

By: Justin Ellingwood  
Basic Linux Navigation and File Ma... ▼

Subscribe

Share

Contents ▼

# Basic Linux Navigation and File Management

Posted November 14, 2014 100.7k LINUX BASICS SYSTEM TOOLS



31

## Introduction

If you do not have much experience working with Linux systems, you may be overwhelmed by the prospect of controlling an operating system from the command line. In this guide, we will attempt to get you up to speed with the basics.

This guide will not cover everything you need to know to effectively use a Linux system. However, it should give you a good jumping-off point for future exploration. This guide will give you the bare minimum you need to know before moving on to other guides.

## Prerequisites and Goals

In order to follow along with this guide, you will need to have access to a Linux server. If you need information about connecting to your server for the first time, you can follow our [guide on connecting to a Linux server using SSH](#).

You will also want to have a basic understanding of how the terminal works and what Linux commands look like. [This guide covers terminal basics](#), so you should check it out if you are new to using terminals.

All of the material in this guide can be accomplished with a regular, non-root (non-administrative) user account. You can learn how to configure this type of user account by following your distribution's initial server setup guide ([Ubuntu 14.04](#), [CentOS 7](#)).

When you are ready to begin, connect to your Linux server using SSH and continue below.

## Navigation and Exploration

The most fundamental skills you need to master are moving around the filesystem and getting an idea of what is around you. We will discuss the tools that allow you to do this in this section.

### Finding Where You Are with the "pwd" Command

When you log into your server, you are typically dropped into your user account's **home directory**. A home directory is a directory set aside for your user to store files and create directories. It is the location in the filesystem where you have full dominion.

To find out where your home directory is in relationship to the rest of the filesystem, you can use the `pwd` command. This command displays the directory that we are currently in:

```
pwd
```

You should get back some information that looks like this:

```
/home/demo
```

The home directory is named after the user account, so the above example is what the value would be if you were logged into the server with an account called `demo`. This directory is within a directory called `/home`, which is itself within the top-level directory, which is called "root" but represented by a single slash `/`.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



✕ will go to another, more populated directory to explore. Type the following command to explore the `/usr/share` directory (we'll use `pwd` to explore other directories in the next section). Afterward, we'll use `pwd` to

confirm that we successfully moved.

```
cd /usr/share
pwd
```

```
/usr/share
```

Now that we are in a new directory, let's look at what's inside. To do this, we can use the `ls` command:

```
ls
```

```
adduser      groff        pam-configs
applications  grub         perl
apport       grub-gfxpayload-lists  perl5
apps         hal          pixmaps
apt          il8n         pkgconfig
aptitude     icons        polkit-1
apt-xapian-index  info        popularity-contest
. . .
```

As you can see, there are *many* items in this directory. We can add some optional flags to the command to modify the default behavior. For instance, to list all of the contents in an extended form, we can use the `-l` flag (for "long" output):

```
ls -l
```

```
total 440
drwxr-xr-x  2 root root  4096 Apr 17  2014 adduser
drwxr-xr-x  2 root root  4096 Sep 24 19:11 applications
drwxr-xr-x  6 root root  4096 Oct  9 18:16 apport
drwxr-xr-x  3 root root  4096 Apr 17  2014 apps
drwxr-xr-x  2 root root  4096 Oct  9 18:15 apt
drwxr-xr-x  2 root root  4096 Apr 17  2014 aptitude
drwxr-xr-x  4 root root  4096 Apr 17  2014 apt-xapian-index
drwxr-xr-x  2 root root  4096 Apr 17  2014 awk
. . .
```

This view gives us plenty of information, most of which looks rather unusual. The first block describes the file type (if the first column is a "d" the item is a directory, if it is a "-", it is a normal file) and permissions. Each subsequent column, separated by white space, describes the number of hard links, the owner, group owner, item size, last modification time, and the name of the item. We will describe some of these at another time, but for now, just know that you can view this information with the `-l` flag of `ls`.

