COMP 206 – Introduction to Software Systems
Linux Intro

Lecture 2

Sept 7th, 2018

Plan for Today

- Introduce Operating Systems and Linux concepts
- Start learning some Linux tools
- Introduce GitHub

What is an Operating System?

- The software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.

- Distinguish Operating System from application programs that operate at a higher level:
  - OS has "control" over application programs by deciding when they start/stop, which users can access, which resources can be accessed (files, network, etc)

- An OS is our first ***interface.*** Let's look at Linux as an example of this interface and try to understand what it does.

Common to every OS: be the first job to run!

- The OS must launch every other application program, so the first interface we can consider is:
  - **COMPUTER HARDWARE** to **OPERATING SYSTEM**

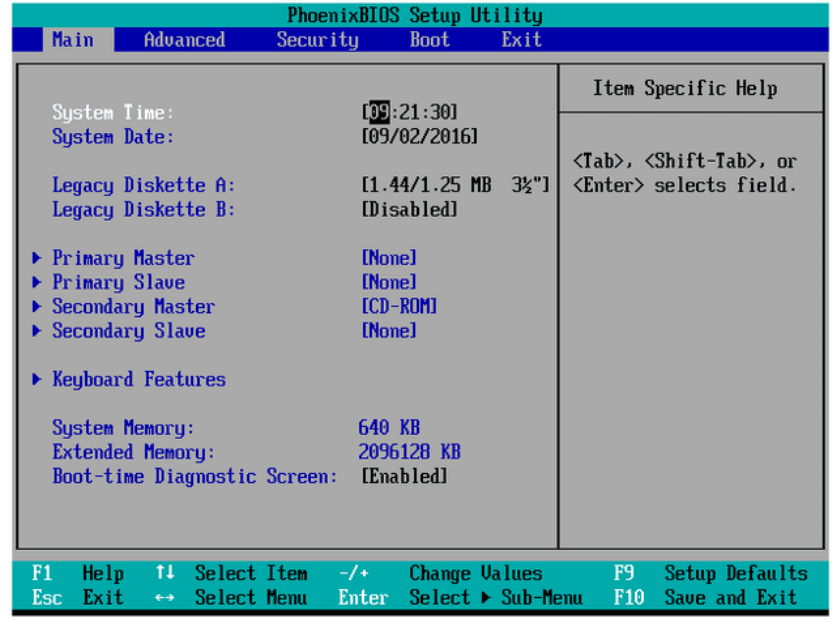
- This requires understanding hardware a bit, although it's not the focus of our courses here in CS. So what's inside your computer?

First job of an OS: be the first job to run!

- The OS must launch every other application program, so the first interface we can consider is:
  - **COMPUTER HARDWARE** to **OPERATING SYSTEM**
- This requires understanding hardware a bit, although it's not the focus of our courses here in CS. So what's inside your computer?
  - Motherboard
  - Processor
  - Short term storage (RAM, memory)
  - Long term storage (disk)
  - BIOS
  - Power supply
  - Graphics Processing Unit (GPU)
  - Many more: data buses (e.g., USB), networking, audio, optical disc dives, cooling, card reader, display, etc

The Beginning: Booting up

- The first device to get power is a BIOS or firmware interface (recently often UEFI). This is a special piece of hardware that manages the lowest level.
  - For the purposes of 206, consider BIOS=UEFI=FI, so we will always say “BIOS”
- The BIOS selects which code to start running next, based on a priority over bootable media (hard drives, USB keys, DVD, the network)



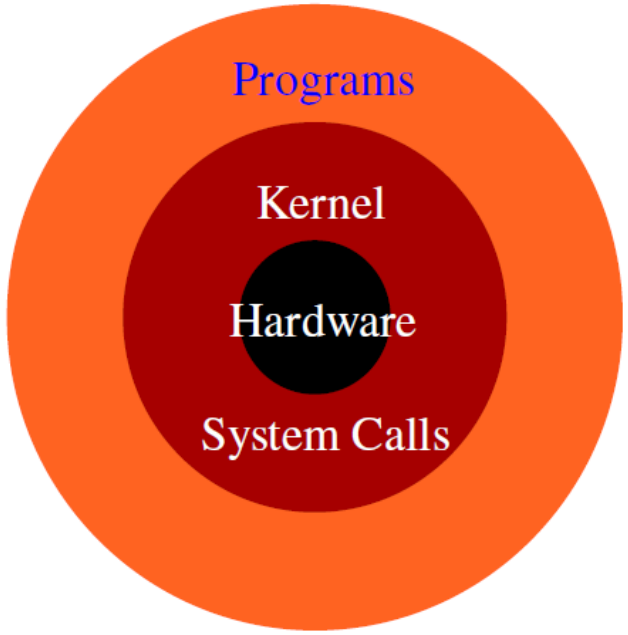
The Middle: A Boot Loader

- A disk is bootable if it has a particular sequence of data at its very beginning, the Master Boot Record (MBR). The BIOS reads each device in order until it finds one that's indicated "bootable"

