NDG Introduction to Linux I - Chapter 1: Introduction

Objectives

Chapter 1: Introduction

This chapter will cover the following exam objectives:

103.1: Work on the command line v2

Weight: 4

Candidates should be able to interact with shells and commands using the command line. The objective assumes the Bash shell.

Key Knowledge Areas:

> Use single shell commands and one line command sequences to perform basic tasks on the command line.
> Section 1.3.1

Key Terms

bash

Bourne Again SHell - an sh-compatible command language interpreter that executes commands read from the standard input or from a file.

Section 1.3.1

ls

Command that will list information about files. The current directory is listed by default.

Section 1.3.1

Content

1.1 Introduction

The definition of the word Linux depends on the context in which it is used. Technically speaking, Linux is the kernel of the system, which is the central controller of everything that happens on the computer. People that say their computer "runs Linux" are usually referring to the kernel and suite of tools that come with it (called a distribution). If someone says they have Linux experience, it might refer to configuring systems, running web servers, or any number of other services and programs that operate on top of Linux. Over time, Linux administration has evolved to encompass just about every task that a modern business, educational, or government institution might use in their daily operations.

What about UNIX? UNIX was originally an operating system developed at AT&T Bell Labs in the 1970s. It has been modified and forked (that is, people modified it, and those modifications served as the basis for other systems) such that now there are many different variants of UNIX. However, UNIX is now both a trademark and a specification, owned by an industry consortium called the Open Group. Only software that has been certified by the Open Group may call itself UNIX. Despite adopting most if not all of the requirements of the UNIX specification, Linux has not been certified, so Linux really isn't UNIX! It's just… UNIX-like.

Note

Much of the early material in this chapter is very similar to what can be found in NDG Linux Essentials. If you have already taken that course, you can use this as an opportunity to refresh your knowledge or feel free to skip ahead a few pages.

1.1.1 Role of the Kernel

The three main components of an operating system are the kernel, shell, and filesystem. The kernel of the operating system is like an air traffic controller at an airport. The kernel dictates which program gets which pieces of memory, it starts and kills programs, it interprets instructions given to it by the user, and it handles more common and simple tasks such as displaying text on a monitor. When an application needs to write to disk, it must ask the kernel to complete the write operation.

The kernel also handles the switching of applications. A computer will have one or more CPUs and a finite amount of memory. The kernel takes care of unloading tasks and loading new tasks, and can manage multiple tasks across multiple CPUs. When the current task has run a sufficient amount of time, the CPU pauses the task so that another may run. This is called preemptive multitasking. Multitasking means that the computer is doing several tasks at once, and preemptive means that the kernel is deciding when to switch focus between tasks. With the tasks so rapidly switching, it appears that the computer is doing many things at once.

Each application may think it has a large block of memory on the system, but it is the kernel that maintains this illusion; remapping smaller blocks of memory, sharing blocks of memory with other applications, or even swapping out blocks that haven't been used in a while to the disk.

When the computer starts up, it loads a small piece of code called a bootloader. The bootloader's job is to give you a choice (if configured) of options to load one or more versions of Linux, or even other operating systems, and then to load the kernel of the chosen option and get it started. If you are more familiar with operating systems such as Microsoft Windows or Apple's OS X, you probably never see the bootloader, but in the UNIX world, it's usually visible so that you can adjust the way your computer boots.

The bootloader loads the Linux kernel and then transfers control. Linux then continues with running the programs necessary to make the computer useful, such as connecting to the network or starting a web server.

1.1.2 Applications

Like an air traffic controller, the kernel is not useful without something to control. If the kernel is the tower, the applications are the airplanes. Applications make requests to the kernel and receive resources, such as memory, CPU, and disk, in return. The kernel also abstracts the complicated details away from the application. The application doesn't know if the block of a disk is on a solid-state drive from manufacturer A, a spinning metal hard drive from manufacturer B, or even a network file share. Applications just follow the kernel's Application Programming Interface (API) and in return don't have to worry about the implementation details

When we, as users, think of applications, we tend to think of word processors, web browsers, and email clients. The kernel doesn't care if it is running something that's user-facing, a network service that talks to a remote computer, or an internal task. So, from this, we get an abstraction called a process. A process is just one task that is loaded and tracked by the kernel. An application may even need multiple processes to function, so the kernel takes care of running the processes, starting and stopping them as requested, and handing out system resources.

1.1.3 Role of Open Source

Software projects take the form of source code, which is a human-readable set of computer instructions. The source code may be written in any of hundreds of different programming languages. The Linux kernel is mostly written in C, which is a language that shares history with the original UNIX.

