// TUTORIAL //

Linux Navigation and File Management

Updated on June 2, 2022

Linux Basics System Tools

Justin Ellingwood

Introduction

Navigating and manipulating files and folders in the filesystem is a key part of working with most computers. Cloud servers mostly use the same common Linux shells, and common Linux commands, for working with files and folders. This terminal will introduce some fundamental skills for using these commands.

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 an understanding of how the terminal works and what Linux commands look like. This guide covers an introduction to the terminal.

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, such as for Ubuntu 22.04.

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. You will review 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 relation to the rest of the filesystem, you can use the `pwd` command. This command displays the directory that you are currently in:

Copy

```
$ pwd
```

```
Output
/home/ sammy
```

The home directory is named after the user account. This directory is within a directory called `/home`, which is itself within the top-level directory, which is usually called the "root" directory, and is represented by a single slash `/`.

Looking at the Contents of Directories with “ls”

Now that you know how to display the directory that you are in, you can look at the contents of a directory.

Currently, your home directory does not have much to look at, so you can go to another, more populated directory to explore. Use `cd` to move to this directory. Afterward, you’ll use `pwd` to confirm that you successfully moved:

Copy

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

```
Output
/usr/share
```

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

Copy

```
$ ls
```

```
Output
adduser            groff                        pam-configs
applications       grub                         perl
apport             grub-gfxpayload-lists        perl5
apps               hal                          pixmaps
apt                i18n                         pkgconfig
aptitude           icons                        polkit-1
apt-xapian-index   info                         popularity-contest
. . .
```

As you can see, there are many items in this directory. You 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, you can use the `-l` flag (for “long” output):

Copy

```
$ ls -l
```

```
Output
total 440
drwxr-xr-x   2 root root  4096 Apr 17  2022 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  2022 apps
drwxr-xr-x   2 root root  4096 Oct  9 18:15 apt
drwxr-xr-x   2 root root  4096 Apr 17  2022 aptitude
drwxr-xr-x   4 root root  4096 Apr 17  2022 apt-xapian-index
drwxr-xr-x   2 root root  4096 Apr 17  2022 awk
. . .
```

This view gives us plenty of information. The first block describes the file type (if the first column is a "d" the item is a directory, and if it is a "-", it is a normal file) and permissions. Each subsequent column, in order, describes the number of hard links to that file elsewhere on the system, the owner, group owner, item size, last modification time, and the name of the item.

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 your home directory and try that command. You can get back to the home directory by typing `cd` with no arguments:

Copy

```
$ cd
$ ls -a
```

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

As you can see, there are three hidden files, along with `.` and `..`, which are special indicators. You will find that often, configuration files are stored as hidden files, as is the case here.

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 indicates the current directory, and the double dot indicates this directory's parent directory. This will come in handy in the next section.

Moving Around the Filesystem with “cd”

You 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:

Copy

```
$ cd /usr/share
```

This is an example of changing a directory by providing 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 that 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 shorter, and it is sometimes beneficial to not need to make assumptions about where a directory is located in the broader filesystem. Any directory within the current directory can be referenced by name without a leading slash. You can change to the `locale` directory within `/usr/share` from your current location by typing:

Copy

```
$ cd locale
```

You can also move multiple directory levels with relative paths by providing the portion of the path that comes after the current directory’s path. From here, you can get to the `LC_MESSAGES` directory within the `en` directory by typing:

Copy

```
$ cd en/LC_MESSAGES
```

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

Copy

```
$ cd ..
```

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

You can always return to your home directory by running `cd` without specifying a directory. You can also use `~` in place of your home directory in any other commands:

```
cd ~
pwd
```

```
/home/ sammy
```

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, you learned how to navigate the filesystem. You probably saw some files when using the `ls` command in various directories. 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 you will view files in this tutorial is with the `less` command. This is what is called a "pager", because it allows you 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.

You will open the `/etc/services` file, which is a configuration file that contains service information that the system knows about:

Copy

```
$ less /etc/services
```

The file will be opened in `less`, allowing you to see the portion of the document that fits in the area of the terminal window:

```
Output
```

```
# 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, 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”, you 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
```

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

```
q
```

There are many other ways of viewing a file that come in handy in certain circumstances. The `cat` command outputs 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. These commands display file contents in a way that is useful for "piping" to other programs. This concept is covered later in this tutorial series.

