


Version 2.0

Intro to Unix/Linux!

Welcome. In this video I will be introducing you to a very brief history of Unix and Linux operating systems and explain a little bit of its design philosophies.



Part 1: History

- Unix released in 1969
- Ken Thompson (sitting) and Dennis Ritchie working together at a PDP-11
- https://en.wikipedia.org/wiki/History_of_Unix

Unix was released in 1969. Ken Thompson and Dennis Ritchie from Bell labs, designed and implemented Unix, on a Digital Equipment PDP-7 machine.

They were initially working on a larger project in collaboration with others from AT&T and Bell labs, called Multix.

Multix was an ambitious OS that worked on much larger, more expensive computers. It was designed to do things like adding memory and CPU on the fly. Basically it was being designed for very special kind of computers.

In the beginning, there was (no) unix

- [On a PDP7], in 1969, a team of Bell Labs researchers led by Thompson and Ritchie, including Rudd Canaday, implemented a hierarchical file system, the concepts of computer processes and device files, a command-line interpreter, and some small utility programs, modeled on the corresponding features in Multics, but simplified.^[3] The resulting system, much smaller and simpler than Multics, was to become Unix.

https://en.wikipedia.org/wiki/History_of_Unix

Ken Thompson and Dennis Ritchie took some of the innovative ideas that they had from working on Multix, and implemented it on a much simpler form, so it could run on a mini-computer.

Some of the innovative approaches they used includes a

- hierarchial filesystem
- Computer Processes
- And a Shell: a command line interpreter to interact with the system.

They also created small utility programs, modeled after their Multix counterparts.

A combination of all these became the Unix Operating system.



Let's look at the type of hardware that Unix was running on.

This is a picture of the actual PDP-7 that Ken Thompson and Dennis Ritchie used to create Unix.

- As you can see, there are no CRT monitors, there is a teletype unit. Teletype is a typewriter connected to the system. User can type the commands that are transmitted to the system, and the output from the system is printed on the paper.
- For loading data and programs, there is a paper tape reader, which uses punch hole to represent the binary data. As you can see, a lot of programming was done by pen and paper, and not actually on the computer itself.
- Additionally there is an Oscilloscope here. This is for troubleshooting the circuits inside.
- There is no CPU or integrated IC's. Inside these cabinets there are logic boards. Each logic board implements logical operations such as "AND" and "OR" gates. These boards are wired together in order to function like a CPU, Memory, and other parts of the computer.



Here is a later model of PDP computer, a PDP11.

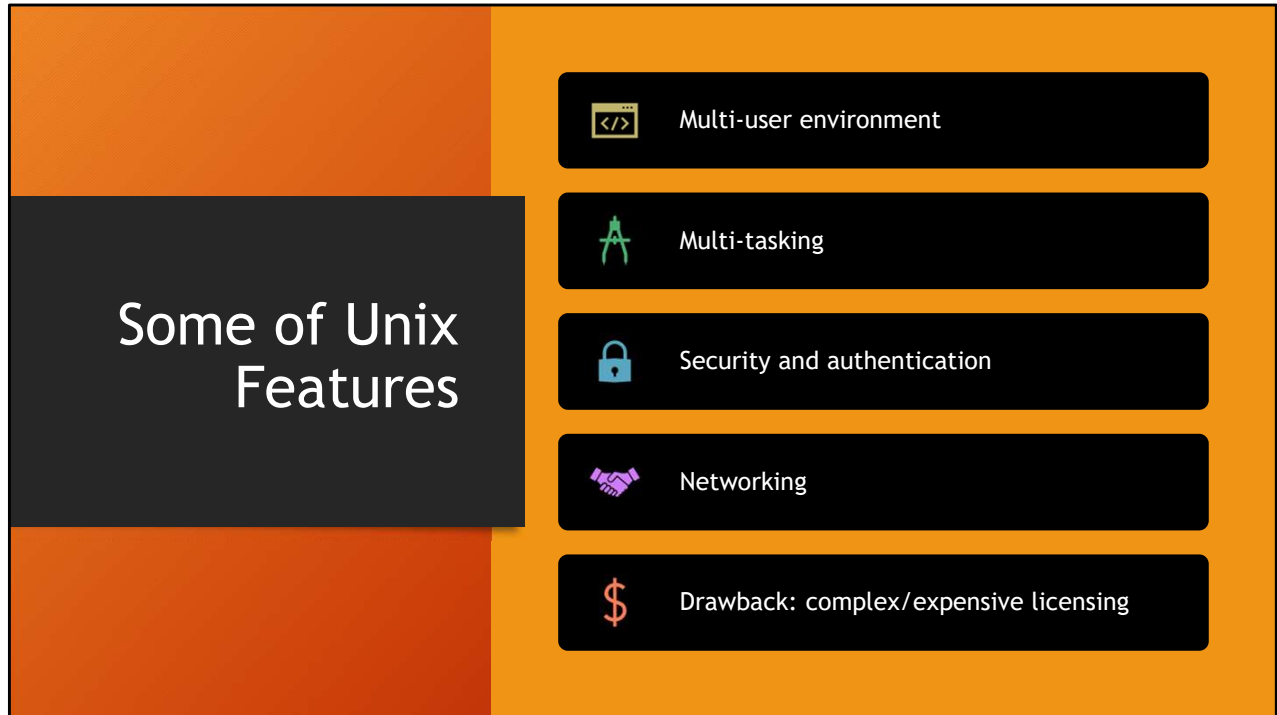
This one does have a CRT console, and magnetic tape drive. Let's take a close look at a later model's control panel.



