LINUX

For

Beginners

The Ultimate Guide To The Linux Operating System & Linux Commands

ADAM VARDY

Linux for Beginners

The Ultimate Guide To The Linux Operating System & Linux Commands

1st Edition

Adam Vardy

Contents

Introduction

Thank you for buying *Linux for Beginners*. Linux In this book, we are going to give you an overview of the concepts that you have to understand before you actually start using Linux. We will explain to you the different elements of it that you ought to know about before you go and delve into the Linux world.

Linux has many benefits. However, it also has numerous little aspects that can leave you perplexed. Not being able to understand these aspects can definitely cause you problems in the future.

In this book, we're going to talk about what those elements are. We are also going to talk about what Linux is, where it came from, and all of the fundamental concepts that you have to understand before you actually start building your own Linux servers and maintaining your own Linux systems. In addition, we'll also teach you basic terminal commands that will get you up and running about within the Linux operating system.

We hope you enjoy this book!

Chapter 1: What is Linux?

The first thing that we need to talk about in this book is about the origins of the Linux. In a nutshell, Linux is an operating system. For those of you who do not know what an operating system is, and for all of you who think you do but may have forgotten what an operating system is, it is actually the software layer that is between your hardware and the software that allows you to get something productive done on a computer.

The operating system is what allows the software to talk to the hardware. It is the one that lets you store information on hard drives, send out print jobs to printers, *etc.* If you are in a normal Windows environment—a Microsoft Windows Operating system—you have your hardware at the bottom, you have the Windows operating system on top of that, and then you have Microsoft Office, for example, that sits on top of the operating system.

Linux is an operating system that acts as an intermediary—i.e. a bridge—between the physical device and the instructional code of a program. The main thing that you just need to realize is that in the Linux world, the software that you will be running is of a completely different type, compared to the ones that you would run in the Windows operating system. Desktop applications like Microsoft Office and Adobe Photoshop are not usually run on a Linux environment. Linux normally runs servers—Apache web servers, database servers, web virtualization servers, *etc.*

However, there are various Linux distributions out there that are specifically made for personal desktop computers. These Linux distributions are, in a way, similar to Windows and Mac OS, in a sense that they run the same types of programs like Word Processing programs, photo and video editing programs, web browsing applications, program development applications, games, *etc.* These Linux distributions are more targeted to home users who just want a free operating system alternative.

Linux did not begin as an operating system, however. Linux was a kernel created by Linus Torvalds while he was a student at the University of Helsinki. The Kernel is essential, but by itself, it is useless. It can only function in the context of a complete operating system. The Linux Kernel was used in combination with

the GNU operating system. Imagine GNU as a big complex puzzle with a big piece in the middle missing—the big piece being the Linux Kernel. The complete puzzle equates to a functional operating system.

It is important to understand what a Kernel is as this is the defining component of Linux. A Kernel is the central part of an operating system that is responsible for interfacing all you applications down to the physical hardware.

There are two major types of Kernels competing in today's market—Windows and Unix-like Kernels. The Linux Kernel falls under the latter as does BSD, Mac OS, and Solaris. The term "Unix-like" refers to the fact that they operate similar to, or are based on the original Bell Labs UNIX operating system.

Kernels tend to fall under three categories:

- Micro Kernel – A Micro Kernel only manages what it has to: CPU, Memory, and IPC or inter-process communications. If it is not an IPC, Memory, or CPU, it is automatically regarded as an accessory and can be handled in user mode.
- Monolithic – Monolithic Kernels like Linux are the opposite of Micro Kernels. They encompass not only the CPU, Memory, and inter-process communications, but other things such as device drivers, file system management, and system server calls.
- Hybrid – The Windows Kernel falls under Hybrid because it has the ability to pick and choose what to run in both user and supervisor mode.

And so, between 1991 and 1994, Linus Torvalds created the Linux operating system by combining the GNU OS with the Linux Kernel. Basically, Linus Torvalds wanted an operating system that is not only free, but also something that he can customize to fit according to his programming needs. Linux was his creative little pet project that he did on the side. The big thing with Linux is, because it has an "ux" suffix, and because most of the commands that you use look a lot like UNIX commands, people think that Linux is a type of UNIX operating system. This is totally not the case. UNIX is its own type of operating system. Linux is its own type of operating system.

Linus Torvalds created the entire Linux operating system from the ground up. The reason he created Linux was that he wanted to create an open source operating system for people to use. Back in the day, UNIX was not open source.

If you wanted to use UNIX, you had to pay somebody in order to use UNIX. Microsoft Windows, of course, is Microsoft Windows. You always have to pay in order to use Microsoft Windows.

So Linus Torvalds, being the computer engineer that he is, wanted an operating system that was completely free. He and his friends at the Massachusetts Institute of Technology, or M.I.T., wanted an operating system that they do not have to pay for to use and will also help them create the computer programs that they wanted to make in a more efficient and easy way. In other words, they wanted an operating system that they can customize to fit their needs as well as completely free to use.

The big thing to remember with Linux is that, even though it is an operating system, it looks totally different than Microsoft Windows or Mac OS.

Distributions

After Linus Torvalds created Linux back in the 1990s, he wanted to stop working for a little bit. So, what he did was he made the source code for his new operating system completely available to the public. This allowed everybody in the world, especially computer geeks, scientists, etc., to start playing with and changing the Linux operating system as they saw fit.

Major companies and educational institutions decided they liked Linux. And since Linux is open source, they are able to see the source code. This gave them the ability to start creating their own versions.

People from University of California, Berkeley, decided to start creating their own version of Linux. People from China also started creating their own version of Linux. People from all over the world—from all walks of life—started making their own versions of Linux that fit their own personal needs. Today, you have Red Hat Linux, Ubuntu Linux, Google Android, and many more.

Making Linux's source code available to the public facilitated the creation of something called distributions or "distros." Distributions are the various versions of Linux that people have created over time. There are many different versions of Linux that are out there. Different distributions have different capabilities. Now, when you need to decide which Linux distribution you want to use, you are going to have to think about what you want your computer to do first with Linux.

It is much more important that you understand what you want your computer to do, before you install the Linux operating system. With Microsoft Windows, you just install it first and then worry about what you want to do with your server later. With Linux, every distribution is built to do things in a certain way.

For example, there is a version of Linux called Trustix. Trustix Linux is considered to be the most secure Linux operating system out there. It is just a brick. You set Trustix Linux up and as long as you do not do anything completely stupid, nobody can hack it and no viruses can get to it. It is just one solid, secure server. But, you have to decide that you want a solid and secure server first, before you go and get that particular distribution to install on the server.

So, if you want a computer that you can use some office applications or you are

going to surf the web with, then you may want Ubuntu Linux's desktop version. If you want a super secure computer, then you might want Trustix Linux. If you want something with enterprise level support, let's say you want to use a Linux distribution that has a tech support center out there to help you if necessary, you may decide to use Red Hat Linux. But again, you have to decide what you need your computer to do in order to determine the exact Linux distribution to install on your computer.

If you install Ubuntu Linux distribution on all your computers, and then you decide you need enterprise support and you call Red Hat Linux, they will not be able to help you. Red Hat Linux does not support Ubuntu Linux. Every distribution does things their own way and is created by different entities. So you must familiarize yourself first with what a particular distribution does, and whether it fits your computing requirements exactly.

Open Source

Now that we have talked basically about where Linux came from, the next thing that we need to talk about is Open source licensing. Of course, at this point in time, you have probably heard of open source software. You are also probably under the wrong idea that open source software is free software. This is not the case.

Open source software is not free software. And if you treat all open source software as if it is free, you are jeopardizing not only your career, but also your company. It is just bad legally. Therefore, it is important that we discuss open source software and the different ways that open source software vendors get paid.

So what does open source exactly mean? What open source software means is that whenever a programmer sits down and begin writing the code for a software, they provide you with the code only so that you can see how the program was written. It doesn't necessarily mean it is free. So how do these programmers earn with an open source license? There are four different ways that these open source vendors or programmers get paid.

The first one is through the open source model where they give the software free of charge, but when you require training or support for the software, that is where you have to pay them a certain amount. Let us say for example you downloaded the mySQL software for your Linux server. You download the mySQL program, tinkered around with it for a little bit, and then you find out that it is extremely useful and powerful.

Even though you already figured out most of the intricacies of the mySQL program, there are still certain aspects of it that you need to learn, or need support with. So, you go approach the software developer and ask for support and training. This is the point where you have to pay them a certain amount. This is one of the ways these developers or programmers get paid for their development efforts.

The second way developers and vendors get paid through an open source license is through a non-commercial, personal-use-only open source license. This is where most people, including veteran system administrators, get into a lot of trouble.

It is true that some open source licensed software will allow you to obtain a program completely free. You can use them in a computer lab or any kind of experimental environment without having to worry about the legal implications. Why? Because it is for personal or non-commercial use only.

The problem, or the part that usually gets an administrator in hot water, is as soon as they take that server from the test lab and screw it into a server rack in the production environment, the commercial use starts to kick in.

If you are a geek at home and you want to play around with the software, there is no problem at all. Once you use it to power up a business server, or maybe host a home business website for example, you now own a licensing fee for that software. The gruesome thing about it is that these licensing fees can be anywhere between $5,000 and $ 10,000. It is that expensive. Therefore, it is only prudent that you be conscious on how you use the software, whether it is for personal, non-commercial, or commercial use.

The third way open source software programmers or vendors get paid is through a paid open source license. Some of you might be asking, how can a software be on an open source license if it is a paid software right off the bat? Well, a paid software will always be considered as open source if they let you see the code.

The idea basically is, if you want the product, you pay the vendor or developer the licensing fee and you buy it just like if you are dealing with Microsoft, Adobe, *etc.* The difference with open source software in this model is, even though you bought the software and can see the code, you may not have the legal right to modify that code.

If you are the type of user who likes modify code to tailor-fit a particular software according to your specific needs, you should lookout and avoid software that has this type of open source model.

The fourth way these open source vendors get paid is through a recurring open source license fee. Again, this is like most open source licenses out there. They let you download and test the software free of charge. They would even let you see the code, just so you know how the software actually works.

However, in order for you to have the legal right to actually use the software, they would have to charge you a yearly fee. This is usually cheaper than a one-time licensing fee, but is expensive nonetheless.

Here is an example: Let us say you downloaded Foxit PDF reader for free. You can use the software, test out its most promising features, and even see the code to figure out how it does what it does. But in order use the software legally, fully unlock its most useful features, and also have technical support for it, you have to pay the developers a yearly recurring fee.

So, as you can see, knowing how open source licensing works is far more important to your business than simply being able to set up a server. This is the kind of thing that can cause massive amounts of damage to you and your company. If you install multiple servers with open source licensed software and you do not know the licensing requirements, that may be a catastrophic problem. Remember, open source does not mean it is free. It has nothing to do with free. A huge amount of open source only happens to be free. However, that doesn't mean open source software is completely free.

What open source means is that you are allowed to see the source code that created the program. This doesn't mean you're allowed to modify the source code. You may not even be able to do a single thing to the source code. But at least you're allowed to see the source code so that you understand what is happening. If there are flaws, or if there are security holes, you can actually see that in the code.

As you can see, Linux can go from being really affordable to being stupidly expensive in an instant. And these maintenance contracts are one of the things that can make it extremely expensive. Open source licensing, as we mentioned, can make or break your career. So make sure you take it seriously.

The Linux Shell

Now that we have the legalities out of the way, let us now talk about the shell of the Linux operating system. So what is a shell? The shell of an operating system is the screen that you use to interact with the operating system. If you're thinking about Microsoft Windows, the Windows shell is that graphical user interface where we have the little mouse pointer which we use to go around and click on the various elements of the desktop, such as folders, icons, *etc.*

The shell is of two types. The first one is the graphical user interface, or GUI. The second one is called the line user interface, or LUI. The LUI basically looks like DOS prompts. So, if you ever played with the Microsoft Windows DOS prompt, the line user interface is exactly that. It is that black and white screen where you type in different commands to get a particular output from the computer.

In Linux, since this is a much more technical operating system preferred by programmers, geeks, engineers, or what have you, they prefer to use the line user interface. So when you go and install Linux, you can either have a graphical component where you can click things in the desktop much like a normal operating system, or you just have that little line user interface.

The main thing that you have to remember about the shell is that the line user interface (LUI) is much more powerful than the graphical user interface (GUI). However, when you install Linux with just a line user interface for the shell, all you are going to get is a prompt. If you don't know what you need to do with the command prompt, like the various shell commands for that operating system, you will be stuck.

There are many cases where people install Linux with an LUI, but doesn't know any Linux shell commands for the LUI. So basically, that's the thing with the Linux line user interface. You are going to have to understand the commands in order to get the computer to do anything that you need it to do.

Root

The next big concept that you have to understand when you are going to be using Linux is the concept of Root. In Linux, root pertains to the highest level of anything. When you hear about the root user, it is referring to the administrator of the computer. The root user is the highest level user that you can be on the computer.

So, if you can log in as root, you can do anything in the world you want to with the computer. Root can also mean the root of the operating system. It is where the operating system installed in the computer hard drive. If you think about this in terms of the Windows operating system, C:/ is the root of the Windows operating system because that is where it is installed.

Root can also mean the highest level that a user can get into. What does this mean? In Linux, users have home folders. The home folder holds all of the user's data, such as documents, settings, programs, *etc.* So the root of the user would be their home directory. The home directory is the highest level for a particular user.

The main thing to understand whenever you are talking about root in Linux is that root is the highest level of anything. There's actually a user account in Linux called root, and that user account is the absolute highest level user that you can begin the operating system in. Root can absolutely do anything. They have total access to everything in the operating system. Once we go into actually starting to type commands and making Linux do certain tasks, this concept of root is going to be crucial.

Capitalization

Let us now talk about something that just messes with every Windows user's head when making the switch to Linux: Capitalization. So basically, you have uppercase and lowercase letters. In the Windows operating system, it doesn't care whether you put in an uppercase or a lowercase letter. If you have a folder named "Home" in Windows, that is going to be the same as "HOME," "home," or "homE." Windows, except when it comes to passwords, does not care about capitalization.

In the Linux world, keep in mind that Linux was created by computer professionals. These computer professionals coded the characters of letters, numbers, punctuations, *etc*. using ASCII text. In ASCII text, an uppercase "H," for example, is actually a different character from a lowercase "h." What this means is that in the Linux world, "Home," "HOME," "home," or "homE" would be considered different folders.

Let us say you have a folder named "USER," but for some reason your typed in "user" when you tried to access it. Linux will not be able to find that folder because the "user" folder does not exist. Only the "USER" folder, with the uppercase letters, does exist. So remember, capitalization matters in Linux.

One of the places this can cause you big problems, and you have probably already seen this with some websites that you use, is when you are typing your username logins. In Linux, all username and passwords are case sensitive. So when you type in your username or password in Linux, make sure that you don't have your caps lock key turned on or you are not accidentally holding down shift when you type in your username and password.

This is actually not complicated. It is just that people are used to using Windows that they totally bring the mannerisms of using Windows over to Linux, where capitalization matters greatly.

Server vs. Desktop

There are generally two versions of Linux that everybody is going to provide. Whether you get Red Hat Linux, Ubuntu Linux, Fedora Linux, or whatever distribution that may be, they will normally have two versions of the distribution. One is going to be the server version while the other is going to be the desktop version.

The main difference between the server versions and the desktop versions of any of these Linux operating systems is that, the server version is a stripped down version if Linux. Why? Because they figured that if you are going to be installing a server, you know specifically what you want installed on the server. What this means is that there will be no graphical user interface in the operating system, and a lot of the tools that you use to administer Linux will not be installed automatically.

