1.1.1 Linux Operating System Introduction

Linux Operating System Introduction

The introduction and adoption of the Linux operating system has been a really interesting thing to observe. Understand that when Linux was first introduced back in the early 1990s, it pretty much went unnoticed. Nobody paid attention to it, and that's because the professionals in the information technology industry at the time were focused on the big operating systems of the day, which included Microsoft Windows, Novell Netware, Mac OS, and Unix.

Since that time, things have changed dramatically. Linux is now a mainstay, especially in the server room, for many major organizations around the world. Using the wide variety of network services that are available for the Linux operating system, it can be configured to perform just about any networking role that any competing server operating system can perform.

In addition to the server room, Linux has also started making inroads into the desktop market. In many cases, it has even replaced Microsoft Windows on user desktops. Many Linux desktop applications are available, and most of them are available free of charge, that allow end users to perform their day-to-day work, such as doing word processing, creating databases, making spreadsheets, making presentations, and so on. There is a growing demand for network administrators who can implement, maintain, and also support the Linux operating system.

If you've had any exposure to Linux at all, you know that it's very different than the other operating systems that most users are already familiar with, such as Windows. Therefore, migrating to Linux requires a degree of expertise. In order to gain that expertise, you first have to understand the role that Linux plays in a computer system. Understand that Linux is an operating system. It's not an application that runs on top of Windows or some other operating system.

Linux provides several key functions in a computer. These are functions that are provided by all operating systems. First of all, the operating system provides a platform where applications can run. Basically, its job is to manage the application's access to the CPU and the memory that's installed in the system. The operating system also serves as a kind of mediator between those applications that are running on the system and the system hardware. For example, one of its key jobs is to make sure that one application running on the system doesn't try to use an area in memory that's already in use by another application. It's also responsible for ensuring that a given application running on the system doesn't use too much CPU time such that other applications running on the system can't use the CPU.

Linux Components

The operating system is also responsible for providing an efficient and reliable means for storing data. This is usually done using some type of storage device, like a hard disk drive. And that storage device has been formatted with a particular file system, and its job is to organize that information in an easily retrievable format. In other words, when we save data on the system, we want to be able to get at it again, and we expect that that data has not been changed in any way in the meantime. The operating system is also responsible for providing some degree of security for that data that's stored on it. Finally, the operating system provides some type of connectivity between computer systems over a network connection. It can do this using a variety of different network media and interfaces. For example, we can create an Ethernet connection between computer systems. In order to fulfill these roles, the Linux operating system employs several key components, and we'll review them here. The first is the Linux kernel. The Linux kernel is, really, the core or the heart of the Linux operating system. It really is the actual operating system itself. It is the component that fulfills the key operating system duties that we just talked about. The Linux operating system also provides libraries. These libraries are very useful for programmers. They contain pre-written code elements that the programmers can use within their programs. This is a big time saver.

Imagine if you're a computer programmer, and when you're writing an application, you have to include code within your application that allows it to work with every type of hard drive that's ever been made so that we can save data back and forth onto the hard disk drive. Using libraries, though, it doesn't matter to the programmer whether the system has a standard SATA drive installed, if it has an IDE drive installed, if it has a SCSI drive installed--it doesn't matter. The programmer simply calls the appropriate library and tells the operating system that it needs to write data to whatever hard drive is installed in the system, and then the operating system takes care of the rest using these libraries.

The Linux operating system also includes a wide variety of utilities that you can use in order to complete operating system management tasks, such as creating files systems, maintaining file systems, editing text files, managing the applications that are running on the system, installing new applications on the system, and so on.

Finally, the Linux operating system provides the end user with a means of interacting with the operating system. That's called a user interface, and Linux actually provides two different user interfaces that you can use. The first one is a graphical user interface, which you see right here. It's a lot like the graphical environment used in other operating systems. When the user wants to do something, they can, say, click on a button right here, click an option, and then the operating system will do whatever it's been programmed to do.

In addition to the graphical user interface, which uses a mouse, Linux also provides a text-based command line interface. In this interface, you type commands and then press Enter, and the operating system does whatever it's been programmed to do. It's actually this interface that we're going to focus on in this course, because a Linux system administrator needs to know how to perform tasks from the command prompt. There's a very good reason for this: because a lot of Linux systems—"in fact, probably a majority of Linux systems--are implemented as servers. Server administrators do not want their system processors and memory to be wasted re-drawing these graphical screens that you see right here. They'll actually disable the graphical user interface and do everything from the command line interface. It's just a much more efficient interface from a hardware perspective, and so you've got to know how to do things using this command line interface.

With this background in mind, let's now talk about how the Linux operating system itself came to be. Understand that Linux is kind of an anomaly in the software development industry. Most software products, whether it's an application or an operating system, are developed as a part of a well-organized design and development effort. I've worked for many years in the software development industry, and I've seen how this works firsthand.

Here's what happens, usually, in most companies. First of all, the organization defines some customer need, and then a design team is put together. It is usually composed of programmers, and project managers, and marketers, and so on. Then that design team hashes out a product requirements document, or PRD. That PRD specifies exactly what the product is going to do. Then the tasks identified in the PRD are assigned to teams of programmers, who then write their assigned code pieces. When that's complete, the code is checked in, and the product is run through a series of testing cycles. When the product has had its bugs worked out, or most of its bugs worked out, then the finished product is actually shipped to the customer.

At that point, the customer then uses the product for a period of time. While they do that, they usually find other bugs that were missed while we were testing the product in-house.

Standard Operating System Development Cycle

In addition, as they use the product, they usually figure out some new features and functionality that they would like to see added to the product. The software company receives feedback from the customers, and the cycle begins all over again.

This is how most commercial software products are developed.

