

Read "Rising Tides of Open Source", the Linux Foundation 2023 Annual Report. **READ NOW**

**ENGLISH** Sign In

JOIN

12 MIN READ

# Classic SysAdmin: The Linux Filesystem Explained

THE LINUX FOUNDATION | 27 FEBRUARY 2022

*This is a classic article written by Paul Brown from the [Linux.com](#) archives. For more great SysAdmin tips and techniques check out our free [intro to Linux course](#).*

Back in 1996 I learned how to install software on my spanking new Linux before really understanding the topography of the filesystem.

## SIMILAR ARTICLES

Classic SysAdmin:  
Absolute Path vs Relative  
Path in Linux/Unix

This turned out to be a problem, not so much for programs, because they would just magically work even though I hadn't a clue of where the actual executable files landed. The problem was the documentation.

You see, back then, Linux was not the intuitive, user-friendly system it is today. You had to read a lot. You had to know things about the frequency rate of your CRT monitor and the ins and outs of your noisy dial-up modem, among hundreds of other things. I soon realized I would need to spend some time getting a handle on how the directories were organized and what all their exotic names like */etc* (not for miscellaneous files), */usr* (not for user files), and */bin* (not a trash can) meant.

This tutorial will help you get up to speed faster than I did.

## Structure

It makes sense to explore the Linux filesystem from a terminal window, not because the author is a grumpy old man and resents new kids and their pretty graphical tools — although there is some truth to that — but because a terminal, despite being text-only, has better tools to show the map of Linux's directory tree.

In fact, that is the name of the first tool you'll install to help you on the way: *tree*. If you are using Ubuntu or Debian, you can do:

```
sudo apt install tree
```

On Red Hat or Fedora, do:

Classic SysAdmin: How to Install an SSL Certificate on Linux Server

Classic SysAdmin: Vim 101: A Beginner's Guide to Vim

### BROWSE CATEGORIES

2023

Cloud Computing

Compliance and Security

Projects

Linux How-To

Diversity & Inclusion

Events

Open Source Best Practices

2022

Open Source

Cross Technology

```
sudo dnf install tree
```

For SUSE/openSUSE use zypper:

```
sudo zypper install tree
```

For Arch-like distros (Manjaro, Antergos, etc.) use:

```
sudo pacman -S tree
```

... and so on.

Once installed, stay in your terminal window and run *tree* like this:

```
tree /
```

The `/` in the instruction above refers to the *root* directory. The root directory is the one from which all other directories branch off from. When you run *tree* and tell it to start with `/`, you will see the whole directory tree, all directories and all the subdirectories in the whole system, with all their files, fly by.

If you have been using your system for some time, this may take a while, because, even if you haven't generated many files yourself, a Linux system and its apps are always logging, cacheing, and storing temporal files. The number of entries in the file system can grow quite quickly.

Don't feel overwhelmed, though. Instead, try this:

```
tree -L 1 /
```

And you should see what is shown in Figure 1.

