Published on Free Software Magazine (http://www.freesoftwaremagazine.com)

A beginner's introduction to the GNU/Linux command line

An introduction to the command line for novices that teaches some simple commands such as ls, cd and pwd and explains how to learn more

By Rosalyn Hunter

So you have decided to try a free software operating system such as GNU/Linux, congratulations. GNU/Linux is not that different from other operating systems on the surface. You point and click using the mouse and call down menus to get programs to work.

However, these icons and windows are just the sweet candy coating on top of a much older system, a system of programs designed to be accessed by the *command line*.

If you know the correct commands, then you can start any program, check your computer's status, and see what files you have stored without having to find the listing in your menu

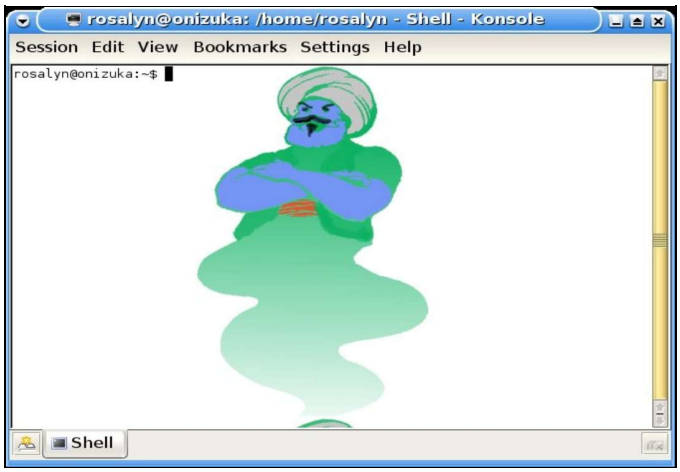
The command line is an interface that allows you to talk directly to your computer using words called *commands*. If you know the correct commands, then you can start any program, check your computer's status, and see what files you have stored without having to find the listing in your menu. Also, some functions can only be accessed through the command line, so if you want to truly understand your new system it is worth learning.

Finding a terminal program

To access the command line you will need to open a *terminal* which will allow you to talk directly to the computer. Look for a program called Xterm, terminal, Konsole, console or something similar. Since different versions of GNU/Linux have different menus, this may take a bit of searching.

The icon for a terminal is usually a computer screen, and different windowing systems will have their preferred terminals. Gnome has a gnome-terminal and KDE has Konsole. On my Debian Linux system, I found a terminal called Xterm under Debian —>Xshells.

First, open a terminal on your computer screen. It will usually list the computer name, or *yourname* @ *yourcomputer* and there will be some kind of punctuation mark like a `$` or a `#` followed by a blinking box or line-shaped cursor. That blinking thing is called a *prompt* because it is prompting you to give it a command. Think of it as a genie coming out of a lamp asking you. "What is your command my master?"



What is your command my master?

If you are at your computer now, you may want to follow along. Open a terminal and then type `whoami`. (Remember to run all of the words together with no spaces, and press enter.) Here's what you get:

```
$ whoami
rosalyn
$
```

This may seem to be a bit of a philosophical question to be asking your computer, but don't worry. Your computer has a very concrete world-view.

It should return your login name. The word that you just typed is a *command*. The computer executed the command and returned a result, your username.

Using commands

Commands are written in a particular way. The command is typed first with no spaces in the name. Then after a space, you can sometimes modify the command by adding what are called *options*. Options change or limit the way the command is executed. Options are usually preceeded by a dash. A command may also include the name of a file or directory that you want the command to work on. The finished command will look something like this.

```
command -option file
```

The best way to understand the command line is to use it. Try typing `ls`.

Remember that GNU/Linux is case sensitive so you must use lower case. When you press enter, the computer will return to you a list of file names in the directory that you are in. The command `ls` stands for *list*.

When I type this command, I get the following response:

```
$ ls
captions.txt  photo1.png
$
```

Files are documents, images or programs on your computer. *Directories* are like boxes that hold the files. You are always considered to be within some directory. Most of the time, commands only act on the files within the directory in which you are located. So, when you list using `ls`, you get a list of all of the files in the

directory that you are in, not all of the files in the computer.

When you use the command line, you are learning to talk "face to face" with your system. You can tell it precisely and concisely what you want it to do. There is something truly freeing about this power

Command names are usually shorthand ways of representing a word, such as `ls` for *list*, or `mv` for *move*. Sometimes, however, they are oddly abbreviated. Using the command line requires you to learn the names of the commands. The advantage is that by learning the language you can talk directly to the computer without needing a point and click interpreter program explaining what you mean.

When you type `ls` the names of all of the files in your current directory are written on the terminal screen, and the prompt returns waiting. It is asking, "What next my master?"