They figured that if you want the tool and you are installing a server, then you know how to install the tool to the server yourself. If you are just beginning to learn Linux, you are probably better off at this point in time to download the desktop version. The desktop versions of these distributions give you the graphical user interface right off the bat.

When you install the desktop version, you'll immediately be able to navigate the operating system using the graphical user interface, much like Microsoft Windows or Mac OS. You will have desktop icons, folders that you can click on, *etc.*

It's going to function differently compared to Windows or Mac, so you still have to learn how to use Linux. But it's going to be an environment that you are probably going to be able to understand as soon as you boot into it. After installing the desktop version, you are going to boot straight into a graphical environment. It is already going to have management tools installed, and you can play around and figure out how to use that graphical environment. That's the main advantage of the desktop version over the server version of Linux.

Why Use Linux?

The reason that you should learn Linux and start deploying Linux is for server functionality. Linux is incredibly rock-solid. Once you install Linux, and once you get through all the quirks and you set up all the configurations, a Linux server will run until the CPU overheats and dies. It would just run non-stop. A Linux server, once installed correctly, can run for a hundred and fifty days continuously without having any problems.

Linux is totally unlike Windows where you have to reboot it weekly to avoid memory leaks or crashes. Linux, as long as you configure it properly, would just run and do its job day in and day out. The reason that you should look at deploying Linux is for server functionality, whether it is for Apache web servers, mySQL database servers, virtualization servers, email servers, *etc.*

When you setup a Linux server, that thing is going to be rock-solid. You are not going to have the same problems that you have with Windows, where you install Windows in a computer today and works really great, but then you keep getting many updates. And two years from now, the computers are working slower because all the updates that Microsoft has delivered actually decreased the performance of the computer itself.

This is not the case with Linux. Once you install Linux, it is going to do its job with the same efficiency as when you first installed it on your computer—as long as you configure it properly, of course.

Linux is a really robust and efficient operating system. At this point, it is important that you have a good foundation and understanding of where exactly Linux came from, and what are some of its basic but important concepts.

Chapter 2: Installing Linux Server Edition

In this chapter, we will talk about how to install Linux so that you can try it out and get a feel on how to use the Linux operating system in general. We are going to discuss how to install both the server and the desktop versions of Linux, so you can just see how the install process for each version works. Also note that we are going to be using the server and desktop version of the Ubuntu distribution of Linux.

Just with all the different distributions that are out there, it is currently the one that seems to be the most popular. With all distributions of Linux, you should understand that they all have their own particular quirks. Ubuntu is no different.

Ubuntu Linux has some particular quirks that other distributions do not have. Through this whole book we're going to be using Ubuntu. Just keep in mind that if you decide to use a different version of Linux, and you go and try to run some of the commands that we will be teaching you in the later chapters, they may be slightly different in those versions. So you may have to do a little Google search to see what those differences are.

For example, in Ubuntu you use the "apt-get" command in order to get and install applications or programs onto the Linux computer. In Red Hat or Fedora, you use the "YUM" command. Basically, at the end of the day, the "apt-get" and "YUM" commands do the same thing. It's just that for each version/distribution, the syntax of the command varies.

Now we are going to learn how to install the Ubuntu Linux server edition. Again, Ubuntu is completely open source; it is completely free. Whether you will be using Ubuntu for personal use or commercial use, you can do so at no cost. All you need to do is go to the Ubuntu website, download the ISO file for the Ubuntu server edition, and then burn it to a disk or put it in a USB thumb drive.

For the sake of conciseness, we will not delve into the disc burning process or bootable USB thumb drive creation process. If you do not know how to burn an image file onto a disk or a USB thumb drive, there are many tutorials on the web on how to do that. Just look it up and follow the steps enumerated in those tutorials.

Since we are installing the server version, keep in mind that at the end, all we are going to get is the line user interface, or LUI, of the shell. In other words, it is a blinking cursor at the command prompt. If you don't know the commands that you need to type in, you will be stuck.

At this point we are not going to discuss those commands yet. What we are going to do is just go through the installation of the operating system so that you see and know how it all works. You may also be asking yourself, if there is a desktop version and there is a server version, then why is the server version so barebones? How come it does not have any graphical user interface at all, like the Microsoft Windows Server operating system?

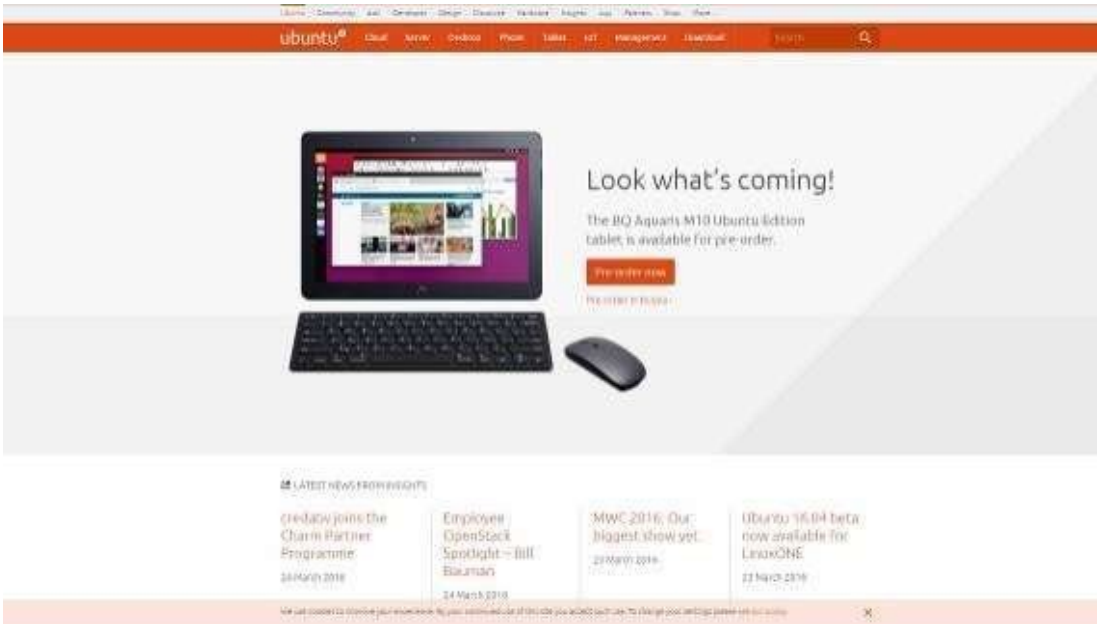
The reason the server version is so barebones is that in all things computer, whether you are dealing with Windows, Mac OS, or Linux, every feature or function is also an attack vector for a hacker. Every additional feature or function that you put into a computer is a potential weakness that a hacker can exploit or manipulate.

Like with the Mac operating system: although the Mac operating system is fairly secure, hackers have learned how to hack into the Adobe Flash applet that the Mac operating system is using, and then take over the computer that way. Even though the Mac operating system is a brick, the Adobe Flash feature of it has now become a security vulnerability, and hackers can take over a lovely Mac computer using the Flash software.

Now, with Linux servers, since these are servers that are going to be major web servers, database servers, or virtualization servers, you want the server to be as secure as possible. One of the ways that you make the server secure, is by not giving hackers the opportunity to hack anything. The less functionality or features your server has, the less opportunity there is for hackers to compromise your system.

With that out of the way, let us proceed with the installation:

Download – The first thing that we need to do is get the Ubuntu Linux server operating system. All you need to do in order to get the Ubuntu Linux server edition is you go to http://www.ubuntu.com/. Look below to see how the home page of the Ubuntu website looks like:



Now with Ubuntu, they are creating numerous versions of Ubuntu to do many different things. There is a desktop version of Ubuntu, there is a netbook version, there is a version for Cloud, and there is even a version for the Phone or Tablet. Now, since we are looking for the server version of Ubuntu, we are going to go and find the link at the top that says "Server." Go ahead and click on "Server."

After clicking on "Server," you will be presented with the screen above. You will also see an orange rectangular button near the middle of the page that says, "Download Ubuntu Server." Go ahead and click that.

After you click on "Download Ubuntu Server," you will see two versions of the server version of Ubuntu: the Ubuntu Server 14.04.4 LTS and the Ubuntu Server 15.10. The difference between the two is that the 14.04.4 LTS version is the most stable release of Ubuntu server. It has Long Term Support, which is important if you are putting this version on an enterprise server and want to have help if ever a problem arises with the operating system.

On the other hand, the 15.10 version is the most up to date version, and it is where most of the newest features are implemented. However, having the newest features does not mean that it is stable. There are still minor quirks within this version that the developers are working to sort out, and support for this version is only up to nine months.

If you are installing Ubuntu server for your business and you need to have the most stable version out there, you should get the Ubuntu Server 14.04.4 LTS version. If you want to play around with the new features of the operating system and you don't mind the stability issues that you may encounter, you may go for the Ubunte Server 15.10 version.

Now, after you have chosen which version of Ubuntu server you want to install, it will then ask you whether you want the 64-bit version, or the 32-bit version. Hopefully, if you are trying to figure out how to use Linux, you understand the difference between 64-bit and 32-bit. If you do not understand the difference between the two, basically all you have to know is for the server version, you

should try to download and install the 64-bit version.

If the 64-bit version does not install onto the computer you are trying to install it on, then download the 32-bit version of the operating system. Once you pick the one that you want to go with, click the download button and then it will start the download process. Keep in mind that the file size is around 600 MB, so it may take a while depending on how fast your Internet speed is.

With this, what is going to happen is the Ubuntu ISO file is going to be downloaded to your computer. In order to install the operating system, you need to take that ISO file and burn it to a CD or to a DVD, and then you put that into your computer. Make sure that you boot off of the CD or DVD. By default, the computer is set to boot directly to the hard drive of the computer where the original operating system is installed.

If you want to install Ubuntu in your computer, make sure it boots off of the CD or DVD first and not the HDD or hard drive. There are many ways on how to do this. However, the most common is by pressing the "DELETE" or "F1" key on your keyboard to get into the BIOS settings of your motherboard, and from there you can set where you want to boot the computer from. Different motherboard manufacturers have different ways to get to the BIOS settings, so make sure to consult the user's manual of your motherboard on how to do this.

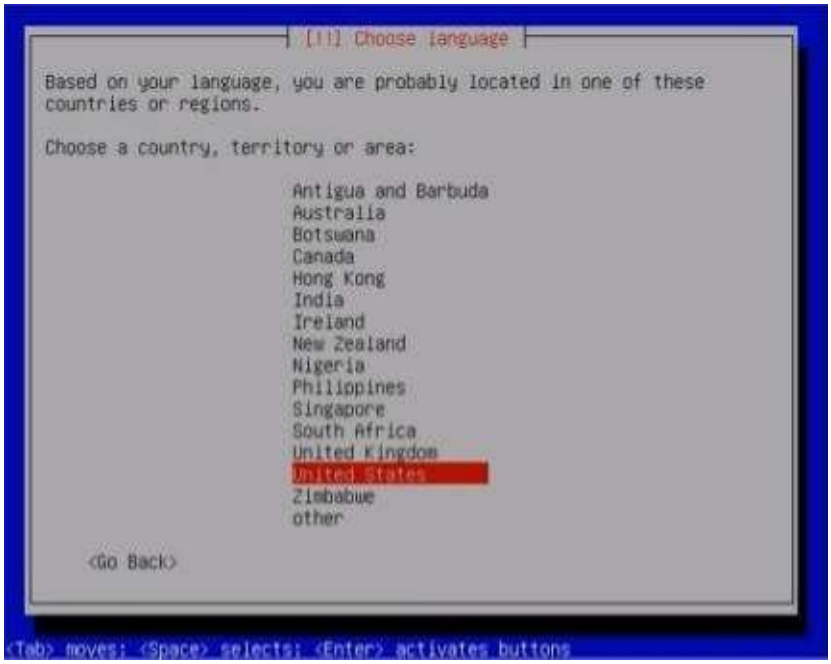
Installation – Once you boot off of the CD or DVD, it will now ask you what language you want to install Ubuntu Linux in.

At this point, just choose whatever language that applies to you and press ENTER. After pressing ENTER, it will now show you the various options for installation.

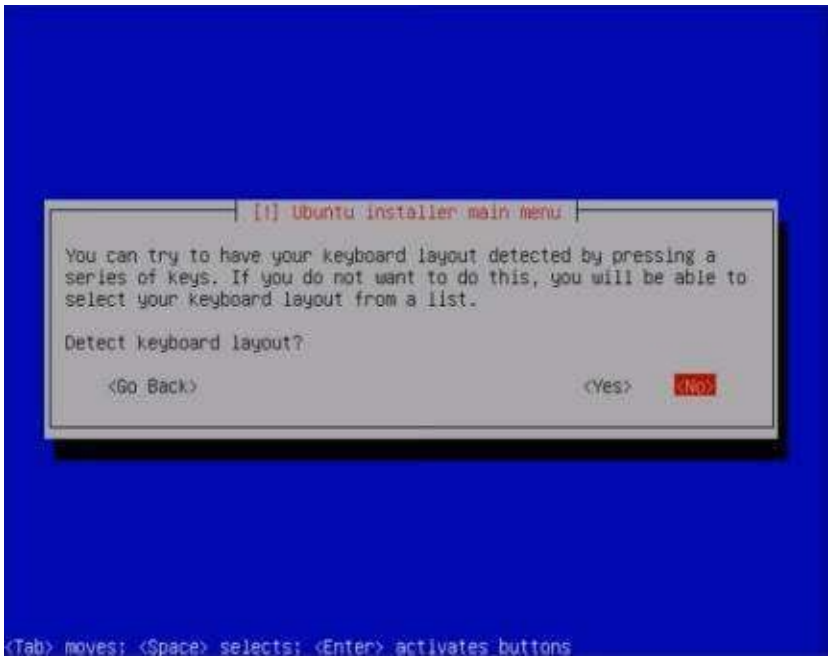
The various options for installation include, Install Ubuntu Server, Install Ubuntu Enterprise Cloud, Check disc for defects, Text memory, Boot from hard disk, and Rescue a broken system. What we are going to do is simply install the Ubuntu Server. Go ahead and highlight that option using the arrow keys on your keyboard and press ENTER.

After you press the ENTER key on your keyboard, the installation process will ask you to choose the language for the installation process itself. Note that the language that we were asked to choose prior was for the language of the operating system itself once it is finished installing.

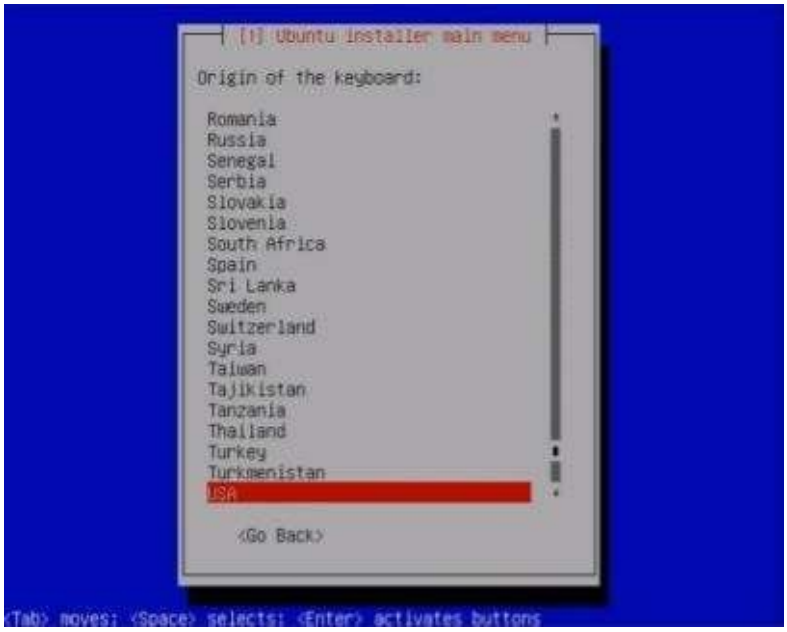
Go ahead and choose the language that applies to you using the arrow keys, and then press ENTER. After that, the installation will now ask you to choose the country, territory, or area you are in right now.



