

man pages comparison

echo

===== suckless =====

NAME

echo - print arguments

SYNOPSIS

echo [-n] [string ...]

DESCRIPTION

echo writes each string to stdout, separated by spaces and terminated by a newline.

OPTIONS

-n Do not print the terminating newline.

STANDARDS

POSIX.1-2013.

The [-n] flag is an extension to that specification.

sbase October 8, 2015

sbase

===== 9front =====

NAME

echo - print arguments

SYNOPSIS

echo [-n] [arg ...]

DESCRIPTION

Echo writes its arguments separated by blanks and terminated by a newline on the standard output. Option -n suppresses the newline.

SOURCE

/sys/src/cmd/echo.c

DIAGNOSTICS

If echo draws an error while writing to standard output, the exit status is Otherwise the exit status is empty.

===== BSD =====

NAME

echo - write arguments to the standard output

SYNOPSIS

echo [-n] [string ...]

DESCRIPTION

The echo utility writes any specified operands, separated by single blank (' ') characters and followed by a newline ('\n') character, to the standard output.

When no operands are given, only the newline is written. The -- operand, which generally denotes an end to option processing, is treated as part of string.

The options are as follows:

-n Do not print the trailing newline character.

EXIT STATUS

The echo utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

csh(1), ksh(1), printf(1)

STANDARDS

The echo utility is compliant with the IEEE Std 1003.1-2008 ("POSIX.1") specification.

The flag [-n] conflicts with the behaviour mandated by the X/Open System Interfaces option of the IEEE Std 1003.1-2008 ("POSIX.1") specification, which says it should be treated as part of string.

Additionally, echo does not support any of the backslash character sequences mandated by XSI.

echo also exists as a built-in to csh(1) and ksh(1), though with a different syntax.

Where portability is paramount, use printf(1).

HISTORY

An echo utility appeared in Version 2 AT&T UNIX.

BSD March 16, 2018

===== GNU =====

NAME

echo - display a line of text

SYNOPSIS

echo [SHORT-OPTION]... [STRING]...

echo LONG-OPTION

DESCRIPTION

Echo the STRING(s) to standard output.

-n do not output the trailing newline

-e enable interpretation of backslash escapes

-E disable interpretation of backslash escapes (default)

--help display this help and exit

--version output version information and exit

If -e is in effect, the following sequences are recognized:

\\ backslash

\a alert (BEL)

\b backspace

\c produce no further output

\e escape

\f form feed

\n new line

\r carriage return

\t horizontal tab

\v vertical tab

\ONNN byte with octal value NNN (1 to 3 digits)

\xHH byte with hexadecimal value HH (1 to 2 digits)

NOTE: your shell may have its own version of echo, which usually supersedes the version described here. Please refer to your shell's documentation for details about the options it supports.

NOTE: printf(1) is a preferred alternative, which does not have issues outputting option-like strings.

AUTHOR

Written by Brian Fox and Chet Ramey.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2022 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

printf(1)

Full documentation <<https://www.gnu.org/software/coreutils/echo>> or available locally via: info '(coreutils) echo invocation'

GNU coreutils 9.1 September 2022

===== POSIX =====

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

echo - write arguments to standard output

SYNOPSIS

echo [string...]

DESCRIPTION

The echo utility writes its arguments to standard output, followed by a <newline>. If there are no arguments, only the <newline> is written.

OPTIONS

The echo utility shall not recognize the "--" argument in the manner specified by Guideline 10 of the Base Definitions volume of POSIX.1-

2017, Section 12.2, Utility Syntax Guidelines; "--" shall be recognized as a string operand.

Implementations shall not support any options.

OPERANDS

The following operands shall be supported:

string A string to be written to standard output. If the first operand is -n, or if any of the operands contain a <backslash> character, the results are implementation-defined.

On XSI-conformant systems, if the first operand is -n, it shall be treated as a string, not an option. The following character sequences shall be recognized on XSI-conformant systems within any of the arguments:

\a Write an <alert>.

\b Write a <backspace>.

\c Suppress the <newline> that otherwise follows the final argument in the output. All characters following the '\c' in the arguments shall be ignored.

\f Write a <form-feed>.

\n Write a <newline>.

\r Write a <carriage-return>.

\t Write a <tab>.

\v Write a <vertical-tab>.

\\ Write a <backslash> character.

\Onum Write an 8-bit value that is the zero, one, two, or three-digit octal number num.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of echo:

LANG Provide a default value for the internationalization variables that are unset or null. (See the Base Definitions volume of POSIX.1-2017, Section 8.2, Internationalization Variables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

LC_MESSAGES Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

The echo utility arguments shall be separated by single <space> characters and a <newline> character shall follow the last argument. Output transformations shall occur based on the escape sequences in the input. See the OPERANDS section.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

It is not possible to use echo portably across all POSIX systems unless both -n (as the first argument) and escape sequences are omitted. The printf utility can be used portably to emulate any of the traditional behaviors of the echo utility as follows (assuming that IFS has its standard value or is unset):

* The historic System V echo and the requirements on XSI implementations in this volume of POSIX.1-2017 are equivalent to:

printf "%b\n\$*"

* The BSD echo is equivalent to:

if ["\$1" = "X-n"]

then

shift

printf "%s\$*"

else

printf "%s\n\$*"

fi

New applications are encouraged to use printf instead of echo.

EXAMPLES

None.

RATIONALE

The echo utility has not been made obsolescent because of its extremely widespread use in historical applications. Conforming applications that wish to do prompting without <newline> characters or that could possibly be expecting to echo a -n, should use the printf utility derived from the Ninth Edition system.

As specified, echo writes its arguments in the simplest of ways. The two different historical versions of echo vary in fatally incompatible ways.

The BSD echo checks the first argument for the string -n which causes it to suppress the <newline> that would otherwise follow the final argument in the output.

The System V echo does not support any options, but allows escape sequences within its operands, as described for XSI implementations in the OPERANDS section.

The echo utility does not support Utility Syntax Guideline 10 because historical applications depend on echo to echo all of its arguments, except for the -n option in the BSD version.

FUTURE DIRECTIONS

None.

SEE ALSO

printf

The Base Definitions volume of POSIX.1-2017, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.opengroup.org/unix/online.html>.

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.kernel.org/doc/man-pages/reporting_bugs.html.

IEEE/The Open Group 2017

ECHO(1POSIX)

