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.

-regextype *type*
: Changes the regular expression syntax understood by **-regex** and **-iregex** tests which occur later on the command line. Currently-implemented types are emacs (this is the default), posix-awk, posix-basic, posix-egrep and posix-extended.

-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.

-xautofs
: Don't descend directories on autofs filesystems.

-xdev

Don't descend directories on other filesystems.

TESTS

Some tests, for example **-newerXY** and **-samefile**, allow comparison between the file currently being examined and some reference file specified on the command line. When these tests are used, the interpretation of the reference file is determined by the options **-H**, **-L** and **-P** and any previous **-follow**, but the reference file is only examined once, at the time the command line is parsed. If the reference file cannot be examined (for example, the **stat**(2) system call fails for it), an error message is issued, and **find** exits with a nonzero status.

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 periods ago the file was last accessed, any fractional part is ignored, so to match **-atime +1**, a file has to have been accessed 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.

-executable
: Matches files which are executable and directories which are searchable (in a file name resolution sense). This takes into account access control lists and other permissions artefacts which the **-perm** test ignores. This test makes use of the **access**(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access**(2) in the client's kernel and so cannot make use of the UID mapping information held on the server. Because this test is based only on the result of the **access**(2) system call, there is no guarantee that a file for which this test succeeds can actually be executed.

-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'. Please note that you should quote patterns as a matter of course, otherwise the shell will expand any wildcard characters in them.

-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 **-path**. 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. Don't forget to enclose the pattern in quotes in order to protect it from expansion by the shell.

-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.

-newerXY *reference*
: Compares the timestamp of the current file with *reference*. The *reference* argument is normally the name of a file (and one of its timestamps is used for the comparison) but it may also be a string describing an absolute time. *X* and *Y* are placeholders for other letters, and these letters select which time belonging to how *reference* is used for the comparison.
Some combinations are invalid; for example, it is invalid for *X* to be *t*. Some combinations are not implemented on all systems; for example *B* is not supported on all systems. If an invalid or unsupported combination of *XY* is specified, a fatal error results. Time specifications are interpreted as for the argument to the **-d** option of GNU **date**. If you try to use the birth time of a reference file, and the birth time cannot be determined, a fatal error message results. If you specify a test which refers to the birth time of files being examined, this test will fail for any files where the birth time is unknown.

   -nogroup

   No group corresponds to file's numeric group ID.

-nouser
: No user corresponds to file's numeric user 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
```

Note that the pattern match test applies to the whole file name, starting from one of the start points named on the command line. It would only make sense to use an absolute path name here if the relevant start point is also an absolute path. This means that this command will never match anything:

