Check Linux file permissions with ls

support.rackspace.com/how-to/checking-linux-file-permissions-with-ls/

This article explains how to use the `ls` command to check Linux file permissions. Being able to check the permissions on a file is useful, especially for troubleshooting. You can ensure that a user can read a particular file, for example, or examine a directory structure to ensure that users can follow the hierarchy to the files that they need.

ls command

Use the `ls` command (the first letter is a lowercase L) to see what files are in a directory. When run by itself, `ls` returns a list of the current working directory. You can also specify a directory to list. The following example shows a list of the first few files in the `/etc` directory on a Gentoo system.

```
$ ls /etc
DIR_COLORS              gentoo-release      man.conf            runlevels
adjtime                 gpm                 mime.types
sandbox.conf
apache2                 group               mke2fs.conf         sandbox.d
bash                    group-              modprobe.d
scsi_id.config
ca-certificates         host.conf           modules.autoload.d  securetty
ca-certificates.conf    hosts               modprobe.d
scsi_id.config
...
```

ls -h

The `-h` option changes the way file sizes are displayed. When you use the -h option, files sizes are displayed in the human-readable format of kilobytes, megabytes, and so on, rather than in raw bytes. Other linux tools such as `df` also support this flag. The command `df -h` shows current disk usage in a easier to read format.

ls -a

To display hidden files (files with names that start with a period), use the `-a` option. For example, if you use only `ls` to look at the root home directory on a clean Linux installation, no files are returned:

```
$ ls
/root
```

However, if you add the `-a` option, the `ls` command returns a list of files:

```
$ ls -a /root
.   ..  .bash_history  .bashrc  .profile
.viminfo
```

Files that start with a period are often system files and application settings files, and you usually don't want them included in directory lists. But it’s important to know that they’re there and how to see them. The `.bashrc` file is especially useful to know about because it contains user environment settings that you can change.

If you combine the `-a` option with the `-l` option (see the next section) into `-la`, you get all the details of the hidden files:

```
$ ls -la /root
total 24
drwxr-xr-x  2 root root 4096 2009-12-16 01:10 .
drwxr-xr-x 23 root root 4096 2010-02-18 10:14 ..
-rw-------  1 root root  123 2010-01-21 15:49
.bash_history
-rw-r--r--  1 root root 2227 2007-10-20 11:51 .bashrc
-rw-r--r--  1 root root  141 2007-10-20 11:51 .profile
-rw-------  1 root root  868 2009-12-16 00:47 .viminfo
```

Consider the single period and double period in both directory lists:

- The single period (.) refers to the directory itself. This is convenient if you want it to run a command and reference your current directory (for example, when you want to copy a file there).
- The double period (..) refers to the parent directory. If you type `cd ..` the directory changes to the one above the one you’re in, in the file system hierarchy. For example, if your current directory is `/root`, typing `cd ..` would take you to `/`, the very top of the hierarchy.

ls -l

To get more information about the files in a directory, use the `-l` option with ls, as shown in the following example.

```
$ ls -l /etc
total 492
-rw-r--r-- 1 root root  4468 Nov 19  2009 DIR_COLORS
-rw-r--r-- 1 root root    10 Jun 30 03:29 adjtime
drwxr-xr-x 4 root root  4096 Jun 30 03:44 apache2
drwxr-xr-x 2 root root  4096 Nov 19  2009 bash
drwxr-xr-x 3 root root  4096 Nov 19  2009 ca-certificates
-rw-r--r-- 1 root root  5955 Nov 19  2009 ca-
certificates.conf
drwxr-xr-x 2 root root  4096 Jul  5 20:37 conf.d
drwxr-xr-x 2 root root  4096 Dec  3  2009 cron.d
drwxr-x--- 2 root root  4096 Dec  3  2009 cron.daily
-rw-r--r-- 1 root root   220 Dec  3  2009 cron.deny
drwxr-x--- 2 root root  4096 Dec  3  2009 cron.hourly
drwxr-x--- 2 root root  4096 Dec  3  2009 cron.monthly
drwxr-x--- 2 root root  4096 Dec  3  2009 cron.weekly
-rw-r--r-- 1 root root   611 Dec  3  2009 crontab
...
```

The file names are on the far right side of each line, and the file details precede the names. The necessary details to check file permissions are (1) the series of letters and dashes on the far left of each line, and (2) the two columns that have `root` in them (in the preceding example). The rest of this article explains how to interpret and use these details.

Permission details

This section explains the series of letters and dashes that define the file permissions.

The first character: file type

In the preceding examples, the first character in each list was either a dash (-) or the letter `d`.

- A dash (-) indicates that the file is a regular file.
- The letter `d` indicates that the file is a directory, which is basically a special kind of file.

A special file type that you might see is a symlink, sometimes called a soft link. It begins with a lowercase `L`, as shown in the following example:

```
lrwxrwxrwx 1 root root       4 Jun 30 03:29 sh ->
bash
```

A symlink is a pointer to another location in the file system.

Permissions abbreviations

Permissions for files are represented by the following letters.

- `r` refers to the read permission.
- `w` refers to the write permission.
- `x` refers to the execute permission.

The permissions characters

Consider the following example:

```
drwxrwxr-x 2 root mail 4096 Dec  3  2009
mail
```

The first trio of letters after the file type in a file list (`rwx`) shows the permissions for the `user`, or file owner.

The next trio of characters (also `rwx`) shows the permissions for the `group` category.

The last trio of characters (`r-x`) shows the permissions for the final category, `other`. In this example, users who are neither the file owner nor in the group have read and execute permissions but not write, as indicated by the dash (-) in the middle position.

Notice the specific order to the permissions in a trio: read, write, execute. A dash in place of a letter for a permission means that category doesn't have that permission.

The first number

The number listed after the permissions indicates the link count of a file or the number of contained directory entries, for a directory. This number is not relevant for permissions.

Owner and group

After the number of links, two names are listed. In the preceding example, the names are `root` and `mail`.

The first name indicates the owner of the file. The `user` permissions apply to owner of the file, so in this case, the user ‘root’ has read, write, and execute permissions for this directory.

The second name is the file’s group. The `group` permissions apply to any user in the same group as the file, so in this case, those permissions apply to anyone in the `mail` group.