Training and Certification

LFX

Newsletter

Legal

Research

LF Research

Networking and Edge

Blog

Data Governance

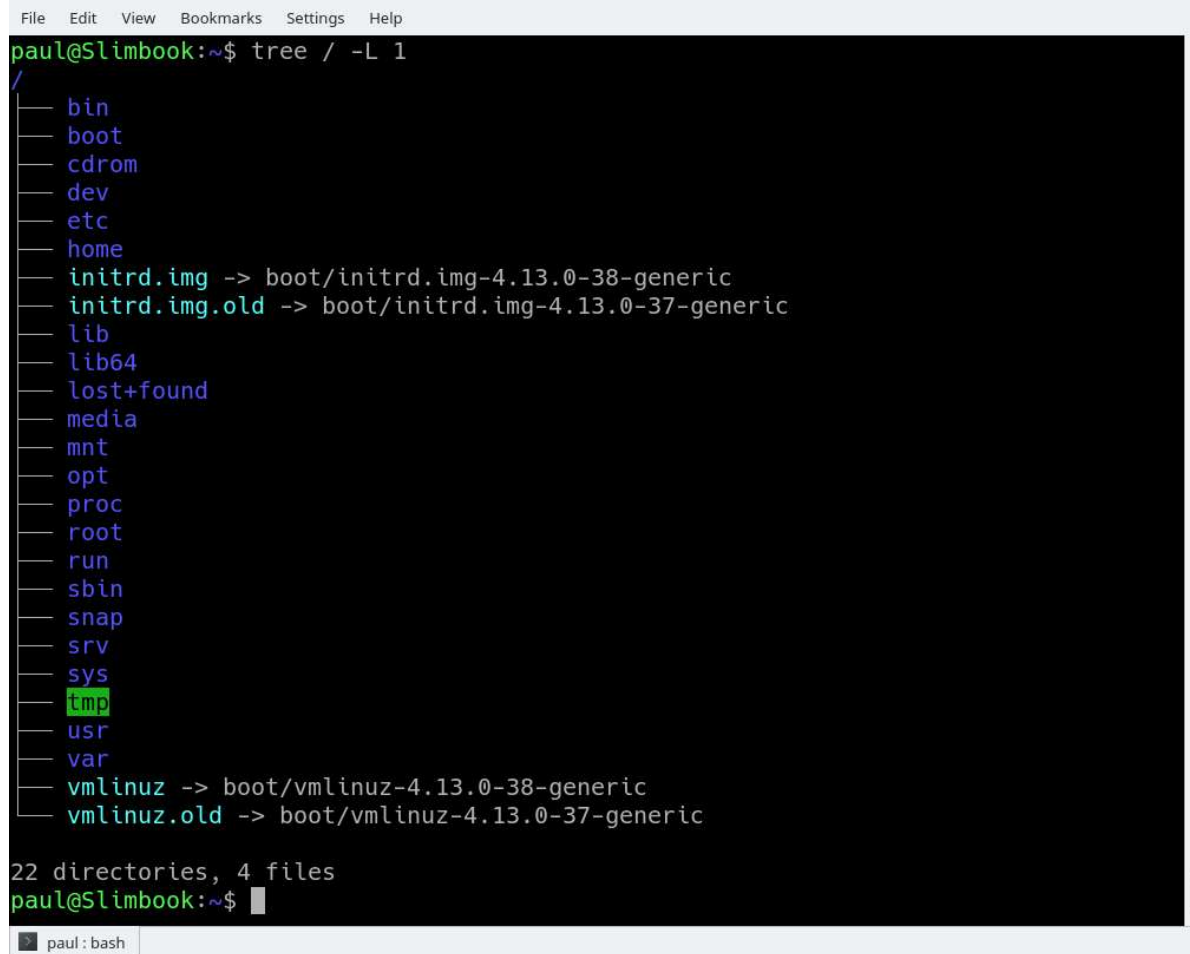
Featured

LF Energy

Open Mainframe

OpenChain

System Administration



```
File Edit View Bookmarks Settings Help
paul@Slimbook:~$ tree / -L 1
/
├── bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── initrd.img -> boot/initrd.img-4.13.0-38-generic
├── initrd.img.old -> boot/initrd.img-4.13.0-37-generic
├── lib
├── lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── snap
├── srv
├── sys
├── tmp
├── usr
├── var
├── vmlinuz -> boot/vmlinuz-4.13.0-38-generic
└── vmlinuz.old -> boot/vmlinuz-4.13.0-37-generic

22 directories, 4 files
paul@Slimbook:~$
```

The instruction above can be translated as *“show me only the 1st Level of the directory tree starting at / (root)”*. The `-L` option tells `tree` how many levels down you want to see.

Most Linux distributions will show you the same or a very similar layout to what you can see in the image above. This means that even if you feel confused now, master this, and you will have a handle on most, if not all, Linux installations in the whole wide world.

To get you started on the road to mastery, let's look at what each directory is used for. While we go through each, you can peek at their contents using *ls*.

## Directories

From top to bottom, the directories you are seeing are as follows.

### */bin*

*/bin* is the directory that contains *binaries*, that is, some of the applications and programs you can run. You will find the *ls* program mentioned above in this directory, as well as other basic tools for making and removing files and directories, moving them around, and so on. There are more *bin* directories in other parts of the file system tree, but we'll be talking about those in a minute.

### */boot*

The */boot* directory contains files required for starting your system. Do I have to say this? Okay, I'll say it: **DO NOT TOUCH!** If you mess up one of the files in here, you may not be able to run your Linux and it is a pain to repair. On the other hand, don't worry too much about destroying your system by accident: you have to have superuser privileges to do that.

