NAME

find − search for files in a directory hierarchy

SYNOPSIS

find [−H] [−L] [−P] [path...] [expression]

DESCRIPTION

This manual page documents the GNU version of **find**. GNU **find** searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left hand side is false for *and* operations, true for *or*), at which point **find** moves on to the next file name.

If you are using **find** in an environment where security is important (for example if you are using it to seach directories that are writable by other users), you should read the "Security Considerations" chapter of the findutils documentation, which is called **Finding Files** and comes with findutils. That document also includes a lot more detail and discussion than this manual page, so you may find it a more useful source of information.

OPTIONS

The '−H', '−L' and '−P' options control the treatment of symbolic links. Command-line arguments following these are taken to be names of files or directories to be examined, up to the first argument that begins with '−', '(', ')', ',', or '!'. That argument and any following arguments are taken to be the expression describing what is to be searched for. If no paths are given, the current directory is used. If no expression is given, the expression '−print' is used (but you should probably consider using '−print0' instead, anyway).

This manual page talks about 'options' within the expression list. These options control the behaviour of **find** but are specified immediately after the last path name. The three 'real' options '−H', '−L' and '−P' must appear before the first path name, if at all.

−P Never follow symbolic links. This is the default behaviour. When **find** examines or prints information a file, and the file is a symbolic link, the information used shall be taken from the properties of the symbolic link itself.

−L Follow symbolic links. When **find** examines or prints information about files, the information used shall be taken from the properties of the file to which the link points, not from the link itself (unless it is a broken symbolic link or **find** is unable to examine the file to which the link points). Use of this option implies −noleaf. If you later use the −P option, −noleaf will still be in effect. If −L is in effect and **find** discovers a symbolic link to a subdirectory during its search, the subdirectory pointed to by the symbolic link will be searched.

When the −L option is in effect, the −type predicate will always match against the type of the file that a symbolic link points to rather than the link itself (unless the symbolic link is broken). Using −L causes the −lname and −ilname predicates always to return false.

−H Do not follow symbolic links, except while processing the command line arguments. When **find** examines or prints information about files, the information used shall be taken from the properties of the symbolic link itself. The only exception to this behaviour is when a file specified on the command line is a symbolic link, and the link can be resolved. For that situation, the information used is taken from whatever the link points to (that is, the link is followed). The information about the link itself is used as a fallback if the file pointed to by the symbolic link cannot be examined. If −H is in effect and one of the paths specified on the command line is a symbolic link to a directory, the contents of that directory will be examined (though of course −maxdepth 0 would prevent this).

If more than one of −H, −L and −P is specified, each overrides the others; the last one appearing on the command line takes effect. Since it is the default, the −P option should be considered to be in effect unless either −H or −L is specified.

GNU **find** frequently stats files during the processing of the command line itself, before any searching has begun. These options also affect how those arguments are processed. Specifically, there are a

number of tests that compare files listed on the command line against a file we are currently considering. In each case, the file specified on the command line will have been examined and some of its properties will have been saved. If the named file is in fact a symbolic link, and the −P option is in effect (or if neither −H nor −L were specified), the information used for the comparison will be taken from the properties of the symbolic link. Otherwise, it will be taken from the properties of the file the link points to. If **find** cannot follow the link (for example because it has insufficient privileges or the link points to a nonexistent file) the properties of the link itself will be used.

When the −H or −L options are in effect, any symbolic links listed as the argument of −newer will be dereferenced, and the timestamp will be taken from the file to which the symbolic link points. The same consideration applies to −anewer and −cnewer.

The −follow option has a similar effect to −L, though it takes effect at the point where it appears (that is, if −L is not used but −follow is, any symbolic links appearing after −follow on the command line will be dereferenced, and those before it will not).

EXPRESSIONS

The expression is made up of options (which affect overall operation rather than the processing of a specific file, and always return true), tests (which return a true or false value), and actions (which have side effects and return a true or false value), all separated by operators. −and is assumed where the operator is omitted. If the expression contains no actions other than −prune, −print is performed on all files for which the expression is true.

OPTIONS

All options always return true. Except for −follow and −daystart, they always take effect, rather than being processed only when their place in the expression is reached. Therefore, for clarity, it is best to place them at the beginning of the expression. A warning is issued if you don't do this.

−daystart

Measure times (for −amin, −atime, −cmin, −ctime, −mmin, and −mtime) from the beginning of today rather than from 24 hours ago. This option only affects tests which appear later on the command line.

−depth Process each directory's contents before the directory itself.

−d A synonym for −depth, for compatibility with FreeBSD, NetBSD, MacOS X and OpenBSD.

−follow

Deprecated; use the −L option instead. Dereference symbolic links. Implies −noleaf. Unless the −H or −L option has been specified, the position of the −follow option changes the behaviour of the −newer predicate; any files listed as the argument of −newer will be dereferenced if they are symbolic links. The same consideration applies to −anewer and −cnewer. Similarly, the −type predicate will always match against the type of the file that a symbolic link points to rather than the link itself. Using −follow causes the −lname and −ilname predicates always to return false.

−help, −−help

Print a summary of the command-line usage of **find** and exit.

−ignore_readdir_race

Normally, **find** will emit an error message when it fails to stat a file. If you give this option and a file is deleted between the time **find** reads the name of the file from the directory and the time it tries to stat the file, no error message will be issued. This also applies to files or directories whose names are given on the command line. This option takes effect at the time the command line is read, which means that you cannot search one part of the filesystem with this option on and part of it with this option off (if you need to do that, you will need to issue two **find** commands instead, one with the option and one without it).

−maxdepth *levels*

Descend at most *levels* (a non-negative integer) levels of directories below the command line arguments. '−maxdepth 0' means only apply the tests and actions to the command line arguments.

−mindepth *levels*
Do not apply any tests or actions at levels less than *levels* (a non-negative integer). '−mindepth 1' means process all files except the command line arguments.

−mount
Don't descend directories on other filesystems. An alternate name for −xdev, for compatibility with some other versions of **find**.

−noignore_readdir_race
Turns off the effect of −ignore_readdir_race.