cat

===== suckless =====

NAME

```

cat - concatenate files
SYNOPSIS
cat [-u] [file ...]
DESCRIPTION
cat reads each file in sequence and writes it to stdout. If no file is
given cat reads from stdin.
OPTIONS
-u Unbuffered output.
STANDARDS
POSIX.1-2013.
sbase October 8, 2015
sbase
===== 9front =====
NAME
cat, read - catenate files
SYNOPSIS
cat [ file ... ]
read [-m | -n nlines | -c nbytes | -r nrunes ] [ file ... ]
DESCRIPTION
Cat reads each file in sequence and writes it on the standard output.
Thus
cat file
prints a file and
cat file1 file2 >file3
concatenates the first two files and places the result on the third.
If no file is given, cat reads from the standard input. Output is
buffered in blocks matching the input.
Read copies to standard output exactly one line from the named file,
default standard input. It is useful in interactive rc(1) scripts.
The -m flag causes it to continue reading and writing multiple lines
until end of file; -n causes it to read no more than nlines lines. The
c and -r flags specify a number of bytes or runes to read instead of
lines.
When reading lines, read always executes a single write for each line
of input, which can be helpful when preparing input to programs that
expect line-at-a-time data. It never reads any more data from the input
than it prints to the output.
SOURCE
/sys/src/cmd/cat.c
/sys/src/cmd/read.c
SEE ALSO
cp(1)
DIAGNOSTICS
Read exits with status eof on end of file or, in the -n case, if it
doesn't read nlines lines.
BUGS
Beware of and which destroy input files before reading them.
===== BSD =====
NAME
cat - concatenate and print files
SYNOPSIS
cat [-benstv] [file ...]
DESCRIPTION
The cat utility reads files sequentially, writing them to the standard
output. The file operands are processed in command-line order. If file
is a single dash ("-") or absent, cat reads from the standard input.
The options are as follows:
-b Number the lines, but don't count blank lines.
-e Print a dollar sign ('$') at the end of each line. Implies the -v
option to display non-printing characters.
-n Number the output lines, starting at 1.
-s Squeeze multiple adjacent empty lines, causing the output to be
single spaced.
-t Print tab characters as 'I'. Implies the -v option to display
non-printing characters.
-u The output is guaranteed to be unbuffered (see setvbuf(3)).
-v Displays non-printing characters so they are visible. Control
characters print as '^X' for control-X, with the exception of the tab
and EOL characters, which are displayed normally. The DEL character
(octal 0177)
prints as '^?'. Non-ASCII characters (with the high bit set) are
printed as '^M' (for meta) followed by the character for the low 7 bits.
EXIT STATUS
The cat utility exits 0 on success, and >0 if an error occurs.
EXAMPLES
Print the contents of file1 to the standard output:
$ cat file1
Sequentially print the contents of file1 and file2 to the file file3,
truncating file3 if it already exists. See the manual page for your
shell (e.g., sh(1)) for more information on redirection.
$ cat file1 file2 > file3
Print the contents of file1, print data it receives from the standard
input until it receives an EOF ('^D') character, print the contents of
file2, read and output contents of the standard input again, then
finally output the contents of file3. Note that if the standard input
referred to a file, the second dash on the command line would have no
effect, since the entire contents of the file would have already been
read and printed by cat when it encountered the first '-' operand.
$ cat file1 - file2 - file3
SEE ALSO
head(1), less(1), more(1), pr(1), sh(1), tail(1), vis(1), setvbuf(3)
Rob Pike, "UNIX Style, or cat -v Considered Harmful", USENIX Summer
Conference Proceedings, 1983.
STANDARDS
The cat utility is compliant with the specification.
The flags [-benstv] are extensions to that specification.
HISTORY
A cat utility appeared in Version 1 AT&T UNIX.
CAVEATS
Because of the shell language mechanism used to perform output
redirection, the following command will cause the original data in file1
to be destroyed:
$ cat file1 file2 > file1
To append file2 to file1, instead use:
$ cat file2 >> file1
BSD August 1, 2024
BSD
===== GNU =====
NAME
cat - concatenate files and print on the standard output
SYNOPSIS
cat [OPTION]... [FILE]...
DESCRIPTION
Concatenate FILE(s) to standard output.
With no FILE, or when FILE is -, read standard input.
-A, --show-all
equivalent to -vET
-b, --number-nonblank
number nonempty output lines, overrides -n
-e, --equivalent to -vE
-E, --show-ends
display $ at end of each line
-n, --number
number all output lines
-s, --squeeze-blank
suppress repeated empty output lines
-t, --equivalent to -vT
-T, --show-tabs
display TAB characters as ^I
-u, --ignored
-v, --show-nonprinting
use ^ and M- notation, except for LFD and TAB
--help display this help and exit
--version
output version information and exit
EXAMPLES
cat f - g
Output f's contents, then standard input, then g's contents.
cat Copy standard input to standard output.
AUTHOR
Written by Torbjorn Granlund and Richard M. Stallman.
REPORTING BUGS
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report any translation bugs to <https://translationproject.org/team/>
COPYRIGHT
Copyright © 2022 Free Software Foundation, Inc. License GPLv3+: GNU
GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
SEE ALSO
tac(1)
Full documentation <https://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'
GNU coreutils 9.1 September
2022
===== POSIX =====
CAT(1)
PROLOG
This manual page is part of the POSIX Programmer's Manual. The Linux
implementation of this interface may differ (consult the corresponding
Linux manual page for details of Linux behavior), or the interface may
not be implemented on Linux.
NAME
cat - concatenate and print files
SYNOPSIS
cat [-u] [file...]
DESCRIPTION
The cat utility shall read files in sequence and shall write their
contents to the standard output in the same sequence.
OPTIONS
The cat utility shall conform to the Base Definitions volume of
POSIX.1-2017, Section 12.2, Utility Syntax Guidelines.
The following option shall be supported:
-u Write bytes from the input file to the standard output without
delay as each is read.
OPERANDS
The following operand shall be supported:
file A pathname of an input file. If no file operands are specified,
the standard input shall be used. If a file is '-', the cat utility
shall read from the standard input at that point in the sequence. The
cat utility shall not close and reopen standard input when it is
referenced in this way, but shall accept multiple occurrences of '-' as
a file operand.
STDIN
The standard input shall be used only if no file operands are
specified, or if a file operand is '-'. See the INPUT FILES section.
INPUT FILES
The input files can be any file type.
ENVIRONMENT VARIABLES
The following environment variables shall affect the execution of cat:
LANG Provide a default value for the internationalization variables
that are unset or null. (See the Base Definitions volume of POSIX.1-
2017, Section 8.2, Internationalization Variables for the precedence of
internationalization variables used to determine the values of locale
categories.)
LC_ALL If set to a non-empty string value, override the values of all
the other internationalization variables.
LC_CTYPE Determine the locale for the interpretation of sequences of
bytes of text data as characters (for example, single-byte as opposed to
multi-byte characters in arguments).
LC_MESSAGES
Determine the locale that should be used to affect the format and
contents of diagnostic messages written to standard error.
NLSPATH Determine the location of message catalogs for the processing
of LC_MESSAGES.
ASYNCHRONOUS EVENTS
Default.
STDOUT
The standard output shall contain the sequence of bytes read from the
input files. Nothing else shall be written to the standard output. If
the standard output is a regular file, and is the same file as any of
the input file operands, the implementation may treat this as an error.
STDERR
The standard error shall be used only for diagnostic messages.
OUTPUT FILES
None.
EXTENDED DESCRIPTION
None.
EXIT STATUS
The following exit values shall be returned:
0 All input files were output successfully.
>0 An error occurred.
CONSEQUENCES OF ERRORS
Default.
The following sections are informative.
APPLICATION USAGE
The -u option has value in prototyping non-blocking reads from FIFOs.
The intent is to support the following sequence:
mkfifo foo
cat -u foo > /dev/tty13 &
cat -u > foo
It is unspecified whether standard output is or is not buffered in the
default case. This is sometimes of interest when standard output is
associated with a terminal, since buffering may delay the output. The
presence of the -u option guarantees that unbuffered I/O is available.
It is implementation-defined whether the cat utility buffers output if
the -u option is not specified. Traditionally, the -u option is
implemented using the equivalent of the setvbuf() function defined in
the System Interfaces volume of POSIX.1-2017.
EXAMPLES
The following command:
cat myfile
writes the contents of the file myfile to standard output.
The following command:
cat doc1 doc2 > doc.all
concatenates the files doc1 and doc2 and writes the result to doc.all.
Because of the shell language mechanism used to perform output
redirection, a command such as this:
cat doc doc.end > doc
causes the original data in doc to be lost before cat even begins
execution. This is true whether the cat command fails with an error or
silently succeeds (the specification allows both behaviors). In order to
append the contents of doc.end without losing the original contents of
doc, this command should be used instead:
cat doc.doc >> doc
The command:
cat start - middle - end > file
when standard input is a terminal, gets two arbitrary pieces of input
from the terminal with a single invocation of cat. Note, however, that
if standard input is a regular file, this would be equivalent to the
command:
cat start - middle /dev/null end > file
because the entire contents of the file would be consumed by cat the
first time '-' was used as a file operand and an end-of-file condition
would be detected immediately when '-' was referenced the second time.

```

RATIONALE

Historical versions of the cat utility include the -e, -t, and -v, options which permit the ends of lines, <tab> characters, and invisible characters, respectively, to be rendered visible in the output. The standard developers omitted these options because they provide too fine a degree of control over what is made visible, and similar output can be obtained using a command such as:

```
sed -n 1 pathname
```

The latter also has the advantage that its output is unambiguous, whereas the output of historical cat -etv is not.

The -s option was omitted because it corresponds to different functions in BSD and System V-based systems. The BSD -s option to squeeze blank lines can be accomplished by the shell script shown in the following example:

```
sed -n '
# Write non-empty lines.
./.' {
```

```
    p
}
```

```
# Write a single empty line, then look for more empty lines.
./.$ p
```

```
# Get next line, discard the held <newline> (empty line),
# and look for more empty lines.
:Empty
./.$ {
```

```
    N
    s././
    b Empty
}
```

```
# Write the non-empty line before going back to search
# for the first in a set of empty lines.
    p
```

The System V -s option to silence error messages can be accomplished by redirecting the standard error. Note that the BSD documentation for cat uses the term ``blank line'' to mean the same as the POSIX ``empty line'': a

line consisting only of a <newline>.

The BSD -n option was omitted because similar functionality can be obtained from the -n option of the pr utility.

FUTURE DIRECTIONS

None.

SEE ALSO

more

The Base Definitions volume of POSIX.1-2017, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines

The System Interfaces volume of POSIX.1-2017, setvbuf()

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.opengroup.org/unix/online.html>.

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.kernel.org/doc/man-pages/reporting_bugs.html.

IEEE/The Open Group

2017

CAT(1POSIX)

grep

===== suckless =====

NAME

grep - search files for patterns

SYNOPSIS

grep [-EFHchlnqsvx] [-e pattern] [-f file] [pattern] [file ...]

DESCRIPTION

grep searches the input files for lines that match the pattern, a regular expression as defined in regex(7) or re_format(7). By default each matching line is printed to stdout. If no file is given grep reads from stdin.

OPTIONS

-E Match using extended regex.
-F Match using fixed strings. Treat each pattern specified as a string instead of a regular expression.
-H Prefix each matching line with its filename in the output. This is the default when there is more than one file specified.
-c Print only a count of matching lines.
-e pattern

Specify a pattern used during the search of the input: an input line is selected if it matches any of the specified patterns. This option is most useful when multiple -e options are used to specify multiple patterns, or when a pattern begins with a dash.

-f file

Read one or more patterns from the file named by the pathname file. Patterns in file shall be terminated 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 shall be treated as a BRE. ('-').

-h Do not prefix each line with 'filename:' prefix.
-i Match lines case insensitively.
-l Print only the names of files with matching lines.
-n Prefix each matching line with its line number in the input.
-q Print nothing, only return status.
-s Suppress the error messages ordinarily written for nonexistent or unreadable files.
-v Select lines which do not match the pattern.
-w The expression is searched for as a word (as if surrounded by '<' and '>').
-x Consider only input lines that use all characters in the line excluding the terminating <newline> to match an entire fixed string or regular expression to be matching lines.

