launch_code

John Woolbright, Technical Curriculum Developer

June, 15, 2022

Linux Terms

- Operating System
- User space applications
- Kernel
- GNU
- Shell
- Terminal
- File System

Operating System

- An operating system contains two major components:
  - a **kernel**
  - collection of **userspace applications**.

End User

- Any person using a computer is considered an end user.
- An end user uses a computer to achieve some goal which varies depending on their specific needs and wants.

OS Servers the End User

- The OS is responsible for allowing the end user to interface with a computer by handling two responsibilities:
- Managing the physical components of the computer (kernel)
- Providing interfaces to the computer that can be used by the end user (user space applications)
- End users **only** interact with end user applications.

User Space Applications

- User space applications are all of the programs on a computer that the end user interacts with directly.
- Every action you have ever completed on a computer is considered an user space application.
- Creating a word document, Writing/Compiling/Executing code, watching movies, editing photos, browsing the internet, sending emails are all user space applications.
- Even the Graphical User Interface (GUI) you use to select your documents and applications is itself a user space application!

OS Provides the User Interface

These applications have been created with people in mind to achieve a specific goal.

In essence one of the primary responsibilities of an OS is to provide a collection of programs that allow interfacing with a computer easy and intuitive.

Under the Hood

Additionally, under the hood the OS is also responsible for:

- Providing to a monitor
- Managing the mouse location
- Determining mouse and keyboard click events
- Opening programs by allocating memory (RAM)
- Writing to the disk when files are created or modified
- Managing CPU access to all running processes.

All of these under the hood actions are responsibilities of the kernel.

Kernel

Software that manages computer operations including all of the physical hardware of a machine.

Kernel Example

Consider all the actions needed to open a word document. From a user perspective we simply double click the file in our file manager or on our desktop.

The computer has to determine which file was clicked based on the mouse position, find that file from hard disk storage, open the contents of the file in memory (RAM), create and send the display of the file to the monitor.

All of these under the hood tasks would burden the user and are therefore handled by the kernel.

Linux Kernel

The Linux kernel is an open source and actively developed monolithic kernel. It was originally created by Linus Torvalds, but has since received commits from 1000s of contributors.

GitHub repo: https://github.com/torvalds/linux

The Linux Kernel Documentation: https://www.kernel.org/doc/html/latest/

GNU

GNU is a recursive acronym that stands for **G**NU is **N**ot **U**nix.

GNU offers a huge collection of free software, that is commonly bundled with the Linux kernel to create a full fledged operating system.

Many of the tools we will use in this class are created and maintained by GNU.

GNU Homepage: https://www.gnu.org/

GNU Package Listing: https://www.gnu.org/software/software.html

Shell

A **shell** is software that is end user interactive.

It's purpose is to serve as the interface between an end user and the kernel.

There are both text-based shells (Bash shell) and graphical shells (gnome-shell).

Terminal

A terminal is the graphical application that powers a text-based shell.

Before graphical shells (like the Xerox Neptune Directory Editor, Windows 1, Apple Lisa & Macintosh, etc) there were only text-based shells.

Terminal Emulators

Terminals which were the monitors connected to these computers only provided a text-based interface.

In our modern era of computing we mostly use **terminal emulators**. They emulate what the original text-based only terminals felt like, but provide additional features, and are run as an application within a graphical shell.

Terminal Emulator Examples

If you have used **git-bash**, the **bash shell**, **powershell**, or the **Windows cmd line** you have used a terminal emulator.

We will be spending the majority of our time in this class using a terminal emulator displaying the **bash shell**.

File System

The filesystem of a computer is defined and managed by the kernel.

The Linux filesystem contains a **root** directory (“/”) which holds all of the files and directories on the computer.

Root Directory

Many of the subdirectories inside of the **root** directory contain configurations, data, and tools that are used directly by the operating system.

The end user can see, and possibly edit or delete these files which are crucial to running the operating system.

To mitigate the risk of having end users mess around with kernel level data Linux has also designated **userspace**.

Userspace

Userspace is a term that describes the areas of the Linux filesystem that contain end user files and directories.

Userspace is usually found in a couple of different locations and the location varies by Linux distribution. In Ubuntu, which is the Linux distribution we will be using, it is predominantly found in the /usr and /home directories.

Bash Introduction

Bash

- Bash is the GNU Project shell.
- Bash stands for Bourne Again Shell.
- Bash is a text based shell.

Shell

- A **shell** is an interface between a user and the kernel.
- There are two different types of shells: text based and graphical.
- The Ubuntu graphical windows management shell is called GNOME. The Ubuntu text based shell is called **Bash**.

Bash Shell

The Bash shell is extraordinarily powerful. Almost everything that can be accomplished using a Linux distribution can be completed using the Bash shell.

Bash Commands

Being a text based shell, Bash is expecting us to invoke various commands.

These commands may include **arguments** and **options**.

General Bash Command Structure

command --option(s) argument

The structure is consistent across the commands and programs we will be invoking.

Example

```
ls -la /home/student
```

The command “ls” is being invoked with the options -l and -a upon the argument /home/student.

Bash Arguments

A Bash command may require no arguments, one argument only, or many arguments.

No Argument Example

“pwd”

The print working directory command takes no arguments

Default Argument Example

“ls”

The list contents command will automatically use the current working directory if no arguments are provided. This is known as a default argument.

One Argument example

```
ls /home/student
```

The ls command has been provided a specific directory in which we want the contents of listed. Instead of using the current working directory, ls will use the provided directory instead.

Multiple Argument Example

```
rename /home/student/example.file new-name.file
```

Two arguments were provided to the **rename** command. The first argument is the existing file we want to modify. The second argument is the new name we want the file to have.

Bash Options

A Bash command may be presented with any number of options. The options may modify the command, provide additional information to the command, or change the output of the command.

Options are indicated by using double hyphens (--option-name) or single short-hand hyphens (-o).

Bash Double Hyphen Option Example

```
ls --all
```

Print all contents of directory, including any hidden files or directories.

Bash Single Hyphen example

ls -a

-a is the shorthand version of the --all option for the ls command.

Not all commands have both a short and full version. You can learn about the various options by viewing the Reference Manual for a given command.

Bash Combining Options

ls -la

You can oftentimes combine multiple options together. The -a flag will list all files/directories including hidden ones.

The -l option displays the output in a long format providing more information about the files and directories.

The shorthand options can be combined together behind a single hyphen.

Bash Shell Variables

The Bash shell has numerous variables that contain information useful to Bash. They are indicated by the following pattern:

$VARIABLE_NAME

Examples:

- The $BASH shell variable contains the absolute path to the shell this session is using.
- The $HOME shell variable contains the absolute path to the home directory of the user that initiated the Bash shell.
- The $PATH shell variable contains a collection of all the tools currently accessible to this current Bash Shell session.

Bash Command Reference Manuals

To learn more about any Bash command you read its Reference Manual.

You can access the Reference Manual by entering man command-name in your Bash Shell.

This will open up the Manual in your terminal. You can explore the provided information with the directional keys on your keyboard. You can exit the Manual by pressing the q key.

Walkthrough

The walkthrough will take us through some of the basic Bash commands you will be using regularly.

Throughout this class you will continue to learn more about Bash as it is a key tool in most Linux distributions.

launchcode.org