−noleaf Do not optimize by assuming that directories contain 2 fewer subdirectories than their hard link count. This option is needed when searching filesystems that do not follow the Unix directory-link convention, such as CD-ROM or MS-DOS filesystems or AFS volume mount points. Each directory on a normal Unix filesystem has at least 2 hard links: its name and its '.' entry. Additionally, its subdirectories (if any) each have a '..' entry linked to that directory. When **find** is examining a directory, after it has statted 2 fewer subdirectories than the directory's link count, it knows that the rest of the entries in the directory are non-directories ('leaf' files in the directory tree). If only the files' names need to be examined, there is no need to stat them; this gives a significant increase in search speed.

−version, −−version
Print the **find** version number and exit.

−warn, −nowarn
Turn warning messages on or off. These warnings apply only to the command line usage, not to any conditions that **find** might encounter when it searches directories. The default behaviour corresponds to −warn if standard input is a tty, and to −nowarn otherwise.

−xdev Don't descend directories on other filesystems.

TESTS

Numeric arguments can be specified as

+*n* for greater than *n*,

−*n* for less than *n*,

n for exactly *n*.

−amin *n*
File was last accessed *n* minutes ago.

−anewer *file*
File was last accessed more recently than *file* was modified. If *file* is a symbolic link and the −H option or the −L option is in effect, the access time of the file it points to is always used.

−atime *n*
File was last accessed *n**24 hours ago. When find figures out how many 24-hour preiods ago the file was last accessed, any fractional part is ignored, so to match **−atime +1**, a file has to have been modified at least *two* days ago.

−cmin *n*
File's status was last changed *n* minutes ago.

−cnewer *file*
File's status was last changed more recently than *file* was modified. If *file* is a symbolic link and the −H option or the −L option is in effect, the status-change time of the file it points to is always used.

−ctime *n*
File's status was last changed *n**24 hours ago. See the comments for **−atime** to understand how rounding affects the interpretation of file status change times.

−empty File is empty and is either a regular file or a directory.

−false Always false.

−fstype *type*
File is on a filesystem of type *type*. The valid filesystem types vary among different versions of Unix; an incomplete list of filesystem types that are accepted on some version of Unix or another is: ufs, 4.2, 4.3, nfs, tmp, mfs, S51K, S52K. You can use −printf with the %F directive to see the types of your filesystems.

−gid *n* File's numeric group ID is *n*.

−group *gname*
File belongs to group *gname* (numeric group ID allowed).

−ilname *pattern*
Like −lname, but the match is case insensitive. If the −L option or the −follow option is in effect, this test returns false unless the symbolic link is broken.

−iname *pattern*
Like −name, but the match is case insensitive. For example, the patterns 'fo*' and 'F??' match the file names 'Foo', 'FOO', 'foo', 'fOo', etc. In these patterns, unlike filename expansion by the shell, an initial '.' can be matched by '*'. That is, **find −name *bar** will match the file '.foobar'.

−inum *n*
File has inode number *n*. It is normally easier to use the **−samefile** test instead.

−ipath *pattern*
Behaves in the same way as −iwholename. This option is deprecated, so please do not use it.

−iregex *pattern*
Like −regex, but the match is case insensitive.

−iwholename *pattern*
Like −wholename, but the match is case insensitive.

−links *n*
File has *n* links.

−lname *pattern*
File is a symbolic link whose contents match shell pattern *pattern*. The metacharacters do not treat '/' or '.' specially. If the −L option or the −follow option is in effect, this test returns false unless the symbolic link is broken.

−mmin *n*
File's data was last modified *n* minutes ago.

−mtime *n*
File's data was last modified *n**24 hours ago. See the comments for **−atime** to understand how rounding affects the interpretation of file modification times.

−name *pattern*
Base of file name (the path with the leading directories removed) matches shell pattern *pattern*. The metacharacters ('*', '?', and '[]') match a '.' at the start of the base name (this is a change in findutils-4.2.2; see section STANDARDS CONFORMANCE below). To ignore a directory and the files under it, use −prune; see an example in the description of −wholename. Braces are not recognised as being special, despite the fact that some shells including Bash imbue braces with a special meaning in shell patterns. The filename matching is performed with the use of the **fnmatch**(3) library function.

−newer *file*
File was modified more recently than *file*. If *file* is a symbolic link and the −H option or the −L option is in effect, the modification time of the file it points to is always used.

−nouser
No user corresponds to file's numeric user ID.

−nogroup
No group corresponds to file's numeric group ID.

−path *pattern*
See −wholename. The predicate −path is also supported by HP-UX **find**.

−perm *mode*
File's permission bits are exactly *mode* (octal or symbolic). Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex mode string. For example '−perm g=w' will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set). It is more likely that you will want to use the '+' or '-' forms, for example '−perm -g=w', which matches any file with group write permission. See the **EXAMPLES** section for some illustrative examples.

−perm −*mode*
All of the permission bits *mode* are set for the file. Symbolic modes are accepted in this form, and this is usually the way in which would want to use them. You must specify 'u', 'g' or 'o' if you use a symbolic mode. See the **EXAMPLES** section for some illustrative examples.

−perm +*mode*
Any of the permission bits *mode* are set for the file. Symbolic modes are accepted in this form. You must specify 'u', 'g' or 'o' if you use a symbolic mode. See the **EXAMPLES** section for some illustrative examples.

−regex *pattern*
File name matches regular expression *pattern*. This is a match on the whole path, not a search. For example, to match a file named './fubar3', you can use the regular expression '.*bar.' or '.*b.*3', but not 'f.*r3'. The regular expressions understood by **find** follow the conventions for the **re_match** system library function where this is present (i.e. on systems using the GNU C Library). On other systems, the implementation within Gnulib is used; by default, Gnulib provides ''basic'' regular expressions.

−samefile *name*
File refers to the same inode as *name*. When -L is in effect, this can include symbolic links.

−size *n*[cwbkMG]
File uses *n* units of space. The following suffixes can be used:

'b' for 512-byte blocks (this is the default if no suffix is used)

'c' for bytes

'w' for two-byte words

'k' for Kilobytes (units of 1024 bytes)

'M' for Megabytes (units of 1048576 bytes)

'G' for Gigabytes (units of 1073741824 bytes)

The size does not count indirect blocks, but it does count blocks in sparse files that are not actually allocated. Bear in mind that the '%k' and '%b' format specifiers of −printf handle sparse files differently. The 'b' suffix always denotes 512-byte blocks and never 1 Kilobyte blocks, which is different to the behaviour of −ls.

