

# Forum:[Hack Forums](#)

Thread: [The Complete Linux Beginner's Guide v1.0](#)

**Faith Connors** - November 6, 2016, 12:35 pm

The Complete Linux Beginner's Guide v1.0

Written by Faith Connors for HackForums under the GPLv3 license

Editted by Faith Connors

Inspired by a Friend

## Table of Contents

### 1. Introduction

Who am I?

What is Linux?

Why should I use it?

### 2. The Basics

The UI and terminal

The Package Manager

Distributions

### 3. Simple Commands

Navigation

Every day tasks

Installing packages

### 4. Advanced Commands

Special Tools

Options

Compiling Software

## 5. Programming on Linux

### Text Editors

IDE

The GCC and Other Compilers

## 6. Customizing Linux

### Window Management

Colors

Application

## 7. Faith's Recommended Programs/Packages

### Daily Use

Development

Customization

Security/Hacking

## 8. Closing Words

### Thanks

Support

Feedback

Introduction

Who Am I?

You may have seen me on the forums before if you browse the lounge occasionally or if you drop into the beginner's hacking section here. I post a lot and try to help people when I can. Sometimes people ask for help with tools I know, sometimes more technical stuff, everyone usually has something to ask or share here. It's a nice forum and I want to give back even more than I already have. So I decided to write up this guide on the joys of Linux.

I started using Linux when I discovered BackTrack, a penetration testing distro which is now outdated. I didn't really know what I was doing most of the time, just repeating commands I had written down somewhere. I didn't really learn anything about the distro or using Linux as a whole. But in school, some of the computer labs started to use Linux for Computer Science and Computer engineering, so I dropped in on some classes, later took them and learned a lot more about Linux. I

grew to like it and Now I Dual boot on all my devices. It's a very simple operating system to use once you understand the workflow.

### What is Linux?

Linux is an operating system for your computer. Think of it as a program like Windows, it runs all your programs on it, and basically houses everything running on your computer. If we want to go more specific, Linux is simply a kernel. A Kernel is a very low-level program that interacts with the CPU in your computer to do all kinds of tasks. It allows the operating system to call up the motherboard, use all the other parts of your computer like the GPU and more. The kernel is the most important part of the operating system and that's what Linux is.

Many people are intimidated when they look at Linux. They usually see a black screen with all kinds of letters, numbers and words that make very little sense to them. It can be hard at first to learn everything, but that's why I decided to write this guide. I wanted to make a good starting point to learn Linux quickly and efficiently, in a friendly manner.

### Why Should I Use It?

Linux is simply more stable, gives you more freedom, and that can help you take control of your system. Windows has all kinds of limitations. You know when you get a program that isn't responding and it just turns white and you can't close it on Windows? In Linux, when you tell it to kill a process, it kills it right away, no questions asked. Linux is also safer to use because less people write malware for Linux than Windows, so you'll have a much smaller chance of getting infected by a RAT or Keylogger.

### The Basics

#### The UI and Terminal

If you are using one of the recommended distros I mentioned a few lines below, I want you to boot it up and start to explore the interface. You will notice that it's very similar to Windows, everything has its own window, you have a Start/Applications menu, and a taskbar/dock somewhere on the screen for doing certain tasks like turning off your computer, setting up WiFi, and volume controls. The main difference with Linux compared to Windows is:

1. How stuff looks
2. The Terminal

Things look different in Linux, generally a little prettier and smoother. Sometimes things will be different than from Windows, like if you are on ElementaryOS, you will see the Close, Minimize and Maximize/Window buttons will all be on the left side of the Window, like MacOS or OS X. But most of the features your current operating system has, Linux will have as well. This is because of two services running in the background. One of those is the X server. This service is responsible for anything visual, it creates Windows and other boxes. The other service running is something we call the Desktop Environment.

The Desktop Environment is also for visuals, but it's a little bit more specific. It's what adds colors to everything, it actually makes the interface that the X server is generating. It's what actually displays stuff on your screen. You may have noticed that when downloading your Distro, you had different options, like KDE, GNOME, LXDE, XFCE, and so on. These are all different types of Window managers and these are also probably the easiest to use.