File and Directory Manipulation

In this section, you'll create and manipulate files and directories.

Create a File with "touch"

Many commands and programs can create files. The most straightforward method of creating a file is with the `touch` command. This will create an empty file using the name and location specified.

First, make sure you are in your home directory, since this is a location where you have permission to save files. Then, you can create a file called `file1` by typing:

Copy

```
$ cd
$ touch file1
```

Now, if you view the files in the directory, you can see your newly created file:

Copy

```
$ ls
```

```
Output
file1
```

If you use the `touch` command on an existing file, it updates the "last modified" time associated with that file. This can be helpful to keep in mind.

You can also create multiple files at the same time. You can use absolute paths as well. For instance, you could type:

```
$ touch /home/ sammy /file2 /home/ sammy /file3
$ ls
```

```
Output
file1  file2  file3
```

Create a Directory with “mkdir”

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

For instance, to create a directory within your home directory called `test`, you could type:

```
$ cd
$ mkdir test
```

You 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. You 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”

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

Copy

```
$ mv file1 test
```

You can move that file *back* to your home directory by using the special dot reference to refer to the current directory. Make sure you're in your home directory, and then run the `mv` command:

Copy

```
$ cd
$ mv test/file1 .
```

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`, you could type:

Copy

```
$ mv test testing
```

Note: The shell will not prevent you from making accidentally 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, you could move or rename a file or directory, but you could not duplicate it. The `cp` command can make a new copy of an existing item.

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

Copy

```
$ cp file3 file4
```

Unlike a `mv` operation, after which `file3` would no longer exist, you 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 entire 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.

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

Copy

```
$ 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:

Copy

```
$ 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 in the shell, so it is possible to accidentally destroy important files permanently.

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

Copy

```
$ cd
$ rm file4
```

Likewise, to remove empty directories, you can use the `rmdir` command. This will only succeed if there is nothing in the directory in question. For instance, to remove the `example` directory within the `testing` directory:

Copy

```
$ rmdir testing/example
```

To remove a non-empty directory, you will use the `rm` command with 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, you can type:

Copy

```
$ rm -r again
```

Editing Files

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

`nano` is one of a few common 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 save your content.

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

Copy

```
$ cd
$ nano file1
```

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

```
  GNU nano 4.8                  file1









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

To get help from within the editor, press `Ctrl+G`.

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

For this example, you can just type these two sentences:

file1

```
Hello there.

Here is some text.
```

To save your work, press `Ctrl+O`.

```
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. To confirm writing to `file1`, press `Enter`.

After saving, If you make additional changes and try to exit the program, you will see a similar prompt. Add a new line, and then try to exit `nano` by pressing `Ctrl+X`.

If you have not saved, you will be asked to save the modifications you made:

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No           ^C Cancel
```

You can press `Y` to save your changes, `N` to discard your changes and exit, or `Ctrl+C` to cancel quitting. 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 press `q` to get back to the terminal.

Copy

```
$ less file1
```

```
Output
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 steep learning curve. If you are ever told to use `vim` or `vi`, feel free to use `nano` instead. To learn how to use `vim`, read our guide to getting started with vim.

Conclusion

By now, you should have an understanding of how to get around your Linux server and how to see the files and directories available. You should also know file manipulation commands that will allow you to view, copy, move, or delete files. Finally, you should be comfortable with some 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, you will understand how to view and understand Linux permissions.

Next in series: An Introduction to Linux Permissions →

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.

Subscribe

Linux Basics System Tools

Browse Series: 4 articles

Expand to view all

About the authors

Justin Ellingwood Author

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

Leave a comment

Leave a comment...

This textbox defaults to using `Markdown` to format your answer.