−true Always true.

−type *c* File is of type *c*:

b block (buffered) special

c character (unbuffered) special

d directory

p named pipe (FIFO)

f regular file

l symbolic link (never true if the −L option or the −follow option is in effect, unless the symbolic link is broken).

s socket

D door (Solaris)

−uid *n* File's numeric user ID is *n*.

−used *n*
File was last accessed *n* days after its status was last changed.

−user *uname*
File is owned by user *uname* (numeric user ID allowed).

−wholename *pattern*
File name matches shell pattern *pattern*. The metacharacters do not treat '/' or '.' specially; so, for example,

find . −wholename './sr*sc'

will print an entry for a directory called './src/misc' (if one exists). To ignore a whole directory tree, use −prune rather than checking every file in the tree. For example, to skip the directory 'src/emacs' and all files and directories under it, and print the names of the other files found, do something like this:

find . −wholename './src/emacs' −prune −o −print

−xtype *c*
The same as −type unless the file is a symbolic link. For symbolic links: if the −H or −P option was specified, true if the file is a link to a file of type *c*; if the −L option has been given, true if *c* is 'l'. In other words, for symbolic links, −xtype checks the type of the file that −type does not check.

ACTIONS

−delete Delete files; true if removal succeeded. If the removal failed, an error message is issued.

−exec *command* ;
Execute *command*; true if 0 status is returned. All following arguments to **find** are taken to be arguments to the command until an argument consisting of ';' is encountered. The string '{}' is replaced by the current file name being processed everywhere it occurs in the arguments to the command, not just in arguments where it is alone, as in some versions of **find**. Both of these constructions might need to be escaped (with a '\') or quoted to protect them from expansion by the shell. See the **EXAMPLES** section for examples of the use of the '−exec' option. The specified command is run once for each matched file. The command is executed in the starting directory. There are unavoidable security problems surrounding use of the -exec option; you should use the -execdir option instead.

−exec *command* {} +
This variant of the -exec option runs the specified command on the selected files, but the command line is built by appending each selected file name at the end; the total number of invocations of the command will be much less than the number of matched files. The command line is built in much the same way that **xargs** builds its command lines. Only one instance of '{}' is allowed within the command. The command is executed in the starting directory.

−execdir *command* ;

−execdir *command* {} +
Like -exec, but the specified command is run from the subdirectory containing the matched file, which is not normally the directory in which you started **find**. This a much more secure method for invoking commands, as it avoids race conditions during resolution of the paths to the matched files. As with the -exec option, the '+' form of -execdir will build a command line to process more than one matched file, but any given invocation of *command* will only list files that exist in the same subdirectory. If you use this option, you must ensure that your **$PATH** environment variable does not reference the current directory; otherwise, an attacker can run any commands they like by leaving an appropriately-named file in a directory in which you will run −execdir.

−fls *file* True; like −ls but write to *file* like −fprint. The output file is always created, even if the predicate is never matched.

−fprint *file*
True; print the full file name into file *file*. If *file* does not exist when **find** is run, it is created; if it does exist, it is truncated. The file names ‘‘/dev/stdout’’ and ‘‘/dev/stderr’’ are handled specially; they refer to the standard output and standard error output, respectively. The output file is always created, even if the predicate is never matched.

−fprint0 *file*
True; like −print0 but write to *file* like −fprint. The output file is always created, even if the predicate is never matched.

−fprintf *file format*
True; like −printf but write to *file* like −fprint. The output file is always created, even if the predicate is never matched.

−ok *command* ;
Like −exec but ask the user first (on the standard input); if the response does not start with ‘y’ or ‘Y’, do not run the command, and return false.

−print True; print the full file name on the standard output, followed by a newline. If you are piping the output of **find** into another program and there is the faintest possibility that the files which you are searching for might contain a newline, then you should seriously consider using the ‘−print0’ option instead of ‘−print’.

−okdir *command* ;
Like −execdir but ask the user first (on the standard input); if the response does not start with ‘y’ or ‘Y’, do not run the command, and return false.

−print0 True; print the full file name on the standard output, followed by a null character (instead of the newline character that ‘−print’ uses). This allows file names that contain newlines or other types of white space to be correctly interpreted by programs that process the **find** output. This option corresponds to the ‘−0’ option of **xargs**.

−printf *format*
True; print *format* on the standard output, interpreting ‘\’ escapes and ‘%’ directives. Field widths and precisions can be specified as with the ‘printf’ C function. Please note that many of the fields are printed as %s rather than %d, and this may mean that flags don’t work as you might expect. This also means that the ‘−’ flag does work (it forces fields to be left-aligned). Unlike −print, −printf does not add a newline at the end of the string. The escapes and directives are:

\a Alarm bell.

\b Backspace.

\c Stop printing from this format immediately and flush the output.

\f Form feed.

\n Newline.

\r Carriage return.

\t Horizontal tab.

\v Vertical tab.

\ ASCII NUL.

\\ A literal backslash (‘\’).

\NNN The character whose ASCII code is NNN (octal).

A ‘\’ character followed by any other character is treated as an ordinary character, so they both are printed.

%% A literal percent sign.

%a File’s last access time in the format returned by the C ‘ctime’ function.

%A*k* File's last access time in the format specified by *k*, which is either '@' or a directive for the C 'strftime' function. The possible values for *k* are listed below; some of them might not be available on all systems, due to differences in 'strftime' between systems.

@ seconds since Jan. 1, 1970, 00:00 GMT.

Time fields:

H hour (00..23)

I hour (01..12)

k hour (0..23)

l hour (1..12)

M minute (00..59)

p locale's AM or PM

r time, 12-hour (hh:mm:ss [AP]M)

S second (00..61)

T time, 24-hour (hh:mm:ss)

+ Date and time, separated by '+', for example '2004−04−28+22:22:05'. The time is given in the current timezone (which may be affected by setting the TZ environment variable). This is a GNU extension.

