NAME

find – find files

SYNOPSIS

find *path* . . . *expression*

AVAILABILITY

SUNWcsu

DESCRIPTION

The **find** command recursively descends the directory hierarchy for each *path* seeking files that match a Boolean *expression* written in the primaries given below.

find will be able to descend to arbitrary depths in a file hierarchy and will not fail due to path length limitations (unless a *path* operand specified by the application exceeds **PATH_MAX** requirements).

OPERANDS

The following operands are supported:

path a path name of a starting point in the directory hierarchy.

The first argument that starts with a –, or is a **!** or a **(**, and all subsequent arguments will be interpreted as an *expression* made up of the following primaries and operators. In the descriptions, wherever *n* is used as a primary argument, it will be interpreted as a decimal integer optionally preceded by a plus (**+**) or minus (–) sign, as follows:

+*n* more than *n*
n exactly *n*
–*n* less than *n*.

Expressions

Valid expressions are:

–**atime** *n* True if the file was accessed *n* days ago. The access time of directories in *path* is changed by **find** itself.

–**cpio** *device* Always true; write the current file on *device* in **cpio** format (5120-byte records).

–**ctime** *n* True if the file's status was changed *n* days ago.

–**depth** Always true; causes descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. This can be useful when **find** is used with **cpio**(1) to transfer files that are contained in directories without write permission.

–**exec** *command* True if the executed *command* returns a zero value as exit status. The end of *command* must be punctuated by an escaped semicolon. A command argument **{ }** is replaced by the current path name.

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

–fstype *type* True if the filesystem to which the file belongs is of type *type*.

–group *gname* True if the file belongs to the group *gname*. If *gname* is numeric and does not appear in the **/etc/group** file, it is taken as a group ID.

–inum *n* True if the file has inode number *n*.

–links *n* True if the file has *n* links.

–local True if the file system type is not a remote file system type as defined in the **/etc/dfs/fstypes** file. **nfs** is used as the default remote filesystem type if the **/etc/dfs/fstypes** file is not present.

–ls Always true; prints current path name together with its associated statistics. These include (respectively):

- inode number
- size in kilobytes (1024 bytes)
- protection mode
- number of hard links
- user
- group
- size in bytes
- modification time.

If the file is a special file the size field will instead contain the major and minor device numbers.

If the file is a symbolic link the pathname of the linked-to file is printed preceded by '→'. The format is identical to that of **ls –gilds** (see **ls**(1)).

Note: Formatting is done internally, without executing the **ls** program.

–mount Always true; restricts the search to the file system containing the directory specified. Does not list mount points to other file systems.

–mtime *n* True if the file's data was modified *n* days ago.

–name *pattern* True if *pattern* matches the current file name. Normal shell file name generation characters (see **sh**(1)) may be used. A backslash (\ ) is used as an escape character within the pattern. The pattern should be escaped or quoted when **find** is invoked from the shell.

–ncpio *device* Always true; write the current file on *device* in **cpio –c** format (5120 byte records).

–newer *file* True if the current file has been modified more recently than the argument *file*.

–nogroup True if the file belongs to a group not in the **/etc/group** file.

–nouser True if the file belongs to a user not in the **/etc/passwd** file.

–ok *command* Like **–exec** except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing **y**.

–perm [–]*mode* The *mode* argument is used to represent file mode bits. It will be identical in format to the <*symbolic*mode> operand described in **chmod**(1), and will be interpreted as follows. To start, a template will be assumed with all file mode bits cleared. An *op* symbol of:

+ will set the appropriate mode bits in the template;

– will clear the appropriate bits;

= will set the appropriate mode bits, without regard to the contents of process' file mode creation mask.

The *op* symbol of – cannot be the first character of *mode*; this avoids ambiguity with the optional leading hyphen. Since the initial mode is all bits off, there are not any symbolic modes that need to use – as the first character.

If the hyphen is omitted, the primary will evaluate as true when the file permission bits exactly match the value of the resulting template.