```
find bar -path /foo/bar/myfile -print
```

The predicate **-path** is also supported by HP-UX **find** and will be in a forthcoming version of the POSIX standard.

-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. If no permission bits in *mode* are set, this test matches any file (the idea here is to be consistent with the behaviour of **-perm -000**).

-perm +*mode*
: Deprecated, old way of searching for files with any of the permission bits in *mode* set. You should use **-perm** */mode* instead. Trying to use the '+' syntax with symbolic modes will yield surprising results. For example, '+u+x' is a valid symbolic mode (equivalent to +u,+x, i.e. 0111) and will therefore not be evaluated as **-perm +***mode* but instead as the exact mode specifier **-perm** *mode* and so it matches files with exact permissions 0111 instead of files with any execute bit set. If you found this paragraph confusing, you're not alone - just use **-perm /***mode*. This form of the **-perm** test is deprecated because the POSIX specification requires the interpretation of a leading '+' as being part of a symbolic mode, and so we switched to using '/' instead.

-readable
: Matches files which are readable. This takes into account access control lists and other permissions artefacts which the **-perm** test ignores. This test makes use of the **access**(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access**(2) in the client's kernel and so cannot make use of the UID mapping information held on the server.

-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** are by default Emacs Regular Expressions, but this can be changed with the **-regextype** option.

-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; this is never true if the **-L** option or the **-follow** option is in effect, unless the symbolic link is broken. If you want to search for symbolic links when **-L** is in effect, use **-xtype**.

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*
: See -path. This alternative is less portable than **-path**.

-writable
: Matches files which are writable. This takes into account access control lists and other permissions artefacts which the **-perm** test ignores. This test makes use of the **access**(2) system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access**(2) in the client's kernel and so cannot make use of the UID mapping information held on the server.

-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.

-context *pattern*
: (SELinux only) Security context of the file matches glob *pattern*.

ACTIONS

-delete

Delete files; true if removal succeeded. If the removal failed, an error message is issued. If **-delete** fails, **find**'s exit status will be nonzero (when it eventually exits). Use of **-delete** automatically turns on the **-depth** option.

Warnings: Don't forget that the find command line is evaluated as an expression, so putting **-delete** first will make **find** try to delete everything below the starting points you specified. When testing a **find** command line that you later intend to use with **-delete**, you should explicitly specify **-depth** in order to avoid later surprises. Because **-delete** implies **-depth**, you cannot usefully use **-prune** and **-delete** together.

-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** action; you should use the **-execdir** option instead.

-exec *command* {} +
: This variant of the **-exec** action 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** action, 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 **$PATH** environment variable does not reference '.'; otherwise, an attacker can run any commands they like by leaving an appropriately-named file in a directory in which you will run **-execdir**. The same applies to having entries in **$PATH** which are empty or which are not absolute directory names.

-fls *file*
: True; like **-ls** but write to *file* like **-fprint**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-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. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-fprint0 *file*
: True; like **-print0** but write to *file* like **-fprint**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-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. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-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. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-ok *command* ;
: Like **-exec** but ask the user first. If the user agrees, run the command. Otherwise just return false. If the command is run, its standard input is redirected from **/dev/null**.

The response to the prompt is matched against a pair of regular expressions to determine if it is an affirmative or negative response. This regular expression is obtained from the system if the 'POSIXLY_CORRECT' environment variable is set, or otherwise from **find**'s message translations. If the system has no suitable definition, **find**'s own definition will be used. In either case, the interpretation of the regular expression itself will be affected by the environment variables 'LC_CTYPE' (character classes) and 'LC_COLLATE' (character ranges and equivalence classes).

-okdir *command* ;

Like **-execdir** but ask the user first in the same way as for **-ok**. If the user does not agree, just return false. If the command is run, its standard input is redirected from **/dev/null**.

-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**. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-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.

\0

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, with fractional part.

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 (00.00 .. 61.00). There is a fractional part.

T

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

+

Date and time, separated by '+', for example '2004-04-28+22:22:05.0'. This is a GNU extension. The time is given in the current timezone (which may be affected by setting the TZ environment variable). The seconds field includes a fractional part.

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). The format is the same as for **ctime**(3) and so to preserve compatibility with that format, there is no fractional part in the seconds field.

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

The amount of disk space used for this file in

512-byte blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/512, but it can also be smaller if the file is a sparse file.

%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. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/1024, but it can also be smaller 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').

%M

File's permissions (in symbolic form, as for **ls**). This directive is supported in findutils 4.2.5 and later.

%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.

%S

File's sparseness. This is calculated as (BLOCKSIZE*st_blocks / st_size). The exact value you will get for an ordinary file of a certain length is system-dependent. However, normally sparse files will have values less than 1.0, and files which use indirect blocks may have a value which is greater than 1.0. The value used for BLOCKSIZE is system-dependent, but is usually 512 bytes. If the file size is zero, the value printed is undefined. On systems which lack support for st_blocks, a file's sparseness is assumed to be 1.0.

%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

%Z

(SELinux only) file's security context.

A '%' character followed by any other character is discarded, but the other character is printed (don't rely on this, as further format characters may be introduced). A '%' at the end of the format argument causes undefined behaviour since there is no following character. In some locales, it may hide your door keys, while in others it may remove the final page from the novel you are reading.

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.

See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-prune

True; if the file is a directory, do not descend into it. If **-depth** is given, false; no effect. Because **-delete** implies **-depth**, you cannot usefully use **-prune** and **-delete together.**

-quit

Exit immediately. No child processes 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.

UNUSUAL FILENAMES

Many of the actions of **find** result in the printing of data which is under the control of other users. This includes file names, sizes, modification times and so forth. File names are a potential problem since they can contain any character except '\0' and '/'. Unusual characters in file names can do unexpected and often undesirable things to your terminal (for example, changing the settings of your function keys on some terminals). Unusual characters are handled differently by various actions, as described below.

-print0, -fprint0

Always print the exact filename, unchanged, even if the output is going to a terminal.

-ls, -fls

Unusual characters are always escaped. White space, backslash, and double quote characters are printed using C-style escaping (for example '\f', '\"'). Other unusual characters are printed using an octal escape. Other printable characters (for **-ls** and **-fls** these are the characters between octal 041 and 0176) are printed as-is.

-printf, -fprintf

If the output is not going to a terminal, it is printed as-is. Otherwise, the result depends on which directive is in use. The directives %D, %F, %g, %G, %H, %Y, and %y expand to values which are not under control of files' owners, and so are printed as-is. The directives %a, %b, %c, %d, %i, %k, %m, %M, %n, %s, %t, %u and %U have values which are under the control of files' owners but which cannot be used to send arbitrary data to the terminal, and so these are printed as-is. The directives %f, %h, %l, %p and %P are quoted. This quoting is performed in the same way as for GNU **ls**. This is not the same quoting mechanism as the one used for **-ls** and **-fls**. If you are able to decide what format to use for the output of **find** then it is normally better to use '\0' as a terminator than to use newline, as file names can contain white space and newline characters. The setting of the 'LC_CTYPE' environment variable is used to determine which characters need to be quoted.

-print, -fprint

Quoting is handled in the same way as for **-printf** and **-fprintf**. If you are using **find** in a script or in a situation where the matched files might have arbitrary names, you should consider using **-print0** instead of **-print**.

The **-ok** and **-okdir** actions print the current filename as-is. This may change in a future release.

OPERATORS

Listed in order of decreasing precedence:

(*expr*)

Force precedence. Since parentheses are special to the shell, you will normally need to quote them. Many of the examples in this manual page use backslashes for this purpose: '\(...\)' instead of '(...)'.