EXIT STATUS

0 One or more lines were matched.
1 No lines were matched.
> 1 An error occurred.

SEE ALSO

sed(1), re_format(7), regex(7)

STANDARDS

POSIX.1-2013.

The [-Hhw] flags are an extension to that specification.

Sbase

===== 9front =====

NAME

grep, g - search a file for a pattern

SYNOPSIS

grep [-bchillnsv] [-e] pattern | -f patternfile [file ...]
g [flags] pattern [file ...]

DESCRIPTION

Grep searches the input files (standard input default) for lines that match the pattern, a regular expression as defined in regex(6) with the addition of a newline character as an alternative (substitute for |) with lowest precedence. Normally, each line matching the pattern is 'selected', and each selected line is copied to the standard output.

The options are:

-c Print only a count of matching lines.
-h Do not print file name tags (headers) with output lines.

-e The following argument is taken as a pattern. This option makes it easy to specify patterns that might confuse argument parsing, such as -n.

-i Ignore alphabetic case distinctions. The implementation folds into lower case all letters in the pattern and input before interpretation. Matched lines are printed in their original form.

-l (ell) Print the names of files with selected lines; don't print the lines.

-L Print the names of files with no selected lines; the converse of -l.

-n Mark each printed line with its line number counted in its file.

-s Produce no output, but return status.

-v Reverse: print lines that do not match the pattern.

-f The pattern argument is the name of a file containing regular expressions one per line.

-b Don't buffer the output: write each output line as soon as it is discovered.

Output lines are tagged by file name when there is more than one input file. (To force this tagging, include /dev/null as a file name argument.)

Care should be taken when using the shell metacharacters \${} and \ and newline in pattern; it is safest to enclose the entire expression in single quotes '...'. An expression starting with '*' will treat the rest of the expression as literal characters.

G invokes grep with -n (plus additional flags, if provided) and forces tagging of output lines by file name. If no files are listed, it recursively searches the current directory for all files matching *.b *.c *.C *.h *.l *.m *.s *.y *.asm *.awk *.cc *.cgi *.cpp *.cs *.cxx *.go *.goc *.hpp *.hs *.hxx *.in *.java *.lua *.lx *.mk *.ml *.mli *.ms *.myr *.pl *.py *.rc *.sh *.tex *.txt *.xy

The recursive search can be suppressed by passing g the -n flag.

SOURCE

/sys/src/cmd/grep

/rc/bin/g

SEE ALSO

ed(1), awk(1), sed(1), sam(1), regexp(6)

DIAGNOSTICS

Exit status is null if any lines are selected, or non-null when no

lines are selected or an error occurs.

===== BSD =====

NAME

grep, egrep, fgrep, rgrep - file pattern searcher

SYNOPSIS

grep [-abcdDEFGHhIiLlmmOopqRSsUVvwxz] [-A num] [-B num] [-C[num]] [-e pattern] [-f file] [--binary-files=value] [--color=when] [--colour=when] [--context=num] [--label] [--line-buffered] [--null] [pattern] [file ...]

DESCRIPTION

The grep utility searches any given input files, selecting lines that match one or more patterns. By default, a pattern matches an input line if the regular expression (RE) in the pattern matches the input line without its trailing newline. An empty expression matches every line. Each input line that matches at least one of the patterns is written to the standard output.

grep is used for simple patterns and basic regular expressions (BREs); egrep can handle extended regular expressions (EREs). See re_format(7) for more information on regular expressions. fgrep is quicker than both grep and egrep, but can only handle fixed patterns (i.e., it does not interpret regular expressions). Patterns may consist of one or more lines, allowing any of the pattern lines to match a portion of the input.

The following options are available:

-A num, --after-context=num

Print num lines of trailing context after each match. See also the -B and -C options.

-a, --text

Treat all files as ASCII text. Normally grep will simply print "Binary file ... matches" if files contain binary characters. Use of this option forces grep to output lines matching the specified pattern.

-B num, --before-context=num

Print num lines of leading context before each match. See also the -A and -C options.

-b, --byte-offset

The offset in bytes of a matched pattern is displayed in front of the respective matched line.

-C[num], --context=[num]

Print num lines of leading and trailing context surrounding each match. The default value of num is "2" and is equivalent to "-A 2 -B 2". Note: no whitespace may be given between the option and its argument.

-c, --count

Only a count of selected lines is written to standard output.

--colour=[when], --color=[when]

Mark up the matching text with the expression stored in the GREP_COLOR environment variable. The possible values of when are "never", "always" and "auto".

-D action, --devices=action

Specify the demanded action for devices, FIFOs and sockets. The default action is "read", which means, that they are read as if they were normal files. If the action is set to "skip", devices are silently skipped.

-d action, --directories=action

Specify the demanded action for directories. It is "read" by default, which means that the directories are read in the same manner as normal files. Other possible values are "skip" to silently ignore the directories, and "recurse" to read them recursively, which has the same effect as the -R and -r option.

-E, --extended-regex

Interpret pattern as an extended regular expression (i.e., force grep to behave as egrep).

-e pattern, --regex=pattern

Specify a pattern used during the search of the input: an input line is selected if it matches any of the specified patterns. This option is most useful when multiple -e options are used to specify multiple patterns, or when a pattern begins with a dash ('-').

--exclude=pattern

If specified, it excludes files matching the given filename pattern from the search. Note that --exclude and --include patterns are processed in the order given. If a name matches multiple patterns, the latest matching rule wins. If no --include pattern is specified, all files are searched that are not excluded. Patterns are matched to the full path specified, not only to the filename component.

--exclude-dir=pattern

If -R is specified, it excludes directories matching the given filename pattern from the search. Note that --exclude-dir and --include-dir patterns are processed in the order given. If a name matches multiple patterns, the latest matching rule wins. If no --include-dir pattern is specified, all directories are searched that are not excluded.

-F, --fixed-strings

Interpret pattern as a set of fixed strings (i.e., force grep to behave as fgrep).

-f file, --file=

Read one or more newline separated patterns from file. Empty pattern lines match every input line. Newlines are not considered part of a pattern. If file is empty, nothing is matched.

-G, --basic-regex

Interpret pattern as a basic regular expression (i.e., force grep to behave as traditional grep).

-H Always print filename headers with output lines.

-h, --no-filename

Never print filename headers (i.e., filenames) with output lines.