Otherwise, if *mode* is prefixed by a hyphen, the primary will evaluate as true if at least all the bits in the resulting template are set in the file permission bits.

–**perm** [–]*onum* True if the file permission flags exactly match the octal number *onum* (see **chmod**(1)). If *onum* is prefixed by a minus sign (–), only the bits that are set in *onum* are compared with the file permission flags, and the expression evaluates true if they match.

–**print** Always true; causes the current path name to be printed.

–**prune** Always yields true. Do not examine any directories or files in the directory structure below the *pattern* just matched. See the examples, below.

–**size** *n*[**c**] True if the file is *n* blocks long (512 bytes per block). If *n* is followed by a **c**, the size is in bytes.

–**type** *c* True if the type of the file is *c*, where *c* is **b**, **c**, **d**, **l**, **p**, or **f** for block special file, character special file, directory, symbolic link, fifo (named pipe), or plain file, respectively.

–**user** *uname* True if the file belongs to the user *uname*. If *uname* is numeric and does not appear as a login name in the **/etc/passwd** file, it is taken as a user ID.

–**xdev** Same as the –**mount** primary.

Complex Expressions

The primaries may be combined using the following operators (in order of decreasing precedence):

1) **(** *expression* **)** — True if the parenthesized expression is true (parentheses are special to the shell and must be escaped).

2) **!** *expression* — The negation of a primary (**!** is the unary *not* operator).

3) *expression* **[−a]** *expression* — Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries).

4) *expression* −**o** *expression* — Alternation of primaries (−**o** is the *or* operator).

Note: When you use **find** in conjunction with **cpio**, if you use the −**L** option with **cpio** then you must use the −**follow** expression with **find** and vice versa. Otherwise there will be undesirable results.

If no *expression* is present, −**print** will be used as the expression. Otherwise, if the given expression does not contain any of the primaries −**exec**, −**ok** or −**print**, the given expression will be effectively replaced by:

(*given_expression*) −print

The −**user**, −**group**, and −**newer** primaries each will evaluate their respective arguments only once.

EXAMPLES

The following commands are equivalent:

```
example% find .
example% find . -print
```

They both write out the entire directory hierarchy from the current directory.

Remove all files in your home directory named **a.out** or ∗**.o** that have not been accessed for a week:

```
example% find $HOME \( −name a.out −o −name '*.o' \
        −atime +7 \ −exec rm {} \;
```

Recursively print all file names in the current directory and below, but skipping SCCS directories:

```
example% find . −name SCCS −prune −o −print
```

Recursively print all file names in the current directory and below, skipping the contents of SCCS directories, but printing out the SCCS directory name:

```
example% find . −print −name SCCS −prune
```

The following command is roughly equivalent to the −**nt** extension to **test**(1):

```
example$ if [ -n "$(find file1 -prune -newer file2)" ]; then
        printf %s\\n "file1 is newer than file2"
fi
```

The descriptions of −**atime**, −**ctime**, and −**mtime** use the terminology *n* ‘‘24-hour periods’’. For example, a file accessed at 23:59 will be selected by:

example% find . -atime -1 -print

at 00:01 the next day (less than 24 hours later, not more than one day ago); the midnight boundary between days has no effect on the 24-hour calculation.

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **find**: **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 All *path* operands were traversed successfully.

>0 An error occurred.

FILES

/etc/passwd password file

/etc/group group file

/etc/dfs/fstypes file that registers distributed file system packages

SEE ALSO

chmod(1), **ls**(1), **sh**(1), **test**(1), **stat**(2), **umask**(2), **environ**(5)

WARNINGS

The following options are obsolete and will not be supported in future releases:

-cpio *device* Always true; write the current file on *device* in **cpio** format (5120-byte records).

-ncpio *device* Always true; write the current file on *device* in **cpio -c** format (5120 byte records).

NOTES

When using **find** to determine files modified within a range of time, one must use the **?time** argument *before* the −**print** argument otherwise **find** will give all files.