Find Cheat Sheet by Linux Training Academy

Find Command Format

```
find <OPTIONS> [PATH...] [EXPRESSION...]
```

Description

The options determine whether find will follow symbolic links (symlinks) as if they are the files they reference.

One or more paths can be listed as arguments to set the directories that find will search.

The expressions determine what find will search for and what actions it will take on items it finds. All path arguments must be listed before the first expression, and expression options should be listed before the test and action expressions. If no action expressions are listed, find will use -print by default.

Note that there are more expressions than those listed on this sheet, these are just the more commonly used expressions. For a full list of expressions and options, see `man find`.

Common Options

Option	Description
-P	Don't follow symlinks (default behavior)
-L	Follow symlinks (process a symlink as if it were the file or directory it points to)
-H	Only follow symlinks named in the command line arguments - other symlinks found in the search won't be followed

Common Expression Options

These options modify the way the expressions are evaluated. Note that "-help" will print a usage summary and exit without conducting a search.

Option	Description
-depth	Process directory contents before the directory itself. Enabled by default when -delete is used to ensure contents are deleted before the directory is deleted
-maxdepth *levels*	The maximum depth below the command line arguments to include in the search. A maxdepth of 0 means that only items specified in the command line arguments will be searched, not their subdirectories or directory contents
-mindepth *levels*	The minimum depth below the command line arguments to be navigated before search checks will be applied. A mindepth of 1 means that the path items in the command line arguments won't be checked, but all of their subdirectories and directory contents will be

Common Expression Tests

Note that when tests call for numeric arguments, you can either specify a specific number *n*, a value greater than the number by putting a plus in front of it as in *+n*, or a value less than the number by putting a minus sign in front of it as in *-n*.

Option	Description
-amin *n*	The file was accessed *n* minutes ago
-cmin *n*	The file status was changed *n* minutes ago (like a change in permissions)
-empty	The file or directory is empty
-executable	The file can be executed, or a directory navigated into, by the current user. Note that this test can include access via ACLs or network shares, so it doesn't just rely on file system permissions
-group *group*	The file or directory is in group *group* (either the group name or numeric ID)
-mmin *n*	The file was last modified *n* minutes ago

-name *pattern*	The file or directory's base name matches the shell pattern *pattern*. Note that you should put the pattern in quotes to prevent the shell from expanding the pattern before find can process it. To include the full path in the check, use -path
-iname *pattern*	Same as -name, but the test is case insensitive
-newer *file*	The file or directory was modified more recently than *file*
-nouser	The owner of the file or directory has a numeric user ID that isn't defined as a user on the system (e.g. isn't listed in /etc/passwd)
-path *pattern*	Like -name, but the pattern is applied to the full path plus the file or directory name. Remember to put the search pattern in quotes
-perm *mode*	The file or directory has the exact permissions specified as *mode*. Note that if symbolic representation is used, the match is still exact - "o=w" will match *only* if permissions on the item are 002 in octal
-perm -*mode*	A permission check with a dash will check to see if all of the specified permissions are included. Using symbolic representation, "o=w" in this case will match any permission set that includes it, like 777 in octal.
-perm /*mode*	A permission check with a forward slash will check to see if any of the specified permissions are set on an item. Using symbolic representation, "o=w,g=w" will match 002, 020, or 777 in octal
-readable	The file can be read by the current user. Note that this test can include access via ACLs or network shares, so it doesn't just rely on file system permissions
-regex *pattern*	Check the file or directory name (with full path) against a regular expression. Remember to put the expression in quotes to prevent the shell from expanding the search before find reads it
-size *n*	The file takes up *n* amount of space. The default unit is 512-byte blocks, but the number can be specified with "k" for kilobytes, "M" for megabytes, or "G" for gigabytes. For example, "-size +1G" will search for files larger than 1 gigabyte
-type *c*	Items is of type *c*, which can be "d" to look for a directory, "f" for a regular file, "l" for a symlink, or "s" for a network socket file
-user *uname*	File or directory is owned by the specified user (either a user name or numeric ID)
-writable	File can be written to by the current user. Note that this test can include access via ACLs or network shares, so it doesn't just rely on file system permissions

Common Operators

If no logical operator is specified, the default operator between expressions is *and* (i.e. all tests must be true for a file to match).