X locale's time representation (H:M:S)

Z time zone (e.g., EDT), or nothing if no time zone is determinable

Date fields:

a locale's abbreviated weekday name (Sun..Sat)

A locale's full weekday name, variable length (Sunday..Saturday)

b locale's abbreviated month name (Jan..Dec)

B locale's full month name, variable length (January..December)

c locale's date and time (Sat Nov 04 12:02:33 EST 1989)

d day of month (01..31)

D date (mm/dd/yy)

h same as b

j day of year (001..366)

m month (01..12)

U week number of year with Sunday as first day of week (00..53)

w day of week (0..6)

W week number of year with Monday as first day of week (00..53)

x locale's date representation (mm/dd/yy)

y last two digits of year (00..99)

Y year (1970...)

%b File's size in 512-byte blocks (rounded up).

%c File's last status change time in the format returned by the C 'ctime' function.

%C*k* File's last status change time in the format specified by *k*, which is the same as for %A.

%d File's depth in the directory tree; 0 means the file is a command line argument.

%D The device number on which the file exists (the st_dev field of struct stat), in decimal.

%f File's name with any leading directories removed (only the last element).

%F Type of the filesystem the file is on; this value can be used for −fstype.

%g File's group name, or numeric group ID if the group has no name.

%G File's numeric group ID.

%h Leading directories of file's name (all but the last element). If the file name contains no slashes (since it is in the current directory) the %h specifier expands to ".".

%H Command line argument under which file was found.

%i File's inode number (in decimal).

%k The amount of disk space used for this file in 1K blocks (rounded up). This is different from %s/1024 if the file is a sparse file.

%l Object of symbolic link (empty string if file is not a symbolic link).

%m File's permission bits (in octal). This option uses the 'traditional' numbers which most Unix implementations use, but if your particular implementation uses an unusual ordering of octal permissions bits, you will see a difference between the actual value of the file's mode and the output of %m. Normally you will want to have a leading zero on this number, and to do this, you should use the **#** flag (as in, for example, '%#m').

%n Number of hard links to file.

%p File's name.

%P File's name with the name of the command line argument under which it was found removed.

%s File's size in bytes.

%t File's last modification time in the format returned by the C 'ctime' function.

%T*k* File's last modification time in the format specified by *k*, which is the same as for %A.

%u File's user name, or numeric user ID if the user has no name.

%U File's numeric user ID.

%y File's type (like in ls −l), U=unknown type (shouldn't happen)

%Y File's type (like %y), plus follow symlinks: L=loop, N=nonexistent

A '%' character followed by any other character is discarded (but the other character is printed).

The %m and %d directives support the **#** , **0** and **+** flags, but the other directives do not, even if they print numbers. Numeric directives that do not support these flags include **G**, **U**, **b**, **D**, **k** and **n**. The '−' format flag is supported and changes the alignment of a field from right-justified (which is the default) to left-justified.

−prune If −depth is not given, true; if the file is a directory, do not descend into it.
If −depth is given, false; no effect.

−quit Exit immediately. No child proceses will be left running, but no more paths specified on the command line will be processed. For example, **find /tmp/foo /tmp/bar −print −quit** will print only **/tmp/foo**. Any command lines which have been built up with **−execdir ... {} +** will be invoked before **find** exits. The exit status may or may not be zero, depending on whether an error has already occurred.

−ls True; list current file in 'ls −dils' format on standard output. The block counts are of 1K blocks, unless the environment variable POSIXLY_CORRECT is set, in which case 512-byte

blocks are used.

OPERATORS

Listed in order of decreasing precedence:

(*expr*) Force precedence.

! *expr* True if *expr* is false.

−not *expr*
Same as ! *expr*, but not POSIX compliant.

expr1 expr2
Two expressions in a row are taken to be joined with an implied "and"; *expr2* is not evaluated if *expr1* is false.

expr1 −a *expr2*
Same as *expr1 expr2*.

expr1 −and *expr2*
Same as *expr1 expr2*, but not POSIX compliant.

expr1 −o *expr2*
Or; *expr2* is not evaluated if *expr1* is true.

expr1 −or *expr2*
Same as *expr1* −o *expr2*, but not POSIX compliant.

expr1 , *expr2*
List; both *expr1* and *expr2* are always evaluated. The value of *expr1* is discarded; the value of the list is the value of *expr2*. The comma operator can be useful for searching for several different types of thing, but traversing the filesystem hierarchy only once. The **−fprintf** action can be used to list the various matched items into several different output files.

STANDARDS CONFORMANCE

The following options are specified in the POSIX standard (IEEE Std 1003.1, 2003 Edition):

−H This option is supported.

−L This option is supported.

−name This option is supported, but POSIX conformance depends on the POSIX conformance of the system's **fnmatch**(3) library function. As of findutils-4.2.2, shell metacharacters ('*'. '?' or '[]' for example) will match a leading '.', because IEEE PASC interpretation 126 requires this. This is a change from previous versions of findutils.

−type Supported. POSIX specifies 'b', 'c', 'd', 'l', 'p', 'f' and 's'. GNU find also supports 'D', representing a Door, where the OS provides these.

−ok Supported. Interpretation of the response is not locale-dependent (see ENVIRONMENT VARIABLES).

−newer Supported. If the file specified is a symbolic link, it is always dereferenced. This is a change from previous behaviour, which used to take the relevant time from the symbolic link; see the HISTORY section below.

Other predicates
The predicates '−atime', '−ctime', '−depth', '−group', '−links', '−mtime', '−nogroup', '−nouser', '−perm', '−print', '−prune', '−size', '−user' and '−xdev', are all supported.

The POSIX standard specifies parentheses '(', ')', negation '!' and the 'and' and 'or' operators ('−a', '−o').

All other options, predicates, expressions and so forth are extensions beyond the POSIX standard.

Many of these extensions are not unique to GNU find, however.

The POSIX standard requires that

> The **find** utility shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file encountered. When it detects an infinite loop, find shall write a diagnostic message to standard error and shall either recover its position in the hierarchy or terminate.