LXDE is lightweight and good on resources

XFCE is a little heavier but also very minimal resource usage.

KDE is for eye-candy. May take up a little more RAM but is worth it.

GNOME is a different KDE basically.

Unity is Ubuntu's desktop environment. It has lots of eyecandy and nice visuals, but is very heavy and personally not my favourite.

Now, getting away from the looks of Linux, we have the Terminal.

The Terminal is the barebones side of Linux. It's that big black window with all kinds of numbers and letters that you might not understand now. Slowly but surely, we'll be working through it and figuring out what everything is.

If you ran Linux without any x server or desktop environment, all you would see is a black screen with all the letters and numbers and words. You would basically just have a terminal that does not support any extra visuals.

Many terminal programs are also very highly customizable so you can make them look a little cooler or less intimidating. I'll talk about that more later.

### The Package Manager

Installing programs on Linux isn't always as easy as double clicking an exe and walking through the install wizard like on Windows. You might have to type some extra stuff in and maybe configure programs later. Most of it is done through the terminal, but there are programs called Package Managers that help you do it quickly, efficiently, and have many more features.

If you've ever tried to uninstall anything on Windows, you'll know you have to go into control panel and find all your programs. This is basically Window's package manager. You can uninstall programs, change them, see everything you have installed, etc. But it's still lacking. From Control Panel you cannot install programs or packages, there are other bugs like not showing how big a program actually is, and you don't always see every program installed on your computer. The Linux Package Manager is much better for all of this.

Now, before I go further, I want to mention there isn't just one package manager. I will try to document them all later on, but I just want you to know that there is more than one. There are popular ones like Apt for Debian and Ubuntu based versions of Linux, and there are less known ones like Pacman for Arch and NPM for Node.JS.

### Distributions

A distribution is basically a version of Linux that comes with all kinds of stuff preinstalled so you don't have to put very much effort into setting it up yourself. Sometimes this is good because you can jump right into it and not have to worry about much. But other people, like me, prefer lighter distros with very little installed so we could spend some time configuring it to our needs and making sure it is built exactly the way we want it.

Some popular Distros I would recommend for this guide are:Ubuntu

Debian

Fedora

Elementary

OpenSUSE

Some lightweight distros I prefer:Arch Linux

Void Linux (HIGHLY NOT RECOMMENDED FOR BEGINNERS)

Raspbian Lite (For Raspberry Pi)

Penetration Testing Distro: (Please do not use these for this guide) Kali Linux

ParrotOS

BlackArch

ArchStrike

## Simple Commands

Before we go into it, open up your terminal. You'll see a line already, and it will look something like:  
Code Faith@RUNNER:~\$

Now don't worry if it's not exactly the same. The first word, right before the @ sign is the user you are logged in as. In my case, I have a user named Faith on my laptop, so it says Faith there. The word after the @ and before the : is the name of your computer. When you set up your computer you will have set a computer name. This is it. In my case, it's RUNNER. Now, after the : is the current path on your hard drive. ~ is the default, and it's basically your home directory for the user. The \$ is the sign put at the very end to show the start of the command.

The terminal puts all this together to understand that a specific user on a specific machine is trying to execute a command in this directory. Not quite making sense? It's okay. I'll elaborate more about everything later.

## Navigation

We're gonna start learning about the terminal, so open up your terminal on your Linux distro.