- Here you can see the control panel of a PDP11/40 system.
- There are physical buttons at the front for manually loading instructions and data into the system.
- When this machine was powered on, there was no such things as plug-and-play, or BIOS. The operator had to manually boot-strap the machine and give instructions – in machine code none-the-less – in order to load program or data, wherever that program and data was stored.
- The program and data could have been loaded from paper tape with punch hole, or much fancier magnetic tapes, or by other methods.
- As you can probably tell by now, these computers are not very powerful. They have equivalent processing power of an Arduino.
- The reason I'm giving you this information is this: There are preconceived ideas that Unix is hard and Linux is complicated. I'm trying to let you know that these computers were very simple. In fact the mini in Mini computer doesn't mean "Small", but rather it stands for minimal.
- Since the computers themselves are very simple and minimalistic, Unix had to be designed to be simple and minimalistic too.
- If you understand the logic and philosophy behind the design of Unix, it will simplify the

learning. You will understand that Unix and Linux are simple, and not as complicated and complex as you might think. You may also come to appreciate the elegance in simplicity of design, that although our systems have evolved in more than 50 years to become more complex, this simplicity still applies and has not changed much.

- Hardware has evolved, but software has largely remained the same.



Unix has turned out to be a great success and is being widely used.

Because of the environment it was created in, it inherently has some features that other “Desktop operating systems” did not have.

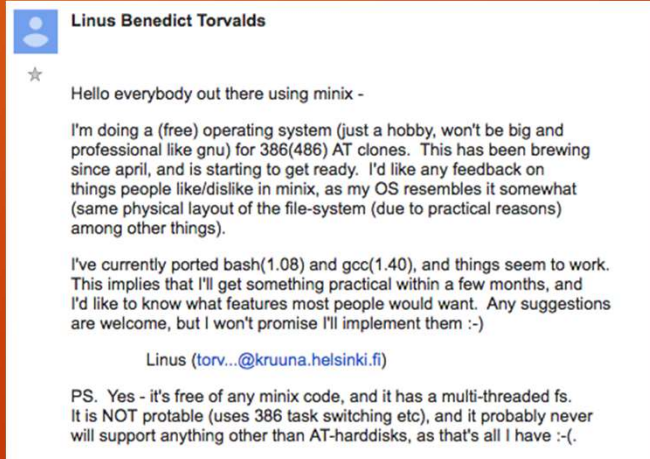
- It was multi-user, with username and password authentication.
 - Of course it would be, because the computers Unix was running on, were shared computers.
- It was capable of multi-tasking.
 - it had to be, because you have multiple users logging in, and running their programs on the machine at the same time.
- Security is built into its file system. Files have permissions assigned to them for the owner of the file, the group, and other users.
- It is capable of networking with other computers also.
- If you compare Unix to other operating systems, such as MS DOS and Windows, or Apple and MacOS of the time, those PC operating systems did not have any of these features. They did not have authentication, they could not connect to networks, and DOS could not multi-task.
- The only big drawback with Unix was basically the licensing fees. The OS was licensed and copyrighted, because it was a commercial product. And it was expensive, because it was marketed to businesses with deeper pockets, whose business operations were large

enough so they could afford to have a computer room or two.

- So for individuals who could afford to buy the “micro-computers” and the PCs of the day, the licensing fees were very steep.

Origin of Linux Kernel

- IBM introduced PC (personal computers) based on Intel 8080 (later on became x86 architecture)
- Linus Torvalds was a student who used Unix at school, and wanted similar OS for himself at home, started writing clone of Unix, initially as a hobby
- Linus Torvalds made "Linux" source code available for free in 1991
- Free software Foundation, GNU started using Linux as their kernel



Back in 1991, Linus Torvalds was a student who had been exposed to Unix at university. When he got his own PC, he was not satisfied with the available solutions in the market. As a hobby project, he decided to put what he had learned from Minix – which was a free educational kernel written for teaching operating systems at schools – into practice, and create his own operating system kernel.