--help Print a brief help message.

```

-I Ignore binary files. This option is equivalent to the "--binary-file=without-match" option.
-i, --ignore-case
    Perform case insensitive matching. By default, grep is case sensitive.
--include pattern
    If specified, only files matching the given filename pattern are searched. Note that --include and --exclude patterns are processed in the order given. If a name matches multiple patterns, the latest matching rule wins. Patterns are matched to the full path specified, not only to the filename component.
--include-dir pattern
    If -R is specified, only directories matching the given filename pattern are searched. Note that --include-dir and --exclude-dir patterns are processed in the order given. If a name matches multiple patterns, the latest matching rule wins.
-L, --files-without-match
    Only the names of files not containing selected lines are written to standard output. Pathnames are listed once per file searched. If the standard input is searched, the string "(standard input)" is written unless a --label is specified.
-l, --files-with-matches
    Only the names of files containing selected lines are written to standard output. grep will only search a file until a match has been found, making it potentially less expensive. Pathnames are listed once per file searched. If the standard input is searched, the string "(standard input)" is written unless a --label is specified.
--label
    Label to use in place of "(standard input)" for a file name where a file name would normally be printed. This option applies to -H, -L, and -l.
--mmap
    Use mmap(2) instead of read(2) to read input, which can result in better performance under some circumstances but can cause undefined behaviour.
-m num, --max-count=num
    Stop reading the file after num matches.
-n, --line-number
    Each output line is preceded by its relative line number in the file, starting at line 1. The line number counter is reset for each file processed. This option is ignored if -c, -L, -l, or -q is specified.
--null
    Prints a zero-byte after the file name.
-O
    If -R is specified, follow symbolic links only if they were explicitly listed on the command line. The default is not to follow symbolic links.
-o, --only-matching
    Prints only the matching part of the lines.
-p
    If -R is specified, no symbolic links are followed. This is the default.
-q, --quiet, --silent
    Quiet mode: suppress normal output. grep will only search a file until a match has been found, making searches potentially less expensive.
-R, -r, --recursive
    Recursively search subdirectories listed. (i.e., force grep to behave as rgrep).
-S
    If -R is specified, all symbolic links are followed. The default is not to follow symbolic links.
-s, --no-messages
    Silent mode. Nonexistent and unreadable files are ignored (i.e., their error messages are suppressed).
-U, --binary
    Search binary files, but do not attempt to print them.
-u
    This option has no effect and is provided only for compatibility with GNU grep.
-V, --version
    Display version information and exit.
-v, --invert-match
    Selected lines are those not matching any of the specified patterns.
-w, --word-regexp
    The expression is searched for as a word (as if surrounded by '[:<:~>:]' and '[:>:~>:]'; see re_format(7)).
-x, --line-regexp
    Only input lines selected against an entire fixed string or regular expression are considered to be matching lines.
-y
    Equivalent to -i. Obsolete.
-z, --null-data
    Treat input and output data as sequences of lines terminated by a zero-byte instead of a newline.
--binary-files=value
    Controls searching and printing of binary files. Options are:
    binary (default) Search binary files but do not print them.
    without-match Do not search binary files.
    text Treat all files as text.
--line-buffered
    Force output to be line buffered. By default, output is line buffered when standard output is a terminal and block buffered otherwise.
    If no file arguments are specified, the standard input is used.
    Additionally, "-" may be used in place of a file name, anywhere that a file name is accepted, to read from standard input. This includes both -f and file arguments.
EXIT STATUS
    The grep utility exits with one of the following values:
    0 One or more lines were selected.
    1 No lines were selected.
    >1 An error occurred.
EXAMPLES
- Find all occurrences of the pattern 'patricia' in a file:
  $ grep 'patricia' myfile
- Same as above but looking only for complete words:
  $ grep -w 'patricia' myfile
- Count occurrences of the exact pattern 'FOO':
  $ grep -c FOO myfile
- Same as above but ignoring case:
  $ grep -c -i FOO myfile
- Find all occurrences of the pattern '.Pp' at the beginning of a line:
  $ grep '^\.Pp' myfile
    The apostrophes ensure the entire expression is evaluated by grep instead of by the user's shell. The caret '^' matches the null string at the beginning of a line, and the '\' escapes the '.', which would otherwise match any character.
- Find all lines in a file which do not contain the words 'foo' or 'bar':
  $ grep -v -e 'foo' -e 'bar' myfile
- Peruse the file 'calendar' looking for either 19, 20, or 25 using extended regular expressions:
  $ egrep '[19|20|25]' calendar
- Show matching lines and the name of the *.h files which contain the pattern 'FIXME'. Do the search recursively from the /usr/src/sys/arm directory:
  $ grep -H -R FIXME --include=*.h /usr/src/sys/arm/
- Same as above but show only the name of the matching file:
  $ grep -l -R FIXME --include=*.h /usr/src/sys/arm/
- Show lines containing the text 'foo'. The matching part of the output is colored and every line is prefixed with the line number and the offset in the file for those lines that matched.
  $ grep -b --colour -n foo myfile
- Show lines that match the extended regular expression patterns read from the standard input:
  $ echo 'FreeBSD\Nall.*reserved' | grep -E -f - myfile
- Show lines from the output of the pciconf(8) command matching the specified extended regular expression along with three lines of leading context and one line of trailing context:
  $ pciconf -lv | grep -B3 -A1 -E 'class.*storage'
- Suppress any output and use the exit status to show an appropriate message:
  $ grep -q foo myfile && echo File matches
SEE ALSO
    ed(1), ex(1), sed(1), zgrep(1), re_format(7)
STANDARDS
    The grep utility is compliant with the IEEE Std 1003.1-2008 ("POSIX.1") specification.
    The flags [-AaBbCdDgHhIlmoPRSUvW] are extensions to that specification, and the behaviour of the -f flag when used with an empty pattern file is left undefined.
    All long options are provided for compatibility with GNU versions of this utility.
    Historic versions of the grep utility also supported the flags [-rui]. This implementation supports those options; however, their use is strongly discouraged.
HISTORY
    The grep command first appeared in Version 6 AT&T UNIX.
BSD
===== GNU =====
NAME
    grep, egrep, fgrep, rgrep - print lines that match patterns
SYNOPSIS
    grep [OPTION...] PATTERNS [FILE...]
    grep [OPTION...] -e PATTERNS ... [FILE...]
    grep [OPTION...] -f PATTERN_FILE ... [FILE...]
DESCRIPTION
    grep searches for PATTERNS in each FILE. PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern. Typically PATTERNS should be quoted when grep is used in a shell command.
    A FILE of "-" stands for standard input. If no FILE is given, recursive searches examine the working directory, and nonrecursive searches read standard input.
    Debian also includes the variant programs egrep, fgrep and rgrep. These programs are the same as grep -E, grep -F, and grep -r, respectively. These variants are deprecated upstream, but Debian provides for backward compatibility. For portability reasons, it is recommended to avoid the variant programs, and use grep with the related option instead.
OPTIONS
    Generic Program Information
    --help Output a usage message and exit.
    -V, --version
        Output the version number of grep and exit.
    Pattern Syntax
    -E, --extended-regexp
        Interpret PATTERNS as extended regular expressions (EREs, see below).
    -F, --fixed-strings
        Interpret PATTERNS as fixed strings, not regular expressions.
    -G, --basic-regexp
        Interpret PATTERNS as basic regular expressions (BREs, see below). This is the default.
    -P, --perl-regexp
        Interpret PATTERNS as Perl-compatible regular expressions (PCREs). This option is experimental when combined with the -z (--null-data) option, and grep -P may warn of unimplemented features.
    Matching Control
    -e PATTERNS, --regexp=PATTERNS
        Use PATTERNS as the patterns. If this option is used multiple times or is combined with the -f (--file) option, search for all patterns given. This option can be used to protect a pattern beginning with "-".
    -f FILE, --file=FILE
        Obtain patterns from FILE, one per line. If this option is used multiple times or is combined with the -e (--regexp) option, search for all patterns given. The empty file contains zero patterns, and therefore matches nothing.
    -i, --ignore-case
        Ignore case distinctions in patterns and input data, so that characters that differ only in case match each other.
    --no-ignore-case
        Do not ignore case distinctions in patterns and input data. This is the default. This option is useful for passing to shell scripts that already use -i, to cancel its effects because the two options override each other.
    -v, --invert-match
        Invert the sense of matching, to select non-matching lines.
    -w, --word-regexp
        Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore. This option has no effect if -x is also specified.
    -x, --line-regexp
        Select only those matches that exactly match the whole line. For a regular expression pattern, this is like parenthesizing the pattern and then surrounding it with ^ and $.
    General Output Control
    -c, --count
        Suppress normal output; instead print a count of matching lines for each input file. With the -v, --invert-match option (see above), count non-matching lines.
    --color[=WHEN], --colour[=WHEN]
        Surround the matched (non-empty) strings, matching lines, context lines, file names, line numbers, byte offsets, and separators (for fields and groups of context lines) with escape sequences to display them in color on the terminal. The colors are defined by the environment variable GREP_COLORS. WHEN is never, always, or auto.
    -L, --files-without-match
        Suppress normal output; instead print the name of each input file from which no output would normally have been printed.
    -l, --files-with-matches
        Suppress normal output; instead print the name of each input file from which output would normally have been printed. Scanning each input file stops upon first match.
    -m NUM, --max-count=NUM
        Stop reading a file after NUM matching lines. If NUM is zero, grep stops right away without reading input. A NUM of -1 is treated as infinity and grep does not stop; this is the default. If the input is standard input from a regular file, and NUM matching lines are output, grep ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the presence of trailing context lines. This enables a calling process to resume a search. When grep stops after NUM matching lines, it outputs any trailing context lines. When the -c or --count option is also used, grep does not output a count greater than NUM. When the -v or --invert-match option is also used, grep stops after outputting NUM non-matching lines.
    -o, --only-matching
        Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.
    -q, --quiet, --silent
        Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected.
    -s, --no-messages
        Suppress error messages about nonexistent or unreadable files.

```

Output Line Prefix Control

-b, --byte-offset

Print the 0-based byte offset within the input file before each line of output. If -o (--only-matching) is specified, print the offset of the matching part itself.

-H, --with-filename

Print the file name for each match. This is the default when there is more than one file to search. This is a GNU extension.

-h, --no-filename

Suppress the prefixing of file names on output. This is the default when there is only one file (or only standard input) to search.

--label=LABEL

Display input actually coming from standard input as input coming from file LABEL. This can be useful for commands that transform a file's contents before searching, e.g., `gzip -cd foo.gz | grep --label=foo -H 'some pattern'`. See also the -H option.

-n, --line-number

Prefix each line of output with the 1-based line number within its input file.

-T, --initial-tab

Make sure that the first character of actual line content lies on a tab stop, so that the alignment of tabs looks normal. This is useful with options that prefix their output to the actual content: -H, -n, and -b. In order to improve the probability that lines from a single file will all start at the same column, this also causes the line number and byte offset (if present) to be printed in a minimum size field width.

-Z, --null

Output a zero byte (the ASCII NUL character) instead of the character that normally follows a file name. For example, `grep -lZ` outputs a zero byte after each file name instead of the usual newline. This option makes the output unambiguous, even in the presence of file names containing unusual characters like newlines. This option can be used with commands like `find -print0`, `perl -0`, `sort -Z`, and `xargs -0` to process arbitrary file names, even those that contain newline characters.

Context Line Control

-A NUM, --after-context=NUM

Print NUM lines of trailing context after matching lines. Places a line containing a group separator (--) between contiguous groups of matches. With the -o or --only-matching option, this has no effect and a warning is given.

-B NUM, --before-context=NUM

Print NUM lines of leading context before matching lines. Places a line containing a group separator (--) between contiguous groups of matches. With the -o or --only-matching option, this has no effect and a warning is given.

-C NUM, --context=NUM

Print NUM lines of output context. Places a line containing a group separator (--) between contiguous groups of matches. With the -o or --only-matching option, this has no effect and a warning is given.

--group-separator=SEP

When -A, -B, or -C are in use, print SEP instead of -- between groups of lines.