We're going to start very simply, first about how to navigate your computer and understand the outputs of different commands. When you open your terminal you'll see that line up there I mentioned. That means you can start typing in a command. Let's start with the basics. Try this:

## Code ls

Commands are case sensitive, so make sure you don't use any capitals here or it will not work. If you typed it right, you should see all kinds of words pop up on your screen. Stuff like Documents, Pictures, Downloads, Desktop, etc. Basically, this command, ls, lists all the files and folders in the current directory. You might also notice that some of the words are in a different color or are in bold. This can be customized from your terminal and it could be different on your machine, but the different colors and typefaces means that these files or folders are different. Now, try this command:

## Code cd Desktop

Again, make sure it is case sensitive. The cd should be lowercase and the Desktop should start with a capital. Once the command is done you'll see now that little line at the beginning is a little longer. It'll say ~/Desktop. That's because now we changed folders to the home directory to the Desktop folder inside the home directory! The cd command is to navigate through the filesystem. Let's try it one more time, but this time, type this in:

## Code cd ..

Instead of Desktop, we'll use two dots, .. and now you'll see the short line at the beginning went back to normal. This is because we can move back directories with the cd command as well! So you can move back and forward, and inside each directory you can use the ls command to see all the files and folders inside of it! Pretty cool, huh.

The Filesystem for Linux is very different from Windows, but rather similar to MacOS. Instead of having a C: drive for booting and all that base stuff, the drive is simply a slash, /. Putting this after the cd command will bring you to the root of the filesystem. Your home folder is always going to be under /home/[USER] where the user is your username. The home folder is right in the root of the filesystem, so we have a slash in front of it, which tells the cd command we want to start the path from the root of the filesystem.

We can also use extended paths in the terminal. Instead of using the command twice to get from / to /home/Faith, we can just write:

```
Code cd /home/Faith
Which will bring you right back to the beginning.
```

## Everyday Tasks and Launching Programs

To do most of these commands, make sure you are in the correct directory/folder, just like you would be in Windows Explorer or whatever file manager you use on your device.

Let's start with some essential commands before going into programs. To make an empty file, we use the touch command.

Code touch [FILE] And the command will create an empty file with the [FILE] as the name. You can also specify extensions, so if your [FILE] was tutorial.txt then it would make a blank text document called tutorial.txt.

You can specify a path to the file as well, like "touch /home/Faith/tutorial.txt".

To open files in terminal to see what they say, we use the cat command.

```
Code cat [FILE]
```

So if I had a file called tutorial.txt which had "hello world" in it, when I type in "cat tutorial.txt" the terminal would say "hello world".

You can specify a path to the file as well, like "cat /home/Faith/tutorial.txt".

To change permissions of a file, we use the chmod command

```
[code]chmod [OPTIONS] [FILE]
```

This one is a little harder, but it's something you will use a lot. You'll see the [OPTIONS] tag which is for adding options to the command. So if I had a file called shell.sh which would be a shell script and it didn't run, I could assign a property like +x as an option to make it an executable file. So my command would look like

```
Code chmod +x shell.sh
```

You can specify a path to a file as well with this command, just like the others.

Now, sometimes you will run into a problem where you can't run a program or command because you need something called "elevation" or more "privileges." This is because the account you are using isn't exactly the administrator account. Linux has a secret account built in called the "root" account which will by default have the same password as the account you are currently using. So to run programs from this account and have maximum privileges, we put sudo before the command

```
Code cat /etc/passwd
```

```
ERR21: Requires Elevation
```

```
sudo cat /etc/passwd
```

Type in password for root:

So to use sudo, you need to have the password to the account. and once you type it in, you won't have to type the password in again until you close and reopen the terminal. (You will still have to type sudo before each command if needed.)

To get around having to type sudo again, we can use the superuser command to log into the root account and run everything elevated by default.

Code `sudo su` The su command is to login to root, but most of the time you will need to run that command with elevation. Once you type in the password, you will see this:

Code `Faith@RUNNER:~$` Change to

Code `root@RUNNER:/home/Faith$` The root home directory is simply the base of the filesystem, just / and no ~ or anything.

I do not recommend running everything from superuser all the time (after typing `sudo su`) as it can be a huge security flaw if your computer gets compromised. Remember, I said Linux can be safe because of less malware for it, but remember, it still exists, and it can be just as bad.

To run a script file, we put a ./ in front of the file name. If I have a shell script called `shell.sh`, I would run it with `./shell.sh`

To run a program, usually you just have to type the package name, and maybe options if needed. I'll explain the basics of the package manager now.

## The Package Manager

Remember what I said about package managers? They will make your life much easier.