The result was Linux kernel, which is an implementation of Unix features and philosophies, but written from ground up, so not subject to Unix copyrights.

He then shared his project for free for anyone to use.

Linux kernel was further adapted when Free software movement, GNU, picked up the project and used the Linux kernel for their free software.

Technically, when we are talking about a Linux operating system, most of the times we mean:

The Linux Kernel, plus GNU utilities and other software, bundled together to create a fully functional operating system.

Linux is the kernel only.



Part 2: What you need to know as a new user

You want to learn how to use Linux, but you are a new user. You may have never used this OS. What are some things that you need to know and also how is this different from Windows?

Unix/Linux is files!

- FHS (Filesystem Hierarchy Standard)
 - / ← root
 - Everything is located under root
- Everything is file!
- File types:
 - Regular files, and directories
 - Configurations, settings, data
 - Device files
 - Storage, Webcams, Network Cards, Serial Ports, ...
 - Links
 - Etc.



Source: <https://commons.wikimedia.org/wiki/File:Standard-unix-filesystem-hierarchy.svg>

First thing you, as a new user, need to know, is how the file system is arranged and organized.

There is a Filesystem Hierarchy Standard for Linux. It is organized like a tree:

- / <-- root
- Everything is listed under the root, including other storage volumes and disks.
 - Unlike DOS or Windows there are no drive letters representing different storage volumes. But I'll explain that farther in a bit.

In Linux everything is stored or represented as a file, and the file types are very generic. Unlike Windows file types do not include things like a document file, an image file, or a video file.

File types are instead things such as:

- regular files, that can contain text or binary data
- directories, which are files containing a list of files
- There are also device files and links (somewhat similar to shortcuts in Windows).

All configurations, settings, attributes of the system and programs are stored in files. There is no registry like there is in Windows.

Even devices have files. Webcams, network cards, serial ports, sound cards, printers, and every other hardware may have a device file.

If you are accessing a device, for example playing a sound through your soundcard, or getting video from a webcam, you just write to, or read the stream of data from the device file. If you have a properly formatted data that your hardware device can understand and process, you can, for example play a wave file on your sound card, or a print a postscript file on your printer, just by copying the data into the device file.

NO DRIVE LETTERS!

- A storage volume is mounted in a directory under / (mount and umount commands)
- Example of Advantages over having drive letters:
 - /home can be on a completely separate physical disk from / or /boot
 - /boot, where the kernel/OS images are loaded from, can be stored anywhere, even at a network location
 - All of these can be transparent to users!

As I just mentioned, there are no drive letters representing storage volumes or removable media.

A storage volume or a filesystem is “mounted” under a directory somewhere under the root. Then you simply navigate to that directory to access the storage volume.

Here are some examples of use cases for this:

- You can completely separate your user data and other system files by mounting another volume in a separate physical disk to /home directory at bootup.
- You can also have a /boot directory that is stored in another machine on the network, so you can control the OS that each host uses to boot, from a central location. This is useful in large corporate environments where they have one image that many computers can boot from. You can also think of AWS, or Microsoft Azure data centers where they launch many thousands of instances of a particular operating system without having to have as many copies of it sitting on as many disks.
- And the best part of this is that the end users won't even know this is happening. From their perspective everything is exactly where they want and need. This is useful for managing and administrating a large number of system and Linux is the operating system of choice for many cloud environments.

NO Add/Remove programs!

- Linux is a "File-based" OS
 - all the configurations, devices, applications are files!
- To run a program:
 - Copy binaries anywhere on the disk (that you have permission to write)
 - change the permission to allow execute and run!
- No installers, No Registry, No add/remove programs.
- Linux Distros use package management software to aid in managing software

Compared to Windows with a sophisticated registry system to keep track of all your configuration, settings, and location of programs and other resources, Linux is just file-based.

As long as all the binary files and data for a program are present, the program can run from anywhere in your filesystem. You just have to make sure that you have set execute permissions for the executable binary file.

Additionally many open source applications come in form of source code. Linux operating systems comes with compilers and you can compile the source code, and once you have the binaries for your specific system, just run the program.

There are No installers, No Registry, No add/remove programs built into Linux by default.

However Linux Distributions ship with a package management software, and they provide a repository for popular software packages. Using the package repository you can very easily install, un-install, and update and upgrade software on your system, so you can keep things tidy and organized.

Linux is Multi-user!