Source code is not understood directly by the computer, so it must be compiled into machine instructions by a compiler. The compiler gathers all of the source files and generates something that can be run on the computer, such as the Linux kernel.

Historically, most software has been issued under a closed-source license, meaning that you get the right to use the machine code, but cannot see the source code. Often the license specifically says that you will not attempt to reverse engineer the machine code back to source code to figure out what it does!

Open source takes a source-centric view of software. The open source philosophy is that you have a right to obtain the software and to modify it for your own use. Linux adopted this philosophy to great success.

In 1991, Linux started out as a hobby project by Linus Torvalds. He made the source freely available, allowing others to join in and shape this fledgling operating system. It was not the first system to be developed by a volunteer group, but since it was built from scratch, early adopters could influence the project's direction. People took the source, made changes, and shared them back with the rest of the group, greatly accelerating the pace of development, and ensuring mistakes from other operating systems were not repeated.

The Linux kernel is licensed under the GNU Public License (GPL) which requires you to make changes available. This guarantees that those who use the code will also contribute to the greater good by making those changes available to anyone.

Alongside this, was the GNU project (GNU's, not UNIX). While GNU (pronounced "guh-noo") was building their own operating system, they were far more successful at building the tools that go along with a UNIX operating system, such as the compilers and user interfaces. The source was all freely available, so Linux was able to target their tools and provide a complete system. As such, most of the tools that are part of the Linux system come from these GNU tools.

There are many different variants on open source. However, all agree that you should have access to the source code, but they differ in how you can, or in some cases, must, redistribute changes.

1.1.4 Linux Distributions

Take the Linux kernel and the GNU tools, add some more user-facing applications like an email client, word processors and other programs and you have a full Linux system. People started bundling all this software into a distribution almost as soon as Linux became usable. The distribution takes care of setting up the storage, installing the kernel, and installing the rest of the software. The full-featured distributions also include tools to manage the system and a package manager to help you add and remove software after the installation is complete.

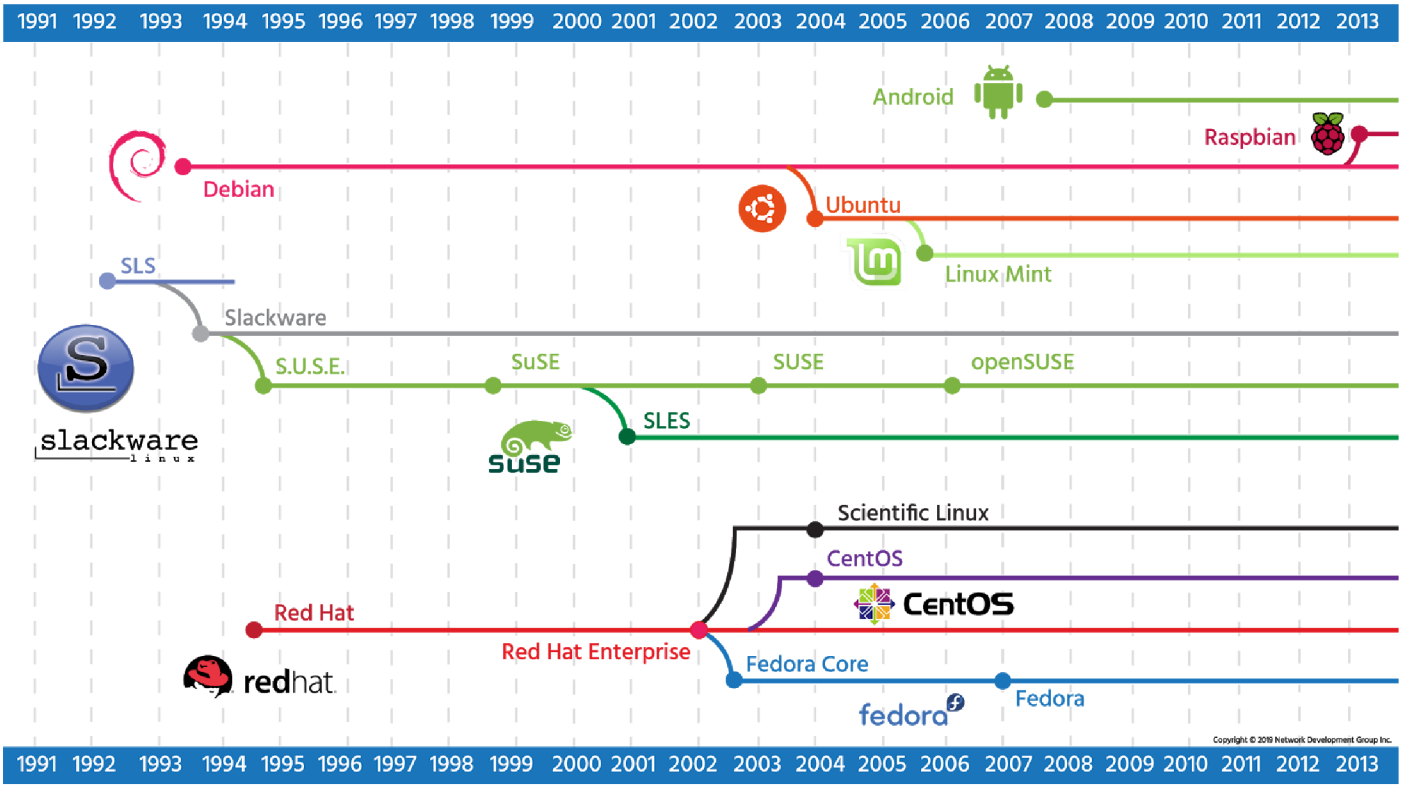
Like UNIX, there are many different flavors of distributions. These days, there are distributions that focus on running servers, desktops, or even industry-specific tools like electronics design or statistical computing. The major players in the market can be traced back to either **Red Hat** or **Debian**. The most visible difference is the software package manager, though you will find other differences on everything from file locations to political philosophies.