Again, using your arrow keys choose the country, territory or area that applies to you and press ENTER. Next, it will ask you if you want the installation to detect your keyboard layout. This is a necessary step since, as you know, Linux is an international operating system. There are countries that utilize a totally different keyboard layout to type text.



Choose the "Yes" option if you are using something like a Japanese, Chinese, or some other Asian keyboard layout. Otherwise, choose "No." After you have made your choice and pressed ENTER, the installation will then ask you the origin of your keyboard.



Go ahead and choose the option that applies to you and press ENTER. Now, depending on your chosen keyboard origin, the installation will then ask you to choose which specific keyboard layout that applies to your keyboard. If your keyboard originated from USA for example, the USA keyboard has many different layouts like the one below:

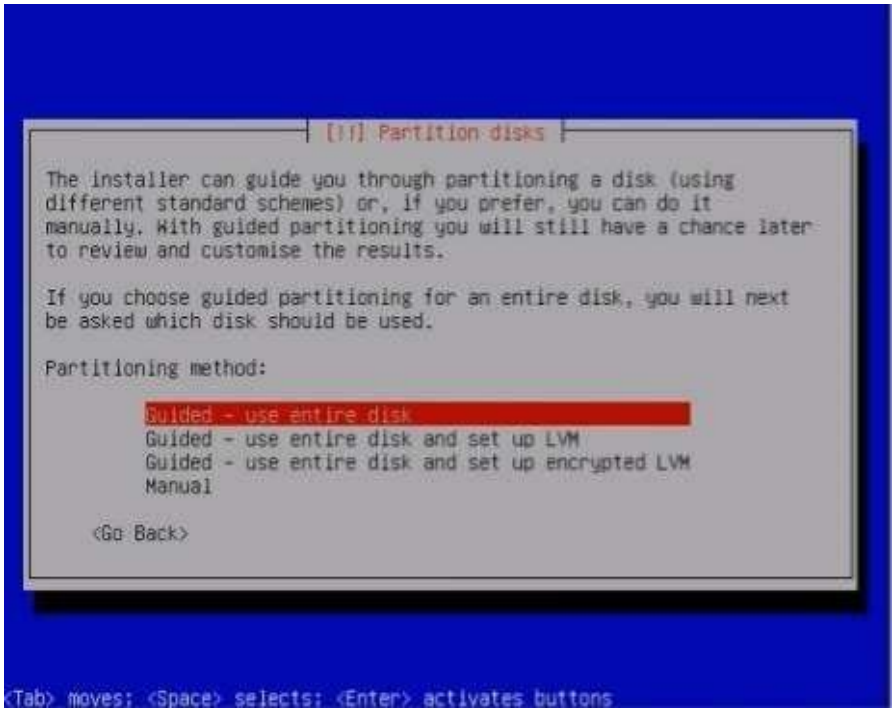
Go ahead and choose the one that applies to your keyboard using the arrow keys and then press ENTER. After you press ENTER, you will see that the installation will run the configuration that you have just chosen. Once that is done, it will now start with the network configuration process.

As you can see in the illustration above, it is now asking you to enter the host name for your particular system. Basically, this is just the computer name. In the

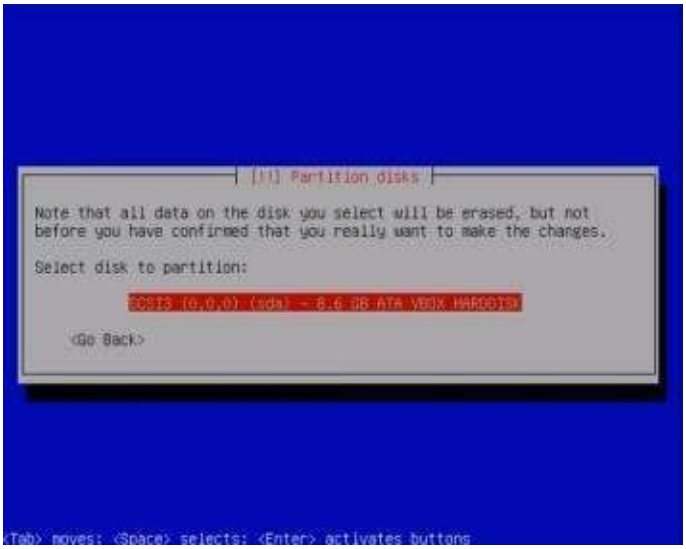
Windows operating system, we normally distinguish each computer by their computer names. In Linux, the equivalent of the computer name is the host name. So, go ahead and just enter the computer name that you like to put and then press ENTER.

After you press ENTER, it will now start configuring the system clock by first asking you to choose your time zone.

After you have selected your time zone, press ENTER. Once you do that, the installation will once again implement the options that you have chosen. Once that is finished, the installation will now ask you how you want your hard drive to be setup prior to the Linux installation.



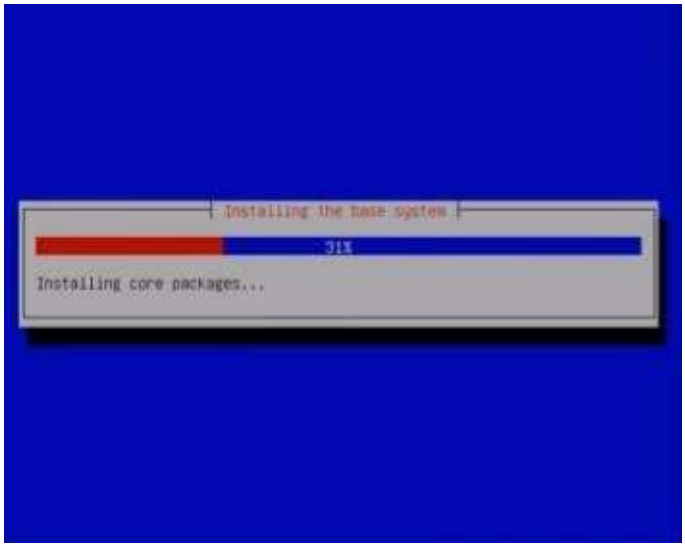
Right now, do not use the option that says, "Guided – use entire disk and set up LVM." We will go over LVM in the succeeding chapters. For now, just go ahead and choose the option that says, "Guided – use entire disk" and press ENTER. After doing that, it will now ask you to select the disk partition.



Depending on how many hard drives you have on your computer, the number of options that will show up here will vary. Just go ahead and choose the correct hard disk where you want to install Linux and press ENTER.

After pressing ENTER, the installation process will now as you whether you

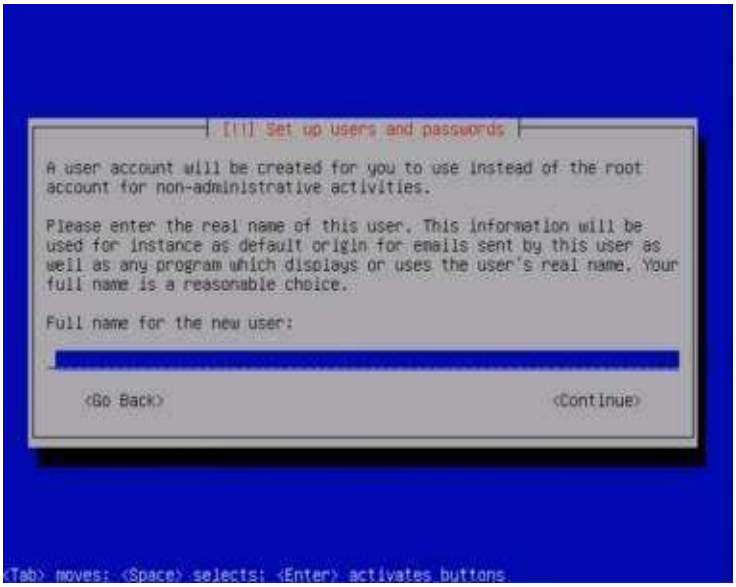
want to write the changes that you've made—the options you chose—to the disk. This is Linux basically asking you whether you are sure about the options you chose prior, and that if everything is as you prefer. If so, choose the "Yes" option and press ENTER.



As you can see in the illustration above, the installation will now implement all the options that you have chosen. It will install all the core essential files necessary for Linux server to function properly on your system. Now, depending on what kind of computer and what kind of hardware you are putting this on, this can either be a really quick process or relatively slow process. And that is the funny thing with Linux. You could be throwing this thing on anything from the most modern computer—a three thousand server computer from Dell--to the server that was three thousand dollars twelve years ago.

Linux, especially with the server version, will work regardless of how old your system specification is. The system requirements of Linux are so low that it can run on what most people today consider outdated systems. Unlike Microsoft Windows where the system requirements are somewhat heavy, Linux's system requirements are extremely light that you can even install it on a USB thumb drive and run the whole operating system off of that on any computer system.

After Linux is done with installing the core essential files, it will now ask you to enter your full name for the Linux user account.



As it says on the screen, your real name will be used as the default original name whenever you send emails, for example, as well as any programs that show the user's real name. You are not actually required to put your full name here. So for those who are not comfortable putting their real names down, do not worry.

Go ahead and enter whatever fictitious name you want to put down for the Linux account. After that, the installation will now ask you to enter a username for the Linux account.

Just enter any username that you want to use and press ENTER again. Now, it will ask you to enter as password for the Linux user account.

Enter the password that you want to your user account to have. After pressing ENTER, it will then ask you to re-verify your password, so go ahead and type your password again and then press ENTER.

Now, take a look at the screen below:

If you receive a screen similar to the one above, this is Linux basically telling you that the password you have chosen is weak. This is how secure Linux is. It will tell you if you have a weak password, and that you have to change it and choose a much harder to guess password that hackers will have difficulty in cracking. If you receive this screen, just go back and replace your password with a much tougher one.

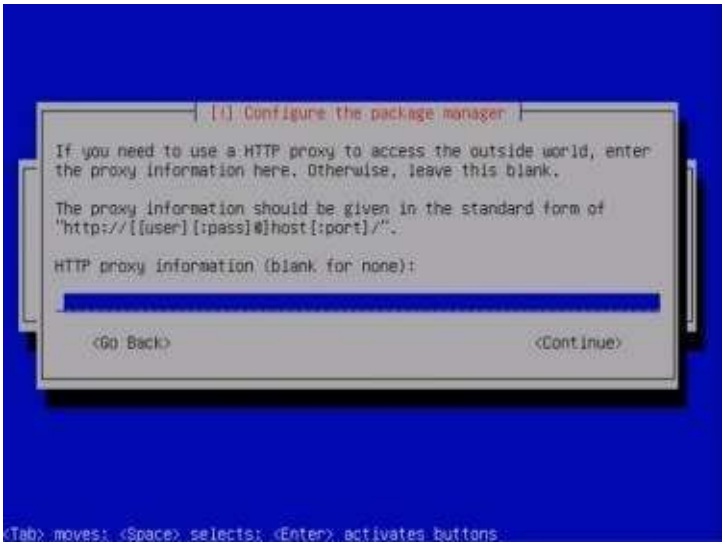
In the next step, the installation process will ask you whether you want to configure your home directory for encryption. What encryption means is that it will actually encrypt—encode—the files and folders that you put into your home directory. Your home directory is basically the Documents and Settings folder in Microsoft Windows. It is where most of your essential files and settings are stored.

Most of you might think that enabling encryption right off the bat is good. However, the issue is that since you are brand new to using Linux, it is suggested that you do not encrypt your home directory. Otherwise, if you encrypt your home directory and then your Linux computer crashes, you will most likely lose all your data. If you do not encrypt your directory and you accidentally do something you are not supposed to do and crash your system, then you can pull the hard drive out and you can recover your data pretty easily.

If you encrypt that directory, it is going to be encoded. When your Linux operating system dies, your data goes with it. Why? Because the system is going to assume that your system has been compromised by a hacker and it will prevent the hackers from gaining access to your files by destroying it. So right now, since you are still beginning to learn how to use Linux, it is recommended t hat you do not encrypt your home directory.

Once you know what you are doing, you can encrypt it. Right now, don't.

In the next step, the installation will ask you to configure your proxy server if you are using one. Since you are still a beginner you are most probably not using any proxy server so just leave this option blank, choose continue and then press ENTER.



Okay. Now the installation is now going to ask how frequently you want updates to happen on your Linux system. Just like Microsoft Windows, Linux needs updates too.

Basically, the installation is asking you if you want automatic updates to be installed. Again, since you are new at this, it is recommended that you do not install automatic updates. As what we have mentioned before, Linux is already stable and extremely secure. If you go in every couple of months and do the updates manually, you will be fine. Doing the updates manually allows you to know what particular features or patches are added to your system for better control.

Because remember the automatic updates, if they install and there is a problem with one, then your web server that was just working fine the other day is now crashing. So at this point, do not install automatic updates.

Next, you will be presented with a Software selection screen.

Within this process, it is actually asking you what kind of server you want your system to be. Do you want it to be a DNS server? Do you want it to be a LAMP server? Do you want it to be a mail server?

Right now we are not going to do anything, partly since we will be discussing how to do this in another chapter. If for example you know what you are doing and you want to make your server a web server, go ahead and highlight the LAMP server, hit the spacebar on your keyboard to select it, and then press ENTER.

LAMP basically means web server. LAMP is an acronym for Linux Apache MySQL PHP server. When you select this option, this means it will automatically install all the components you need for that Apache web server. Because like what we have mentioned before, in the Linux world, they do not want you to install anything that you are not going to use.

In the Windows world, when you install their operating system, they throw in everything including the kitchen sink, your neighbor's kitchen sink, and your grandma's kitchen sink. They just throw everything in when you install the operating system. The good part is, is that it is there so you do not have to install a lot of stuff later on. The bad part is again, as we have mentioned before, every single component that gets installed onto a computer is a potential security vulnerability. In the Linux world, they are much more worried about security so they do not install anything unless you want it to be installed.

Going back to our Software selection screen, unselect anything if you have selected a particular option. Make sure nothing is selected and then press ENTER. This is just going to be a barebones server.

Next, you will be presented with a Grub-PC configuration screen.

This is also called a Grub boot loader. Many people like using Linux, they may adore Linux, and they may even think that the entire world should run Linux. However, these people are also realists. They also know that they need to use Microsoft Windows once in a while. So, what they did was they created dual boot computers.

Dual boot means you can boot either into Windows, or you can boot into Linux. You can subdivide your computer—your computer hard disk—to accommodate 10 different operating systems if you want. And when you boot your computer, you can select between those operating systems.

What the Grub configuration screen is asking now is whether you want to do a dual boot on this particular machine. Dual booting is basically out of the scope of this book since it involves a plethora of operating systems, not just Linux. So if you want to know more about the specifics of Dual booting, there are many resources online that can help you with that. But for now, just go with the default option, which is "Yes," and press ENTER.

Once that is done, you will now get a screen that says that the installation is now finished.