### */dev*

*/dev* contains *device* files. Many of these are generated at boot time or even on the fly. For example, if you plug in a new webcam or a USB pendrive into your machine, a new device entry will automatically pop up here.

## */etc*

*/etc* is the directory where names start to get confusing. */etc* gets its name from the earliest Unixes and it was literally “et cetera” because it was the dumping ground for system files administrators were not sure where else to put.

Nowadays, it would be more appropriate to say that *etc* stands for “Everything to configure,” as it contains most, if not all system-wide configuration files. For example, the files that contain the name of your system, the users and their passwords, the names of machines on your network and when and where the partitions on your hard disks should be mounted are all in here. Again, if you are new to Linux, it may be best if you don’t touch too much in here until you have a better understanding of how things work.

## */home*

*/home* is where you will find your users’ personal directories. In my case, under */home* there are two directories: */home/paul*, which contains all my stuff; and */home/guest*, in case anybody needs to borrow my computer.

## */lib*

*/lib* is where *libraries* live. Libraries are files containing code that your applications can use. They contain snippets of code that applications use to draw windows on your desktop, control peripherals, or send files to your hard disk.

There are more *lib* directories scattered around the file system, but this one, the one hanging directly off of */* is special in that, among other things, it contains the all-important kernel modules. The kernel modules are drivers that make things like your video card, sound card, WiFi, printer, and so on, work.

## */media*

The */media* directory is where external storage will be automatically mounted when you plug it in and try to access it. As opposed to most of the other items on this list, */media* does not hail back to 1970s, mainly because inserting and detecting storage (pendrives, USB hard disks, SD cards, external SSDs, etc) on the fly, while a computer is running, is a relatively new thing.

## */mnt*

The */mnt* directory, however, is a bit of remnant from days gone by. This is where you would manually mount storage devices or partitions. It is not used very often nowadays.

## */opt*

The */opt* directory is often where software you compile (that is, you build yourself from source code and do not install from your distribution repositories) sometimes lands. Applications will end up in the */opt/bin* directory and libraries in the */opt/lib* directory.

A slight digression: another place where applications and libraries end up in is */usr/local*. When software gets installed here, there will also be */usr/local/bin* and */usr/local/lib* directories. What determines which software goes where is how the developers have configured the files that control the compilation and installation process.

## */proc*

*/proc*, like */dev* is a virtual directory. It contains information about your computer, such as information about your CPU and the kernel your Linux system is running. As with */dev*, the files and directories are generated when your computer starts, or on the fly, as your system is running and things change.

## */root*

*/root* is the home directory of the superuser (also known as the "Administrator") of the system. It is separate from the rest of the users' home directories BECAUSE YOU ARE NOT MEANT TO TOUCH IT. Keep your own stuff in your own directories, people.



## */run*

*/run* is another new directory. System processes use it to store temporary data for their own nefarious reasons. This is another one of those DO NOT TOUCH folders.

## */sbin*

*/sbin* is similar to */bin*, but it contains applications that only the superuser (hence the initial *s*) will need. You can use these applications with the `sudo` command that temporarily concedes you superuser powers on many distributions. */sbin* typically contains tools that can install stuff, delete stuff and format stuff. As you can imagine, some of these instructions are lethal if you use them improperly, so handle with care.

## */usr*

The */usr* directory was where users' home directories were originally kept back in the early days of UNIX. However, now */home* is where users kept their stuff as we saw above. These days, */usr* contains a mish-mash of directories which in turn contain applications, libraries, documentation, wallpapers, icons and a long list of other stuff that need to be shared by applications and services.

You will also find *bin*, *sbin* and *lib* directories in */usr*. What is the difference with their root-hanging cousins? Not much nowadays. Originally, the */bin* directory (hanging off of root) would contain very

basic commands, like `ls`, `mv` and `rm`; the kind of commands that would come pre-installed in all UNIX/Linux installations, the bare minimum to run and maintain a system. `/usr/bin` on the other hand would contain stuff the users would install and run to use the system as a work station, things like word processors, web browsers, and other apps.