Red Hat started out as a simple distribution that introduced the Red Hat Package Manager (RPM) based on the .rpm file format. The developer eventually formed a company around it, which tried to commercialize a Linux desktop for business. Over time, Red Hat started to focus more on the server applications such as web and file serving and released Red Hat Enterprise Linux, which was a paid service on a long release cycle. The release cycle dictates how often software is upgraded. A business may value stability and want long release cycles, while a hobbyist or a startup may want the latest software and opt for a shorter release cycle. To satisfy the latter group, Red Hat sponsors the **Fedora Project** which makes a personal desktop comprising the latest software but still built on the same foundations as the enterprise version.

Because everything in Red Hat Enterprise Linux is open source, a project called **CentOS** came to be, that recompiled all the RHEL packages and gave them away for free. CentOS and others like it (such as **Scientific Linux**) are largely compatible with RHEL and integrate some newer software, but do not offer the paid support that Red Hat does.

Debian is more of a community effort, and as such, also promotes the use of open source software and adherence to standards. Debian came up with its own package management system based on the .deb file format. While Red Hat leaves non-Intel and AMD platform support to derivative projects, Debian supports many of these platforms directly.

Ubuntu is the most popular Debian-derived distribution. It is the creation of **Canonical**, a company that was made to further the growth of Ubuntu and makes money by providing support.



1.2 Hardware Platforms

Linux started out as something that would only run on a computer like Linus': a 386 with a specific hard drive controller. The range of support grew, as support for other hardware was built. Eventually, Linux started supporting other chipsets, including hardware that was made to run competitive operating systems!

The types of hardware grew from the humble Intel chip up to supercomputers. Smaller sized Linux-supported chips were eventually developed to fit in consumer devices (called embedded devices). The support for Linux became ubiquitous such that it is often easier to build hardware to support Linux and then use Linux as a springboard for your custom software than it is to build the custom hardware and software from scratch.

Eventually, cellular phones and tablets adopted Linux. A company, later bought by Google, came up with the Android platform which is a bundle of Linux and the software necessary to run a phone or tablet. This means that the effort to get a phone to market is significantly less. Instead of long development on a new operating system, companies can spend their time innovating on the user-facing software. Android is now one of the market leaders in the phone and tablet space.

Aside from phones and tablets, Linux can be found in many consumer devices. Wireless routers often run Linux because it has a rich set of network features. The TiVo is a consumer digital video recorder built on Linux. Even though these devices have Linux at the core, the end users don't have to know how to use Linux. The custom software interacts with the user and Linux provides the stable platform.

1.3 Shell

An operating system provides at least one shell that allows the user to communicate with the operating system. A shell is sometimes called an interpreter because it takes the commands that a user issues and interprets them into a form that the kernel can then execute on the hardware of the computer. The two most common types of shells are the Graphical User Interface (GUI) and Command Line Interface (CLI).