Well, interestingly, Linux didn't conform to this cycle when it was originally developed. Instead, a graduate student at the University of Helsinki in Finland named Linus Torvalds developed the Linux kernel. Way back in the early 1990s, Torvalds became interested in a minimal, freeware, Unix-like operating system called MINIX. There was a professor named Dr. Andrew S. Tanenbaum who taught computer programming in the Netherlands, and he developed MINIX as a clone of the commercial Unix operating system.

You see, at one point in time, the source code to the Unix operating system had been made freely available to universities for educational purposes; however, in the late 1980s, this practice had actually been stopped. That left Tanenbaum without an effective tool to teach his students about the inner workings of an operating system. Therefore, Tanenbaum decided to make his own operating system, and he would use it in class to teach his students with. He developed a small clone of the Unix kernel called MINIX, and his goal was, basically, just to provide his students with real operating system code that they could work with.

Now, Torvalds was inspired by Tanenbaum and MINIX, so he developed his own Unix clone in 1991, and he dubbed it Linux.

Linux Development Cycle

This first version of Linux was not much of an operating system. It was very minimal in nature. It didn't have the full set of applications and utilities that you would expect an operating system to have. Instead, Linux version 0.02, which was released in October of 1991, consisted of the Linux kernel and the three basic utilities. It had a Bash shell, which provides a command line interface. It had an update utility, which is just used for flushing the file system buffers, and it had GCC, which is a C++ compiler that allowed you to write your own programs.

Then, in an unprecedented move, Torvalds actually took the source code for the Linux operating system, and he posted it on the internet. He made it freely available to anyone else who wanted to download it.

With that, the corporate software development model we just talked about was completely broken. Torvalds even took things one step further. He actually invited other programmers around the world to go ahead and just modify that source code that he had posted, enhance it, and make it better. At this point, Linux took on a life of its own, and it became a worldwide collaborative development project. There was no secrecy. There were no tightly guarded copyrights. Access to the source code was open to anyone who wanted it. Most corporate software developers did not do this thing.

This collaborative development project on Linux continued for several years until 1994, when Linux version 1.0 was finally ready for release.

The results since then have been really, really cool. It's gained a lot of traction around the world. At this point, you might be wondering, "Why did Torvalds give away Linux to anyone who just wanted it? Why didn't he follow the standard corporate model and try to make a mountain of money off of it?"

To understand this, you need to be familiar with the GNU's Not Unix movement, or GNU. Back in the early 1980s, there was a programmer named Richard Stallman who worked at the Massachusetts Institute of Technology, and he proposed an alternative to the standard corporate software development model. He objected to the proprietary nature of the process and the final product. In 1983, Stallman launched the GNU project centered on the idea that the source code for applications, as well as operating systems, should be freely distributable to anyone who wants it. He felt that the source code for programs should be free from all restrictions that prevent copying, modification, and redistribution.

The idea behind this was that allowing programmers around the world to modify an application's source code would produce higher quality software.

Software developed under GNU is frequently referred to as free software, and a variation on the free software concept is called open-source software.

Torvalds, who made Linux, was heavily influenced by the GNU project, and released the source code for his Linux operating system kernel to the world.

GNU's not Unix

As a result, the Linux kernel itself is licensed under the GNU general public license, or the GPL. The key point to remember about the GPL is that it requires the source code to remain freely available to anybody who wants it. As a result, you can actually download the Linux kernel source code, you can modify it, you can re-compile it, and you can run it. You can even create your own custom version of Linux, which would be called a Linux distribution.

At this point, we need to stop and talk about the concept of a Linux distribution, because it's really confusing to those who are new to Linux. Perhaps the best way to think of a distribution is to compare the Linux operating system to ice cream. As you know, ice cream comes in a wide variety of flavors; however, the basic formula for ice cream itself is the same. Most ice cream is made from cream or milk, depending on the quality of the ice cream, sugar, and eggs. Companies that sell ice cream will take this same basic ice cream recipe, and then they'll customize it by adding in other ingredients. They'll throw in, maybe, some chocolate, or some vanilla, or fruit, or cookies, or nuts, or candy, whatever it is that they want to do. By doing this, they create their own custom flavors of ice cream.

Well, Linux distributions work in much the same way. The kernel source code really is comparable to this basic ice cream recipe. Because this kernel source code is freely distributable, other programmers are free to download it.

Linux Distributions

Just as ice cream companies add additional ingredients to the basic ice cream recipe, these programmers can then modify and enhance the Linux source code and create a customized kernel. They can also add specialized tools. They can add additional utilities. They can write new applications for the operating system in order to enhance its usefulness.

The result is a Linux distribution.

Currently, there are many different Linux distributions available. Programmers around the world have taken the basic Linux kernel, and they've modified it to suit some particular purpose. They may have also created their own customized powerful applications to run on their distributions. For example, some distributions may be customized to provide high-end network services to remote computer users. Other distributions might be customized to run productivity applications on an end user's desktop. Either way, the result is a customized Linux distribution. Today, there are literally hundreds of different distributions available.

Some of the more popular ones are shown here, such as OpenSUSE, Ubuntu, Fedora, Red Hat Enterprise Linux, Oracle Linux, Debian Linux, CentOS.

This is just a short list. There are literally hundreds and hundreds of different distributions that you can choose from.

Popular Linux Distributions

One question I always get asked by students is, which distribution is the best one? That's actually a dangerous question because a lot of folks who use Linux can almost come to blows while debating which distribution is the best. That's because the distribution that works best for you may not meet the needs of somebody else. The key here is to try out several different Linux distributions and then pick the one that you like.

Summary

That's it for this lesson.

In this lesson, we talked about what Linux is and how it came to be. We also talked about the components that make up the Linux operating system. We talked about the historical development of Linux, and then we ended this lesson by talking about the concept of a distribution.