The most common ones I find are apt and yum, although different distros might have their own, like Pacman for Arch and Nix for NixOS.

So let's start with apt because it's the most common and probably what you're using.

Running the package manager will almost always need elevation, so remember to type "sudo" before each command.

To install packages, we have to find the name of a package first. You can just look up the package on google and you might find it. Many of them are very simple. To install them, we type in:

Code `apt-get install [PACKAGE]`

OR

`apt install [PACKAGE]` Either using apt-get or apt work, but have a look here if you want to learn the difference: <http://askubuntu.com/questions/445384/what-is-the-difference-between-apt-and-apt-get>

To remove a package that we have installed, we use

Code `apt-get remove [PACKAGE]` Again, either apt-get or apt work.

To update the list of packages we have in case there are updates or new packages in the sources, we use:

Code `apt-get update`

To upgrade or update all the packages we have installed, type:

Code `apt-get upgrade`

To see all the packages we have installed, type:

Code `apt-get list --installed`

Now, this one is important, and not many of my friends use it often for some reason. It's called autoremove. Sometimes when upgrading a package, the dependencies and libraries it needed are no longer required or older versions are left installed on your system. They take up extra space and can sometimes even be security flaws, so to get rid of all these libraries being used, we have a special command:

Code `apt autoremove` apt-get will not work for this one, you must use just apt.

To launch a program you installed through apt or whatever package manager you are using, try just typing the package name alone. If I installed a text editor called Atom (which I'm using to write this and I highly recommend,) I would simply type "atom" into the terminal and it will launch.

If you've made it this far, congratulations! You should have learned enough to use Linux as a normal operating system. However, if you are more of a power user and want to learn more, keep reading. It's only starting to get fun.

## Advanced Commands

### Special Tools

Okay, you decided to keep going, and that's great!

Some tools and packages will be a little harder to use. For example, let's use Python, the programming language. Python itself doesn't do anything, when you install Python from the terminal or from the site or whatever, you can run it as "python" but it won't do anything. This is because it needs a script or program to run. So we would have to type "python script.py" to run the script, and we might even need to add more to that if the script has arguments and options too.

Now, other tools like IRC clients or dstat will also run from the terminal and will run in the terminal, but will look different and maybe be harder to use. For these kinds of tools, I suggest reading their documentation and manuals. I can't cover every tool there is, I still have a character limit here ;)

### Options

You'll have seen that I have thrown around the words "options" and "arguments" a little bit. These are just extra pieces of text you type in with a command to tell it exactly what you want it to do. Options and arguments usually go after a command and start with a - or /. For instance, with the ls command we used before, it can run with options too. "ls -l" will show the permissions of all files and folders, "ls -a" will show hidden or ignored files and folders, and you can combine them as "ls -l -a" or the shorthand version, "ls -la"

Different commands and different packages/programs will have different options. The only way to know what options are available to use is if you read their manual or documentation. Remember, wherever their home page is, or google is your friend. Sometimes they will also have a help dialog, by default it's -h or -help.

### Compiling Software From Source

Linux is an open source operating system, meaning you can modify the code in any way you want. As a result, many of the programs and packages you get are also open source. But if you can't get the package from apt or your package manager, then you might have to download the source and build it yourself.

Usually the place you download the source from will have a guide on compiling from source, but if there isn't one, chances are you will use the GCC, the GNU Cross Compiler, built into most versions of Linux by default. So when you download your source, it will probably be in a zip file or a tar.gz file. Extract it and navigate into the folder with your terminal.

make is a utility to compile source easily and usually hassle free. You might get errors sometimes, but once in the folder with all the source (use ls to make sure you're in the right folder first ;) ) simply type "make" and you will start to compile it.

If all goes well, now you just have to type "make install" The install option at the end lets the compiler know you want to be able to run that program from anywhere on your system, and not just in its folder. Sometimes you might not need this or want this, it's optional.

## Programming on Linux

You might hear people always say that programming is better on Linux. If you're a developer and you've always used Windows or MacOS, you might wonder why you should switch and what makes it better and how come you're doing perfectly fine.

