an example that shows all of them:

```
$ host sun4
sun4.rutgers.edu is a nickname for ATHOS.RUTGERS.EDU
ATHOS.RUTGERS.EDU has address 128.6.5.46
ATHOS.RUTGERS.EDU has address 128.6.4.4
ATHOS.RUTGERS.EDU mail is handled by ARAMIS.RUTGERS.EDU
```

In this example, the user typed the command `host sun4`. The first line indicates that the name `sun4.rutgers.edu` is actually a nickname. The official host name is `ATHOS.RUTGERS.EDU`. The next two lines show the address. (If a system has more than one network interface, there is a separate address for each.)

The last line indicates that `ATHOS.RUTGERS.EDU` does not receive its own mail; mail for it is taken by `ARAMIS.RUTGERS.EDU`. More than one of these lines can exist because some systems have more than one other system that handles mail for them. Technically, each system that can receive mail must have an entry of this type. If the system receives its own mail, there must be an entry that mentions the system itself (for example, "*XXX* `mail is handled by` *XXX*"). However, many systems that receive their own mail do not mention that fact. If a system has a `mail is handled by` entry, but no address, it is not actually part of the Internet, but a system that is on the network will forward mail to it. Systems on Usenet, Bitnet, and several other networks have this type of entry.

BUGS

Unexpected results can be obtained when you type a name; for example, a specification for a host that is not part of the local domain.

The `-l` option tries only the first name server that is listed for the domain that you have requested. If this server is not operational, you may have to specify a server manually. For example, to get a listing of `foo.edu`, you can try `host -t ns foo.edu` to get a list of all the name servers for `foo.edu`, and then try `host -l foo.edu` *xxx* for all *xxx* on the list of name servers until you find one that works.

EXAMPLES

Example 1: The following example gives a listing of all hosts in the `rutgers.edu` domain. Without the `-t` option, `host -l` fetches only address information, which also includes PTR and NS records.

```
host -l rutgers.edu
```

Example 2: The following example gives a complete listing of the zone data for `rutgers.edu`, in the official master file format. (However, the SOA record is listed twice, for arcane reasons.) `-l` is implemented by doing a complete zone transfer and then filtering the information for which you have asked through the `-t` option. The `-t any` option in the following example specifies that all information in the domain is listed; this command must be used only if it is absolutely necessary.

```
host -l -v -t any rutgers.edu
```

SEE ALSO

`nslookup`(1)

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

NAME

`hostid` – Sets or prints identifier of current host system

SYNOPSIS

`hostid` [*identifier*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The `hostid` utility sets or prints the *identifier* of the current host in hexadecimal.

By default, the value of *identifier* is 0. This numeric value is expected to be unique across all hosts and is usually set to the host's Internet address. An appropriately authorized user can set the *identifier* by giving a hexadecimal argument or an address in Internet dot format.

NOTES

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

Active Category	Action
`system`, `secadm`	Allowed to set the current host ID.
`sysadm`	Allowed to set the current host ID. Shell-redirected I/O is subject to security label restrictions.

If the `PRIV_SU` configuration option is enabled, the super user is allowed to set the current host ID.

SEE ALSO

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

NAME

hostname – Prints the name of current host system

SYNOPSIS

hostname [*nameofhost*]

IMPLEMENTATION

All Cray Research systems

DESCRIPTION

The hostname utility prints the name of the current host system. An appropriately authorized user can set the host name by giving a *nameofhost* argument to hostname.

NOTES

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

Active Category	Action
system, secadm	Allowed to set the current host name.
sysadm	Allowed to set the current host name. Shell-redirected I/O is subject to security label restrictions.

If the PRIV_SU configuration option is enabled, the super user is allowed to set the current host name.

SEE ALSO

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

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