- A BIOS can load an operating system and begin its execution, but very often, there is a small piece of software known as a boot loader indicated in the MBR.
  - A common boot loader for use with Linux is called **GNU GRand Unified Bootloader (GRUB)**. You might have seen GRUB if you tried dual-booting from the "dangerous" instructions last time.
  - The boot loader can provide some more flexibility and a nice interface to let you choose an OS, which the boot loader then executes

The End (of the beginning)

- The **Kernel** is the name for the part of an OS that runs first and always
- It is responsible for loading and running all other programs, so it gets to create the specification of what's allowed, to enforce security, to kill other jobs, to provide access to hardware etc (if all goes well)



Unix Systems

SunOS/Solaris	Sun Microsystems
Digital Unix (Tru64)	Digital/Compaq
HP-UX	Hewlett Packard
Irix	SGI
UNICOS	Cray
NetBSD, FreeBSD	UC Berkeley / the Net
Linux	Linus Torvalds / the Net

What is Linux?

- (For the purposes of 206) Any operating system based upon the Linux kernel written by Linux Torvalds in 1991
  - The Linux kernel was a re-implementation of Unix designed from the start to be **free and open source**
  - Why is this hard to define? Some people have made new versions starting from this kernel that Linus or others don't agree on. Are they still Linux?
- Linux is still developed by Linus Torvalds, along with a worldwide community
  - The Linus kernel uses https://github.com/torvalds/linux for collaboration
  - With the tools you learn in 206, it is quite possible for you to join the community!

Unix/Linux Philosophy

- Multiuser / Multitasking
- Toolbox approach
- Flexibility / Freedom
- Conciseness
- Everything is a file
- File system has places, processes have life
- Designed by programmers for programmers

IN RAM

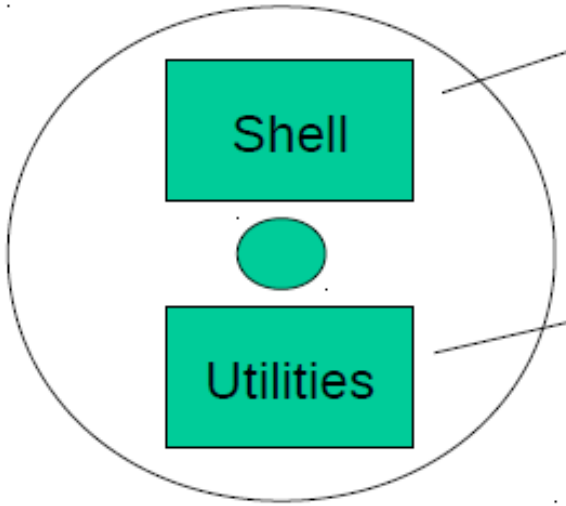
Kernel

- Login
- Task switching, multi-processing
- Basic interface with user and programs
- Drivers, run-time stack, heap