--no-group-separator

When -A, -B, or -C are in use, do not print a separator between groups of lines.

File and Directory Selection

-a, --text

Process a binary file as if it were text; this is equivalent to the --binary-files=text option.

--binary-files=TYPE

If a file's data or metadata indicate that the file contains binary data, assume that the file is of type TYPE. Non-text bytes indicate binary data; these are either output bytes that are improperly encoded for the current locale, or null input bytes when the -z option is not given.

By default, TYPE is binary, and grep suppresses output after null input binary data is discovered, and suppresses output lines that contain improperly encoded data. When some output is suppressed, grep follows any output with a message to standard error saying that a binary file matches.

If TYPE is without-match, when grep discovers null input binary data it assumes that the rest of the file does not match; this is equivalent to the -I option.

If TYPE is text, grep processes a binary file as if it were text; this is equivalent to the -a option.

When type is binary, grep may treat non-text bytes as line terminators even without the -z option. This means choosing binary versus text can affect whether a pattern matches a file. For example, when type is binary the pattern `q$` might match `q` immediately followed by a null byte, even though this is not matched when type is text. Conversely, when type is binary the pattern `.` (period) might not match a null byte.

Warning: The -a option might output binary garbage, which can have nasty side effects if the output is a terminal and if the terminal driver interprets some of it as commands. On the other hand, when reading files whose text encodings are unknown, it can be helpful to use -a or to set `LC_ALL='C'` in the environment, in order to find more matches even if the matches are unsafe for direct display.

-D ACTION, --devices=ACTION

If an input file is a device, FIFO or socket, use ACTION to process it. By default, ACTION is read, which means that devices are read just as if they were ordinary files. If ACTION is skip, devices are silently skipped.

-d ACTION, --directories=ACTION

If an input file is a directory, use ACTION to process it. By default, ACTION is read, i.e., read directories just as if they were ordinary files. If ACTION is skip, silently skip directories. If ACTION is recurse, read all files under each directory, recursively, following symbolic links only if they are on the command line. This is equivalent to the -r option.

--exclude=GLOB

Skip any command-line file with a name suffix that matches the pattern GLOB, using wildcard matching; a name suffix is either the whole name, or a trailing part that starts with a non-slash character immediately after a slash (/) in the name. When searching recursively, skip any subfile whose base name matches GLOB; the base name is the part after the last slash. A pattern can use *, ?, and [...] as wildcards, and \ to quote a wildcard or backslash character literally.

--exclude-from=FILE

Skip files whose base name matches any of the file-name globs read from FILE (using wildcard matching as described under --exclude).

--exclude-dir=GLOB

Skip any command-line directory with a name suffix that matches the pattern GLOB. When searching recursively, skip any subdirectory whose base name matches GLOB. Ignore any redundant trailing slashes in GLOB.

-I Process a binary file as if it did not contain matching data; this is equivalent to the --binary-files=without-match option.

--include=GLOB

Search only files whose base name matches GLOB (using wildcard matching as described under --exclude). If contradictory --include and --exclude options are given, the last matching one wins. If no --include or --exclude options match, a file is included unless the first such option is --include.

-r, --recursive

Read all files under each directory, recursively, following symbolic links only if they are on the command line. Note that if no file operand is given, grep searches the working directory. This is equivalent to the -d recurse option.

-R, --dereference-recursive

Read all files under each directory, recursively. Follow all symbolic links, unlike -r.

Other Options

--line-buffered

Use line buffering on output. This can cause a performance penalty.

-U, --binary

Treat the file(s) as binary. By default, under MS-DOS and MS-Windows, grep guesses whether a file is text or binary as described for the --binary-files option. If grep decides the file is a text file, it strips the CR characters from the original file contents (to make regular expressions with ^ and \$ work correctly). Specifying -U overrules this guesswork, causing all files to be read and passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some regular expressions to fail. This option has no effect on platforms other than MS-DOS and MS-Windows.

-Z, --null-data

Treat input and output data as sequences of lines, each terminated by a zero byte (the ASCII NUL character) instead of a newline. Like the -Z or --null option, this option can be used with commands like `sort -Z` to process arbitrary file names.

REGULAR EXPRESSIONS

A regular expression is a pattern that describes a set of strings.

Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions. grep understands three different versions of regular expression syntax: "basic" (BRE), "extended" (ERE) and "perl" (PCRE). In GNU grep there is no difference in available functionality between basic and extended syntax. In other implementations, basic regular expressions are less powerful. The following description applies to extended regular expressions; differences for basic regular expressions are summarized afterwards. Perl-compatible regular expressions give additional functionality, and are documented in `pcresyntax(3)` and `pcr2pattern(3)`, but work only if PCRE support is enabled.

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any meta-character with special meaning may be quoted by preceding it with a backslash.

The period `.` matches any single character. It is unspecified whether it matches an encoding error.

Character Classes and Bracket Expressions

A bracket expression is a list of characters enclosed by [and]. It matches any single character in that list. If the first character of the list is the caret `^` then it matches any character not in the list; it is unspecified whether it matches an encoding error. For example, the regular expression `[0123456789]` matches any single digit.

Within a bracket expression, a range expression consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters, inclusive, using the locale's collating sequence and character set. For example, in the default C locale, `[a-d]` is equivalent to `[abcd]`. Many locales sort characters in dictionary order, and in these locales `[a-d]` is typically not equivalent to `[abcd]`; it might be equivalent to `[aBbCcDd]`, for example. To obtain the traditional interpretation of bracket expressions, you can use the C locale by setting the `LC_ALL` environment variable to the value C.

Finally, certain named classes of characters are predefined within bracket expressions, as follows. Their names are self explanatory, and they are `[:alnum:]`, `[:alpha:]`, `[:blank:]`, `[:cntrl:]`, `[:digit:]`, `[:graph:]`, `[:lower:]`, `[:print:]`, `[:punct:]`, `[:space:]`, `[:upper:]`, and `[:xdigit:]`. For example, `[:alnum:]` means the character class of numbers and letters in the current locale. In the C locale and ASCII character set encoding, this is the same as `[0-9A-Za-z]`. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket expression.) Most meta-characters lose their special meaning inside bracket expressions. To include a literal `]` place it first in the list. Similarly, to include a literal `^` place it anywhere but first. Finally, to include a literal `-` place it last.

Anchoring

The caret `^` and the dollar sign `$` are meta-characters that respectively match the empty string at the beginning and end of a line. The Backslash Character and Special Expressions

The symbols `<` and `>` respectively match the empty string at the beginning and end of a word. The symbol `\b` matches the empty string at the edge of a word, and `\B` matches the empty string provided it's not at the edge of a word. The symbol `\w` is a synonym for `[:alnum:]` and `\W` is a synonym for `[^[:alnum:]]`.

Repetition

A regular expression may be followed by one of several repetition operators:

? The preceding item is optional and matched at most once.

* The preceding item will be matched zero or more times.

+ The preceding item will be matched one or more times.

{n} The preceding item is matched exactly n times.

{n,} The preceding item is matched n or more times.

{n,m} The preceding item is matched at most m times. This is a GNU extension.

{n,m} The preceding item is matched at least n times, but not more than m times.

Concatenation

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated expressions.

Alternation

Two regular expressions may be joined by the infix operator `|`; the resulting regular expression matches any string matching either alternate expression.

Precedence

Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole expression may be enclosed in parentheses to override these precedence rules and form a subexpression. Back-references and Subexpressions

The back-reference `\n`, where n is a single digit, matches the substring previously matched by the nth parenthesized subexpression of the regular expression.

Basic vs Extended Regular Expressions

In basic regular expressions the meta-characters `?`, `+`, `{`, `|`, `(`, and `)` lose their special meaning; instead use the backslashed versions `\?`, `\+`, `\{`, `\|`, `\(`, and `\)`.

EXIT STATUS

Normally the exit status is 0 if a line is selected, 1 if no lines were selected, and 2 if an error occurred. However, if the -q or --quiet or -s or --silent is used and a line is selected, the exit status is 0 even if an error occurred.

ENVIRONMENT

The behavior of grep is affected by the following environment variables.

The locale for category `LC_foo` is specified by examining the three environment variables `LC_ALL`, `LC_foo`, `LANG`, in that order. The first of these variables that is set specifies the locale. For example, if `LC_ALL` is not set but `LC_MESSAGES` is set to `pt_BR` then the Brazilian Portuguese locale is used for the `LC_MESSAGES` category. The C locale is used if none of these environment variables are set, if the locale catalog is not installed, or if grep was not compiled with national language support (NLS). The shell command `locale -a` lists locales that are currently available.