There are a lot of answers. Linux is very lightweight and has much more freedom over what you can do with it, so naturally you can experiment more with memory allocation, get down and dirty really well and have fun with it. Many IDEs that you are probably already using (like Eclipse) are gonna be compatible with Linux so you won't be missing there, but at the same time it also depends on what language you want to program in. Linux is, I find, the best for writing in C and native languages because of the GNU Cross Compiler which I'll talk about later. It's quick, simple and well optimized, but there aren't any compilers available for any .NET languages simply because the runtimes aren't available. If you are a .NET developer, maybe you should stay with Windows, but for almost every other language, trust me, Linux will be your best friend.

### Text Editors

Text editors are nothing new if you've programmed before. Actually you'll probably need them for doing all kinds of things on your computer. Anyway, as far as text editors go, I've classified them into two categories.

We have command-line text editors, so they run in the terminal, and we have the GUI text editors. Personally I like GUI text editors more but terminal based ones are good for just doing a quick edit or just getting down and dirty for something.

For terminal based editors, the two best ones I know of that are very widely used are:

Vim (Vi)

Nano

Vim is much harder to use, everything is based on keyboard shortcuts, like pressing x to remove a character, i to insert text and so on. It's harder and more complicated, but power users who use other shortcut based applications will like this better.

Nano is much more userfriendly, basically the notepad from windows but it runs in the terminal. It has some shortcuts, like to exit/save/rename, etc. but you won't have any trouble picking up on them.

Utilities like Git will require you have a terminal text editor, so you should have one installed even if you don't like using them.

For GUI based text editors, there is only one that I really like. It's a little bit big, but completely modular, looks great, has syntax highlighting, extra packages from the community, and runs fast. It's called Atom. I'm using it to write this guide before I upload it. It's really great. But there are alternatives available, I'll give you some of the best ones.

Sublime Text

Atom

Gedit

Try one of these, they're all okay. Sublime is very popular as well. I'm just a hipster so I use Atom. (Also I like all the community packages :^))

## IDEs

Of course, you might want an IDE rather than a text editor to program with. For that, there are three main options you have.

- Eclipse
- Geany
- Brackets

Eclipse and Geany have support for a ton of languages and are great, but Brackets is more centered for web development and design.

Atom is also able to manage classes and multiple files in Git and all that so it's great for doing IDE stuff.

## Compilers

These text editors and IDEs don't have any compilers installed, so if you want to write something in Rust or C, you will need a compiler to do it.

The GCC (GNU Cross Compiler) is the default compiler for anything C-related. It is incredibly well optimized and is pretty much what you will be using for a lot of things.

For other languages you will need to install the compiler or interpreter or support for your language, like ruby, python, or Rust.

## Customizing Linux

Remember what I said about Linux being extremely customizable because it's open source and you can change the interface however you want? I'm gonna start talking about that here. You're gonna have a lot of fun here, making sure everything is exactly the way you want it. Sometimes you will need to know some programming knowledge or how to configure stuff in the config files, and you most likely will run into problems sometimes, but in the end, you will end up with a masterpiece that looks as beautiful as it is functional.

## Desktop Environments and Window Managers

I listed some Desktop Environments at the beginning, but now we're gonna go deeper. I want to clear up something first. Desktop Environments are just compilations of packages to get the job done. It will include a Window Manager, a Display Manager, a File Manager and a Session Manager, along with some other defaults like a terminal. Let's look at Window Managers first (abbreviated WM). There are three types of Window Managers:

- Tiling
- Stacking
- Composting

Tiling WMs don't allow windows to overlap, they will be border to border and will almost never require mouse interaction, as much of it will be based on keyboard shortcuts. They can still be very pretty but hard to use for beginners.

- i3
- Awesome
- Dwm
- bspwm
- Xmonad

## WMFS2

i3 is personally my favourite Tiling manager. Awesome is good too.

Stacking WMs are similar to Windows, or MacOS, in the sense your desktop is literally a desk and each window is a sheet of paper, and you can "stack them" so they can overlap. These are the most popular as they resemble interfaces that the user understands better.