Part in RAM
Part **IS** disk

File System

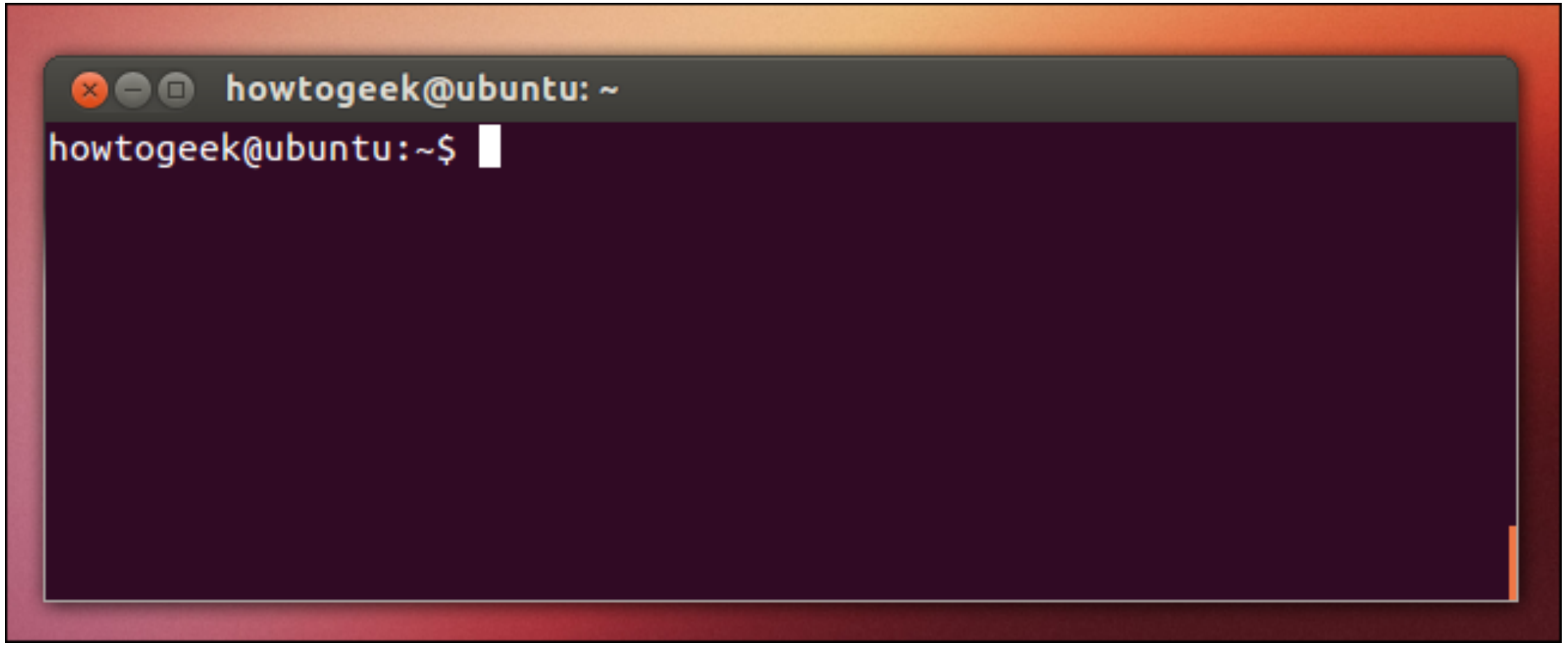
- Defines the way the disk drive is formatted
- The file allocation table (FAT)
- Data structure on disk that makes files real
- Reading and writing to disk and peripherals
- User commands to interact with files



- A more “advanced” user interface
- Has a global memory
- Scriptable to produce complex behaviour

- Additional OS commands and programs
- Third party commands and programs
- Drivers

The Shell (aka terminal)