To get a listing of all files, including *hidden* files and directories, you can add the `-a` flag. Since there are no real hidden files in the `/usr/share` directory, let's go back to our home directory and try that command. You can get back to the home directory by typing `cd` with no arguments:

```
cd
ls -a
```

```
.  .. .bash_logout .bashrc .profile
```

For the dot and double dot entries, these aren't exactly directories as much as built-in methods of referring to related directories. The single dot

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



actory. This will come in handy in the next section.

[Sign Up](#)

## Moving Around the Filesystem with "cd"

We have already made two directory moves in order to demonstrate some properties of `ls` in the last section. Let's take a better look at the command here.

Begin by going back to the `/usr/share` directory by typing this:

```
cd /usr/share
```

This is an example of changing a directory by giving an **absolute path**. In Linux, every file and directory is under the top-most directory, which is called the "root" directory, but referred to by a single leading slash `/`. An absolute path indicates the location of a directory in relation to this top-level directory. This lets us refer to directories in an unambiguous way from any place in the filesystem. Every absolute path **must** begin with a slash.

The alternative is to use **relative paths**. Relative paths refer to directories in relation to the *current* directory. For directories close to the current directory in the hierarchy, this is usually easier and shorter. Any directory within the current directory can be referenced by name without a leading slash. We can change to the `locale` directory within `/usr/share` from our current location by typing:

```
cd locale
```

We can likewise move multiple directory levels with relative paths by providing the portion of the path that comes after the current directory's path. From here, we can get to the `LC_MESSAGES` directory within the `en` directory by typing:

```
cd en/LC_MESSAGES
```

To go back up, travelling to the parent of the current directory, we use the special double dot indicator we talked about earlier. For instance, we are now in the `/usr/share/locale/en/LC_MESSAGES` directory. To move up one level, we can type:

```
cd ..
```

This takes us to the `/usr/share/locale/en` directory.

A shortcut that you saw earlier that will always take you back to your home directory is to use `cd` without providing a directory:

```
cd
pwd
```

```
/home/demo
```

To learn more about how to use these three commands, you can check out [our guide on exploring the Linux filesystem](#).

## Viewing Files

In the last section, we learned a bit about how to navigate the filesystem. You probably saw some files when using the `ls` command in various directories. In this section, we'll discuss different ways that you can use to view files. In contrast to some operating systems, Linux and other Unix-like operating systems rely on plain text files for vast portions of the system.

The main way that we will view files is with the `less` command. This is what we call a "pager", because it allows us to scroll through pages of a file. While the previous commands immediately executed and returned you to the command line, `less` is an application that will continue to run and occupy the screen until you exit.

```
less /etc/services
```

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

fits in the area of the terminal window:

```
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.
#
# Updated from http://www.iana.org/assignments/port-numbers and other
# sources like http://www.freebsd.org/cgi/cvsweb.cgi/src/etc/services .
# New ports will be added on request if they have been officially assigned
# by IANA and used in the real-world or are needed by a debian package.
# If you need a huge list of used numbers please install the nmap package.

tcpmux      1/tcp                # TCP port service multiplexer
echo        7/tcp
. . .
```

To scroll, you can use the up and down arrow keys on your keyboard. To page down one whole screens-worth of information, you can use either the space bar, the "Page Down" button on your keyboard, or the `CTRL-f` shortcut.

To scroll back up, you can use either the "Page Up" button, or the `CTRL-b` keyboard shortcut.

To search for some text in the document, you can type a forward slash `/` followed by the search term. For instance, to search for "mail", we would type:

```
/mail
```

This will search forward through the document and stop at the first result. To get to another result, you can type the lower-case `n` key:

```
n
```

To move backwards to the previous result, use a capital `N` instead:

```
N
```

When you wish to exit the `less` program, you can type `q` to quit:

```
q
```

While we focused on the `less` tool in this section, there are many other ways of viewing a file that come in handy in certain circumstances. The `cat` command displays a file's contents and returns you to the prompt immediately. The `head` command, by default, shows the first 10 lines of a file. Likewise, the `tail` command shows the last 10 lines by default. These commands display file contents in a way that is useful for "piping" to other programs. We will discuss this concept in a future guide.

Feel free to see how these commands display the `/etc/services` file differently.