But many modern Linux distributions just put everything into `/usr/bin` and have `/bin` point to `/usr/bin` just in case erasing it completely would break something. So, while Debian, Ubuntu and Mint still keep `/bin` and `/usr/bin` (and `/sbin` and `/usr/sbin`) separate; others, like Arch and its derivatives just have one “real” directory for binaries, `/usr/bin`, and the rest or *\*bins* are “fake” directories that point to `/usr/bin`.

## `/srv`

The `/srv` directory contains data for servers. If you are running a web server from your Linux box, your HTML files for your sites would go into `/srv/http` (or `/srv/www`). If you were running an FTP server, your files would go into `/srv/ftp`.

## `/sys`

`/sys` is another virtual directory like `/proc` and `/dev` and also contains information from devices connected to your computer.

In some cases you can also manipulate those devices. I can, for example, change the brightness of the screen of my laptop by

modifying the value stored in the `/sys/devices/pci0000:00/0000:00:02.0/drm/card1/card1-eDP-1/intel_backlight/brightness` file (on your machine you will probably have a different file). But to do that you have to become superuser. The reason for that is, as with so many other virtual directories, messing with the contents and files in `/sys` can be dangerous and you can trash your system. DO NOT TOUCH until you are sure you know what you are doing.

## */tmp*

*/tmp* contains temporary files, usually placed there by applications that you are running. The files and directories often (not always) contain data that an application doesn't need right now, but may need later on.

You can also use */tmp* to store your own temporary files — */tmp* is one of the few directories hanging off `/` that you can actually interact with without becoming superuser.

## */var*

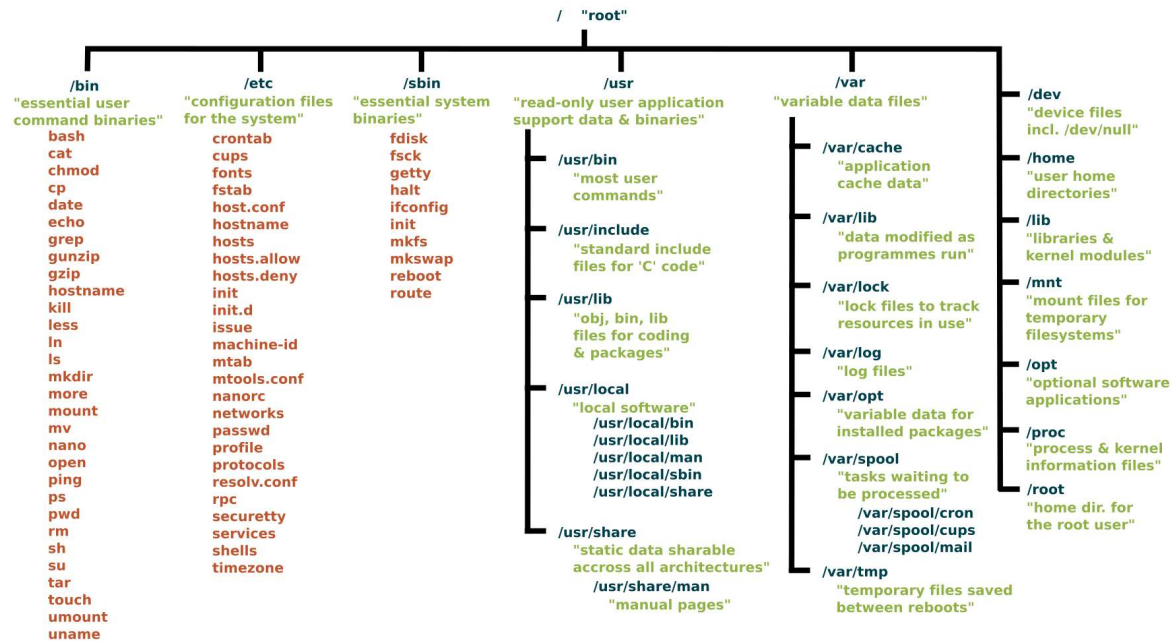
*/var* was originally given its name because its contents was deemed *variable*, in that it changed frequently. Today it is a bit of a misnomer because there are many other directories that also contain data that changes frequently, especially the virtual directories we saw above.