Once you get the screen above, read the instructions carefully and press ENTER to boot into your newly installed Linux server operating system. After you press ENTER, your computer will restart basically and then you will get the screen below:

As you can see, you only have a blinking cursor that is asking you to enter the login credentials for the server. Just go ahead and enter the username and password that you made during the installation and press ENTER.

As soon as you enter your login credentials successfully, it will give you a brief welcoming text on your screen. You are now logged into Linux server itself. So that is the Ubuntu server. So, as we have talked about before, people who install Linux server will get stuck if they do not know the specific commands to navigate around the command prompt of Linux. There is no graphical user interface—nothing. All you get is something that looks like a DOS/Terminal screen.

Chapter 3: Installing Linux Desktop Version

Now we will talk about how to install the desktop version of the Ubuntu distribution of Linux. When you install this version, at the end of it, you will get a nice graphical user interface—you will be able to see files and folder icons that you can click and use to navigate around the Linux operating system. It will also have basic applications like web browsers to navigate websites online, and word processing applications such as OpenOffice for when you need to type documents, *etc.* Again, installation is a pretty simple procedure; you just click a few buttons and you are good to go.

Download – Just like with the server version, go to http://www.ubuntu.com/ and download the desktop version of Ubuntu Linux.

Click the option on top that says "Desktop" to proceed to the page where you can download the desktop version of Ubuntu.

Once you are there, go ahead and click on the orange button that says "Download Ubuntu." After you click that, you will be presented with two versions of the desktop version of Ubuntu Linux: Ubuntu 14.04.4 LTS and Ubuntu 15.10.

We have already explained the difference between the two versions in the last chapter, so go ahead and choose the version that you want to work with. Also, if you notice on the right hand side, it will ask you to choose your flavor of that particular version. This is them asking you whether you want to 32-bit or the 64-bit version of the Ubuntu Linux desktop. Just choose the version that applies to your system and then click "Download."

Go ahead and wait until ISO file download is finished. Once that's done, burn that ISO file to a CD or DVD, put that into the system where you want to install

Linux, and then boot off of it. Now we'll proceed with the installation process.

The interesting part that you will probably notice during the desktop version's installation is that it will give you two options: try Ubuntu or install Ubuntu. The curious thing about Linux is that they have things called Live CDs. What a Live CD means is that the entire operating system is installed and can be run off of the CD.

When you are dealing with Windows, Mac OS, or even Linux server, you always have to install the operating system for it to work on the computer. With a Live CD, you can actually boot straight off of the CD and use all of the functionality of the operating system without installing it. You can surf the web, manage files, edit text, *etc.* by just running the entire operating system from the CD itself.

In addition, if you have a Windows computer, you can put in a Linux Live CD, boot from it, and do whatever you want to do with it without affecting the computer at all. This is a great way for a user to test a particular Linux distribution first without going through the lengthy installation process.

So, going back to our installation process, just go ahead and click install Ubuntu for now.

Next, you will be asked to choose your region and time zone. Go ahead and choose the region and time zone that applies to you and click "Forward."

Again, just like in the server edition, the installation will now ask you to choose your keyboard layout. Go ahead and choose the one that applies to you and click "Forward."

Next, the installation will ask you to prepare the disk space in your hard drive prior to installation. If you want to use your entire disk for Ubuntu alone, go ahead and choose the option that says, "Erase and use the entire disk." However, if you have another operating system on your hard drive and you want to install Linux inside a different disk partition, then choose the option that says, "Specify partitions manually."

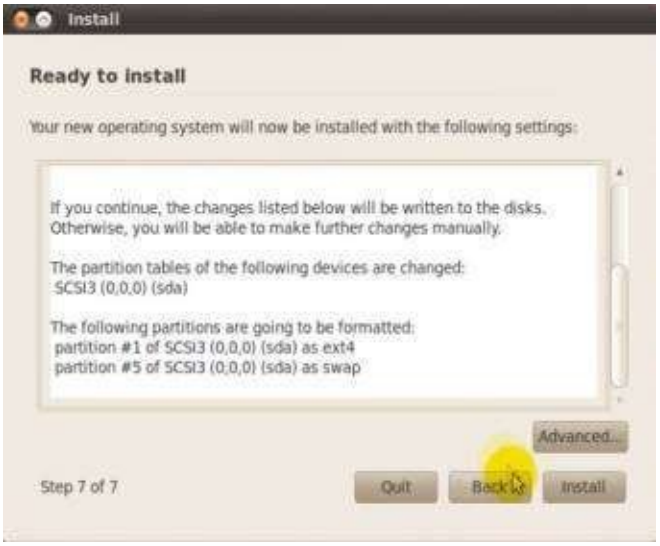
Specifying partitions manually involves telling the installation process how much disk space you want to allocate for the Linux installation. For now, since

you are still a beginner, let us assume that you are just installing Linux alone in your hard drive, and that you are not going to do any partitioning. So go ahead and choose, "Erase and use the entire disk" and press forward.

After you press "Forward," the installation will ask you what your name is, what name you want to use to log in, the password that you want to put for the user account, and the name you want to give the Linux computer.

Also, if you scroll down a little bit, you will see that the installation is asking you whether you want to log into the computer automatically once it boots up, require a password during login, and require a password during login and decrypt your home folder. The latter option only applies if you installed Linux with the encryption feature enabled.

Go ahead and enter the information needed and then press "Forward." Next, the installation process will show you the all the settings and options you have chosen so far and what it is about to do. If everything looks okay, go ahead and press "Install."



Once you click install, the installation will now proceed and you will see a window that has a progress bar at the bottom. At this point, just exercise patience and wait until the progress bar reaches 100%.

Once the progress bar reaches 100%, it will tell you that the installation is complete and that you need to restart your computer to finalize and use the new installation. Go ahead and click "Restart Now."

Again, the actual time that it will take for the process to finish will depend on your hardware. Do realize you should look at the hardware requirements for the desktop version of the Ubuntu Linux operating system. Remember, the server version is extremely stripped down. There is almost nothing to it really.

The desktop version has all these fancy graphics and user interfaces, as you can evidently see even from the installation itself. Where the installation of the server version basically is text based, the installation for the desktop version actually has graphical user interface windows and option buttons that you can actually click on.

All of these fancy graphical niceties actually does require pretty decent hardware. It is therefore recommended that if you are going to use the desktop version of Ubuntu Linux, your computer should be at least five years or younger with at least 1Gigabyte of RAM. If you have any less than that, it may not have the resources it requires to run.

But in the grander scope of things, the desktop version of Linux still has a much lighter system requirement compared to Microsoft Windows or Mac OS.

After the restart sequence of your computer is done and you finish logging into Linux using the credentials that you entered during installation, you will now reach the desktop of Ubuntu Linux.

Just like what we have mentioned prior, you now have something that vaguely looks like Microsoft Windows or the Mac OS operating system. As you can see right off the bat, you have your applications on the left hand side such as Firefox for web browsing, OpenOffice Suite for word processing tasks, your home folder, Linux settings, Amazon, and much, much more.

So, that is the Ubuntu Linux desktop version.

Chapter 4: Basic Linux Tasks/Commands

In this chapter, we are going to go over the basic tasks that you can perform in Linux. We are going to show you how to install applications, how to update applications, how to look at the Linux task manager and even terminate processes if necessary, how to start services, and a whole lot more. These are the basic tasks that you will need to understand in order to do anything else in Linux.

Everything that will be shown here will be on the Ubuntu Linux server version. This is not the desktop version. You are going to be looking at the LUI, or the line user interface. You will be typing in commands on the keyboard instead of clicking a bunch of options in a graphical user interface.

While this may seem a little daunting and tedious, keep in mind that in Linux, all types of serious administration have to be done at the command line. This is an operating system developed by programmers for programmers primarily after all.

Even if you are using the desktop version of Ubuntu Linux, it does not give you the full power of the operating system unless you use the terminal or command line to execute certain tasks. A lot of times, even if you are using a desktop version, you still have to open up a terminal screen and type out all the commands that you need to type out to administer the system.

Sudo

The first command that we need to talk about before you start doing any of the other commands is "*sudo*." Sudo basically means “super user do.” In the previous chapters, we discussed how different distributions do things slightly differently, and that every single distribution of Linux has its own quirks; its own little ways of doing things depending on what the creators are worried about.

One of the things that the creators of Ubuntu Linux were worried about was security. As what we talked about before, in every Linux computer, there is a user called *Root*. Root is the highest level user on the computer. It is kind of like the administrator in a Microsoft Windows computer.

Just like on a Windows computer, if somebody logged in as the administrator, or somebody logged in as root on Linux, they can do absolutely anything they want to that computer. They can install viruses, malware, or spyware, or basically just cause a lot of havoc. Hackers, using special programs and scripts, can also try to login as *Root* and cause all these problems.

To alleviate the possibility of a hacker obtaining *root* access, the Ubuntu creators decided they never want anybody to login straight as *Root*. So in Ubuntu Linux, you cannot login as the user *Root*.

Now, here comes the problem. Since you cannot login as Root, how do you do all these administrative tasks then? How do you execute administrative processes? What they have is this program called *sudo*. It is basically a command prefix that tells the operating system that you want to run a particular process as the super user or *root*.

Sudo temporarily gives a user administrative access—root access—to execute an essential command in Linux. In the Windows operating system, *sudo* is the equivalent of the "Run as Administrator" option each time you want to run a program with administrative rights in Windows.

Man Pages

The next thing we need to talk about are "*man*" pages. Man pages stand for manual pages. What you have to remember with man pages is that if you do not understand how a command is supposed to work or what command you are supposed to run, the man pages is where you look up information about any command.

Let us say for example that you want to look up information about the *ping* command. What you do is you type in *man* followed by the command that you want to figure out. Look at the syntax of this below: $ man ping What this will do in Linux is it will open up a page that will describe to you the ping command and everything that you need to know about the ping command. So if you are trying to figure out how a particular command works, all you need do is type in *man*, a space, and then the name of the command, that will then open up a manual page for you where you will be able to read about whatever it is you want to know about that particular command.

Nowadays, in the age of the Internet, doing a simple Google search is a whole lot easier than trying to do it with the *man* page. However, if Internet access is down, it is good to have the man pages handy.

In Windows, the man pages is basically the old "question mark" command where it shows you the "Help" pages. The only downside about the Man pages in Linux is that it does not explain a lot of things to you. It does not make things necessarily straightforward.

So when you go to the man pages and look up the ping command, you are going to get this whole page of text that is going to tell you about the ping command. However, once you are there, you are not going to know how to get out of that page. It is not user-friendly and straightforward. You can hit the *Escape* key, but that is not going to do anything. You can try to hit *Enter* or *Backspace*, but they are not going to do anything, either.

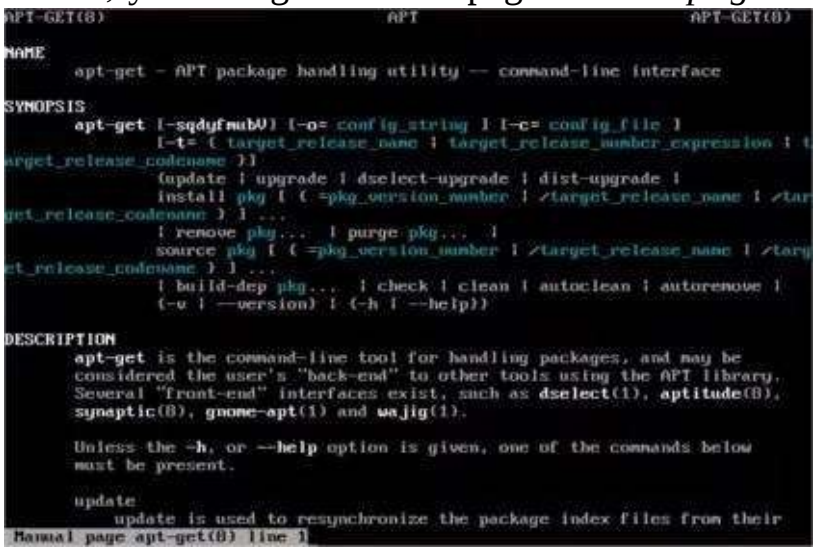
In order to exit out of the man pages, you need to type in the letter "Q" for quit. When you type the letter "Q," you will drop out of the man pages. So unless you press every key in your keyboard and experiment, you will not find out that the command to exit out of the man page is by typing the letter "Q." As you can see, this is counter-intuitive since most people are used to pressing the *Escape* key in

Windows to exit out of anything.

So now, let us say you want to know what the *ping* command is all about. Go ahead and type the command below: $ man ping After you type that in and press ENTER, you will see the *man* page for the ping command just like the one below:

Looking at the *man* page for the ping command, you will see the command description, and also the various options associated with the ping command. The options are basically ping command suffixes which further augment what the basic ping command can do. One example of this is typing the suffix "-*t*" after the ping command to continuously ping a particular host.

Now, let us say you want to know what the *apt-get* command is all about. Go ahead and type the command below: $ man apt-get Once you type that in and press ENTER, you will get the *man* page for the *apt-get* command like the one below:



And now, this *man* page for the *apt-get* command will tell you everything you need to know about the *apt-get* command. Like what we have mentioned earlier, some of the *man* pages for these commands are about as long as a whole book. Remember, if you want to get out of the *man* page for any reason at all, just press *Q* on your keyboard to exit.

Tasksel

The next command that we are going to talk about is *tasksel*. This stands for "task select." Now, Ubuntu is trying to become the premier provider of Linux distributions. They have really done a lot of work to improve Ubuntu Linux and they know how to make things as user-friendly as possible. With that in mind, one of the things they came up with was this program called *tasksel* or task select.

When you are trying to set up an Apache web server, an email server, or a virtualization server, for example, note that each of these servers requires a number of different programs in order for it to work. So if you are trying to set up a web server, you need to install Apache, you need to install mySQL, you need to install PHP, and you need to install the connecters so that they can actually work together.

The Ubuntu creators then figured that if you need to install those things, they can create a script that would just install all those things for you. So what will happen is, if you know exactly what you want your server to do and you are really not worried about making it really customized, you can run the *tasksel* command and choose which pre-packaged server type you want your server to be.

The creators of Ubuntu came up with around ten to twenty different server installation packages that you could choose from. And what happens is if you run the *tasksel* command, you will get a DOS screen where all of these server installation packages are listed. If you want to make your server an email server for example, all you need to do is select *Email server* and then it will install all the necessary software you need for that particular server type.

This facilitates for a hassle-free installation. It is quick and extremely user-friendly. You can have a full Apache web server running in 30 minutes or less with this command.