GREP COLORS

Controls how the --color option highlights output. Its value is a colon-separated list of capabilities that defaults to `ms=01;31;mc=01;31;sl=;cx=;fn=35;ln=32;bn=32;se=36` with the `rv` and `ne` boolean capabilities omitted (i.e., false). Supported capabilities are as follows:

`sl=` SGR substring for whole selected lines (i.e., matching lines when the -v command-line option is omitted, or non-matching lines when -

v is specified). If however the boolean rv capability and the -v command-line option are both specified, it applies to context matching lines instead. The default is empty (i.e., the terminal's default color pair).

cx= SGR substring for whole context lines (i.e., non-matching lines when the -v command-line option is omitted, or matching lines when -v is specified). If however the boolean rv capability and the -v command-line option are both specified, it applies to selected non-matching lines instead. The default is empty (i.e., the terminal's default color pair).

rv Boolean value that reverses (swaps) the meanings of the sl= and cx= capabilities when the -v command-line option is specified. The default is false (i.e., the capability is omitted).

mt=01;31 SGR substring for matching non-empty text in any matching line (i.e., a selected line when the -v command-line option is omitted, or a context line when -v is specified). Setting this is equivalent to setting both ms= and mc= at once to the same value. The default is a bold red text foreground over the current line background.

ms=01;31 SGR substring for matching non-empty text in a selected line. (This is only used when the -v command-line option is omitted.) The effect of the sl= (or cx= if rv) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

mc=01;31 SGR substring for matching non-empty text in a context line. (This is only used when the -v command-line option is specified.) The effect of the cx= (or sl= if rv) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

fn=35 SGR substring for file names prefixing any content line. The default is a magenta text foreground over the terminal's default background.

ln=32 SGR substring for line numbers prefixing any content line. The default is a green text foreground over the terminal's default background.

bn=32 SGR substring for byte offsets prefixing any content line. The default is a green text foreground over the terminal's default background.

se=36 SGR substring for separators that are inserted between selected line fields (:), between context line fields, (-), and between groups of adjacent lines when nonzero context is specified (--). The default is a cyan text foreground over the terminal's default background.

ne Boolean value that prevents clearing to the end of line using Erase in Line (EL) to Right (\33[K) each time a colorized item ends. This is needed on terminals on which EL is not supported. It is otherwise useful on terminals for which the back color erase (bce) boolean terminfo capability does not apply, when the chosen highlight colors do not affect the background, or when EL is too slow or causes too much flicker. The default is false (i.e., the capability is omitted).

Note that boolean capabilities have no =... part. They are omitted (i.e., false) by default and become true when specified.

See the Select Graphic Rendition (SGR) section in the documentation of the text terminal that is used for permitted values and their meaning as character attributes. These substring values are integers in decimal representation and can be concatenated with semicolons. grep takes care of assembling the result into a complete SGR sequence (\33[...m). Common values to concatenate include 1 for bold, 4 for underline, 5 for blink, 7 for inverse, 39 for default foreground color, 30 to 37 for foreground colors, 90 to 97 for 16-color mode foreground colors, 38;5;0 to 38;5;255 for 88-color and 256-color modes foreground colors, 49 for default background color, 40 to 47 for background colors, 100 to 107 for 16-color mode background colors, and 48;5;0 to 48;5;255 for 88-color and 256-color modes background colors.

LC_ALL, LC_COLLATE, LANG

These variables specify the locale for the LC_COLLATE category, which determines the collating sequence used to interpret range expressions like [a-z].

LC_ALL, LC_CTYPE, LANG

These variables specify the locale for the LC_CTYPE category, which determines the type of characters, e.g., which characters are whitespace. This category also determines the character encoding, that is, whether text is encoded in UTF-8, ASCII, or some other encoding. In the C or POSIX locale, all characters are encoded as a single byte and every byte is a valid character.

LC_ALL, LC_MESSAGES, LANG

These variables specify the locale for the LC_MESSAGES category, which determines the language that grep uses for messages. The default C locale uses American English messages.

POSIXLY_CORRECT

If set, grep behaves as POSIX requires; otherwise, grep behaves more like other GNU programs. POSIX requires that options that follow file names must be treated as file names; by default, such options are permuted to the front of the operand list and are treated as options. Also, POSIX requires that unrecognized options be diagnosed as "illegal", but since they are not really against the law the default is to diagnose them as "invalid". POSIXLY_CORRECT also disables _N_GNU_nonoption_argv_flags_, described below.

_N_GNU_nonoption_argv_flags_

(Here N is grep's numeric process ID.) If the ith character of this environment variable's value is 1, do not consider the ith operand of grep to be an option, even if it appears to be one. A shell can put this variable in the environment for each command it runs, specifying which operands are the results of file name wildcard expansion and therefore should not be treated as options. This behavior is available only with the GNU C library, and only when POSIXLY_CORRECT is not set.

NOTES

This man page is maintained only fitfully; the full documentation is often more up-to-date.

COPYRIGHT

Copyright 1998-2000, 2002, 2005-2022 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

BUGS

Reporting Bugs

Email bug reports to the bug-reporting address (bug-grep@gnu.org). An email archive (https://lists.gnu.org/mailman/listinfo/bug-grep) and a bug tracker (https://debbugs.gnu.org/cgi/pkgreport.cgi?package=grep) are available.

Known Bugs

Large repetition counts in the {n,m} construct may cause grep to use lots of memory. In addition, certain other obscure regular expressions require exponential time and space, and may cause grep to run out of memory.

Back-references are very slow, and may require exponential time.

EXAMPLE

The following example outputs the location and contents of any line containing "f" and ending in ".c", within all files in the current directory whose names contain "g" and end in ".h". The -n option outputs line numbers, the -- argument treats expansions of "*g.h" starting with "-" as file names not options, and the empty file /dev/null causes file names to be output even if only one file name happens to be of the form "*g.h".

```
$ grep -n -- 'f.*.c$' *g.h /dev/null
argmatch.h:1:/* definitions and prototypes for argmatch.c
```

The only line that matches is line 1 of argmatch.h. Note that the regular expression syntax used in the pattern differs from the globbing syntax that the shell uses to match file names.

SEE ALSO

Regular Manual Pages

awk(1), cmp(1), diff(1), find(1), perl(1), sed(1), sort(1), xargs(1), read(2), pcre2(3), pcre2syntax(3), pcre2pattern(3), terminfo(5), glob(7), regex(7)

Full Documentation

A complete manual (https://www.gnu.org/software/grep/manual/) is available. If the info and grep programs are properly installed at your site, the command

info grep

should give you access to the complete manual.

GNU grep 3.8

true

===== suckless =====

NAME

true - return success

SYNOPSIS

true

DESCRIPTION

true returns a status code indicating success.

STANDARDS

POSIX.1-2013.

sbase

===== BSD =====

NAME

true - return true value

SYNOPSIS

true

DESCRIPTION

The true utility always returns with an exit code of zero.

Some shells may provide a builtin true command which is identical to this utility. Consult the builtin(1) manual page.

SEE ALSO

builtin(1), csh(1), false(1), sh(1)

STANDARDS

The true utility is expected to be IEEE Std 1003.2 ("POSIX.2") compatible.

BSD

===== GNU =====

NAME

true - do nothing, successfully

SYNOPSIS

true [ignored command line arguments]

true OPTION

DESCRIPTION

Exit with a status code indicating success.

--help display this help and exit

--version

output version information and exit

NOTE: your shell may have its own version of true, which usually supersedes the version described here. Please refer to your shell's documentation for details about the options it supports.

AUTHOR

Written by Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>

Report any translation bugs to <https://translationproject.org/team/>

COPYRIGHT

Copyright © 2022 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <https://www.gnu.org/software/coreutils/true>

or available locally via: info '(coreutils) true invocation'

GNU coreutils 9.1

===== POSIX =====

PROLOG

...

NAME

true - return true value

SYNOPSIS

true

DESCRIPTION

The true utility shall return with exit code zero.

OPTIONS

None.

OPERANDS

None.

...

EXIT STATUS

Zero.

CONSEQUENCES OF ERRORS

None.

The following sections are informative.

APPLICATION USAGE

This utility is typically used in shell scripts, as shown in the

EXAMPLES section. The special built-in utility : is sometimes more

efficient than true.

EXAMPLES

This command is executed forever:

```
while true
```

```
do
```

```
command
```

```
done
```

RATIONALE

The true utility has been retained in this volume of POSIX.1-2017, even though the shell special built-in : provides similar functionality, because true is widely used in historical scripts and is less cryptic to novice script readers.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.9, Shell Commands, false

COPYRIGHT

...

busybox (digest)

NAME

BusyBox - The Swiss Army Knife of Embedded Linux

SYNTAX

busybox <applet> [arguments...] # or

<applet> [arguments...] # if symlinked

DESCRIPTION

BusyBox combines tiny versions of many common UNIX utilities into a single small executable. It provides minimalist replacements for most of the utilities you usually find in GNU coreutils, util-linux, etc. The utilities in BusyBox generally have fewer options than their full-featured GNU cousins; however, the options that are included provide the expected functionality and behave very much like their GNU counterparts.

BusyBox has been written with size-optimization and limited resources in mind. It is also extremely modular so you can easily include or exclude commands (or features) at compile time. This makes it easy to customize your embedded systems. To create a working system, just add /dev, /etc, and a Linux kernel. BusyBox provides a fairly complete POSIX environment for any small or embedded system.

...