- Openbox
- Blackbox
- Fluxbox
- WindowMaker

I personally use Openbox because it is very customizable and has a large variety of themes available online.

Compositing WMs are a little bit more complicated. They give a buffer for each window to draw on and compose the buffers to make the desktop image. If at first you don't understand it, don't worry about it. It's just a little slow and I don't recommend using this kind of WM.

## Colors

Colors are what will give your Distro some flair.

I like to keep it very minimal, using dark colors to be easier on my eyes. The way to customize your environment will be based on the environment or WM you have installed, you will have to configure it there. I can't guide you very far with this, only giving you some good places to find good colors to use.

I like to use <https://flatuicolorpicker.com> as they have a huge collection of colors that make a great flat interface.

Dafont is a great place to find good fonts.

The rest, well, look up how to customize your WM or DE and find some graphics tutorials. Sorry.

## Applications

Many applications will have preferences and options to be able to customize their appearance as well. This isn't limited to text editors, you can do it for your terminal as well. Want that green on black hacker look? Add some transparency to it too to make the window see-through too! Who knows, maybe change the font to something easy to read for the hell of it! Terminator is a great terminal that allows you to do all of this. It's probably the most widely used Terminal package/program you can install.

So you can customize certain applications too, and if you can, make use of it!

**Faith Connors** - November 6, 2016, 12:37 pm

## Faith's Recommended Packages and Programs

### Daily Use Programs

- Office Suite (LibreOffice)
- Gedit (text)
- GIMP (photo editing and drawing)
- Terminator (Terminal)

## Development

- Terminator (terminal)
- Atom (Text Editor IDE)
- Python Interpreter
- Any compilers
- PHP5
- Any additional language runtimes you need (Java, Ruby, C++)
- Eclipse

## Customizing

- OpenBox
- i3
- Terminator
- feh

## Security/Hacking

- Wireshark
- nmap
- Metasploit (requires Ruby)
- oclHashcat
- Atom
- Bettercap (or MITM framework of your choice)
- OpenVPN
- TOR
- IceCat
- VeraCrypt

## Closing Words

### Thanks

Thank you for reading this guide, congratulations if you made it to the end. Maybe you learned something, maybe you're a new user who still has no idea what they're doing, but that's for reading, I really put a lot of time and effort into making this and I appreciate the fact that you took the time and effort to read through it.

### Support

I'm a little bit of a Wizard when it comes to a lot of Linux things, so I could potentially help you out if any problems arise. Just send me a PM.

### Feedback

Hey, let me know what you thought of the guide, I'd love to hear from you. Also, give me any suggestions and I'll add them maybe. Also if you notice any errors or bad info, don't be afraid to call me out. Either post here or PM me, everything is appreciated.

**05317832892** - November 6, 2016, 12:39 pm

Nice tutorial Faith, seen you around a lot recently. Some beginners will love this tutorial, they will

probably beg for a kali one though.

**Faith Connors** - November 6, 2016, 12:41 pm

(Nov 6, 2016 01:39 PM)?Cam? Wrote: [\[CITE\]](#) Nice tutorial Faith, seen you around a lot recently. Some beginners will love this tutorial, they will probably beg for a kali one though. Sadly, but oh well. Kali is only as good as how well you use the tools.

**semendaemon** - November 6, 2016, 12:58 pm

When you talk about linux commands, one thing to know is what actually IS a command and what's a program. Is is a program and not a physical command. you can view it at /bin/lis machine-wide or /usr/bin/lis gid-specific. You're only able to execute it due the path variables (which you can print using /usr/bin/env, and add using export) that dictate /bin/ & /usr/bin to be localized for program calling regardless of your terminal PWD.

How do we make the distinction? We undersrtand that a shell like /bin/sh, /bin/dash or /bin/bash is just a nice little wrapper to execute a langauge called Bash, and inside Bash you have an extensive amount of assosiated tools and programs that can easily coorelate themselves via Bash's STDIN/STDOUT piping - But make no mistake, bash doesn't actually implement Is or echo or cat. Bash simply calls these programs, all of which are GNU-made and part of the userland gnuutils signed GPL3.