Let's ask the same question again, but we will modify it using an option. Type `ls -a`:

```
$ ls -a
.  ..  captions.txt  photo1.png
$
```

Remember to add an empty space between the command and the option. The `-a` option in this instance stands for *all*. This command prints a list of files and directories, only it adds some files at the beginning that start with a period. (We call that period a "dot".) These "dot" files are hidden files, and you normally are not meant to see them. That's why they didn't come up the first time you typed `ls`. Even though they have been hidden, nothing malicious is meant by hiding them. Dot files are usually configuration files or other files used by the computer. They are hidden to keep you from modifying them by accident.

Now that you can speak the computer's language, you can ask it to show you things as they really are. This is part of the power of the command line, and the reason that most advanced GNU/Linux users prefer it. When you use the command line, there is no intermediary program only telling you what you need to know. The command line does not treat you like an idiot.

The command line does not treat you like an idiot

The option `-a` showed you all of your files. There are many other options for `ls`. The `-p` option helps you tell the difference between a file and a directory by writing a slash beside each directory. The `-l` option will print a table of your files that describes not only the name, but the size, the date created, the permissions, and who owns the file. You can type each of these options separately, or you can combine them. To see `ls` with all of the options that I described, type `ls -alp`. The following is what I get:

```
$ ls -alp
total 192
drwxr-xr-x   2 rosalyn rosalyn   4096 2006-01-13 00:45 ./
drwxr-xr-x  48 rosalyn rosalyn   4096 2006-01-13 00:45 ../
-rw-r--r--   1 rosalyn rosalyn      8 2006-01-13 00:12 captions.txt
-rw-r--r--   1 rosalyn rosalyn 178839 2006-01-13 00:45 photo1.png
```

If you want to know what all of the options for a command are, and what they stand for, type `--help` after the command. It won't work for every command, but it is often the easiest way to learn. When you type `ls --help` the computer tells you how to order the parts of the command and it lists the options for `ls` explaining what each option does. You may notice that many options have two forms. The short form has one dash and a letter, and a longer form has two dashes and a word. You use whichever one is easier for you to remember. Therefore you could have typed `ls --all` to get a list of all files including the dot files. It is an

identical command to `ls -a`.

Getting help

You can learn commands in many ways. Sometimes you will see them mentioned in articles, or someone might tell you of them. When you learn of a command, it is best to try to find out what it does BEFORE you use it.

The computer assumes that you know what you are doing, and so it will happily erase the entire disk if you ask it to, without pausing to ask if you really meant to.

My name is Mud

Back in the cave days of personal computers, while trying to learn the Apple Disk Operating System, I created a file called *mud* and did all kinds of commands on it. To learn what the command `init` did, I typed `init mud`. I expected it to do whatever `init` did to a throw away file named "mud". Instead, it erased my drive, and the only thing that came up when I started my computer was the sentence, "my name is mud".

Don't make the same mistake I did. Always look up a command before you use it for the first time.

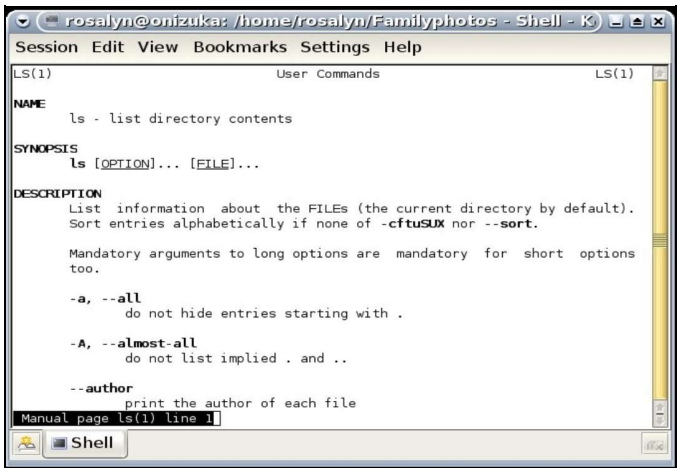
So how exactly can you look up what a command does? One of the simplest ways is to use the command `whatis`. Try typing `whatis` and the name of the command in question. It will return a short explanation of a command. For example, when you type `whatis mv` you get:

```
$ whatis mv
mv (1)               - move (rename) files
$
```

This is fine if you want to know simply what a command does. But if you want to know all of the options as well as seeing examples and more complete descriptions, then you need to look at the manual pages.

Type `man` and a command name and a page will open up giving you lots of information about that command written in paragraph form. `man` stands for *manual* and most commands will have a manual page, although some are more complete than others.

If you type `man ls` and press enter, then the terminal will fill the screen with the `ls` manual page. Hit the space bar to cycle through the pages. When you reach the end of the file, it will usually drop you back onto the command line, but if it doesn't just hit the letter `Q` (quit) to exit.



The man page for the command ls

There is also another system of help pages found on your computer called `info` pages. `man` and `info` pages are similar in content, although they vary a bit in how they are displayed. The display system in `info` pages is based on an editor called emacs. `man` pages are an older system. Even so, you might want to try both. I suggest that you try the following commands:

```
man man

info info
```

The first command takes you to the manual page for the command `man`, and the second takes you to the information page for the command `info`.