COMMAND DESCRIPTIONS

...

```

cat cat [-nbvteA] [FILE]...
Print FILES to stdout
-n Number output lines
-b Number nonempty lines
-v Show nonprinting characters as ^x or M-x
-t ...and tabs as ^I
-e ...and end lines with $
-A Same as -vte

...
chmod
chmod [-Rcvf] MODE[,MODE]... FILE...
MODE is octal number (bit pattern sstrwxrwxrwx) or [ugoa][+|-|=]
[rxwXst]
-R Recurse
-c List changed files
-v Verbose
-f Hide errors

...
cmp cmp [-ls] [-n NUM] FILE1 [FILE2 [SKIP1 [SKIP2]]]
Compare FILE1 with FILE2 (or stdin)
-l Write the byte numbers (decimal) and values (octal)
  for all differing bytes
-s Quiet
-n NUM Compare at most NUM bytes
cp cp [-arPLHpfiHlsu] SOURCE DEST or: cp [-arPLHpfiHlsu] SOURCE...
{ -t DIRECTORY | DIRECTORY }
Copy SOURCES to DEST
-a Same as -dpr
-R,-r Recurse
-d,-P Preserve symlinks (default if -R)
-L Follow all symlinks
-H Follow symlinks on command line
-p Preserve file attributes if possible
-f Overwrite
-i Prompt before overwrite
-n Don't overwrite
-l,-s Create (sym)links
-T Refuse to copy if DEST is a directory
-t DIR Copy all SOURCES into DIR
-u Copy only newer files

...
date
date [OPTIONS] [+FMT] [[-s] TIME]
Display time (using +FMT), or set time
-u Work in UTC (don't convert to local time)
[-s] TIME Set time to TIME
-d TIME Display TIME, not 'now'
-D FMT FMT (strftime format) for -s/-d TIME conversion
-r FILE Display last modification time of FILE
-R Output RFC-2822 date
-I[SPEC] Output ISO-8601 date
      SPEC=date (default), hours, minutes, seconds or ns
Recognized TIME formats:
@seconds_since_1970
hh:mm[:ss]
[YYYY.]MM.DD-hh:mm[:ss]
YYYY-MM-DD hh:mm[:ss]
[[[YY]YY]MM]DD[hh]mm[:ss]
'date TIME' form accepts MMDDhhmm[YY]YY[.ss] instead

...
dd dd [if=FILE] [of=FILE] [ibs=N obs=N/bs=N] [count=N] [skip=N]
[seek=N] [conv=notrunc|noerror|sync|fsync] [iflag=skip_bytes|
count_bytes|fullblock|direct] [oflag=seek_bytes|append|direct]
Copy a file with converting and formatting
if=FILE Read from FILE instead of stdin
of=FILE Write to FILE instead of stdout
bs=N Read and write N bytes at a time
ibs=N Read N bytes at a time
obs=N Write N bytes at a time
count=N Copy only N input blocks
skip=N Skip N input blocks
seek=N Skip N output blocks
conv=notrunc Don't truncate output file
conv=noerror Continue after read errors
conv=sync Pad blocks with zeros
conv=fsync Physically write data out before finishing
conv=swab Swap every pair of bytes
iflag=skip_bytes skip=N is in bytes
iflag=count_bytes count=N is in bytes
oflag=seek_bytes seek=N is in bytes
iflag=direct O_DIRECT input
oflag=direct O_DIRECT output
iflag=fullblock Read full blocks
oflag=append Open output in append mode
status=noxfer Suppress rate output
status=none Suppress all output
N may be suffixed by c (1), w (2), b (512), kB (1000), k (1024), MB,
M, GB, G
...
diff
diff [-abBdiNqrTstw] [-L LABEL] [-S FILE] [-U LINES] FILE1 FILE2
Compare files line by line and output the differences between them.
This implementation supports unified diffs only.
-a Treat all files as text
-b Ignore changes in the amount of whitespace
-B Ignore changes whose lines are all blank
-d Try hard to find a smaller set of changes
-l Ignore case differences
-L Use LABEL instead of the filename in the unified header
-N Treat absent files as empty
-q Output only whether files differ
-r Recurse
-R Start with FILE when comparing directories
-T Make tabs line up by prefixing a tab when necessary
-s Report when two files are the same
-u Expand tabs to spaces in output
-U Output LINES lines of context
-w Ignore all whitespace

...
df df [-PkmbTail] [-B SIZE] [-t TYPE] [FILESYSTEM]...
Print filesystem usage statistics
-P POSIX output format
-k 1024-byte blocks (default)
-m 1M-byte blocks
-h Human readable (e.g. 1K 243M 2G)
-t Print filesystem type
-T TYPE Print only mounts of this type
-a Show all filesystems
-i Inodes
-B SIZE Blocksize

...
du du [-aHdclsxhmk] [FILE]...
Summarize disk space used for FILES (or directories)
-a Show file sizes too
-b Apparent size (including holes)
-L Follow all symlinks
-H Follow symlinks on command line
-d N Limit output to directories (and files with -a) of depth <
N
-c Show grand total
-l Count sizes many times if hard linked
-s Display only a total for each argument

-x Skip directories on different filesystems
-h Sizes in human readable format (e.g., 1K 243M 2G)
-m Sizes in megabytes
-k Sizes in kilobytes (default)

...
echo
echo [-neE] [ARG]...
Print ARGs to stdout
-n No trailing newline
-e Interpret backslash escapes (\t=tab etc)
-E Don't interpret backslash escapes (default)

...
find
find [-HL] [PATH]... [OPTIONS] [ACTIONS]
Search for files and perform actions on them. First failed action
stops processing of current file. Defaults: PATH is current directory,
action is '-print'
-L,-follow Follow symlinks
-H ...on command line only
-xdev Don't descend directories on other filesystems
-maxdepth N Descend at most N levels. -maxdepth 0 applies
actions to command line arguments only
-mindepth N Don't act on first N levels
-depth Act on directory *after* traversing it
Actions:
( ACTIONS ) Group actions for -o / -a
! ACT Invert ACT's success/failure
ACT1 [-a] ACT2 If ACT1 fails, stop, else do ACT2
ACT1 -o ACT2 If ACT1 succeeds, stop, else do ACT2
Note: -a has higher priority than -o
-name PATTERN Match file name (w/o directory name) to PATTERN
-iname PATTERN Case insensitive -name
-path PATTERN Match path to PATTERN
-ipath PATTERN Case insensitive -path
-regex PATTERN Match path to regex PATTERN
-type X File type is X (one of: f,d,l,b,c,s,p)
-executable File is executable
-perm MASK At least one mask bit (+MASK), all bits (-MASK),
or exactly MASK bits are set in file's mode
-mtime DAYS mtime is greater than (+N), less than (-N),
or exactly N days in the past
-atime DAYS atime +N/-N/N days in the past
-ctime DAYS ctime +N/-N/N days in the past
-mmin MINS mtime is greater than (+N), less than (-N),
or exactly N minutes in the past
-amin MINS atime +N/-N/N minutes in the past
-cmin MINS ctime +N/-N/N minutes in the past
-newer FILE mtime is more recent than FILE's
-inum N File has inode number N
-samefile FILE File is same as FILE
-user NAME/ID File is owned by given user
-group NAME/ID File is owned by given group
-size N[bck] File size is N (c:bytes,k:kbytes,b:512
bytes(def.))
+/-N: file size is bigger/smaller than N
-links N Number of links is greater than (+N), less than (-
N),
or exactly N
-empty Match empty file/directory
-prune If current file is directory, don't descend into it
If none of the following actions is specified, -print is assumed
-print Print file name
-print0 Print file name, NUL terminated
-exec CMD ARG ; Run CMD with all instances of {} replaced by
file name. Fails if CMD exits with nonzero
-exec CMD ARG + Run CMD with {} replaced by list of file names
-quit Exit

...
true; do case "$1" in -a) echo A; shift;; -b|--bb) echo "B:$2";
shift 2;; -c) case "$2" in "") echo C; shift 2;; *) echo
"C:$2"; shift 2;; esac;; --) shift; break;; *) echo Error;
exit 1;; esac done

...
grep
grep [-HhnlLqovsrRwFEz] [-m N] [-A|B|C N] { PATTERN | -e PATTERN...
| -f FILE... } [FILE]...
Search for PATTERN in FILES (or stdin)
-H Add 'filename:' prefix
-h Do not add 'filename:' prefix
-n Add 'line no:' prefix
-l Show only names of files that match
-L Show only names of files that don't match
-c Show only count of matching lines
-o Show only the matching part of line
-q Quiet. Return 0 if PATTERN is found, 1 otherwise
-v Select non-matching lines
-s Suppress open and read errors
-r Recurse
-R Recurse and dereference symlinks
-i Ignore case
-w Match whole words only
-x Match whole lines only
-F PATTERN is a literal (not regexp)
-E PATTERN is an extended regexp
-z NUL terminated input
-m N Match up to N times per file
-A N Print N lines of trailing context
-B N Print N lines of leading context
-C N Same as '-A N -B N'
-e PTRN Pattern to match
-f FILE Read pattern from file

...
halt
halt [-d DELAY] [-nfw]
Halt the system
-d SEC Delay interval
-n Do not sync
-f Force (don't go through init)
-w Only write a wtmp record

head
head [OPTIONS] [FILE]...
Print first 10 lines of FILES (or stdin). With more than one FILE,
precede each with a filename header.
-n N[bkm] Print first N lines
-n N[bkm] Print all except N last lines
-c [-]N[bkm] Print first N bytes
(b:*512 k:*1024 m:*1024*2)
-q Never print headers
-v Always print headers

hexdump
hexdump [-bcdoxCv] [-e FMT] [-f FMT_FILE] [-n LEN] [-s OFS]
[FILE]...
Display FILES (or stdin) in a user specified format
-b 1-byte octal display
-c 1-byte character display
-d 2-byte decimal display
-o 2-byte octal display
-x 2-byte hex display
-C hex+ASCII 16 bytes per line
-v Show all (no dup folding)
-e FORMAT_STR Example: '16/1 "%02x|""\n"'
-f FORMAT_FILE