The link count of directories which contain entries which are hard links to an ancestor will often be lower than they otherwise should be. This can mean that GNU find will sometimes optimise away the visiting of a subdirectory which is actually a link to an ancestor. Since **find** does not actually enter such a subdirectory, it is allowed to avoid emitting a diagnostic message. Although this behaviour may be somewhat confusing, it is unlikely that anybody actually depends on this behaviour. If the leaf optimisation has been turned off with **-noleaf**, the directory entry will always be examined and the diagnostic message will be issued where it is appropriate. Symbolic links cannot be used to create filesystem cycles as such, but if the −L option or the −follow option is in use, a diagnostic message is issued when **find** encounters a loop of symbolic links. As with loops containing hard links, the leaf optimisation will often mean that **find** knows that it doesn't need to call *stat()* or *chdir()* on the symbolic link, so this diagnostic is frequently not necessary.

The −d option is supported for comatibility with various BSD systems, but you should use the POSIX-compliant predicate −depth instead.

ENVIRONMENT VARIABLES

LANG Provides a default value for the internationalization variables that are unset or null.

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

LC_COLLATE
> The POSIX standard specifies that this variable affects the pattern matching to be used for the '−name' option. GNU find uses the **fnmatch**(3) library function, and so support for 'LC_COLLATE' depends on the system library.
>
> POSIX also specifies that the 'LC_COLLATE' environment variable affects the interpretation of the user's response to the query issued by '−ok', but this is not the case for GNU find.

LC_CTYPE
> This variable affects the treatment of character classes used with the '−name' option, if the system's **fnmatch**(3) library function supports this. It has no effect on the behaviour of the '−ok' expression.

LC_MESSAGES
> Determines the locale to be used for internationalised messages.

NLSPATH
> Determines the location of the internationalisation message catalogues.

PATH Affects the directores which are searched to find the executables invoked by '−exec' and '−ok'.

POSIXLY_CORRECT
> Determines the block size used by '-ls'.

TZ Affects the time zone used for some of the time-related format directives of -printf and -fprintf.

EXAMPLES

find /tmp −name core −type f −print | xargs /bin/rm -f

Find files named **core** in or below the directory **/tmp** and delete them. Note that this will work incorrectly if there are any filenames containing newlines, single or double quotes, or spaces.

find /tmp −name core −type f −print0 | xargs −0 /bin/rm -f

Find files named **core** in or below the directory **/tmp** and delete them, processing filenames in such a

way that file or directory names containing single or double quotes, spaces or newlines are correctly handled. The **−name** test comes before the **−type** test in order to avoid having to call **stat(2)** on every file.

find . −type f −exec file '{}' \;

Runs 'file' on every file in or below the current directory. Notice that the braces are enclosed in single quote marks to protect them from interpretation as shell script punctuation. The semicolon is similarly protected by the use of a backslash, though ';' could have been used in that case also.

**find / (−perm +4000 −fprintf /root/suid.txt '%#m %u %p\n') , **
(−size +100M −fprintf /root/big.txt '%−10s %p\n')

Traverse the filesystem just once, listing setuid files and directories into **/root/suid.txt** and large files into **/root/big/txt**.

find $HOME -mtime 0

Search for files in your home directory which have been modified in the last twenty-four hours. This command works this way because the time since each file was last accessed is divided by 24 hours and any remainder is discarded. That means that to match **-atime 0**, a file will have to have a modification in the past which is less than 24 hours ago.

find . -perm 664

Search for files which have read and write permission for their owner, and group, but which the rest of the world can read but not write to. Files which meet these criteria but have other permissions bits set (for example if someone can execute the file) will not be matched.

find . -perm -664

Search for files which have read and write permission for their owner, and group, but which the rest of the world can read but not write to, without regard to the presence of any extra permission bits (for example the executable bit). This will match a file which has mode 0777, for example.

find . -perm +222

Search for files which are writeable by somebody (their owner, or their group, or anybody else).

find . -perm +022
find . -perm +g+w,o+w
find . -perm +g=w,o=w

All three of these commands do the same thing, but the first one uses the octal representation of the file mode, and the other two use the symbolic form. These commands all search for files which are writeable by either their owner or their group. The files don't have to be writeable by both the owner and group to be matched; either will do.

find . -perm -022
find . -perm -g+w,o+w

Both these commands do the same thing; search for files which are writeable by both their owner and their group.

EXIT STATUS

find exits with status 0 if all files are processed successfully, greater than 0 if errors occur. This is deliberately a very broad description, but if the return value is non-zero, you should not rely on the correctness of the results of **find**.

SEE ALSO

locate(1), **locatedb**(5), **updatedb**(1), **xargs**(1), **fnmatch**(3), **regex**(7), **stat**(2), **lstat**(2), **ls**(1), **printf**(3), **strftime**(3), **ctime**(3), **Finding Files** (on-line in Info, or printed),

HISTORY

As of findutils-4.2.2, shell metacharacters ('*'. '?' or '[]' for example) used in filename patterns will match a leading '.', because IEEE POSIX interpretation 126 requires this.

BUGS

There are security problems inherent in the behaviour that the POSIX standard specifies for **find**, which therefore cannot be fixed. For example, the -exec action is inherently insecure, and -execdir should be used instead. Please see **Finding Files** for more information.

The best way to report a bug is to use the form at http://savannah.gnu.org/bugs/?group=findutils. The reason for this is that you will then be able to track progress in fixing the problem. Other comments about **find**(1) and about the findutils package in general can be sent to the *bug−findutils* mailing list. To join the list, send email to *bug−findutils−request@gnu.org*.

NAME

locate − list files in databases that match a pattern

SYNOPSIS

locate [−d path | −−database=path] [−e | −−existing] [−i | −−ignore-case] [−0 | −−null] [−c | −−count] [−w | −−wholename] [−b | −−basename] [−l N | −−limit=N] [−S | −−statistics] [−r | −−regex] [−P | −H | −−nofollow] [−L | −−follow] [−−version] [−−help] pattern...

DESCRIPTION

This manual page documents the GNU version of **locate**. For each given pattern, **locate** searches one or more databases of file names and displays the file names that contain the pattern. Patterns can contain shell-style metacharacters: '*', '?', and '[]'. The metacharacters do not treat '/' or '.' specially. Therefore, a pattern 'foo*bar' can match a file name that contains 'foo3/bar', and a pattern '*duck*' can match a file name that contains 'lake/.ducky'. Patterns that contain metacharacters should be quoted to protect them from expansion by the shell.