## File and Directory Manipulation

We learned in the last section how to view a file. In this section, we'll demonstrate how to create and manipulate files and directories.

### Create a File with "touch"

First, we should make sure we are in our home directory, since this is a location where we have permission to save files. Then, we can create a file

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.

[Sign Up](#)

```
cd
touch file1
```

Now, if we view the files in our directory, we can see our newly created file:

```
ls
```

```
file1
```

If we use this command on an existing file, the command simply updates the data our filesystem stores on the time when the file was last accessed and modified. This won't have much use for us at the moment.

We can also create multiple files at the same time. We can use absolute paths as well. For instance, if our user account is called `demo`, we could type:

```
touch /home/demo/file2 /home/demo/file3
ls
```

```
file1 file2 file3
```

## Create a Directory with "mkdir"

Similar to the `touch` command, the `mkdir` command allows us to create empty directories.

For instance, to create a directory within our home directory called `test`, we could type:

```
cd
mkdir test
```

We can make a directory *within* the `test` directory called `example` by typing:

```
mkdir test/example
```

For the above command to work, the `test` directory must already exist. To tell `mkdir` that it should create any directories necessary to construct a given directory path, you can use the `-p` option. This allows you to create nested directories in one step. We can create a directory structure that looks like `some/other/directories` by typing:

```
mkdir -p some/other/directories
```

The command will make the `some` directory first, then it will create the `other` directory inside of that. Finally it will create the `directories` directory within those two directories.

## Moving and Renaming Files and Directories with "mv"

We can move a file to a new location using the `mv` command. For instance, we can move `file1` into the `test` directory by typing:

```
mv file1 test
```

Command:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up

This may seem unintuitive at first, but the `mv` command is also used to *rename* files and directories. In essence, moving and renaming are both just adjusting the location and name for an existing item.

So to rename the `test` directory to `testing`, we could type:

```
mv test testing
```

**Note:** It is important to realize that your Linux system will not prevent you from certain destructive actions. If you are renaming a file and choose a name that *already* exists, the previous file will be **overwritten** by the file you are moving. There is no way to recover the previous file if you accidentally overwrite it.

## Copying Files and Directories with "cp"

With the `mv` command, we could move or rename a file or directory, but we could not duplicate it. The `cp` command can make a new copy of an existing item.

For instance, we can copy `file3` to a new file called `file4`:

```
cp file3 file4
```

Unlike a `mv` operation, after which `file3` would no longer exist, we now have both `file3` and `file4`.

**Note:** As with the `mv` command, it is possible to **overwrite** a file if you are not careful about the filename you are using as the target of the operation. For instance, if `file4` already existed in the above example, its content would be completely replaced by the content of `file3`.

In order to copy directories, you must include the `-r` option to the command. This stands for "recursive", as it copies the directory, plus all of the directory's contents. This option is necessary with directories, regardless of whether the directory is empty.

For instance, to copy the `some` directory structure to a new structure called `again`, we could type:

```
cp -r some again
```

Unlike with files, with which an existing destination would lead to an overwrite, if the target is an *existing directory*, the file or directory is copied *into* the target:

```
cp file1 again
```

This will create a new copy of `file1` and place it inside of the `again` directory.

## Removing Files and Directories with "rm" and "rmdir"

To delete a file, you can use the `rm` command.

**Note:** Be extremely careful when using any destructive command like `rm`. There is no "undo" command for these actions so it is possible to accidentally destroy important files permanently.

To remove a regular file, just pass it to the `rm` command:

```
cd
rm file4
```

instance, to remove the `example` directory within the `testing` directory, we can type:

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.





If you wish to remove a *non-empty* directory, you will have to use the `rm` command again. This time, you will have to pass the `-r` option, which removes all of the directory's contents recursively, plus the directory itself.

For instance, to remove the `again` directory and everything within it, we can type:

```
rm -r again
```

Once again, it is worth reiterating that these are permanent actions. Be entirely sure that the command you typed is the one that you wish to execute.

## Editing Files

Currently, we know how to manipulate files as objects, but we have not learned how to actually edit them and add content to them.