- Every file has “mode”, owner, and group
- Permissions are assigned for:
 - Owner
 - Group owner
 - Others
- Notations:
 - Octal (777, 644, 650)
 - Alphabetic

Command: `ls -l`

Permission	owner	Group	size	Date	name
<code>drwxr-xr-x</code>	3	root root	4096	Apr 26 2012	acpi
<code>-rw-r--r--</code>	1	root root	2981	Apr 26 2012	adduser.conf
<code>drwxr-xr-x</code>	2	root root	4096	Jul 5 20:53	alternatives
<code>-rw-r--r--</code>	1	root root	395	Jun 20 2010	readme

You, as a Linux user, will have user account on a system. This is the account that you use to authenticate and log into the system.

There maybe other users who also have user accounts on the same system. Your user account can also be member of at least one, but many groups. System administrators use the user and groups of users to mage the access and permissions to the system resources.

The user and group permissions for each file is stored with the file and is built into the filesystem itself.

Firstly, each file has an owner, and is also owned by a group. You can specify permissions for the owner, the group and the others for each file.

In order to explain this very quickly, I have an example output from “`ls -l`” command. This basically lists files, and `-l` is a flag to list the files in “long” format. Long format give us more details on each file rather than just printing the name of each file.

I have added the row at the top to make things clearer. I have colored some of the text as well.

If we look at each entry on the list of files here, there are permissions, the owner of the

file, the group that the file belongs to, and other information such as the size and the name of the file.

In the permissions column there are some letters. The first letter is the file type. For example, the “d” means directory, and the “-” is simply a regular file. The second 3 letters are permissions for the owner of the file.

For example, the “acpi” file here is a directory, and the owner can read, write, and execute the file. Letter “r” is for “read”, “w” for “write”, and “x” is for execute.

The second 3 letter are permissions for the group. Here any user belonging to the group called “root” can also read the file and execute it, but they cannot write to this file. This means that this file cannot be modified by the group.

The last 3 characters are permissions for all the other users on this system. They can also read and execute, but they cannot modify the file.

This is not complicated at all. Knowing this you can simply look at another file on this list and right away you can tell who can do what to this file.

For example the file “readme” cannot be executed by anyone. Also nobody but the owner of the file could modify it. Nobody else has the “write” permissions, they could only read the file.

In the shell or the command line interface these permissions could also be represented in octal format. For example, the binary “111” corresponding to “rwx” is equal 7, “rw-” is 6, and “r--” is 4 when converted from binary to octal. So the permissions for the file named readme is set to “644”, while acpi directory is set to “755”.

su/sudo command

- Switch/substitute User!
- su: “makes it possible to change a login session's owner (i.e., the user who originally created that session by logging in to the system) without the owner having to first log out of that session.” (<http://www.linfo.org/su.html>)
- Some Linux Distros use “sudo” instead

```
#----Example 1----
user@host$ su alice
alice@host$ su
root@host#

#-----Example 2-----
# The “passwd” command is used to
change password
user@host$ passwd
user@host$ sudo passwd
```

Since we assign permissions to individual files, your user account may not have permissions to read, modify, or execute a file. In that case you may want to switch to another user account that does have permission to perform the task that you would like to perform.

- Su or sudo commands are used to accomplish this without having to log out of your session and log back in as the other user.
- One thing that new users seem to get confused about is knowing when to use sudo or su. The answer is simple if you understand fundamental concept of file owners and permissions. To explain this I'll use an example of the command passwd.
- Passwd is the command that is used by user to change their own password. You type passwd and it asks you to enter the old password, then it prompts you to enter the new password.
- Typically I ask this question from my students:
 - If you want to change a password, would you run passwd or sudo passwd?
 - The answer is, well, it depends whose password you want to change:
 - If you want to change the password for the user that you are logged in as, in this case the username is “user” here, then just type passwd. You run this command as “user”, so you get to change the password for the user.
 - If you run sudo passwd, you are changing the password for root, because

you are executing this command as root, and not as “user”. So you get to change the password for the root account.

- This concept applied to everything in a Linux system. If you want to do something as another user, use su or sudo depending on your Linux distribution. But if your user already has permission to do that task there is no need to run the command with “sudo”.
- In fact I would encourage you to use permissions correctly and discourage you from using the root account to do everything. The reason is because root has absolute power to do almost anything on the system. In case you make a mistake, you can end up modifying somebody else’s file.
- Even worse than that, is running programs, games, services, or applications on the system as root. If in case that program has a malicious code, virus, or malware in it, or somehow its process is compromised or crashes, then running it as root with root permissions gives it unlimited power to do all kinds of damage to your system. So please use sudo and su responsibly!