If a pattern is a plain string — it contains no metacharacters — **locate** displays all file names in the database that contain that string anywhere. If a pattern does contain metacharacters, **locate** only displays file names that match the pattern exactly. As a result, patterns that contain metacharacters should usually begin with a '*', and will most often end with one as well. The exceptions are patterns that are intended to explicitly match the beginning or end of a file name.

The file name databases contain lists of files that were on the system when the databases were last updated. The system administrator can choose the file name of the default database, the frequency with which the databases are updated, and the directories for which they contain entries; see **updatedb**(1).

OPTIONS

−c, −−count

Instead of printing the matched filenames, just print the total number of matches we found.

−d path, −−database=path

Instead of searching the default file name database, search the file name databases in *path*, which is a colon-separated list of database file names. You can also use the environment variable **LOCATE_PATH** to set the list of database files to search. The option overrides the environment variable if both are used. Empty elements in the path are taken to be synonyms for the file name of the default database.

The file name database format changed starting with GNU **find** and **locate** version 4.0 to allow machines with diffent byte orderings to share the databases. This version of **locate** can automatically recognize and read databases produced for older versions of GNU **locate** or Unix versions of **locate** or **find**. Support for the old locate database format will be discontinued in a future release.

−e, −−existing

Only print out such names that currently exist (instead of such names that existed when the database was created). Note that this may slow down the program a lot, if there are many matches in the database. If you are using this option within a program, please note that it is possible for the file to be deleted after **locate** has checked that it exists, but before you use it.

−L, −−follow

If testing for the existence of files (with the −e option), omit broken symbolic links. This is the default.

−P, −H, −−nofollow

If testing for the existence of files (with the −e option), treat broken symbolic links count as if they were exiting files. The −H form of this option is provided purely for similarity with **find**; the use of −P is recommended over −H.

−i, −−ignore-case

Ignore case distinctions in both the pattern and the file names.

−l N, −−limit=N

Limit the number of matches to N. If a limit is set via this option, the number of results printed for the −c option will never be larger than this number.

−m, −−mmap
Accepted but does nothing, for compatibility with BSD **locate**.

−0, −−null
Use ASCII NUL as a separator, instead of newline.

−w, −−wholename
Match against the whole name of the file as listed in the database. This is the default.

−b, −−basename
Results are considered to match if the pattern specified matches the final component of the name of a file as listed in the database. This final component is usually referred to as the 'base name'.

−r, −−regex
The pattern specified on the command line is understood to be a POSIX exteded regular expression, as opposed to a glob pattern. Filenames whose full paths match the specified regular expression are printed (or, in the case of the -c option, counted). If you wish to anchor your regular expression at the ends of the full path name, then as is usual with regular expressions, you should use the characters ^ and $ to signify this. Newline is not considered to be special.

−s, −−stdio
Accepted but does nothing, for compatibility with BSD **locate**.

−S, −−statistics
Print various statistics about each locate database and then exit without performing a search. Any patterns given on the command line are ignored. For compatibility with BSD, −S is accepted as a synonym for −−statistics.

−−help Print a summary of the options to **locate** and exit.

−−version
Print the version number of **locate** and exit.

ENVIRONMENT

LOCATE_PATH
Colon-separated list of databases to search. If the value has a leading or trailing colon, or has two colons in a row, you may get results that vary between different versions of **locate**.

SEE ALSO

find(1), **locatedb**(5), **updatedb**(1), **xargs**(1), **glob**(3),egex(7), **Finding Files** (on-line in Info, or printed)

BUGS

The locate database correctly handles filenames containing newlines, but only if the system's sort command has a working *−z* option. If you suspect that **locate** may need to return filenames containing newlines, consider using its *−−null* option.

The best way to report a bug is to use the form at http://savannah.gnu.org/bugs/?group=findutils. The reason for this is that you will then be able to track progress in fixing the problem. Other comments about **locate**(1) and about the findutils package in general can be sent to the *bug-findutils* mailing list. To join the list, send email to *bug-findutils-request@gnu.org*.

NAME

updatedb − update a file name database

SYNOPSIS

updatedb [*options***]**

DESCRIPTION

This manual page documents the GNU version of **updatedb**, which updates file name databases used by GNU **locate**. The file name databases contain lists of files that were in particular directory trees when the databases were last updated. The file name of the default database is determined when **locate** and **updatedb** are configured and installed. The frequency with which the databases are updated and the directories for which they contain entries depend on how often **updatedb** is run, and with which arguments.

In networked environments, it often makes sense to build a database at the root of each filesystem, containing the entries for that filesystem. **updatedb** is then run for each filesystem on the fileserver where that filesystem is on a local disk, to prevent thrashing the network. Users can select which databases **locate** searches using an environment variable or command line option; see **locate**(1). Databases can not be concatenated together.

The file name database format changed starting with GNU **find** and **locate** version 4.0 to allow machines with diffent byte orderings to share the databases. The new GNU **locate** can read both the old and new database formats. However, old versions of **locate** and **find** produce incorrect results if given a new-format database.

OPTIONS

−−findoptions='*−option1 −option2...***'**
Global options to pass on to **find**. The environment variable **FINDOPTIONS** also sets this value. Default is none.

−−localpaths='*path1 path2...***'**
Non-network directories to put in the database. Default is /.

−−netpaths='*path1 path2...***'**
Network (NFS, AFS, RFS, etc.) directories to put in the database. The environment variable **NETPATHS** also sets this value. Default is none.

−−prunepaths='*path1 path2...***'**
Directories to not put in the database, which would otherwise be. The environment variable **PRUNEPATHS** also sets this value. Default is /tmp /usr/tmp /var/tmp /afs.

−−prunefs='*path...***'**
File systems to not put in the database, which would otherwise be. Note that files are pruned when a file system is reached; Any file system mounted under an undesired file system will be ignored. The environment variable **PRUNEFS** also sets this value. Default is *nfs NFS proc*.