The `nano` command is one of the simplest command-line Linux text editors, and is a great starting point for beginners. It operates somewhat similarly to the `less` program discussed above, in that it occupies the entire terminal for the duration of its use.

The `nano` editor can open existing files, or create a file. If you decide to create a new file, you can give it a name when you call the `nano` editor, or later on, when you wish to save your content.

We can open the `file1` file for editing by typing:

```
cd
nano file1
```

The `nano` application will open the file (which is currently blank). The interface looks something like this:

```
GNU nano 2.2.6
```

```
File: file1
```

```
[ Read 0 lines ]
```

```
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Along the top, we have the name of the application and the name of the file we are editing. In the middle, the content of the file, currently blank, is displayed. Along the bottom, we have a number of key combinations that indicate some basic controls for the editor. For each of these, the `^` character means the `CTRL` key.

To get help from within the editor, type:

```
CTRL-G
```

When you are finished browsing the help, type `CTRL-X` to get back to your document.

Type in or modify any text you would like. For this example, we'll just type these two sentences:

Here is some text.

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.




Sign Up

CTRL-O

This is the letter "o", not a zero. It will ask you to confirm the name of the file you wish to save to:

File Name to Write: file1

^G Get Help

M-D DOS Format

M-A Append

M-B Backup File

^C Cancel

M-M Mac Format

M-P Prepend

As you can see, the options at the bottom have also changed. These are contextual, meaning they will change depending on what you are trying to do. If `file1` is still the file you wish to write to, hit "ENTER".

If we make some additional changes and wish to save the file and exit the program, we will see a similar prompt. Add a new line, and then try to exit the program by typing:

CTRL-X

If you have not saved after making your modification, you will be asked whether you wish to save the modifications you made:

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?

Y Yes

N No                    ^C Cancel

You can type "Y" to save your changes, "N" to discard your changes and exit, or "CTRL-C" to cancel the exit operation. If you choose to save, you will be given the same file prompt that you received before, confirming that you want to save the changes to the same file. Press ENTER to save the file and exit the editor.

You can see the contents of the file you created using either the `cat` program to display the contents, or the `less` program to open the file for viewing. After viewing with `less`, remember that you should hit `q` to get back to the terminal.

```
less file1
```

```
Hello there.
```

```
Here is some text.
```

```
Another line.
```

Another editor that you may see referenced in certain guides is `vim` or `vi`. This is a more advanced editor that is very powerful, but comes with a very steep learning curve. If you are ever told to use `vim` or `vi`, feel free to use `nano` instead. If you wish to learn how to use `vim`, read our [guide to getting started with vim](#).

## Conclusion

By now, you should have a basic understanding of how to get around your Linux server and how to see the files and directories available. You should also know some basic file manipulation commands that will allow you to view, copy, move, or delete files. Finally, you should be comfortable with some basic editing using the `nano` text editor.

With these few skills, you should be able to continue on with other guides and learn how to get the most out of your server. In our next guide, we will discuss [how to view and understand Linux permissions](#).



Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Enter your email address

Sign Up

## Tutorial Series

### Getting Started with Linux

If you are new to Linux and its command line interface, it may seem like a daunting task to get started with it. This series will bring you up to speed with essential Linux basics, and provide a solid foundation for working with Linux servers. If you have little to no knowledge about using Linux, this is where you will want to start.

Show Tutorials

## New Droplets: More RAM, More SSD Storage, More Flexibility

New Droplets on DigitalOcean include 2x Memory for the same price, new High-CPU Optimized Plans, and a new class of Flexible \$15 plans. The \$5 Droplet now has 1GB RAM and 25GB SSD.

[READ ABOUT NEW DROPLETS AND PRICES](#)

### Related Tutorials

How to Add and Delete Users on Ubuntu 16.04

How To Download Software and Content onto your Linux VPS

An Introduction to Useful Bash Aliases and Functions

An Introduction To Regular Expressions

How To Use Bash History Commands and Expansions on a Linux VPS

## 0 Comments

Leave a comment...

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Enter your email address

Sign Up



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2018 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DigitalOcean](#) [Shop](#)

Sign up for our newsletter. Get the latest tutorials on SysAdmin and open source topics.



Sign Up