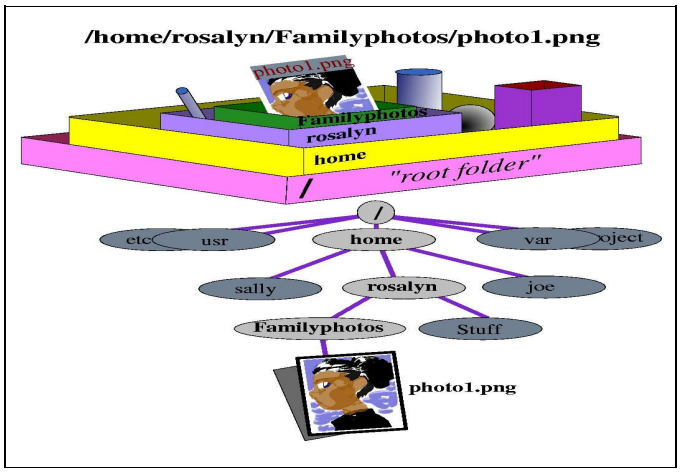
If you want to know all of the options as well as seeing examples and more complete descriptions, then you need to look at the manual pages

You may need to type a special command to exit from an `info` page. Try pressing the *control* button and the `C` button at the same time (Ctrl-C) or press *control* and `Q` (Ctrl-Q) to exit.

Navigating the file system

The files in a GNU/Linux system branch from large to small like the branching roots of a tree. The trunk of the tree is a *directory* called root represented by a forward slash (`/`). This directory will have a number of subdirectories with names such as *bin*, *lib*, *usr*, *home*, and *etc*.

Another way to think of a directory is as a box. The box may contain objects called files, or it may contain other boxes. Each of those boxes may contain other boxes as well. Any location in this "file system" can be described by listing the boxes in order from the biggest box (`/`) to the smallest.



Your file tree is like a set of nested boxes.

Let's say I have a file called `photo1.png` which is a picture of me. I can put it in one of the boxes. Then if I want to explain where the photo is, I might write a list of box names such as

```
/home/rosalyn/familyphotos/photo1.png
```

This is the complete filename of *photo1.png*. The list of boxes that tells where it is located (`/home/rosalyn/familyphotos/`) is called the *path*. If you are coming from a Microsoft Windows system, please notice that there is no drive letter in the path name.

To find your current location type the `pwd` command:

```
$ pwd
/home/rosalyn/Familyphotos
```

`pwd` stands for "print working directory". It will print your current path telling you where you are in the file system.

You can navigate around the file system using the `cd` (change directory) command. Type `cd` followed by the name of the the directory that you wish to go to. If you type it without a destination it will return you to your home directory.

Type `ls -a` again:

```
$ ls -a
.  ..  captions.txt  photo1.png
$
```

The first two files listed are "`.`" and "`..`" called "dot" and "dot-dot". The directory that you are currently in is the "dot" file, and the directory one above you is called "dot-dot". Therefore, to go up to the next higher directory, you type `cd ..` (the letters "cd" followed by a space and then dot-dot).

Once you learn the language, a whole new world is open to you

First, let's check our location with `pwd`. Then we will move up the file system and check it again:

```
$ pwd
/home/rosalyn/Familyphotos
$ cd ..
$ pwd
```

```
/home/rosalyn
```

Notice that your pathname is shorter. If you started at `/home/username` now you will be at `/home`. Using `pwd` (print working directory), `ls` (list), and `cd` (change directory) you can now go all around your file system finding where everything is located.

More command line help

Try some of the following websites for more command line help.

- Linux command.org
- Tuxfiles Linux command line tutorials for newbies
- Wikibooks Linux for Newbies guide
- Introduction to Unix
- UNIXhelp for Users
- Digital Unix Command and Shell User’s Guide

Enjoying the command line

When you use the command line, you are learning to talk “face to face” with your system. You can tell it precisely and concisely what you want it to do. There is something truly freeing about this power. You can `kill` programs that hang up your system. You can view the `history` of the last several commands that you have typed, or a listing of the `top` twenty jobs running on your computer at the time. Once you learn the language, a whole new world is open to you.

The point and click graphical interface limits you to what the designer of the system believes that you need. Language is still a much more powerful tool. When you learn to talk the language that your computer understands, there is no middle man. Try it and in time you too may learn to love the command line.

Biography

Rosalyn Hunter (/user/36" title="View user profile.): Rosalyn Hunter has been on the internet since before the web was created. Born into a family of instructors, she has made it her life's goal to teach others about the important things in life, such as how to type kill -9 when a process is dead. She lives in a little house on the prairie in the American West with her husband, her three beautiful children, a cat and a dog.

Copyright information

Source URL:
http://www.freesoftwaremagazine.com/articles/command_line_intro

---