Won't be pedantic, but clarifying Linux being the kernel and not addressing GNU/Linux is a miss. Also,  
Code root@RUNNER:/home/Faith\$ is wrong, bash\_profile / .bashrc defaults uid=0 to possess the shebang sign, that'll be root@RUNNER: /home/faith~#

Anyway, solid tutorial. I'd argue everything was written hundreds of times before but hey if you wanna write it'll always be probably helpful.

**Faith Connors** - November 6, 2016, 1:02 pm

(Nov 6, 2016 01:58 PM)semendaemon Wrote: [\[CITE\]](#) When you talk about linux commands, one thing to know is what actually IS a command and what's a program. Is is a program and not a physical command. you can view it at /bin/lis machine-wide or /usr/bin/lis gid-specific. You're only able to execute it due the path variables (which you can print using /usr/bin/env, and add using export) that dictate /bin/ & /usr/bin to be localized for program calling regardless of your terminal PWD.

How do we make the distinction? We undersrtand that a shell like /bin/sh, /bin/dash or /bin/bash is just a nice little wrapper to execute a langauge called Bash, and inside Bash you have an extensive amount of assosiated tools and programs that can easily coorelate themselves via Bash's STDIN/STDOUT piping - But make no mistake, bash doesn't actually implement Is or echo or cat. Bash simply calls these programs, all of which are GNU-made and part of the userland gnuutils signed GPL3.

Good find, I'll update when I get home from work (which is like at least 6hrs)  
Admittedly I did use it interchangeably a few times.

(Nov 6, 2016 01:58 PM)semendaemon Wrote: [\[CITE\]](#) Won't be pedantic, but clarifying Linux being the kernel and not addressing GNU/Linux is a miss.

Also,  
Code root@RUNNER:/home/Faith\$ is wrong, bash\_profile / .bashrc defaults uid=0 to possess the shebang sign, that'll be root@RUNNER: /home/faith~#  
Yeah yeah I could paste the whole fookin GNU/Linux interjection too, Imao  
And I'll fix that last bit now

(Nov 6, 2016 01:58 PM)semendaemon Wrote: [\[CITE\]](#) Anyway, solid tutorial. I'd argue everything was written hundreds of times before but hey if you wanna write it'll always be probably helpful. I could say you're guilty of this one too ;)

EDIT: Ooooooh boy looks like I broke the character limit by about 18000 so I'm just gonna leave that and hope people read your reply.

**renzor** - November 6, 2016, 3:57 pm

Great guide, as a new Ubuntu user I wish I'd found this post earlier, it would have spared me lot of time on google. Very well written and easy to understand.

One thing I would add is the Code man command, it helps me to understand the other commands, not just copy/paste them.

**Faith Connors** - November 6, 2016, 7:24 pm

(Nov 6, 2016 04:57 PM)renzor Wrote: [\[CITE\]](#) Great guide, as a new Ubuntu user I wish I'd found this post earlier, it would have spared me lot of time on google. Very well written and easy to understand.

One thing I would add is the Code man command, it helps me to understand the other commands, not just copy/paste them.

Yeah reading the man page is always great and if you really need to most well known and documented tools have it available online as well.

Unfortunately I can't make edits to the post because it's 30k characters long and I can only make a 12k post so it doesn't let me edit :/ oh well

**Aakash\_Choudhary** - November 6, 2016, 8:19 pm

You know what ? You are doing absolutely great among hackforum communities. Before i had always problem that 2010-14 was the best for hackforums community as i digged lots of threads and found useful things and great helping members of that time

But nowadays i things changes. I again see some members going on those great helpful member's on footprints

Its good to see

Thanks

**Faith Connors** - November 7, 2016, 12:46 pm

(Nov 6, 2016 09:19 PM)Aakash\_Choudhary Wrote: [\[CITE\]](#) You know what ? You are doing absolutely great among hackforum communities. Before i had always problem that 2010-14 was the best for hackforums community as i digged lots of threads and found useful things and great helping members of that time

But nowadays i things changes. I again see some members going on those great helpful member's on footprints

Its good to see

Thanks

I appreciate the kind words, thanks.