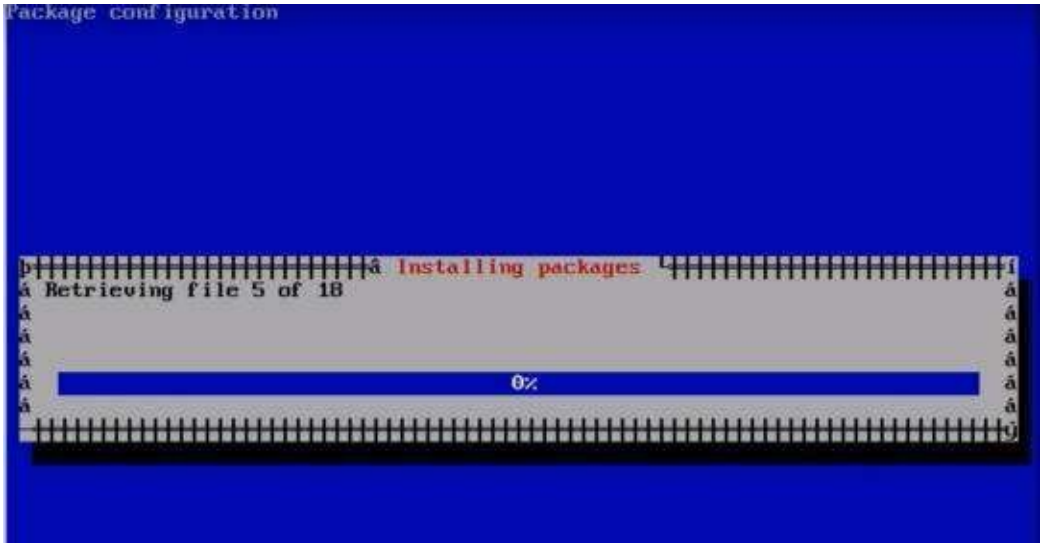
The main thing that you have to remember with this command is that you have to put *sudo* in front of the command. You might be wondering, how come we did not put *sudo* when we executed the *man* command? Well, the *man* command does not necessarily pose any security risk when executed. It is therefore one of those commands where *sudo* is not necessary.

With the *tasksel* command, however, remember that we are installing applications with this command. Installation requires modifying certain files in the operating system to accommodate a certain program. If a hacker someone manages to get access and execute the *tasksel* command to install viruses, you will be in having real big problems.

The creators of Ubuntu realize this as well. They are worried about people installing software that should not be installed onto the server. So, if you are trying to install software, or do any administrative task, you have to put *sudo* before the command that you are trying to initiate. With that in mind, to properly execute the *tasksel* command, you have to type in the syntax below: $ sudo tasksel Once you type in the command above and press ENTER, you will get the screen below:

So now, you are in the *tasksel* screen. If you look closely, you will see various options that you can select. These are all the different packages that you can install easily onto the server. If you want your server to be used in cloud computing, you can select one of the cloud computing options available. If you want your server to be a DNS server, you can go ahead and select DNS server, and so forth.

Once you have highlighted the package of your choice, you simply hit the spacebar on your keyboard. That will put an asterisk, and in some cases a check mark, on the box on the left and then you can press ENTER. When you hit ENTER, what will happen is the server will now install all of the software packages that are required in order for it to work.



This process will take a few minutes depending on how fast your computer is. Once the process is done, you will then have a fully functional server.

Apt-get

Now we are going to talk about how you can install individual programs. The thing that you have to remember with Linux is that this is an open source world. Most of the software is free, or you pay for it in weird ways like service agreements and such. In the Windows world, everything has to go through activation procedures.

So, whether you are using Quickbooks, Adobe, or Microsoft software itself, everybody is worried that somebody is going to pirate or steal a product. To prevent their software from being pirated, they put in these insane activation procedures that require you to have the CD with the correct CD key, *etc.* And then beyond that, once it is installed with the right code, you then have to go off and hit some activation server so that the creators of the software can verify your installation.

In Linux, they are able to create things called repositories. Repositories are places on the Internet that just house thousands upon thousands of Linux programs. So instead of having to have a disk of some sort, you can just go to that repository and install the application from there.

In Windows or Mac OS, you have to have a CD or DVD of the software. If you lose that disk, you are screwed. In Linux, all the software, or a huge portion of the software is out sitting in these repositories. You can just go and grab software from these repositories as long as you have Internet connection. This is the easiest way to install software on the Linux platform.

There are other ways to install applications in Linux, and we will talk about them in the later chapters. But for now, since we are still discussing the basics, we will just focus on installing from repositories.

So basically, within a Linux computer, there is already a configuration file that tells that Linux computer where the repositories are. When your run the *apt-get* command, this *apt-get* command will go out to the repository and it will get whatever program it is that you want to get and install it for you.

To correctly execute an *apt-get* command, just type in the syntax below: $ sudo apt-get install <name of program> Let’s say you want to install the *Apache2* program in your server. Simply type in the command below: $ sudo apt-get install Apache2

After you type the above command and press ENTER, Linux will then go out to the repository sitting on the Internet, find the *Apache2* program, and then install it on your Linux computer. It’s that easy.

Now, let’s say you decide that you do not want *Apache2* and you want to use a different web server instead. Well, the command the uninstall *Apache2* is: $ sudo apt-get remove Apache2

This will go in and uninstall *Apache2* from your Linux computer. It’s that simple. This is how you install and uninstall most of the software that you are going to need for your Linux server. Once you get better and you gain more experience in using Linux, you will start buying proprietary software in the Linux world.

You may buy special backup software, or maybe a special security software. Some of these types of programs may not be in the repositories, and you may have to go through different steps in order to install those programs. But for 99% of the programs that anybody ever installs for Linux, this *apt-get* command will work.

There are thousands and thousands of Linux programs in these repositories. Figuring out what programs you want to install can be a little bit tricky. If you do not know what are the best programs to have in your Linux computer, the best thing to do would be to make a Google search about it. You'll find many recommended Linux programs out there that are great for beginners.

As you probably know by now, everybody pokes fun at the people behind Microsoft Windows because they always have these updates. Every third day of the week or every month you get at least 10 updates for your system. Many people say that these updates are proof that Microsoft is crap. However, the reality is every single operating system or software needs to get updated once in a while. The same is true with Linux.

So, once you install all the software that you want on your Linux server and you want to update them, all you need to do is use the *upgrade* command. See the syntax below: $ sudo apt-get upgrade What the above command will do is it will see whether the program in the repository is newer than the one you have on the server. If the version of the program that you have is older than the ones in the repository, it will bring down that information and it will ask you whether you want to update your software or not. If you say *Yes*, it will automatically update

all the software that you have on your computer.

This is basically all you have to understand at this point with regard to installing and maintaining your software.

Services

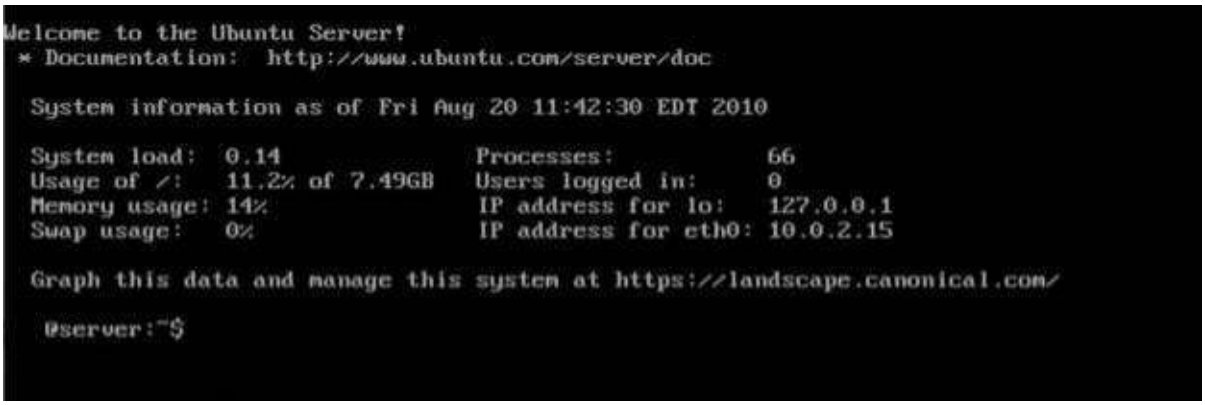
Like what we have previously mentioned, Linux is best for servers. Once you turn a Linux server on, they can just go until the hardware burns out, literally. Once you turn a Linux system on, there are very few times you actually need to reboot it. When we say very few times, we are talking once a year; and that is already overkill.

However, when you go in and change your configuration files in the software that you have running on your server, you may need to restart that individual software or that individual service. Let us say you have *Apache2* installed onto your Linux server and you change some of the configuration files. Those configuration files do not get loaded until your restart the *Apache2* service.

Although the computer stays on all the time, you do have to restart the services every once in a while just to make sure they are up-to-date. What we are going to see now is how to start, stop, and restart services.

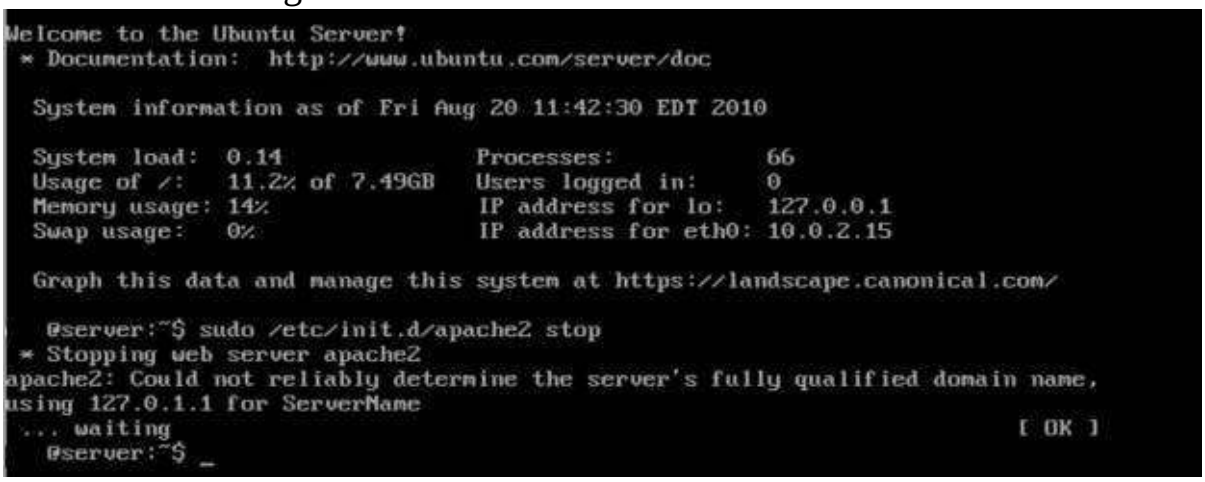
With this, you need to type in the syntax below: $ sudo *etc*init.d/<name of the program or service> start $ sudo *etc*init.d/<name of the program or service > stop $ sudo *etc*init.d/<name of the program or service > restart So, let's say you changed the configuration files in the *Apache2* program in your server and you need to restart it. Simply type: $ sudo *etc*init.d/Apache2 restart This would restart the *Apache2* service, which would bring online whatever configuration changes that you have made. Now, let's say you are playing with the web server, you are making some changes, and you do not want anybody from the outside world coming in while you do the changes. So you want to stop the services entirely, which basically makes the web server go offline. All you do is: $ sudo *etc*init.d/Apache2 stop The above command stops the web service. However, it does not stop anything else. This means that while nobody from the outside world is able to reach the website that you are hosting, you are still able to make changes to the configuration files, edit options, modify settings, *etc.* Only the actual web server component is not functioning at this point.

Let's now go ahead and look at how this looks like when you actually execute it at the command prompt of Linux.



So right now we're at the command prompt or terminal of our Linux server operating system. In this example, let us assume that you have *Apache2* installed on your system, and that you want to stop that service. The first thing that you should do is type in the word *sudo*, put a space after the *sudo*, then type in *etcinit.d/Apache2*, put a space again after that, and then the word *stop*.

As soon as you hit ENTER after typing in the aforementioned command, you will get the below screen:



As you can see in the screen above, Linux has successfully stopped the service. At this point, the domain that this web server is hosting is not accessible over the Internet. The website is currently offline. To start the service, all you need to do is type the same command that we typed when stopping the service, with the exception of typing *start* instead of stop at the end.

```
Welcome to the Ubuntu Server!
 * Documentation:  http://www.ubuntu.com/server/doc

  System information as of Fri Aug 20 11:42:30 EDT 2010

  System load:  0.14              Processes:           66
  Usage of /:   11.2% of 7.49GB   Users logged in:     0
  Memory usage: 14%               IP address for lo:   127.0.0.1
  Swap usage:   0%                IP address for eth0: 10.0.2.15

  Graph this data and manage this system at https://landscape.canonical.com/

   @server:~$ sudo /etc/init.d/apache2 stop
 * Stopping web server apache2
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
 ... waiting                                                          [ OK ]
   @server:~$ sudo /etc/init.d/apache2 start
 * Starting web server apache2
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
                                                                      [ OK ]
   @server:~$
```

When restarting a service—starting and stopping a service automatically—simply enter the same command. The only exception is instead of typing *start* or *stop* at the end, you type *restart*. Again, what is incredibly nice with this in Linux is that you do not have to restart the entire server.

If you restart an entire server, no matter how fast the operating system is, it is going to take a couple of minutes before everything is online. If you only have to restart the service, everything will be back online in seconds.

```
Welcome to the Ubuntu Server!
 * Documentation:  http://www.ubuntu.com/server/doc

  System information as of Fri Aug 20 11:42:30 EDT 2010

  System load:  0.14              Processes:           66
  Usage of /:   11.2% of 7.49GB   Users logged in:     0
  Memory usage: 14%               IP address for lo:   127.0.0.1
  Swap usage:   0%                IP address for eth0: 10.0.2.15

  Graph this data and manage this system at https://landscape.canonical.com/

   @server:~$ sudo /etc/init.d/apache2 stop
 * Stopping web server apache2
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
 ... waiting                                                          [ OK ]
   @server:~$ sudo /etc/init.d/apache2 start
 * Starting web server apache2
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
                                                                      [ OK ]
   @server:~$ sudo /etc/init.d/apache2 restart
 * Restarting web server apache2
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
apache2: Could not reliably determine the server's fully qualified domain name,
using 127.0.1.1 for ServerName
                                                                      [ OK ]
   @server:~$
```

Top

The next thing that we need to talk about is the command called *top*. If you are coming from the Windows world, think of the top command as basically your task manager. So, if you go the Linux command prompt and type in *top*, what happens is you will see how much your memory is being used, how much of your CPU is being used, and then all of the processes that are currently running on your system.

You will see something called the processor ID or the PID, the CPU usage per process, the memory usage per process, how long the process has been running, *etc*. So basically, the *top* command is the task manager of the Linux operating system.

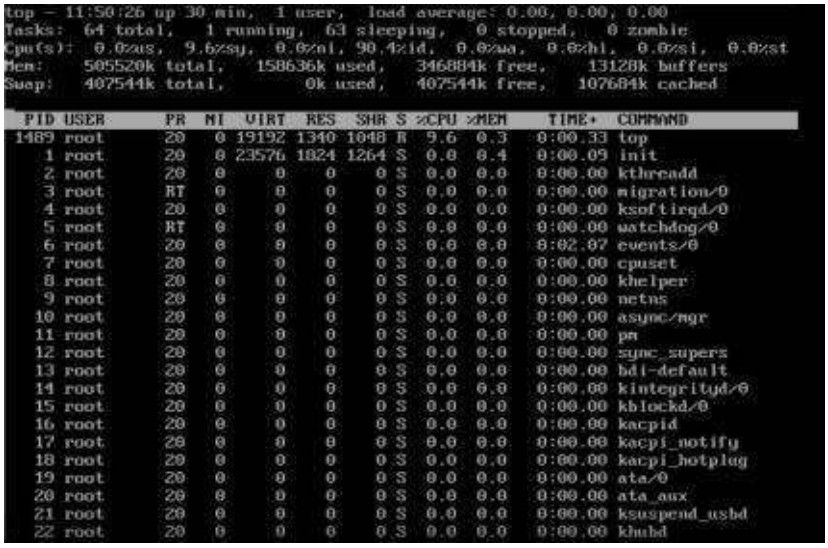
In the Windows task manager, if a process is running and you need to terminate it, you can just do a right-click using your mouse and do *End process*. In Linux with *top*, if you need to terminate or kill a particular process, you just need to type in the syntax below: $ K <PID> So as you can see from the command syntax above, all you need to do is type in the letter *K* and the process ID number. So if you see process *1578* doing something stupid for example, you type in: $ K 1578

Typing that command in will kill that particular process. All the processes in Linux get their own unique process ID number. So if one particular process is doing something that it is not supposed to be doing, it is extremely easy for the user to identify that process through the PID and terminate it.

