NAME

find − search for files in a directory hierarchy

SYNOPSIS

find [path...] [expression]

DESCRIPTION

This manual page documents the GNU version of **find**. **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.

The first argument that begins with '−', '(', ')', ',', or '!' is taken to be the beginning of the expression; any arguments before it are paths to search, and any arguments after it are the rest of the expression. If no paths are given, the current directory is used. If no expression is given, the expression '−print' is used.

find exits with status 0 if all files are processed successfully, greater than 0 if errors occur.

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. 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.

−daystart
: Measure times (for −amin, −atime, −cmin, −ctime, −mmin, and −mtime) from the beginning of today rather than from 24 hours ago.

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

−follow
: Dereference symbolic links. Implies −noleaf.

−help, −−help
: Print a summary of the command-line usage of **find** and exit.

−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**.

−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.

−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. −anewer is affected by −follow only if −follow comes before −anewer on the command line.

−atime *n*
File was last accessed *n**24 hours 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. −cnewer is affected by −follow only if −follow comes before −cnewer on the command line.

−ctime *n*
File's status was last changed *n**24 hours ago.

−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.

−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.

−inum *n*
File has inode number *n*.

−ipath *pattern*
Like −path, but the match is case insensitive.

−iregex *pattern*
Like −regex, 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.

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

−mtime *n*
File's data was last modified *n**24 hours ago.

−name *pattern*
Base of file name (the path with the leading directories removed) matches shell pattern *pattern*. The metacharacters ('*', '?', and '[]') do not match a '.' at the start of the base name. To ignore a directory and the files under it, use −prune; see an example in the description of −path.

−newer *file*
File was modified more recently than *file*. −newer is affected by −follow only if −follow comes before −newer on the command line.

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

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

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

find . −path './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 . −path './src/emacs' -prune -o -print

−perm *mode*
File's permission bits are exactly *mode* (octal or symbolic). Symbolic modes use mode 0 as a point of departure.

−perm −*mode*
All of the permission bits *mode* are set for the file.

−perm +*mode*
Any of the permission bits *mode* are set for the file.

−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 'b.*r3'.

−size *n*[bckw]
File uses *n* units of space. The units are 512-byte blocks by default or if 'b' follows *n*, bytes if 'c' follows *n*, kilobytes if 'k' follows *n*, or 2-byte words if 'w' follows *n*. The size does not count indirect blocks, but it does count blocks in sparse files that are not actually allocated.

−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

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).

−xtype *c*
The same as −type unless the file is a symbolic link. For symbolic links: if −follow has not been given, true if the file is a link to a file of type *c*; if −follow 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.

−context *scontext*

−-context *scontext*
(SELinux only) File has the security context *scontext*.

ACTIONS

−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. The command is executed in the starting directory.

−fls *file* True; like −ls but write to *file* like −fprint.

−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.

−fprint0 *file*
True; like −print0 but write to *file* like −fprint.

−fprintf *file format*
True; like −printf but write to *file* like −fprint.

−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.

−print0 True; print the full file name on the standard output, followed by a null character. This allows file names that contain newlines to be correctly interpreted by programs that process the **find** output.

−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. 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.

\\ 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 specifi ed 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 fi elds:

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)

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

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

Date fi elds:

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.

%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).

%H Command line argument under which file was found.

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

%k File's size in 1K blocks (rounded up).

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

%m File's permission bits (in octal).

%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.

%Z (SELinux only) file's security context.

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

−prune If −depth is not given, true; do not descend the current directory.
If −depth is given, false; no effect.

−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*.

expr1 expr2
And (implied); *expr2* is not evaluated if *expr1* is false.

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

expr1 −and *expr2*
Same as *expr1 expr2*.

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

expr1 −or *expr2*
Same as *expr1* −o *expr2*.

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*.

SEE ALSO

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