Microsoft Windows™ typically uses a GUI shell, primarily using the mouse to indicate what you want to accomplish. While using an operating system in this way might be considered easy, there are many advantages to using a CLI, including:

- **Command Repetition**: In a GUI shell, there is no easy way to repeat a previous command. In a CLI there is an easy way to repeat (and also modify) a previous command.
- **Command Flexibility**: The GUI shell provides limited flexibility in the way the command executes. In a CLI, options are specified with commands to provide a more flexible and powerful interface.
- **Resources**: A GUI shell typically uses a relatively large amount of resources (RAM, CPU, etc.). This is because a great deal of processing power and memory is needed to display graphics. By contrast, a CLI uses very little system resources, allowing more of these resources to be available to other programs.
- **Scripting**: In a GUI shell, completing multiple tasks often requires multiple mouse clicks. With a CLI, a script can be created to execute many complex operations by just typing the name of the script. A script is a series of commands placed into a single file. When executed, the script runs all of the commands in the file.
- **Remote Access**: While it is possible to execute commands in a GUI shell remotely, this feature isn't typically set up by default. With a CLI shell, gaining access to a remote machine is easy and typically available by default.
- **Development**: Normally a GUI-based program takes more time for the developers to create when compared to CLI-based programs. As a result, there are usually thousands of CLI programs on a typical Linux OS compared to only a couple hundred programs in a primarily GUI-based OS like Microsoft Windows. More programs mean more power and flexibility.

The Microsoft Windows operating system was designed to primarily use the GUI interface because of its simplicity, although there are several CLI interfaces available, too. For simple commands, there is the Run dialog box, where you can type or browse to the commands that you want to execute. If you want to type multiple commands or if you want to see the output of the command, you can use the Command Prompt, also called the DOS shell. Recently, Microsoft realized how important it is to have a powerful command line environment and introduced Powershell.

Like Windows, Linux also has both a CLI and GUI. Unlike Windows, Linux lets you easily change the GUI shell (also called the desktop environment) that you want to use. The two most common desktop environments for Linux are GNOME and KDE; however, there are many other GUI shells available.

To access the CLI from within the GUI on a Linux operating system, the user can open a software program called a terminal. Linux can also be configured only to run the CLI without the GUI; this is typically done on servers that don't require a GUI, primarily to free up system resources.

1.3.1 Bash Shell

Not only does the Linux operating system provide multiple GUI shells, but also multiple CLI shells are available. Normally, these shells are derived from one of two older UNIX shells: the Bourne Shell and the C Shell. In fact, the Bash shell, a default CLI shell used in modern Linux operating systems, derives its name from the Bourne Shell: **B**ourne **A**gain **SH**ell. In this course, you will focus upon learning how to use the CLI for Linux with the Bash shell, arguably the most popular CLI in Linux.

Users interact with a system by executing commands which are interpreted by the shell and transformed into actions by the kernel. These actions may or may not return information to the command line depending on the command issued and its result. For example, when the `ls` command is typed into the console, it will return the contents of whichever directory the user is currently in.

```
sysadmin@localhost:~$ ls

Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

A command can be followed by options that modify how the command is executed, and arguments, that are typically the files to be operated on:

```
command [options] [arguments
```

Note

Some commands require options and arguments while others, like `ls`, can be used alone.

Commands entered are considered standard input, (stdin) whether they are typed by an operator, entered by a script, or as the result of another command. Text returned to the console can be either standard output (stdout), or standard error (stderr).

This deceptively simple method of communicating with the Linux kernel is the basis for almost every interaction a Linux administrator has with their systems. It can be confusing at first for users who have only experienced GUI interfaces, but ultimately it gives the experienced operator far more power than any graphical interface can.

The Bash shell has numerous built-in commands and features that you will learn including:

- **Aliases**: Give a command a different or shorter name to make working with the shell more efficient.
- **Re-Executing Commands**: To save retyping long command lines.
- **Wildcard Matching**: Uses special characters like ?, *, and [] to select one or more files as a group for processing.
- **Input/Output Redirection**: Uses special characters for redirecting input, < or <<, and output, >.
- **Pipes**: Used to connect one or more simple commands to perform more complex operations.
- **Background Processing**: Enables programs and commands to run in the background while the user continues to interact with the shell to complete other tasks.

The shell that your user account uses by default is set at the time your user account was created. By default, many Linux distributions use Bash for a new user's shell. Typically, a user learns one shell and sticks with that shell; however, after you have learned the basics of Linux, you may want to explore the features of other shells

Consider This

An administrator can use the usermod command to specify a different default shell after the account has been created.

As a user, you can use the chsh command to change your default shell. Most of the time the default shell a system offers will be adequate for basic tasks. Occasionally, an administrator will want to change the shell to have access to more advanced features, or simply because they are more familiar with a different shell and the features it offers. On systems that are near capacity, it may be advisable not to change shells as it could require additional resources and slow processing for all users.

The location where the system stores the default shell for user accounts is the /etc/passwd file.

1.3.2 Accessing the Shell

How you access the command line shell depends on whether your system provides a GUI login or CLI login:

- **GUI-based systems**: If the system is configured to present a GUI, then you will need to find a software application called a terminal. In the GNOME desktop environment, the terminal application can be started by clicking the Applications menu, then the System Tools menu and Terminal icon.
- **CLI-based systems**: Many Linux systems, especially servers, are not configured to provide a GUI by default, so they present a CLI instead. If the system is configured to present a CLI, then the system runs a terminal application automatically after you log in.

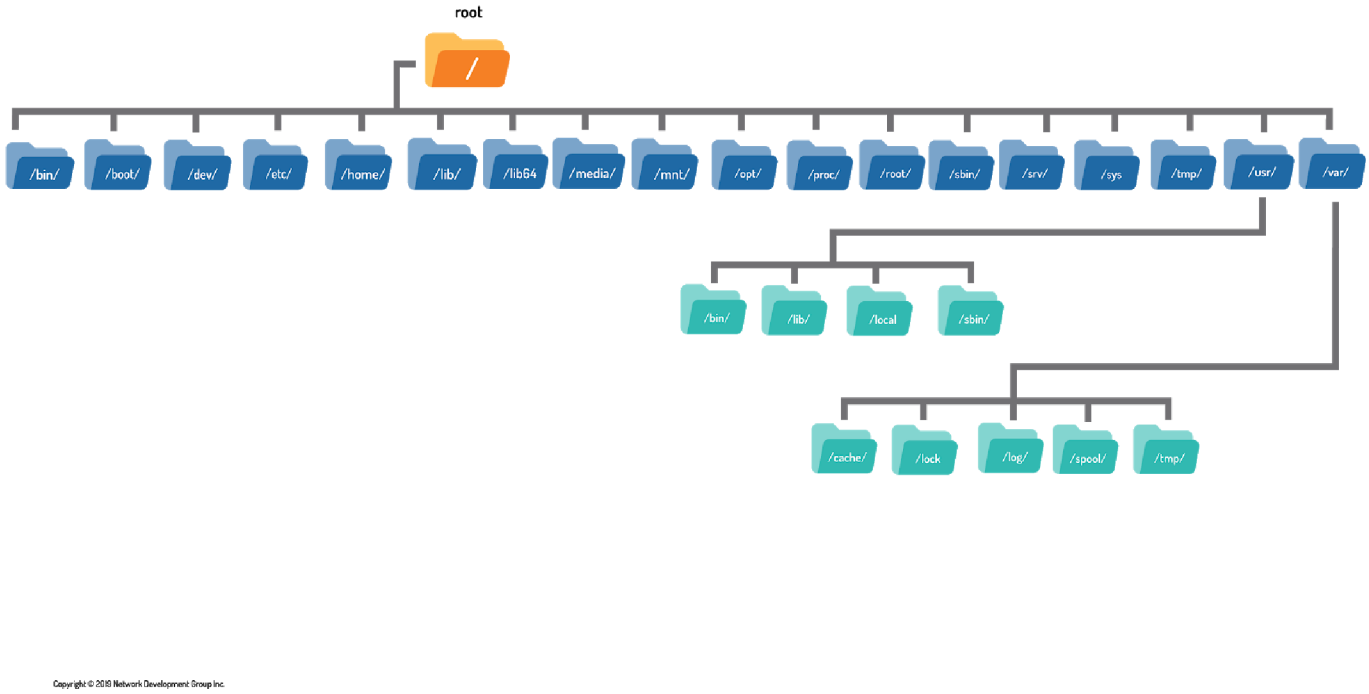
In the early days of computing, terminal devices were large machines that allowed users to provide input through a keyboard and displayed output by printing on paper. Over time, terminals evolved and their size shrank down into something that looked similar to a desktop computer with a video display monitor for output and a keyboard for input.

Ultimately, with the introduction of personal computers, terminals became software emulators of the actual hardware. Whatever you type in the terminal is interpreted by your shell and translated into a form that can then be executed by the kernel of the operating system.

If you are in a remote location, then pseudo-terminal connections can also be made across the network using several techniques. Insecure connections could be made using protocols such as telnet and programs such as rlogin, while secure connections can be established using programs like putty and protocols such as ssh.

1.4 Filesystems

In addition to the kernel and the shell, the other major component of any operating system is the filesystem. To the user, a filesystem is a hierarchy of directories and files with the root / directory at the top of the directory tree. To the operating system, a filesystem is a structure created on a disk partition consisting of tables defining the locations of directories and files.



In this course, you will learn about the different Linux filesystems, filesystem benefits and how to create and manage filesystems using commands. https://lms.netacad.com