Remember, the kill or terminate command will only work within *top*. If you are not within the *top* task manager, the kill/terminate command will not work on its own. If the particular process that you are trying to terminate is something important for the Linux server to function properly, it will tell you that the process cannot be killed/terminated. If that is the case, you may have to do your "once a year" reboot in order to kill the process.

So, in order to see the *top* task manager, all you need to do is type in the command below: $ sudo top Once you type that in, you will have the screen

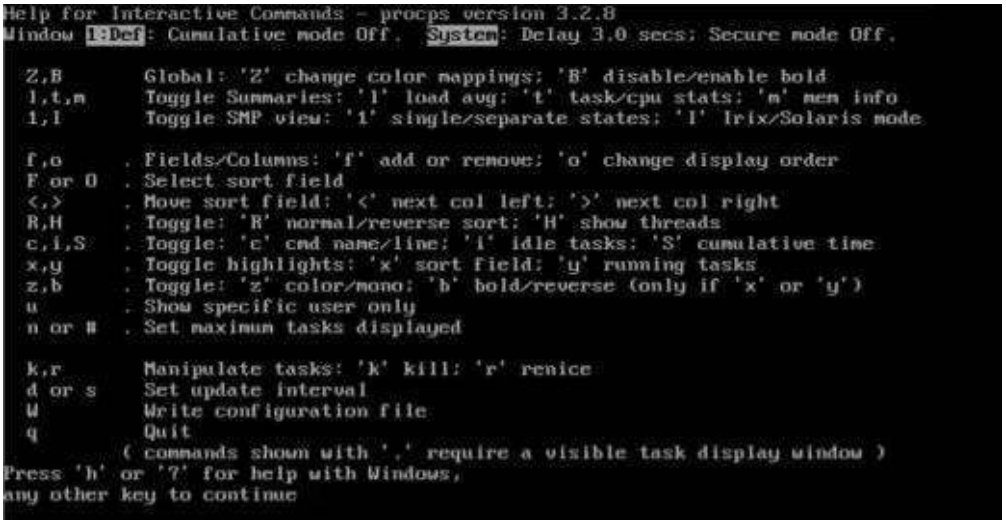
below:



The illustration above shows how the *top* task manager looks like. If you look closely, you can see what the up time for the computer is, the total number of tasks, what the current CPU usage is, the memory usage, buffers swap files, *etc.* This basically gives you the same information a task manager would in the Microsoft Windows operating system.

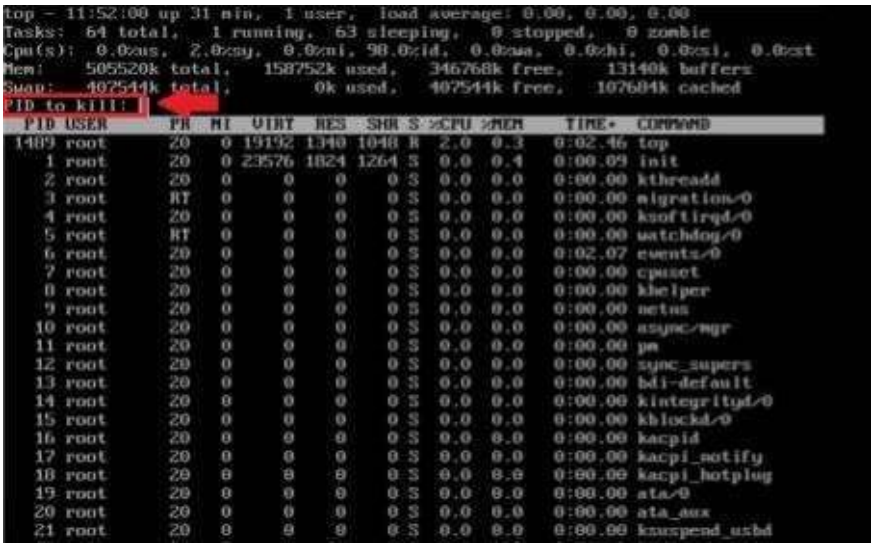
You can see right now, up at the top, there is process ID 1489 and the user is root. And if you look to the far right of that, you will see that process ID 1489 is actually for the *top* task manager process. You can also see the CPU usage, the memory usage, and the time that the *top* process is up and running.

Now, if you need to understand how to use the *top* command better, you can type in the letter "H." Typing in the letter *H* will show you the different commands that you can use with the *top* command.

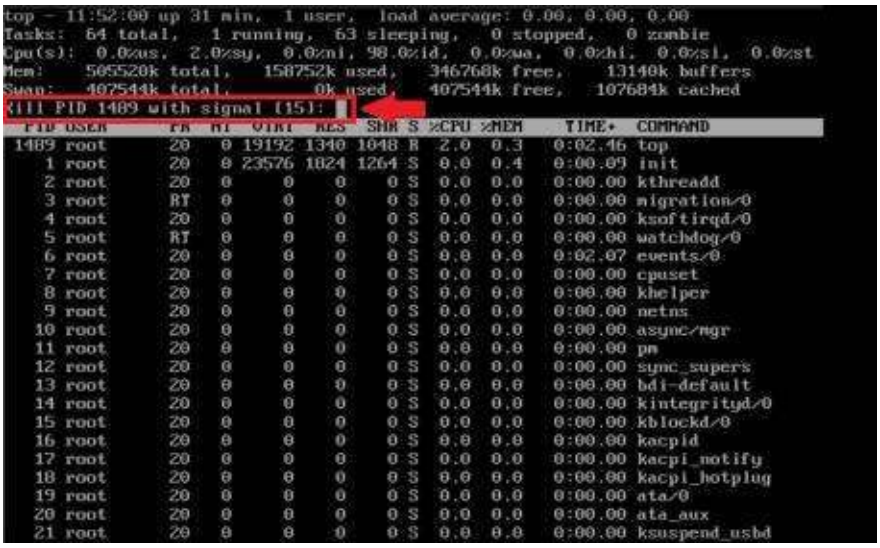


What you see after typing *H* is basically the help page for the *top* command. It displays all the sub-commands that you can use within the *top* environment. To go back to the main *top* environment, just press any key.

Now, let's say you want to kill/terminate a process. Let us say you want to kill the *top* task manager in order to exit out of it. So again, as you can see the process ID for *top* is 1489. First, type in *K*, and then it will ask you which PID to kill.



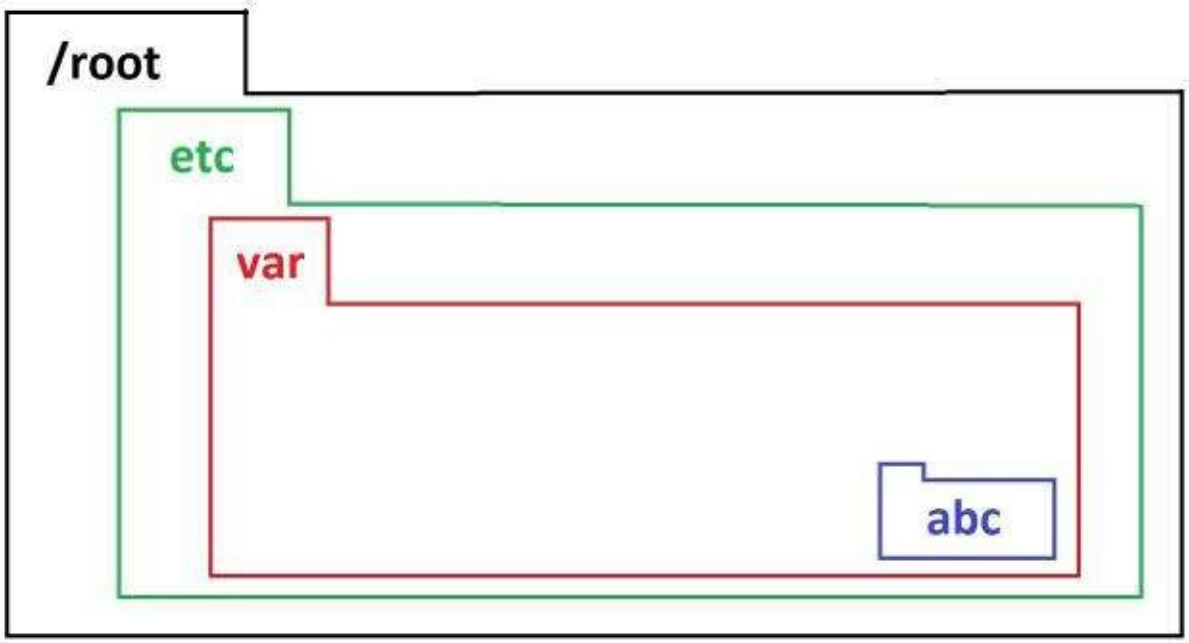
Once it asks you which PID to kill, type *1489* and then press the ENTER key. It will then ask you to re-verify that you want to kill process *1489*.



Go ahead and press ENTER again to terminate the *top* task manager and bring you back to the Linux command prompt. It's that simple.

Chapter 5: Basic Linux Navigation

Now we need to talk about basic folder navigation in the Linux operating system. This is slightly different from Microsoft Windows. However, the funny thing is it looks close enough to Windows that when things do not work right, people have the urge to want to pick up their computer and throw it out the window.



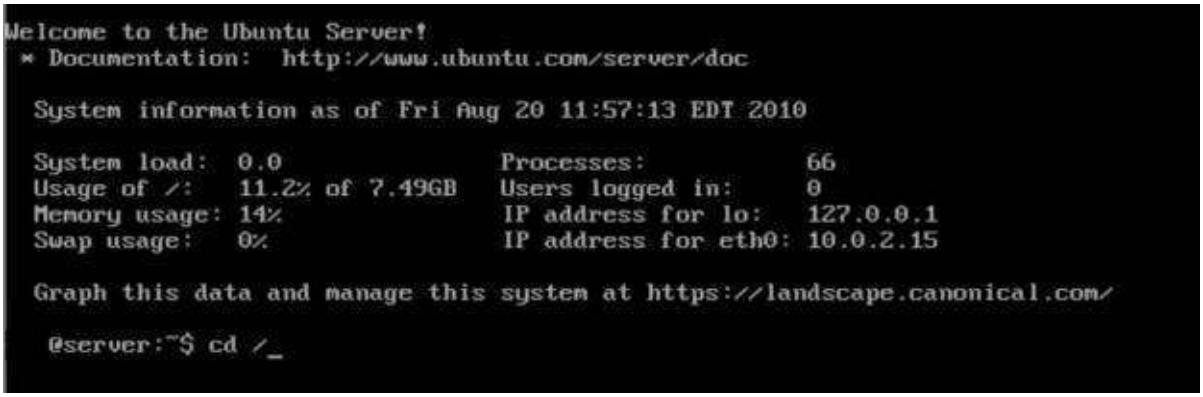
Take the above diagram for example. In Linux, you also have the *cd* command. Just like in Windows, the *cd* command stands for change directory. If you want to go to a different directory or folder, you type in *cd*. If you are in a folder, let us say you are in the *var* folder in our illustration, and you want to go to a folder within the *var* folder, which in this case is the *abc* folder, all you need to do is type in the command below: $ cd abc

So, that is *cd*, a space, and then the name of the folder that you want to go to. That will drop you in the folder that you are trying to go into. Now, the problem is, people in the Windows world are used to following the *cd* command with a forward slash symbol. If we were to apply this in our previous command, it will look like the one below: $ cd /abc What Linux does in this case is it interprets the forward slash symbol as the *root* folder. Linux will think that you want to change to the *root* folder, and then look for the *abc* folder within the *root* folder.

Of course, once Linux executes this command, it will not be able to find an existing *abc* folder since in actuality, the *abc* folder is located in the *var* folder. That is where you have to be extremely careful.

This is one reason why a lot of Linux administrators always type in the full path to wherever they are trying to go, no matter where they are at. If a Linux administrator is trying to go in and change something in the *abc* folder, they will just type in the command below: $ cd /etc/var/abc Linux is finicky about this whole change directory syntax. As we have mentioned before as well, capitalization matters in Linux. If you are trying to reach a folder whose name is written in all lowercase, and you type the folder in all capital letters, Linux will not be able to find it; it will say that the directory cannot be found.

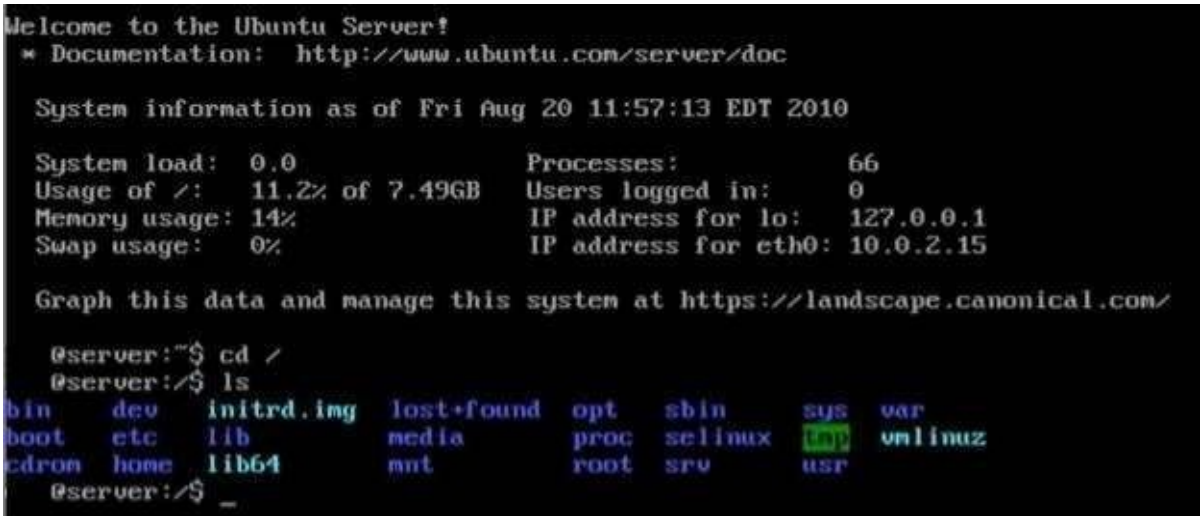
All of this is important for you to understand. It can be extremely frustrating if you do not grasp it.



So, the first thing you want to do when navigating files and folders in Linux is make sure that you are in the root directory. To do this, you must type the command below: $ cd /

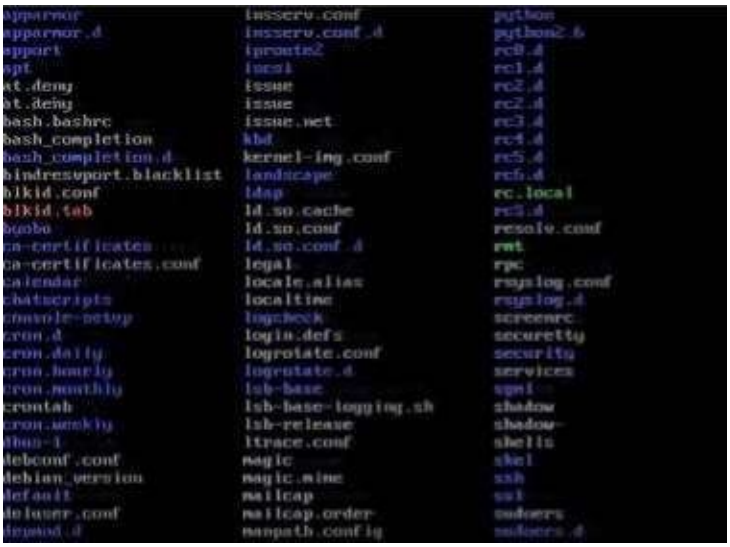
That is *cd*, a space, and then a forward slash. This is automatically bring you to do the *root* directory if you are not already there. Now the next logical thing to do would be to see what folders are within the *root* folder itself. To do this, you must type the command below: $ ls

The *ls*, or list command will list all the files and folders within a particular directory--*root*, in this case.