Popular Shells

`sh` Bourne Shell

`ksh` Korn Shell

`csh` C Shell

`tcsh` Programmable C Shell

`bash` Bourne-Again Shell

Popular Shells

`sh` Bourne Shell

`ksh` Korn Shell

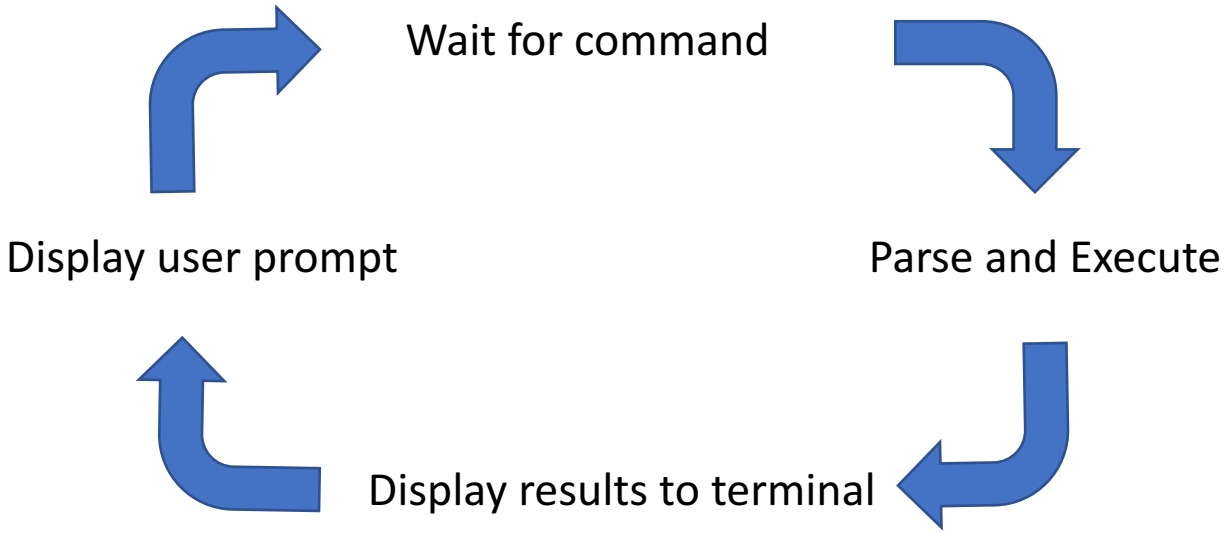
`csh` C Shell

`tcsh` Programmable C Shell

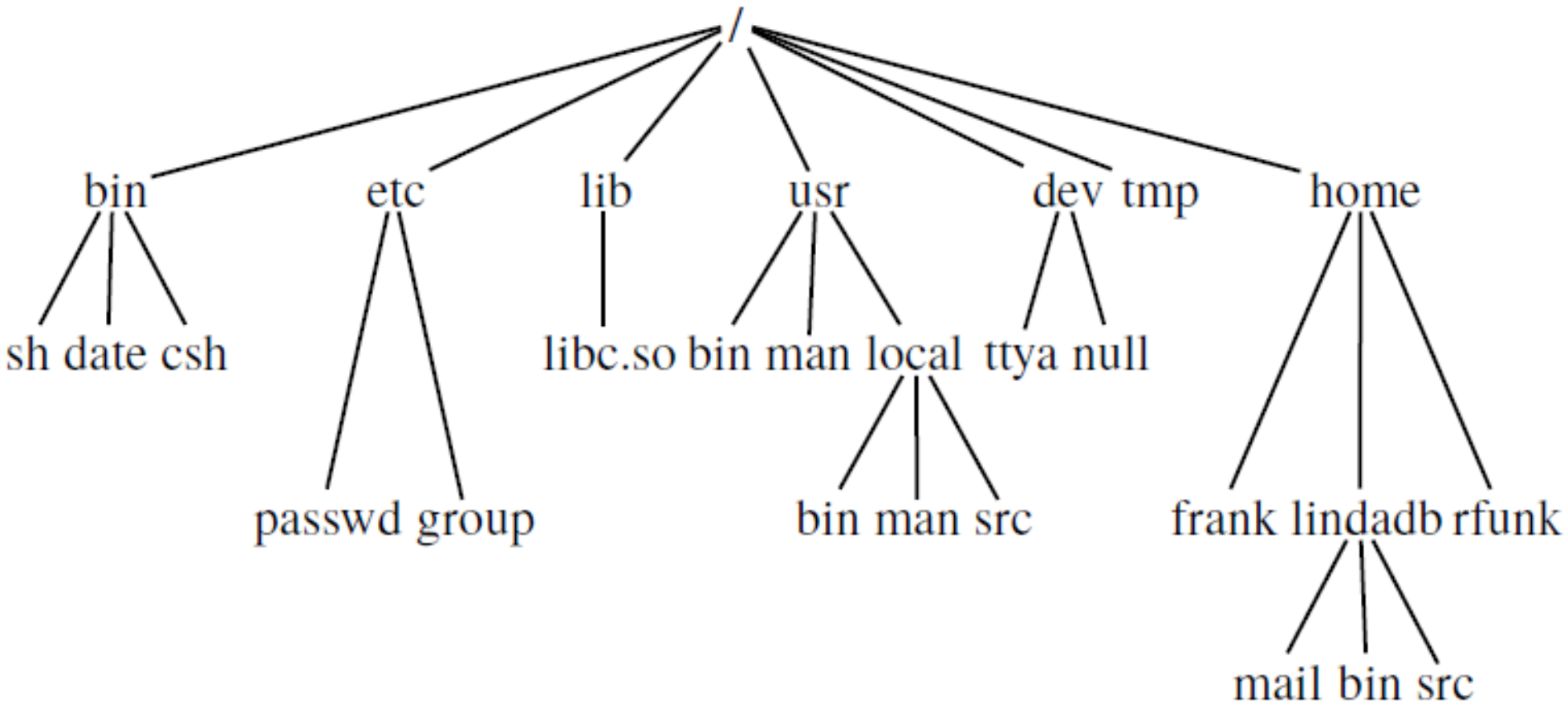
`bash` Bourne-Again Shell

Shell as a user interface

- A shell is a command ***interpreter*** turns text that you type (at the command line) in to actions:
  - **runs a program (aka command), perhaps the `ls` program.**
  - *allows you to edit a command line*.
  - opens the door for complex ways of chaining commands into larger programs, scripting, etc.