Bumping back up

**haster macker** - November 7, 2016, 12:59 pm

i don't understand much about this, but when i do, i'll be back, thanks :)

**NapoleonicCode** - November 7, 2016, 1:27 pm

Very nice and informative tutorial Faith! Thanks bro, I learned some new stuff ?

**Faith Connors** - November 8, 2016, 7:49 pm

Bumping to help out new users <3

**Kundra** - November 9, 2016, 8:37 pm

Wow thx OP this is gonna be very helpful?

**Last\_viking** - January 23, 2017, 11:45 pm

Thanx faith nice tut

Cheers brotha

**TheChi?f** - January 27, 2017, 9:22 am

This is an amazing guide, Faith. Keep up the good work.

**Mom** - January 27, 2017, 9:23 am

I don't use Linux at all, anymore, but this guide will be helpful as fuck to other users, great job man!

**mirana\_** - February 27, 2017, 7:56 am

i ran to my room and got a pen and paper and started crying writing to this.

i have ubuntu on a pendrive flash and im afraid to dual boot till i get clear and concise information. also simple non-egotistical information with visual that arent videos with people who like to hear the sound of thier own voice.

i really want to thank you, we could all go a bit further with this concise logic of information in the world in any subject,module or orientation.

there may be a thousand like this, but i understood it immediately and want to give my token of appreciation. (im new so dont really know how: guilty as charged)

**Maschil\_Crane** - February 27, 2017, 11:11 am

Wow. Thanks a ton. This is very useful. I'd pay at least \$1 for this :) But really. Thanks a bunch.

**random\_stranger** - March 7, 2017, 7:27 am

I'm gonna tell you something you don't want to hear. I'm gonna show you where it's dumped but have no fear.

**chrispa** - March 8, 2017, 6:10 am

Great thank you for a job well done, Honestly the time you put in writing this material is not wasted. It is of great help to all not only beginners. This is good!

**Levi** - March 8, 2017, 6:23 am

I love this thead but I do wish you would stop self-banning out of nowhere!!

**CeejLq** - March 8, 2017, 2:51 pm

Wow, thanks for bumping this, it's pretty amazing. This explains excellently a couple of qualms I had using Ubuntu and makes the it seem a lot less complex to use Linux commands and especially the terminal than I'd thought.

**Dread111** - March 8, 2017, 8:57 pm

Really nice guide. It will really help people new at linux

**Maschil\_Crane** - March 11, 2017, 6:34 pm

I just want to thank you once again because I am now browsing hackforums on my new ubuntu OS and it feels oh so good. Ive been wanting to start using linux for a long time now Ive just been putting off learning it, but youre guide made it so simple and straightforward it's almost hard not to use it now. Thank you. Youre the greatest.

```
Code root@crane-linux:/home/crane/Downloads# sudo chmod 400
~/home/crane/Downloads/mykey.pem
chmod: cannot access '/root/home/crane/Downloads/mykey.pem': No such file or directory
root@crane-linux:/home/crane/Downloads# ls
discord-0.0.1.deb mykey.pem torguard-latest-amd64.deb
root@crane-linux:/home/crane/Downloads#
```

Someone help. I am having trouble with cmd not recognizing my file. I am using the provided tutorial to launch an instance through aws, but I keep getting a "No such file or directory" error, but as you can see when I did ls, the file does exist. Am I missing something here? Ive tried googling for the answer and tried entering the command in a bunch of different formats, but I keep getting the same problem.

No one said this would be easy....

P.S. I dont know how to make the quote box show up without the word "quote" like I see people do when they share something from the console. Its probably something simple IM just looking over but....

P.P.S. I figured out the code box bit. I just need to click the button next to "quote"ahaha.

P.P.P.S. I got it Seems like I was using the ~ function wrong. Score for me.