As you can see after typing in the *ls* command, Linux will now show you all the folders that are within in the *root* directory. You can see lots of folders here like the *bin, dev, opt, sbin, sys, var, boot, etc lib*, and so forth.

Now, let's say we want to go to inside the *etc* folder. All you need to do is type in the command below: $ cd etc The command above will automatically drop you inside the *etc* folder. If you look beside the blinking cursor of the command prompt, it will say */etc$.* That is the main indication that you are indeed inside the actual folder that you want to go into. If you do the *ls* command again, it will show you all the files within the *etc* folder.



In order to go back to the *root* folder, all you need to do is type: $ cd /

This will automatically return you to the *root* directory no matter how deep you

are within the folders.

Now let’s say you want to go back to the *etc* folder. But this time, instead of typing *etc* in all lowercase, you typed it in all uppercase.

As you can see, Linux will not be able to find the directory with an uppercase *ETC*. Why? Because Linux cares about capitalization. Uppercase letters are different from lower case letters in the Linux world. Basically, that is all there is to it in Linux basic navigation.

Chapter 6: Editing Linux Files with Vim

This chapter is about Vim for file editing within Linux. When you are dealing with Linux systems, editing configuration files is an extremely important thing. If you want a piece of software to do something specific, or if you want to customize something, you almost always have to open up a little configuration file and actually edit it. Whether it is a PHP.ini file, whether it is a password file, whether it is the .ht access file, basically all the configurations in a Linux system are held within a simple text file that you have to edit in order to make the system do different things.

Now, there are a lot of different editors out there that many people use to edit configuration files within Linux. We are going to be talking about *Vim*, which is one of the most powerful file editors available for Linux.

One of the big things that you have to understand (since most of you are coming from the Windows operating system) is that in Linux, there are no file associations. What we mean by this is that in Windows, you always have the file extension—association—after the filename. For example, if you have a Microsoft Word document named *Notes*, you will notice that the file extension that it will have is either *.doc* or *.docx*. Therefore, its complete filename would be either *Notes.doc* or *Notes.docx*.

The file extension is what tells the Windows operating system what program to use to open that file. In Linux, there are no file associations. This is one of the main reasons why most people are afraid of using Linux; no file associations at all. Basically, all you have is just a filename. That's it.

Now you may be asking, "How do you know if it is a text file or any other file type?" That is actually the weird thing about Linux. They expect you, as the system administrator, to know what that file is. So if you are going to modify text files, understand that they are not going to have *.txt* file extensions. It will just be the filename. You must understand what file it is you need to modify first before you use a file editor software to edit that file.

All the configuration files in Linux have to be edited using a file editor. If you do not understand how to edit documents or text in Linux, you are not going to get anywhere.

Starting Vim

Now, we need to talk about how to start *Vim*, particularly how do you open or create a file with them. It is pretty straightforward. The first thing that you want to do is type the command *sudo*, followed by a space, followed by the word *vim*, followed by another space, and then the name of the file that you want to open.

> $ sudo vim <filename> Notice that the word *vim* is in all lowercase. Remember: capitalization is essential in Linux, so make sure to type *vim* in all lowercase. The same thing applies when typing in the name of the file that you want to open.

That is all that you have to do to open a file using *Vim*. If, for example, you are trying to open a PHP.ini file, simply type the command below: $ sudo vim php.ini Now, if you are trying to create a new file, the "*$ sudo vim <filename>"* syntax also creates files. So if you type that command in the command prompt, that will create the file and open it at the same time. With that in mind, opening files and creating files uses the exact same command.

For some of you users with experience in using other versions of Linux who are also reading this book, you may notice that you can run the *vim* command without *sudo*. The problem with running *vim* without *sudo*, at least in the Ubuntu distribution, is sometimes it will work right and sometimes it will not. It is a case-to-case basis wherein some of the configuration files will open and can be edited without using *sudo* while others will not edit properly.

And you can run into problems where, if you open a configuration file with simply *vim* and the filename, you will not be able to save the file once you are finished editing it. Why? Because you did not open that file as an administrator. This is why it is always considered good practice to use *sudo* whenever you are doing critical tasks in Linux.

Changing File Ownership

File ownership is another important thing that we need to take note of since we are using *Vim* to edit configuration files. So what do we basically mean by file ownership? Keep in mind that most of the software that you install on your Linux system came from their respective developers. In other words, they came from an external source. When you install them onto your Linux system, the ownership of all of the files of that software technically still falls on the developer.

Now, here is the problem: In Linux, you cannot edit any file unless you are the owner of that file. For example, when you install *Apache2, mySQL,* or *PHP* on your Linux web server, all the files associated with that software is owned by *root* of the source of the file. If you try to go in and edit some of those files, many times you will not be able to edit them because you are not the owner.

So, if you are going to edit configuration files, the first thing that you ought to do is change the ownership from whoever it is to you, just to make your life easy. To do that, you simply type in this command: $ sudo chown <username> <filename> The *username* in the above command pertains to the name of the user that you want to change to the permission to. *Filename,* on the other hand, pertains to the name of the file whose ownership you want to change.

Let us say your username is *reader* and you want to change the ownership of the file named *notes.* All you need to do is type the command syntax below: $ sudo chown reader notes Once you do this, you will now become the owner of the file and thus will have the ability or permission to edit it.

Editing and Navigating

Now that you know how to open a file using *Vim,* the next thing that you need to understand is how to modify a file using *Vim.* The first thing that you will notice when editing a *vim* file is that, if you use the arrow keys to move the cursor and start typing to edit the file, nothing happens. The reason is that you have to enter the *Insert* mode in *Vim*.

In order to enter the *Insert* mode in *Vim*, you simply type the letter "*a.*" Once you do that, you will be able to edit and start typing your modifications on the file. When you are done making your modifications to the file, what you do to get out of the *Insert* mode is hit the ESC key on your keyboard.

So the next thing is, if you edit a configuration file, especially like the *php.ini* or any other main configuration file, keep in mind that these are really long documents. If you are trying to edit a single word or value inside that extremely long file, finding that single word or value alone can be a tedious task.

In order to find something in *Vim,* there are two simple commands that you can use to make your life easy. First, make sure that you are out of the *Insert* mode. Once you are out of *Insert* mode, go ahead and type the command below: :/ <name> That is colon, followed by a forward slash, and then a space, and then word of value that you are looking for within the file that you are trying to modify. Again, remember that capitalization is crucial. Make sure you capitalize the name correctly, or else Linux will not able to find what you are looking for. And if it did find something, it will be the incorrect one.

What this command does is it looks from the place that you are at—your cursor —all the way down through the rest of the document. It will look for the word of value that you typed in your *find* command. Now, the important thing with searching is that you can put in wildcard characters.

Let us say you are looking for the word *max* within the configuration file. To look for *max,* you need to type in the *find* command below: :/ max Now, one of the things most Linux experts do is whenever they are looking for a specific string in a file, they put a wildcard character, specifically an asterisk symbol, before and after the name of the string. The command would then look like this: :/ *max*

What the above command says is you are looking for anything that has *max* in

the middle; whether it is a part of a string or a standalone string. Because sometimes if there is a space or there is a little character before or after the string, it may not show up when you do the *find* command. This is the purpose of the wildcard in cases like these.

Now take note that with the forward slash, *Vim* will search from where your cursor is at going down. So, what if you have reached the end of the file and you did not find what you are looking for? At this point, your cursor is at the end of the file. In order to make a search upwards from where your cursor is at, simply type the command below: :/? <name> That is colon, followed by a forward slash, followed by a question mark, a space, and then the string or value that you are looking for. This is what will make the search go upwards from where your cursor is at.

Now, what if there are multiple instances of the string or value that you are looking for in a *vim* file? Well, when you do a search, *Vim* will automatically halt on the first instance of the string or value that you are looking for. To move to the next instance, you just need to hit the letter "*n*" on the keyboard. This is the equivalent of "Find Next" in the Windows operating system. As you can see, navigating in *Vim* is really easy. The above commands are all it takes to open, navigate, search, and edit files in *Vim.*

Exiting and Saving

The next thing we need to talk about is how open, exit, and save files within *Vim.* So far, what we discussed opening *Vim* files from the command prompt. So now, let us discuss how to open up other *Vim* files while inside *Vim* itself.

If you need to open a file and you are already in *Vim*—so you need to switch to another file—just type the command below: :e <filename> That is a colon, followed by a lowercase letter *e*, a space, and then the file name. Again, make sure you are out of *Insert* mode when you do this. If you are in *Insert* mode and you type in this command, what will happen is *Vim* will just type in the command within the file itself. It will not be interpreted as a command to open up a different file.

That is all you have to do to open up another *Vim* file from within *Vim*. Now, let us say you open up a file and you look around in them. But then, you decide that you do not want to change anything and you need to get out of that file. What you do in this instance is type in the command below: :q

That is a colon, and then followed by the lowercase letter *q*. This command will exit you out of *Vim.* Of course, you have to press ENTER after typing in each command in order to execute them.

There will be times when you also have to force quit out of *Vim.* There may be times when *Vim* has become unresponsive or it does not want to accept the normal quit command. In cases like these, you want to force *Vim* to exit. To do that you have to put an exclamation mark after the lowercase letter *q.*

:q!

Whether *Vim* got stuck or has become unresponsive to other commands, putting an exclamation mark after the lowercase letter *q* will definitely get you out of the *Vim* application. Only use this command if nothing else works. Keep in mind that if you are quitting *Vim* normally using the *:q* or *:q!* command, no changes will be saved in the file that you are trying to edit—unless you saved the changes first before trying to execute the quit command.

Now, let us say *Vim* locked up for some reason and you have already entered numerous changes in the file. Of course, in order for you not to go through all the changes again, you want to save the file before quitting. In this instance,

what you do is type in the command below: :wq

That is a colon, followed by the lowercase letter *w,* and then followed by the lowercase letter *q.* The lowercase letter *w* stands for "write", while the lowercase letter *q* stands for quit. So this will save whatever changes you made to the file first before quitting *Vim*.

The next thing is, let us say you made some changes to the file but you do not want it to be saved to the original file. So you are basically trying to do a "save as" command here. To do that, just type in the command below: :w <new filename> That is colon, followed by a lowercase letter *w,* a space, and then the name of the new file that you want to create and save into. This command will save the file that you edited to a new file. This is particularly useful when you do not want to overwrite the original file and you want to preserve it for some reason.

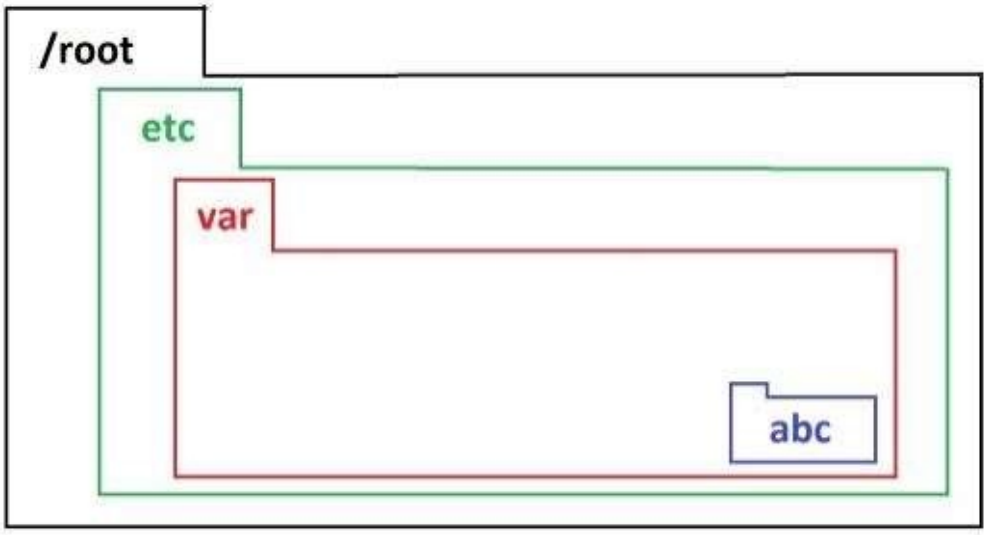
Chapter 7: Advanced Linux Navigation

In this chapter, we will talk about advanced Linux navigation. In chapter 5, we have already talked about how to change directories in Linux. Now, we will discuss how to find folders in Linux, how you make and remove directories, how you copy files, and then finally how you mount drives in Linux.

Changing Directories and Finding Files

In the last chapter, we talked about how you edit configuration files using *Vim*, which is pretty essential in Linux. Now, as you sit there and look at the file system, you probably have no idea where those configuration files are in the first place. If you wanted to edit the *php.ini* file for example, the question that you may be asking since you are a beginner in Linux is, where is the *php.ini* file located?

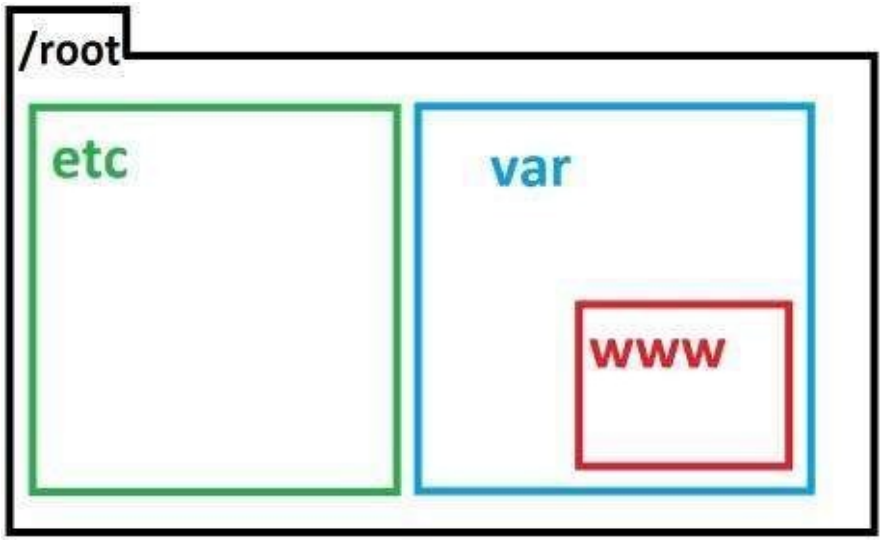
So, to find any particular file that you want to edit, you must first learn how to get to the folder or directory where that particular file is located. Again, to change directories, all you need to do is make use of the *cd* command just like in Windows. The syntax to change directory is: $ cd <name of folder> Now, there is more to this command than meets the eye. Remember, Linux is exceptionally literal when it comes to the interpretation of commands syntaxes. Again, let us make use of the diagram below:



If for example you are in the *var* directory/folder and you want to go to the *abc* folder, all you need to do is type: $ cd abc Now, if for example you are in the *var* folder and you want to go to the *etc* folder, just typing in *$ cd etc* from within *var* will not work. Because then Linux will think that you want to go to the *etc* folder that is within the *var* folder, which does not exist.