The File System: An interface to the disk

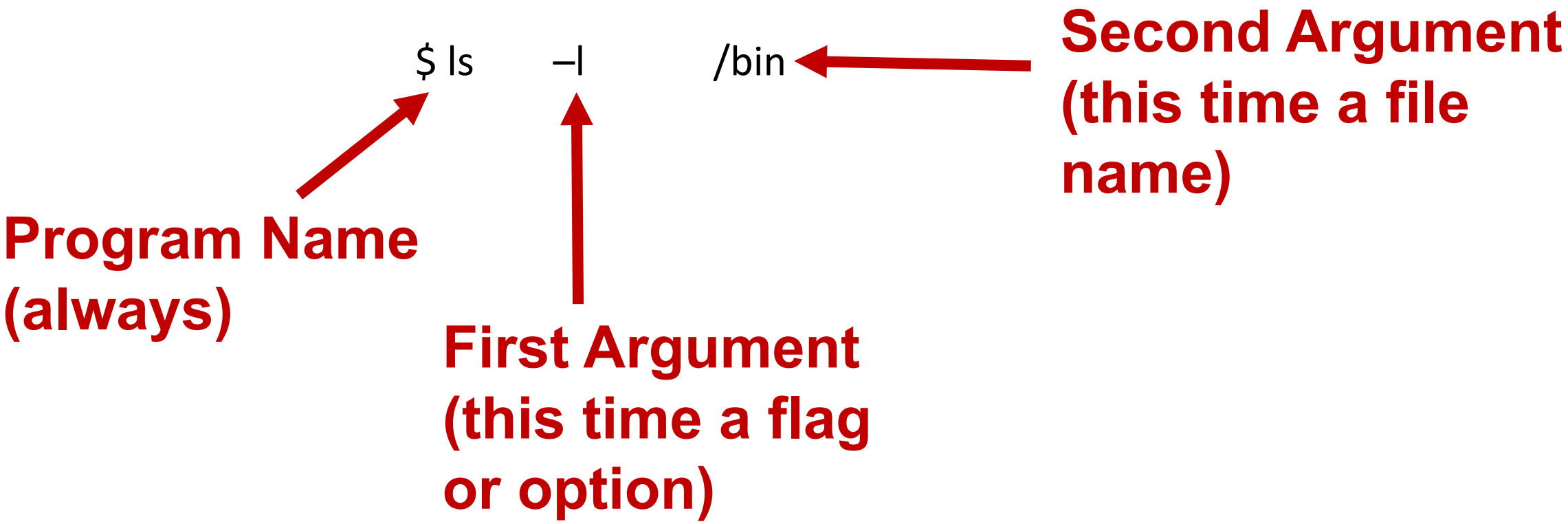


Commands to Know: File related

- ls – list files
- cd – change directory
- pwd – where am I now? (present working directory)
- mv – move files or directories
- find – search for files with given properties
- chmod – change permissions
- cp – copy files or directories
- cat – concatenate input files

Running A Program: Simple or complex

- Type the program's name followed command-line arguments, the shell executes this program, feeding the options you specified:



More Examples

- Example 1, only a program's name. Such as a built-in: $ ls
- Example 2, a program, and one file argument: $ ls /usr
- Example 3, a program and one option flag: $ ls –l
- Example 4, a program with 3 args: $ du –h –-max-depth=1 /dev

More commands to know

- echo: copy input to output (why is this needed?)
- grep: filter input based on a pattern
- tr: translate inputs to outputs
- sort: sort inputs, then output
- ps: display running proceses (once)
- top: display the running processes (continuous) and resource usage
- uname: print system information (which Linux version)
- ssh: remotely connect to another computer

Exercises

- Ensure you can log-in to a Linux system (follow exercises from Lecture 1), be able to open a terminal, know how to type basic commands
- Create a directory, fill it with some files and use the terminal to interact with it:
  - Navigate to the new folder
  - List the files, get their details
  - Move some of the files into a new folder
- Read ahead on the more advanced Shell features which lets us “program” Linux through the Shell