Linux/Unix System Programming

CSCI 2153

David L. Sylvester, Sr., Professor

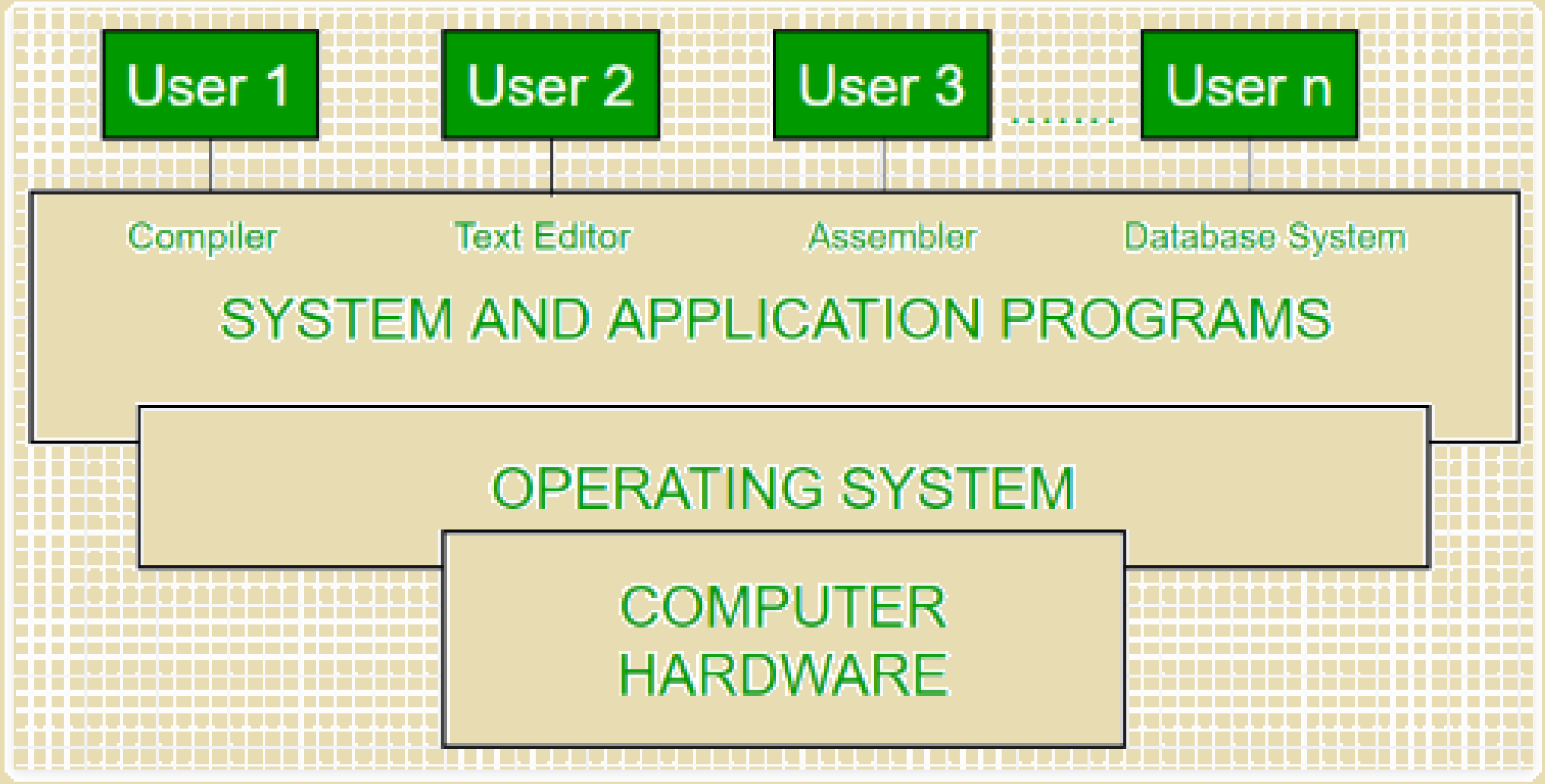
What is an Operating System?

- Software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.
- A program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
- The one program running at all times on the computer (usually called the kernel), with all else being application programs.

Functions of an Operating System?

- **Convenience:** An OS makes a computer more convenient to use.
- **Efficiency:** An OS allows the computer system resources to be used in an efficient manner.
- **Ability to Evolve:** An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions at the same time without interfering with service.

Computer System Conceptual View



Computer System Conceptual View

- Every general-purpose computer consists of:

1. **Hardware**, which consists of memory, CPU, ALU, and I/O devices, peripheral device, and storage device.
2. **System Programs,** which consists of compilers, loaders, editors, OS, etc.
3. **Application Programs**, which consists of business programs, database programs.

Then we have the **User(s)** who interacts with the computer.

I/O System Management

- The module that keeps track of the status of devices is called the I/O traffic controller. Each I/O device has a device handler that resides in a separate process associated with that device.