−−output=*dbfile*
The database file to build. Default is system-dependent. In Debian GNU/Linux, the default is /var/cache/locate/locatedb.

−−localuser=*user*
The user to search non-network directories as, using **su**(1). Default is to search the non-network directories as the current user. You can also use the environment variable **LOCALUSER** to set this user.

−−netuser=*user*
The user to search network directories as, using **su**(1). Default is **daemon**. You can also use the environment variable **NETUSER** to set this user.

−−old−format
Create the database in the old format instead of the new one.

−−version
Print the version number of **updatedb** and exit.

−−*help* Print a summary of the options to **updatedb** and exit.

SEE ALSO

find(1), **locate**(1), **locatedb**(5), **xargs**(1) **Finding Files** (on-line in Info, or printed)

BUGS

The **updatedb** program correctly handles filenames containing newlines, but only if the system's sort command has a working −*z* option. If you suspect that **locate** may need to return filenames containing newlines, consider using its −−*null* option.

The best way to report a bug is to use the form at http://savannah.gnu.org/bugs/?group=findutils. The reason for this is that you will then be able to track progress in fixing the problem. Other comments about **updatedb**(1) and about the findutils package in general can be sent to the *bug-findutils* mailing list. To join the list, send email to *bug-findutils-request@gnu.org*.

NAME

xargs − build and execute command lines from standard input

SYNOPSIS

xargs [−0prtx] [−E[eof-str]] [−e[eof-str]] [−−eof[=eof-str]] [−−null] [−I[replace-str]] [−i[replace-str]] [−−replace[=replace-str]] [−l[max-lines]] [−L[max-lines]] [−−max−lines[=max-lines]] [−n max-args] [−−max−args=max-args] [−s max-chars] [−−max−chars=max-chars] [−P max-procs] [−−max−procs=max-procs] [−−interactive] [−−verbose] [−−exit] [−−no−run−if−empty] [−−arg−file=file] [−−version] [−−help] [command [initial-arguments]]

DESCRIPTION

This manual page documents the GNU version of **xargs**. **xargs** reads items from the standard input, delimited by blanks (which can be protected with double or single quotes or a backslash) or newlines, and executes the *command* (default is /bin/echo) one or more times with any *initial-arguments* followed by items read from standard input. Blank lines on the standard input are ignored.

Because Unix filenames can contain blanks and newlines, this default behaviour is often problematic; filenames containing blanks and/or newlines are incorrectly processed by **xargs**. In these situations it is better to use the '−0' option, which prevents such problems. When using this option you will need to ensure that the program which produces the input for **xargs** also uses a null character as a separator. If that program is GNU **find** for example, the '−print0' option does this for you.

If any invocation of the command exits with a status of 255, **xargs** will stop immediately without reading any further input. An error message is issued on stderr when this happens.

OPTIONS

−−arg−file=file, −a file
: Read items from *file* instead of standard input. If you use this option, stdin remains unchanged when commands are run. Otherwise, stdin is redirected from */dev/null*.

−−null, −0
: Input items are terminated by a null character instead of by whitespace, and the quotes and backslash are not special (every character is taken literally). Disables the end of file string, which is treated like any other argument. Useful when input items might contain white space, quote marks, or backslashes. The GNU find −print0 option produces input suitable for this mode.

−−eof[=eof-str], −E[eof-str]
: Set the end of file string to *eof-str*. If the end of file string occurs as a line of input, the rest of the input is ignored. If *eof-str* is omitted, there is no end of file string. If this option is not given, no end of file string is used.

−e[eof-str]
: This option is a synonym for the '−E' option. Use '−E' instead, because it is POSIX compliant while this option is not.

−−help Print a summary of the options to **xargs** and exit.

−−replace[=replace-str], −i[replace-str]
: Replace occurences of *replace-str* in the initial-arguments with names read from standard input. Also, unquoted blanks do not terminate input items; instead the separator is the newline character. If *replace-str* is omitted, it defaults to "{}" (like for 'find −exec'). Implies *−x* and *−l 1*.

−−max−lines[=max-lines], −L[max-lines]
: Use at most *max-lines* nonblank input lines per command line; *max-lines* defaults to 1 if omitted. Trailing blanks cause an input line to be logically continued on the next input line. Implies *−x*.

−l[max-lines]
: Deprecated; non-POSIX-compliant synonym for the *−L* option.

−−max−args=max-args, −n max-args
Use at most *max-args* arguments per command line. Fewer than *max-args* arguments will be used if the size (see the −s option) is exceeded, unless the −x option is given, in which case **xargs** will exit.

−−interactive, −p
Prompt the user about whether to run each command line and read a line from the terminal. Only run the command line if the response starts with ‘y’ or ‘Y’. Implies *−t*.

−−no−run−if−empty, −r
If the standard input does not contain any nonblanks, do not run the command. Normally, the command is run once even if there is no input. This option is a GNU extension.

−−max−chars=max-chars, −s max-chars
Use at most *max-chars* characters per command line, including the command and initial-arguments and the terminating nulls at the ends of the argument strings. The default is 131072 characters, not including the size of the environment variables (which are provided for separately so that it doesn’t matter if your environment variables take up more than 131072 bytes). The operating system places limits on the values that you can usefully specify, and if you exceed these a warning message is printed and the value actually used is set to the appropriate upper or lower limit.

−−verbose, −t
Print the command line on the standard error output before executing it.

−−version
Print the version number of **xargs** and exit.

−−exit, −x
Exit if the size (see the *−s* option) is exceeded.

−−max−procs=max-procs, −P max-procs
Run up to *max-procs* processes at a time; the default is 1. If *max-procs* is 0, **xargs** will run as many processes as possible at a time. Use the *−n* option with *−P*; otherwise chances are that only one exec will be done.

EXAMPLES

find /tmp −name core −type f −print | xargs /bin/rm −f

Find files named **core** in or below the directory **/tmp** and delete them. Note that this will work incorrectly if there are any filenames containing newlines or spaces.

find /tmp −name core −type f −print0 | xargs −0 /bin/rm −f

Find files named **core** in or below the directory **/tmp** and delete them, processing filenames in such a way that file or directory names containing spaces or newlines are correctly handled.

cut −d: −f1 < /etc/passwd | sort | xargs echo

Generates a compact listing of all the users on the system.