What you must do is type the command below: $ cd /etc Take note of the forward slash before the name of the directory that you want to go to. This forward slash will tell Linux to first go to the *root* directory and then find the *etc* folder or directory. Now, Linux will now be able to successfully bring you to the

etc folder since it is able to locate it in the *root* directory.



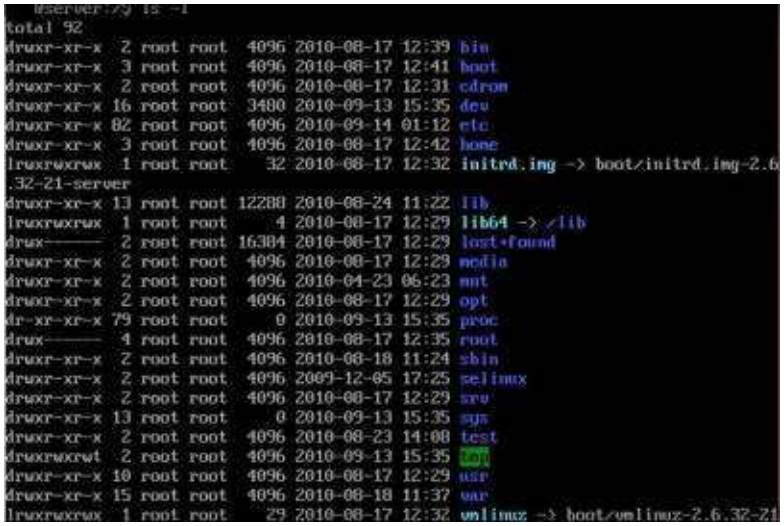
Now, consider the example diagram above. Let us say that you are in the *root* directory and you want to go to the *www* directory. If you type: $ cd www Linux will not be able to bring you to the *www* directory; it will say that the directory does not exist. Why? Because remember, you are in the *root* directory. As far as Linux is concerned, there are only two files within the *root* directory at this point: *etc* and *var*. To be able to go to the *www* directory from *root*, you must type: $ cd /var/www Now you might be asking, won't the second forward slash in the above command bring you back to *root*? No, it will not. Linux only interprets the first instance of the forward slash as the *root* directory. The second forward slash will be interpreted as an instruction to look for the directory that succeeds it, from the directory that precedes it.

Here is another example: Let us say you are in the *www* directory and you want to go to the *etc* directory. If you type in: $ cd etc This command will not work. Keep in mind that you are currently in the *www* directory. As far as Linux is concerned, there is no *etc* directory inside the *www* directory. To make this work, you have to type the command below: $ cd /etc So again, the first instance of the forward slash will be interpreted by Linux as an instruction to go to the *root* directory first. Once it is in the *root* directory, it will then find the *etc* directory. Of course, this command will successfully bring you inside the *etc* directory since it exists within the *root* folder.

Listing/Displaying Files

Now, once you are in a directory, it is always important to find out what else is in that directory. To list files in Linux, you simply use the command *ls*.

> $ ls And then, depending on what you want to do, you can apply one of two arguments. If you do: $ ls -l What will happen is that all the files and folders will get listed. In addition to that, it will also show you the permissions for those files and folders, the date they were modified, the group ownership of the files and folders, and the individual owner.



A lot of times that can turn into a really big file. So, what you can also do is type: $ ls -m This time, instead of Linux giving you a really long list, it just types everything into a nice block so you can see every bit of information about the files and folders. That is all you need to do to list files and folders in Linux.

Now, let's say you want to edit a critical configuration file, but you have absolutely no idea where that file is. Thankfully, Linux does have a search option. In order to search for files and folders, all you do is type: $ sudo find -iname <file/folder name> That is *sudo*, a space, followed by the word *find*, another space, the argument -*iname*, another space, and then finally the name of the file or folder that you are looking for. Now you might be askin, what does that -*iname* argument do? Well, what it does is it makes the search case insensitive.

Remember, just like what we have been mentioning since chapter 1, capitalization matters in Linux. Since you are new to Linux, you may not know

which files have uppercase letters and which files have lowercase letters. If you put in the *-iname* argument in your search parameter, it will locate the file or folder that you are looking for regardless of their capitalization.

If you are looking for a folder named *home* for example, and you use the search command with the *-iname* argument, it will come back with all the folder that has *home* as the folder name. Whether that folder is name as *Home*, *homE, hOme*, *HOme*, *hoMe,* or *HOME,* it will come up in the search result.

Whereas if you mess up the capitalization of the name of the file or folder that you are looking for, and you did not use the *-iname* argument, the search may come back with no results because no file or folder match the capitalization.

Also, do not forget about the *sudo* command when executing *find* commands in Linux. If you do not put in *sudo* at the beginning, the *find* command will fail in the most obnoxious way in that it will not tell you that it failed. It will simply not give you any results. So you will think that the file does not exist on the computer, when in actuality it does exist but you just did not use *sudo.*

With whatever file or folder that you are looking for, you can also use wildcard characters together with their name when doing a search. In chapter 6, we briefly talked about making use of the asterisk wildcard when searching for strings in *Vim.* When finding files and folders within the Linux file system, you can also make use of the asterisk wildcard.

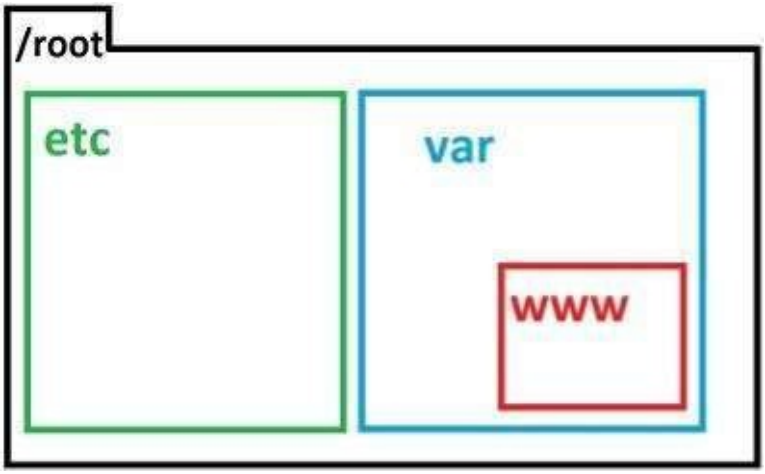
If you put an asterisk before a file or folder name, that means that you are looking for "*anything before*" that particular file or folder name. Let us say you are trying to locate a configuration file in Linux. In Linux, all configuration files have *.conf* suffix. If you are looking for a configuration file but you forgot what the exact name is, you may want to pull up all the files that have *.conf* as their suffix. In this case, what you do is type the command below: $ sudo find -iname *.conf What this command will do is look for a file or folder name that begins with anything and ends with *.conf.* Now, what if you know the filename of the file that you are looking for, but you do not know the suffix. In this case, you need to type the command below: $ sudo find -iname php*

What this command will do is look for a file or folder name that begins with *php* and ends with anything.

Making, Deleting, Moving, Copying, Renaming

Now we need to talk about making, deleting, moving, copying, and renaming files and folders in Linux. The first thing that we are going to do is talk about creating a folder. It is pretty straightforward. All you do is type: $ sudo mkdir <name of directory/folder> That is *sudo*, followed by a space, and then the *mkdir* parameter, another space, and then the name of the folder or directory that you want to make. As you can probably already guess, *mkdir* stands for "make directory."

Keep in mind that if you do not put the full path of the directory, this command will simply create that directory or folder inside the directory that you are currently in.



For example, let us say you are in the *var* directory and you want to create a *Notes* folder inside the *etc* directory. If you type: $ sudo mkdir Notes This command will just create the *Notes* folder in the *var* directory. To create the *Notes* folder in the *etc* directory from within *var*, you must type in the full path just like the one below: $ sudo mkdir *etc*Notes This way, Linux will go to the *root* directory first, and then into the *etc* directory, and within the *etc* directory create a *Notes* folder. Typing in the full file path will ensure that no matter where you are in the file system, you would be able to create the folder in the right location. That is all you need to do to create a folder or directory in Linux.

Deleting files or folders is just as simple. What you do is type the command below: $ sudo rm <file or folder name> That is *sudo*, a space, followed by the letters *rm*, another space, and then finally the name of the file or folder that you want to delete. As you may have already guessed, *rm* stands for remove. Let us

say for example you want to delete a file named *Notes*. All you need to do is type: $ sudo rm Notes Deleting folders is a little bit different, however. You actually still follow the same syntax as when you are deleting a file. The only difference is that you have to add the *-R* argument at the end of the command when deleting folders. The argument *-R* stands for recursive. By using the recursive argument, you are basically telling Linux to do the same task to the contents of the folder.

Let us say you want to delete the *www* folder that has three files in it. To do this, you just type: $ sudo rm www -R

This command will not only delete the folder itself, but also the three files that are inside it. If you do not put a recursive argument at the end of a delete folder command, Linux will not be able to execute it successfully, since you cannot delete a folder without deleting the contents as well. The only time you would be able to delete a folder successfully without the recursive argument is when the folder is empty to begin with.

In Linux, there is no command to rename files and folders per se. Instead of rename, we have the *move* command in Linux. Let us say for example you have a file named *file1*, and you want to change the name to *file2*. To do that you have to type the command syntax below: $ sudo mv <name of old file> <name of new file> That is *sudo*, a space, followed by the letters *mv*, another space, the name of the original file, another space again, and then followed by the name of the new file. So again, going back to our example, if you want to change the name *file1* to *file2*, you have to type: $ sudo mv *file1 file2*

This command will change the name of *file1* to *file2*. It is still technically renaming a file. But in Linux, we use the term *move*. Now, how about moving a file from folder to folder? Well, the same syntax still applies. However, instead of just typing in the name of the files and folders, you must indicate the full path.

Let's say you want to move a file named *file1* inside the *var* folder to the *etc* folder in the *root* directory. To do this, you must type the command below: $ sudo mv *etc*var/file1 *etc*file1

That is *sudo*, a space, followed by the letters *mv* for move, another space, the exact file path of the source file, another space, and then lastly followed by the exact file path to the destination. It is as simple as that.

Copying is another important task in Linux. A lot of times, especially if you are dealing with a configuration file, it is essential that you make a backup file before you start messing around with the original file. Otherwise, if you mess up the original file, you may end up damaging the program that that file is associated with.

To copy files in Linux, all you have to do is type: $ sudo cp <name of the file> <name of the copied file> That is *sudo*, a space, followed by the letters *cp* for copy, another space, the name of the file you want to make a copy of, another space, and then finally the name of the duplicate file. So, let us say you want to make a copy of a file named *file10*. All you need to do is type: $ sudo cp file10 file10.bak Here, *file10.bak* is the name of the duplicate file of *file10*. As you can see, it is easy to make copies of files in Linux.

Mounting Drives

The last big thing that you have to understand in order to navigate Linux is mounting drives. This is where you have an external hard drive, you plug it into the computer, and you need to mount the drive in order for you to browse its contents. That external hard drive could be a normal hard drive, a flash drive, or a CD-ROM drive. Flash drives and CD-ROM drives are considered as hard drives in Linux.

Basically, any drive that you will connect to the computer is a drive that would have to be mounted. The first thing that you have to understand is how the mounting process works in Linux. The mounting process is as follows: Step 1 – Connect the hard drive.

Step 2 – Create a folder for your mount point.

Step 3 – Grab some specific information about the hard drive and then point the mount point folder to that hard drive.

That is essentially how you mount the drive. The mount point folder now gets tied to the hard drive. Since they are tied together, you can now browse the contents of the hard drive through the mount point folder.

So how do you actually make the mount point folder in the command prompt? Simple: just type the command below: $ sudo mkdir *mnt*<name of mount point folder> Take note of the */mnt* directory that we are creating here. Know that */mnt* is the standard mounting directory in Linux. This mounting directory is universal. You can use a different mounting directory if you want. However, using */mnt* is considered as a best practice since most Linux administrators and programmers will use it by default.

Now, let's say you want to mount a connected hard drive to a mount point named *drive1*. To do this, you just type: $ sudo mkdir *mnt*drive1

By typing the command above, you are creating the folder that your hard drive will be mounted to. Once you do this, the next thing that you have to do is find out the information about the hard drive that you are trying to mount. To do this you just run the command below: $ sudo fdisk -l The above command will list all of the physical hard drives that are connected to your system. This list is going to show you the different hard drives that are connected to your system,

how much space they have, *etc.* In there, you are looking for something that is going to look like *devsda1*. These are called disk names in Linux.

Linux disk names are always arranged in an alphabetical order. If you have two hard drives connected to your Linux system, for example, they will be named *devsda* and *devsdb*, respectively. The numbers that succeed the disk names refer to the partitions within that particular disk.

If you have two partitions in your first hard drive for example, they will show up as *devsda1* and *devsda2* respectively. If you have three partitions in your second hard drive, they will show up as *devsdb1*, *devsdb2*, and *devsdb3* respectively.

So, in order to mount the hard drive, all we are going to do once we find the disk name is type the following: $ sudo mount <disk name> <mount point> If we were to apply this command syntax to our example, the command should look like this: $ sudo mount *dev*sda *mnt*drive1

This is all you to do mount your hard drive. Now, you can go and change directory to *mntdrive1* by typing: $ sudo cd *mnt*drive1

And once you are in your mount point directory, you can type: $ ls -l The above command will list all the contents of that hard drive. You can now apply all the Linux navigation commands that you have learned so far in your mounted drive. That is basically how you mount a hard drive in Linux.

Now, if you are done with that hard drive and you want to un-mount it, all you need to do is type: $ sudo umount <disk name> <mount point> As you may have already guessed, *umount* stands for un-mount. Once you have successfully un-mounted a hard drive, you can repeat the same process of mounting a drive if you want to use a new hard drive.

Conclusion

There you have it. That is all there is to the basic installation, basic task commands, basic and advanced navigation, and editing files in Linux. Again, like what we have mentioned in the other chapters, the only reason this is intimidating and the only reason anybody is nervous about this is that they do not know what commands to run. Once you understand the commands that you need to run, it all becomes easy.

Once again, these commands, though they might seem complicated to you at this point, are just pretty basic. If what we have showed you in this book does not help you do everything that you need to do, then by all means use the *man* pages in order to learn the more complicated commands that you need.

With Linux, the sky's the limit on the things that you can do with the operating system. We are basically just bringing this down to a level that beginners like you can learn and understand quickly.

We would like to thank you for buying this book. We hope that you learned a lot about the Linux operating system and its basic commands. Feel free to make this book your beginner's quick guide as you explore the intricacies of this fantastic operating system.

At this point, we would like to encourage you to tinker and play around with the Linux operating system and its numerous distributions. Keep in mind that different distributions have different features. Some of the commands that are indicated in this book may already be an automatic process in another distribution. For example, desktop distributions of Linux such as Linux Mint, Arch Linux, Peppermint Linux, Kali Linux, Puppy Linux, *etc.* all perform the hard drive mounting task automatically. Not only that, they also allow you to apply various themes to the GUI, make use of widgets, *etc*. to give the operating system a more customized feel and look.

Linux is a truly wonderful operating system, and it has now reached a level where it can compete with the more popular operating systems out there. We hope that this book becomes your first steppingstone to what may be the future of computer operating systems—Linux.