```

```

-n LENGTH      Show only first LENGTH bytes
-s OFFSET      Skip OFFSET bytes

...
ifconfig
ifconfig [-a] [IFACE] [ADDRESS]
Configure a network interface
[add ADDRESS[/PREFIXLEN]]
[del ADDRESS[/PREFIXLEN]]
[[-]broadcast [ADDRESS]] [[-]pointopoint [ADDRESS]]
[netmask ADDRESS] [dstaddr ADDRESS]
[outfill NN] [keepalive NN]
[hw ether|infiniband ADDRESS] [metric NN] [mtu NN]
[[-]trailers] [[-]arp] [[-]allmulti]
[multicast] [[-]promisc] [txqueuelen NN] [[-]dynamic]
[mem_start NN] [lo_addr NN] [irq NN]
[up|down] ...

...
kill
kill [-l] [-SIG] PID...
Send a signal (default: TERM) to given PIDs
-l List all signal names and numbers

...
less
less [-EFIMmNSRh-] [FILE]...
View FILE (or stdin) one screenful at a time
-E Quit once the end of a file is reached
-F Quit if entire file fits on first screen
-I Ignore case in all searches
-M, -m Display status line with line numbers
and percentage through the file
-N Prefix line number to each line
-S Truncate long lines
-R Remove color escape codes in input
-- Suppress -s displayed past EOF

...
ls ls [-lAaCxdLHRFplnshrsXvctu] [-w WIDTH] [FILE]...
List directory contents
-l One column output
-a Include names starting with .
-A Like -a, but exclude . and ..
-x List by lines
-d List directory names, not contents
-L Follow symlinks
-H Follow symlinks on command line
-R Recurse
-p Append / to directory names
-F Append indicator (one of */=#|) to names
-l Long format
-i List inode numbers
-n List numeric UIDs and GIDs instead of names
-s List allocated blocks
-lc List ctime
-lu List atime
--full-time List full date/time
-h Human readable sizes (1K 243M 2G)
--group-directories-first
-S Sort by size
-X Sort by extension
-v Sort by version
-t Sort by mtime
-tc Sort by ctime
-tu Sort by atime
-r Reverse sort order
-w N Format N columns wide
--color[=always,never,auto]

...
mkdir
mkdir [-m MODE] [-p] DIRECTORY...
Create DIRECTORY
-m MODE Mode
-p No error if exists; make parent directories as needed

...
mv mv [-finT] SOURCE DEST or: mv [-fin] SOURCE... {-t DIRECTORY |
DIRECTORY }
Rename SOURCE to DEST, or move SOURCES to DIRECTORY
-f Don't prompt before overwriting
-i Interactive, prompt before overwrite
-n Don't overwrite an existing file
-T Refuse to move if DEST is a directory
-t DIR Move all SOURCES into DIR

...
patch
patch [-RNE] [-p N] [-i DIFF] [ORIGFILE [PATCHFILE]]
-p N Strip N leading components from file names
-i DIFF Read DIFF instead of stdin
-R Reverse patch
-N Ignore already applied patches
-E Remove output files if they become empty
--dry-run Don't actually change files

...
ping
ping [OPTIONS] HOST
Send ICMP ECHO REQUESTs to HOST
-4,-6 Force IP or IPv6 name resolution
-c CNT Send only CNT pings
-s SIZE Send SIZE data bytes in packets (default 56)
-i SECS Interval
-A Ping as soon as reply is received
-t TTL Set TTL
-I IFACE/IP Source interface or IP address
-W SEC Seconds to wait for the first response (default 10)
(after all -c CNT packets are sent)
-w SEC Seconds until ping exits (default:infinite)
(can exit earlier with -c CNT)
-q Quiet, only display output at start/finish
-p HEXBYTE Payload pattern

...
printf
printf FORMAT [ARG]...
Format and print ARG(s) according to FORMAT (a-la C printf)

...
rm rm [-irf] FILE...
Remove (unlink) FILES
-i Always prompt before removing
-f Never prompt
-R,-r Recurse

...
sed sed [-i[SFX]] [-nrE] [-f FILE]... [-e CMD]... [FILE]... or: sed [-
i[SFX]] [-nrE] CMD [FILE]...
-e CMD Add CMD to sed commands to be executed
-f FILE Add FILE contents to sed commands to be executed
-i[SFX] Edit files in-place (otherwise write to stdout)
Optionally back files up, appending SFX
-n Suppress automatic printing of pattern space
-r,-E Use extended regex syntax
If no -e or -f, the first non-option argument is the sed command
string. Remaining arguments are input files (stdin if none).

...
shuf
shuf [-n NUM] [-o FILE] [-z] [FILE | -e [ARG...]] | -i L-H
Randomly permute lines
-n NUM Output at most NUM lines

...
sleep
sleep [N]...
Pause for a time equal to the total of the args given, where each
arg can have an optional suffix of (s)econds, (m)inutes, (h)ours, or
(d)ays
sort
sort [-nrgMcszdbfiokt] [-o FILE] [-k START[.OFS] [OPTS] [,END[.OFS]
[OPTS]] [-t CHAR] [FILE]...
Sort lines of text
-o FILE Output to FILE
-c Check whether input is sorted
-b Ignore leading blanks
-f Ignore case
-i Ignore unprintable characters
-d Dictionary order (blank or alphanumeric only)
-n Sort numbers
-g General numerical sort
-M Sort month
-V Sort version
-t CHAR Field separator
-k N[,M] Sort by Nth field
-r Reverse sort order
-s Stable (don't sort ties alphabetically)
-u Suppress duplicate lines
-z NUL terminated input and output

...
tail
tail [OPTIONS] [FILE]...

...
tar tar c|xt [-ZzJjahmvokO] [-f TARFILE] [-C DIR] [FILE]...
Create, extract, or list files from a tar file
c Create
x Extract
t List
-f FILE Name of TARFILE ('-' for stdin/out)
-C DIR Change to DIR before operation
-v Verbose
-O Extract to stdout
-m Don't restore mtime
-o Don't restore user:group
-k Don't replace existing files
-Z (De)compress using compress
-z (De)compress using gzip
-J (De)compress using xz
-j (De)compress using bzip2
--lzma (De)compress using lzma
-a (De)compress based on extension
-h Follow symlinks
--overwrite Replace existing files
--strip-components NUM NUM of leading components to strip
--no-recursion Don't descend in directories
--numeric-owner Use numeric user:group
--no-same-permissions Don't restore access permissions
--to-command COMMAND Pipe files to COMMAND

...
tr tr [-cds] STRING1 [STRING2]
Translate, squeeze, or delete characters from stdin, writing to
stdout
-c Take complement of STRING1
-d Delete input characters coded STRING1
-s Squeeze multiple output characters of STRING2 into one
character

...
uniq
uniq [-cdulz] [-f,s,w N] [FILE [OUTFILE]]
Discard duplicate lines
-c Prefix lines by the number of occurrences
-d Only print duplicate lines
-u Only print unique lines
-i Ignore case
-z NUL terminated output
-f N Skip first N fields
-s N Skip first N chars (after any skipped fields)
-w N Compare N characters in line

...
vi vi [-c CMD] [-R] [-H] [FILE]...
Edit FILE
-c CMD Initial command to run ($EXINIT and ~/.exrc also
available)
-R Read-only
-H List available features

...
wc wc [-cmlwL] [FILE]...
Count lines, words, and bytes for FILES (or stdin)
-c Count bytes
-m Count characters
-l Count newlines
-w Count words
-L Print longest line length

...
wget
wget [-cqs] [--spider] [-O FILE] [-o LOGFILE] [--header STR] [--
post-data STR] [--post-file FILE] [-Y on/off] [--no-check-certificate]
[-P DIR] [-U AGENT] URL...
Retrieve files via HTTP or FTP
--spider Only check URL existence: $? is 0 if exists
--header STR Add STR (of form 'header: value') to headers
--post-data STR Send STR using POST method
--post-file FILE Send FILE using POST method
--no-check-certificate Don't validate the server's certificate
-c Continue retrieval of aborted transfer
-g Quiet
-P DIR Save to DIR (default .)
-S Show server response
-O FILE Save to FILE ('-' for stdout)
-o LOGFILE Log messages to FILE
-U STR Use STR for User-Agent header
-Y on/off Use proxy

...
xargs
xargs [OPTIONS] [PROG ARGS]
Run PROG on every item given by stdin
-o NUL terminated input
-a FILE Read from FILE instead of stdin
-r Don't run command if input is empty
-t Print the command on stderr before execution
-p Ask user whether to run each command
-E STR,-c[STR] STR stops input processing
-I STR Replace STR within PROG ARGS with input line
-n N Pass no more than N args to PROG
-s N Pass command line of no more than N bytes
-P N Run up to N PROGS in parallel
-x Exit if size is exceeded

...
yes yes [STRING]
Repeatedly print a line with STRING, or 'y'

...
version 1.35.0

```