EXIT STATUS

xargs exits with the following status:
0 if it succeeds
123 if any invocation of the command exited with status 1-125
124 if the command exited with status 255
125 if the command is killed by a signal
126 if the command cannot be run
127 if the command is not found
1 if some other error occurred.

Exit codes greater than 128 are used by the shell to indicate that a program died due to a fatal signal.

STANDARDS CONFORMANCE

As of GNU xargs version 4.2.9, the default behaviour of **xargs** is not to have a logical end-of-file marker. POSIX (IEEE Std 1003.1, 2004 Edition) allows this.

SEE ALSO

find(1), **locate**(1), **locatedb**(5), **updatedb**(1), **Finding Files** (on-line in Info, or printed)

BUGS

It is not possible for **xargs** to be used securely, since there will always be a time gap between the production of the list of input files and their use in the commands that **xargs** issues. If other users have access to the system, they can manipulate the filesystem during this time window to force the action of the commands **xargs** runs to apply to files that you didn't intend. For a more detailed discussion of this and related problems, please refer to the ''Security Considerations'' chapter in the findutils Texinfo documentation. The -execdir option of **find** can often be used as a more secure alternative.

When you use the −i option, each line read from the input is buffered internally. This means that there is an upper limit on the length of input line that **xargs** will accept when used with the −i option. To work around this limitation, you can use the −s option to increase the amount of buffer space that **xargs** uses, and you can also use an extra invocation of **xargs** to ensure that very long lines do not occur. For example:

somecommand | xargs −s 50000 echo | xargs −i −s 100000 rm '{}'

Here, the first invocation of **xargs** has no input line length limit because it doesn't use the −i option. The second invocation of **xargs** does have such a limit, but we have ensured that the it never encounters a line which is longer than it can handle. This is not an ideal solution. Instead, the −i option should not impose a line length limit, which is why this discussion appears in the BUGS section. The problem doesn't occur with the output of **find**(1) because it emits just one filename per line.

The best way to report a bug is to use the form at http://savannah.gnu.org/bugs/?group=findutils. The reason for this is that you will then be able to track progress in fixing the problem. Other comments about **xargs**(1) and about the findutils package in general can be sent to the *bug−findutils* mailing list. To join the list, send email to *bug−findutils−request@gnu.org*.

NAME

locatedb − front-compressed file name database

DESCRIPTION

This manual page documents the format of file name databases for the GNU version of **locate**. The file name databases contain lists of files that were in particular directory trees when the databases were last updated.

There can be multiple databases. Users can select which databases **locate** searches using an environment variable or command line option; see **locate**(1). The system administrator can choose the file name of the default database, the frequency with which the databases are updated, and the directories for which they contain entries. Normally, file name databases are updated by running the **updatedb** program periodically, typically nightly; see **updatedb**(1).

updatedb runs a program called **frcode** to compress the list of file names using front-compression, which reduces the database size by a factor of 4 to 5. Front-compression (also known as incremental encoding) works as follows.

The database entries are a sorted list (case-insensitively, for users' convenience). Since the list is sorted, each entry is likely to share a prefix (initial string) with the previous entry. Each database entry begins with an offset-differential count byte, which is the additional number of characters of prefix of the preceding entry to use beyond the number that the preceding entry is using of its predecessor. (The counts can be negative.) Following the count is a null-terminated ASCII remainder — the part of the name that follows the shared prefix.

If the offset-differential count is larger than can be stored in a byte (+/−127), the byte has the value 0x80 and the count follows in a 2-byte word, with the high byte first (network byte order).

Every database begins with a dummy entry for a file called 'LOCATE02', which **locate** checks for to ensure that the database file has the correct format; it ignores the entry in doing the search.

Databases can not be concatenated together, even if the first (dummy) entry is trimmed from all but the first database. This is because the offset-differential count in the first entry of the second and following databases will be wrong.

There is also an old database format, used by Unix **locate** and **find** programs and earlier releases of the GNU ones. **updatedb** runs programs called **bigram** and **code** to produce old-format databases. The old format differs from the above description in the following ways. Instead of each entry starting with an offset-differential count byte and ending with a null, byte values from 0 through 28 indicate offset-differential counts from -14 through 14. The byte value indicating that a long offset-differential count follows is 0x1e (30), not 0x80. The long counts are stored in host byte order, which is not necessarily network byte order, and host integer word size, which is usually 4 bytes. They also represent a count 14 less than their value. The database lines have no termination byte; the start of the next line is indicated by its first byte having a value <= 30.

In addition, instead of starting with a dummy entry, the old database format starts with a 256 byte table containing the 128 most common bigrams in the file list. A bigram is a pair of adjacent bytes. Bytes in the database that have the high bit set are indexes (with the high bit cleared) into the bigram table. The bigram and offset-differential count coding makes these databases 20-25% smaller than the new format, but makes them not 8-bit clean. Any byte in a file name that is in the ranges used for the special codes is replaced in the database by a question mark, which not coincidentally is the shell wildcard to match a single character.

EXAMPLE

Input to **frcode**:
/usr/src
/usr/src/cmd/aardvark.c
/usr/src/cmd/armadillo.c
/usr/tmp/zoo

Length of the longest prefix of the preceding entry to share:
0 /usr/src
8 /cmd/aardvark.c

14 rmadillo.c
5 tmp/zoo

Output from **frcode**, with trailing nulls changed to newlines and count bytes made printable:
0 LOCATE02
0 /usr/src
8 /cmd/aardvark.c
6 rmadillo.c
−9 tmp/zoo

(6 = 14 − 8, and −9 = 5 − 14)

SEE ALSO

find(1), **locate**(1), **locatedb**(5), **xargs**(1) **Finding Files** (on-line in Info, or printed)

BUGS

The best way to report a bug is to use the form at http://savannah.gnu.org/bugs/?group=findutils. The reason for this is that you will then be able to track progress in fixing the problem. Other comments about **locate**(1) and about the findutils package in general can be sent to the *bug-findutils* mailing list. To join the list, send email to *bug-findutils-request@gnu.org*.