Be that as it may, */var* contains things like logs in the */var/log* subdirectories. Logs are files that register events that happen on the system. If something fails in the kernel, it will be logged in a file in */var/log*; if someone tries to break into your computer from outside, your firewall will also log the attempt here. It also contains *spools* for tasks. These “tasks” can be the jobs you send to a shared printer when you have to wait because another user is printing a long document, or mail that is waiting to be delivered to users on the system.

Your system may have some more directories we haven’t mentioned above. In the screenshot, for example, there is a */snap* directory. That’s because the shot was captured on an Ubuntu system. Ubuntu has recently incorporated [snap](#) packages as a way of distributing software. The */snap* directory contains all the files and the software installed from snaps.

## Digging Deeper

That is the root directory covered, but many of the subdirectories lead to their own set of files and subdirectories. Figure 2 gives you an overall idea of what the basic file system tree looks like (the image is kindly supplied under a CC By-SA license by Paul Gardner) and [Wikipedia has a break down with a summary of what each directory is used for](#).



To explore the filesystem yourself, use the `cd` command:

`cd`

will take you to the directory of your choice (*cd* stands for *change directory*).

If you get confused,

`pwd`

will always tell you where you (*pwd* stands for *print working directory*).

Also,

`cd`

with no options or parameters, will take you back to your own home directory, where things are safe and cosy.

Finally,

```
cd ..
```

will take you up one level, getting you one level closer to the `/root` directory. If you are in `/usr/share/wallpapers` and run `cd ..`, you will move up to `/usr/share`.

To see what a directory contains, use

```
ls
```

or simply

```
ls
```

to list the contents of the directory you are in right now.

And, of course, you always have `tree` to get an overview of what lays within a directory. Try it on `/usr/share` — there is a lot of interesting stuff in there.

## Conclusion

Although there are minor differences between Linux distributions, the layout for their filesystems are mercifully similar. So much so that you could say: once you know one, you know them all. And the best way to know the filesystem is to explore it. So go forth with `tree`, `ls`, and `cd` into uncharted territory.

You cannot damage your filesystem just by looking at it, so move from one directory to another and take a look around. Soon you'll discover that the Linux filesystem and how it is laid out really makes a lot of sense, and you will intuitively know where to find apps, documentation, and other resources.

**Ready to continue your Linux journey?** Check out our free [intro to Linux course!](#)

## Stay Connected with the Linux Foundation

First name

Last name

Email\*

In addition, I would like to receive marketing emails about news, events and training from The Linux Foundation and its Projects. I understand that I can unsubscribe at any time.

☐ I agree to receive other communications from The Linux Foundation.

By submitting this form, I acknowledge that my information is subject to The Linux Foundation's [Privacy Policy](#).

By clicking submit below, you consent to allow The Linux Foundation to store and process the personal information submitted above to provide you the content requested.

protected by reCAPTCHA

[Privacy](#) - [Terms](#)[SUBSCRIBE](#)

## ABOUT THE LINUX FOUNDATION

The Linux Foundation provides a neutral, trusted hub for developers to code, manage, and scale open technology projects.

[About the LF](#)[Leadership](#)[Careers](#)[Corporate Members](#)[Diversity & Inclusivity](#)[Brand Guidelines](#)[Contact Us](#)[Store](#)

## PROJECTS

[View All Projects](#)[Host Your Project](#)[Security](#)

## NEWSROOM

[Press Releases](#)

## LF RESEARCH

[Latest Research](#)[Leadership &  
Advisory Board](#)

## LFX PLATFORM

[LFX Home](#)[LFX Tools](#)[LFX Community  
Forum](#)[Create an LFX  
Account](#)

## RESOURCES

[Blog](#)[Publications](#)[Open Source Guides](#)[Webinars](#)[Case Studies](#)

## EVENTS

[Upcoming Events](#)[Sponsor an Event](#)[Submit a Talk](#)[Code of Conduct](#)

## LF TRAINING

[Training Home](#)[Course Catalog](#)[Training Resources](#)



Copyright © 2024 The Linux Foundation®. All rights reserved. The Linux Foundation has registered trademarks and uses trademarks. For a list of trademarks of The Linux Foundation, please see our [Trademark Usage](#) page. Linux is a registered trademark of Linus Torvalds. [Terms of Use](#) | [Privacy Policy](#) | [Bylaws](#) | [Antitrust Policy](#) | [Good Standing Policy](#) | [Generative AI Policy](#)