Parentheses can be used to control operator precedence.

Option	Description
! *expr*	"Not" operator. Only match if the test *expr* is not true
expr1 -o *expr2*	"Or" operator. Match if one or both of the expressions are true

Common Actions

Option	Description
-delete	Delete files and directories found. It's wise to test the find command with -print before having it delete everything it finds. Automatically enables -depth
-exec *command* ;	Execute the listed commands on matched files. Where {} is encountered, the matched file name will be substituted. The command will be run from the directory from which find was run. Everything after -exec will be considered a command until ; is encountered. You should escape the ; symbol to prevent the shell from evaluating it, like ";". The commands will run for each item separately
-exec *command* +	Execute the listed commands on matched files. Where {} is encountered, the matched file name will be substituted. The command will be run from the directory from which find was run. Ending the command list with a + instead of ; will cause all matched files to be combined into a single line before commands are executed. When using +, only one instance of {} can be used and it must be by itself without additional symbols attached
-execdir *command* ; -execdir *command* +	Execute the listed commands as in -exec ; and -exec +, but run from the subdirectory containing the file rather than the directory find was run from. Safer than -exec because of the subdirectory constraint.
-fprint *file*	Write the full file name and path into *file*
-fls *file*	Write the output of `ls -dils` for the match into *file*
-fprintf *file format*	Write the full file name and path using the *format* string to format the output (see `man find` for format options) into *file*

-ls	Print the output of `ls -dils` for the file
-ok *command* ;	Like -exec, but ask the user for confirmation for every match
-okdir *command* ;	Like -execdir, but ask the user for confirmation for every match
-print	Print the full file name and path. This is the default if no action is specified
-printf *format*	Print the full file name and path using the *format* string to format the output (see `man find` for format options)
-prune	If a matched item is a directory, do not descend into it for more checks. Is not active if -depth is also enabled

Command Examples

Find a file by name

This command looks for files in /usr/bin that start with the letters "sha". Note that the search pattern is enclosed in quotes - without quotes, the shell would replace that part of the command with any files in the current directory starting with "sha".

```
find /usr/bin -name "sha*"
```

Find large files in home directories

This command searches for files larger than 100 megabytes in user home directories, /tmp, and /var/log, then writes the full file details for anything that matched into a file in the current directory named "bigfiles.txt".

```
find /home /tmp /var/log -size +100M -fls bigfiles.txt
```

Find multiple file extensions at a minimum depth

This command will look for files ending in .htm, .html, or .php (using -o as the logical or operator). The search is limited to a minimum depth of 3 so that files specifically in the home directory are ignored, but any files in htdocs or public_html directories are.

```
find /home -mindepth 3 -name "*.html" -o -name "*.htm" -o -name "*.php"
```

Restrict permissions on user home directories

This command will find user directories that don't have permissions set to octal 700 (i.e. can be accessed by anyone other than the user) and change their permissions to 700. The prune option ensures that only user home directories are checked, not their contents.

```
find /home/* -prune ! -perm 700 -execdir chmod 700 {} \;
```

Find recently modified files

This command prints the names of files in the file system that have been modified within the last 10 minutes. Note that the files in /proc are modified all the time, so this command excludes them by using ! and parentheses to exclude the /proc path (and -prune to ensure that the /proc subdirectories are omitted as well).

```
find / ! \( -path /proc -prune \) -mmin -10
```

Using - and / with -perm

When specifying permissions in a search, putting the - symbol before the permissions will return items that include all listed permissions, while the / symbol returns items that list any of the listed permissions.

This command using - will return any regular files in user home directories that are both readable and writeable by users other than the owner.

```
find /home -perm -o=rx -type f
```

Change the - to a / and the command will return files that are readable by other users, executable by other users, or both readable and executable by other users.

```
find /home -perm /o=rx -type f
```

The difference -execdir makes

A difference between how exec and execdir work with results can be illustrated by using exec and execdir in the same command, using echo to print the result. The + symbol is used instead of ; to put all the results on one line. The first line of output is created by execdir and the second line is from exec.

```
find /usr/bin/ -name 't*' -exec echo {} + -execdir echo {} +
```

The execdir results show every result with a "./" prefix because find essentially uses `cd` to switch to the match's parent directory before working with the file. The exec results use the full path because exec works with matches without changing directories.