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

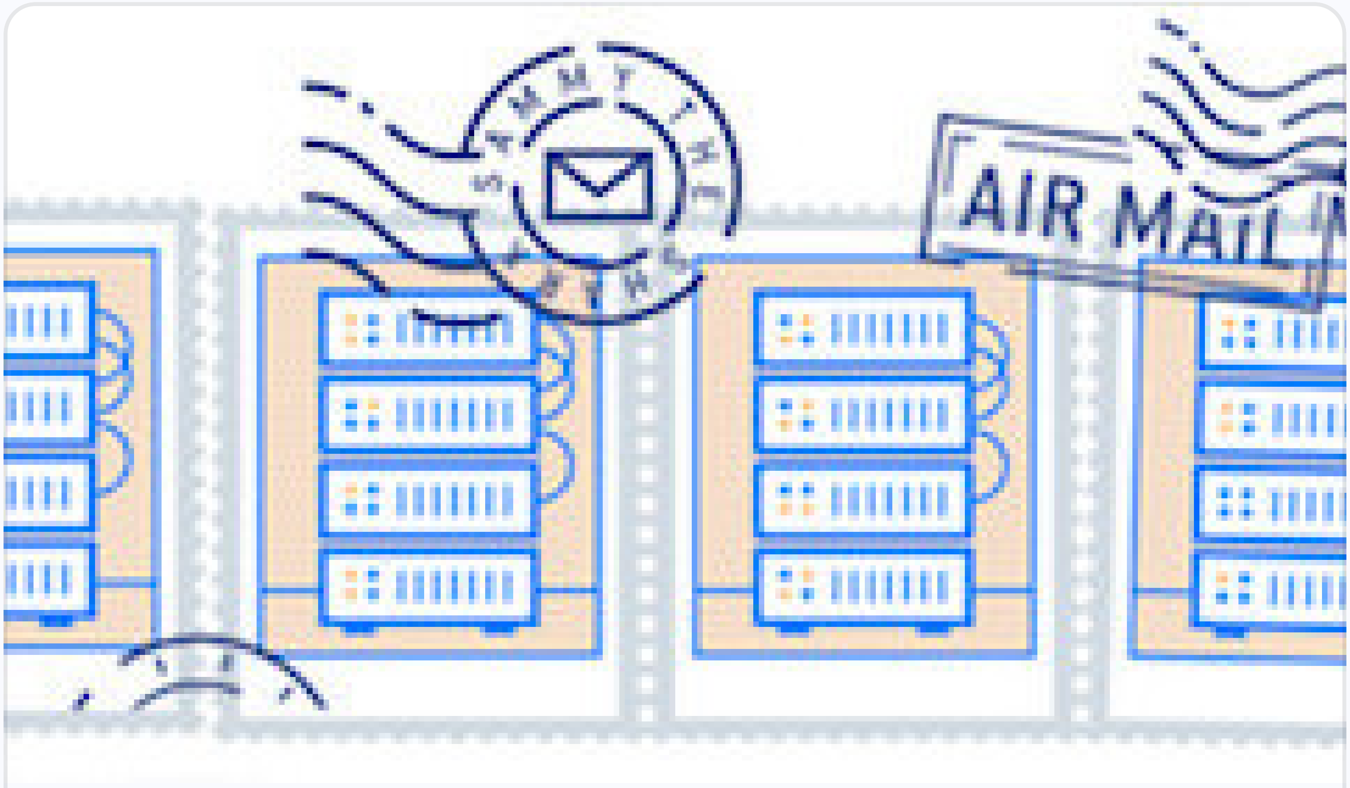
MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →



Get our biweekly newsletter

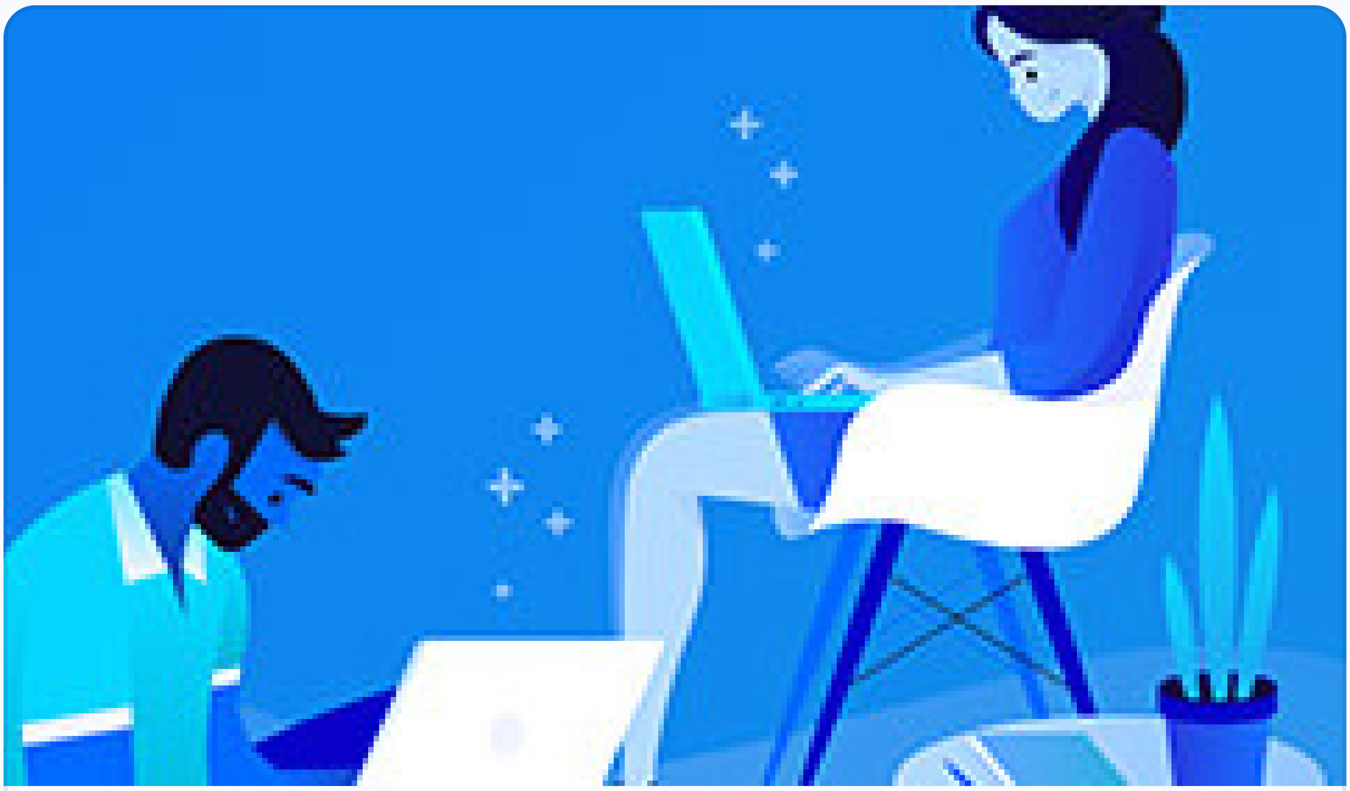
Sign up for Infrastructure as a Newsletter.

Sign up →

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

Learn more →



Become a contributor

Get paid to write technical tutorials and select a tech-focused charity to receive a matching donation.

Learn more →

Featured Tutorials

Kubernetes Course Learn Python 3 Machine Learning in Python

Getting started with Go Intro to Kubernetes

DigitalOcean Products

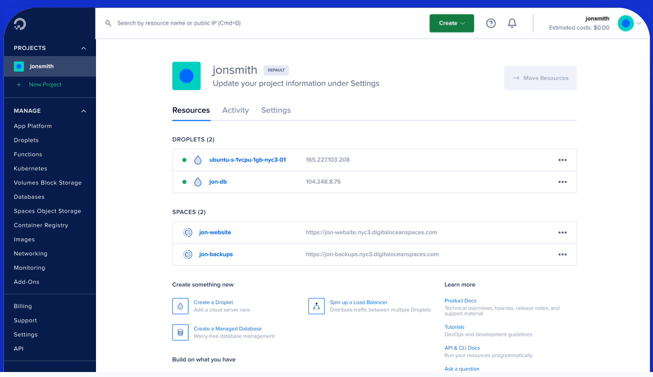
App Platform Virtual Machines Managed Databases Managed Kubernetes

Block Storage Object Storage Marketplace VPC Load Balancers

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow — whether you're running one virtual machine or ten thousand.

Learn more →



Company