The I/O subsystem consists of

- A memory Management component that includes buffering caching and spooling.
- A general device driver interface.

Drivers for specific hardware devices.

Assignment (Spooling vs Buffering)

- Write a one-page document elaborating on the differences between spooling and buffering. Also include a cover page that includes: (your name, course title, course number, course day/time, instructor's name and due date) and a work cited page. Assignment is to be uploaded to CANVAS.

Assignment (Device Drivers and Their Purpose)

- Write a one-page document pertaining to device drivers and their purpose within a operating system. Also include a cover page that includes: (your name, course title, course number, course day/time, instructor’s name and due date) and a work cited page. Assignment is to be uploaded to CANVAS.

I/O System Management

- Assembler - program that takes basic computer instructions and converts them into a pattern of bits that the computer's processor can use to perform its basic operations.
- Compiler - program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses.
- Loader - the part of an **operating system** that is responsible for loading programs and libraries.

Evolution of Operating Systems

Generation	Year	Electronic Device Used	Types of OS Device
First	1945-55	Vaccum Tubes	Plug Boards
Second	1955-65	Transistors	Batch Systems
Third	1965-80	Integrated Circuits(CI)	Multiprogramming
Fourth	1980-82	Large Scale Integration	PC
Fifth	1982-	Parallel Processing and AI	Smartphones, Tablets

Types of Operating Systems

- **Batch Operating Systems** - Sequence of jobs in a program on a computer without manual interventions.
- **Time-Sharing operating Systems** - allows many users to share the computer resources.
- **Distributed operating System**- Manages a group of different computers and make appear to be a single computer.
- **Distributed operating System**- Manages a group of different computers and make appear to be a single computer.
- **Network operating system**- computers running in different operating system can participate in common network .
- **Real time operating system** – meant applications to fix the deadlines.

What is UNIX?

- **UNIX** is an operating system developed in the **Bell Laboratories** of **AT&T** and is an example a multi-tasking, multi-user operating system. It provides its users with: program development tools; electronic communications facilities, such as an electronic mail; text editors and text formatters. There are also many development tools available as standard within the UNIX operating system that other operating systems require as add-ons.

The Process Table

Managing processes is one of the kernel's biggest responsibilities. It decides which process actually gets to run on the CPU (or CPUs) at any point in time. The kernel keeps a data structure (in kernel space) to track information about processes: the *process table*. Each process has an entry in this table (entries are called *process control blocks* that include all sorts of information. The processes ID (pid) is one piece of info. You can actually look at all this process table information by examining files (at least they look like files) in the /proc directory.

```
$ ps                        (list running processes)
$ ls /proc                  (list processes)
$ pidof  process-name       (finds the PID number)
$ ls /proc/1                (lists system processes with PID 1)
```

Stopping Processes

Ending a process can be done in several different ways. Often, from a console-based command, sending a CTRL + C keystroke (the default interrupt character) will exit the command. This works when the process is running in the foreground mode.

If a process is running in the background, you should get its Job ID using the **ps** command. After that, you can use the **kill** command to kill the process as follows −

```
$ps -f
UID PID PPID C STIME TTY TIME CMD
amrood 6738 3662 0 10:23:03 pts/6 0:00 first_one
```

The **kill** command terminates the **first_one** process. If a process ignores a regular kill command, you can use **kill -9** followed by the process ID as follows:

```
$  kill -9 6738
```

Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

Most of the commands that you run have the shell as their parent. Check the **ps -f** example where this command listed both the process ID and the parent process ID.

Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

Deamon Processes

Daemons are system-related background processes that often run with the permissions of root and services requests from other processes.
A daemon has no controlling terminal. It cannot open **/dev/tty**. If you do a **"ps -ef"** and look at the **tty** field, all daemons will have a **?** for the **tty**.
To be precise, a daemon is a process that runs in the background, usually waiting for something to happen that it is capable of working with. For example, a printer daemon waiting for print commands.
If you have a program that calls for lengthy processing, then it’s worth to make it a daemon and run it in the background.

The top Command

The **top** command is a very useful tool for quickly showing processes sorted by various criteria.

It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

Here is the simple syntax to run top command and to see the statistics of CPU utilization by different processes:

```
$top
```

Network Communication Utilities

The **ping** command sends an echo request to a host available on the network. Using this command, you can check if your remote host is responding well or not.

The ping command is useful for the following –

- Tracking and isolating hardware and software problems.
- Determining the status of the network and various foreign hosts.
- Testing, measuring, and managing networks.

```
$ ping hostname or ip-address
```

Network Communication Utilities

The ftp Utility

ftp stands for **F**ile **T**ransfer **P**rotocol. This utility helps you upload and download your file from one computer to another computer.

The ftp utility has its own set of Unix-like commands. These commands help you perform tasks such as:

- Connect and login to a remote host.
- Navigate directories.
- List directory contents.
- Put and get files.
- Transfer files as **ascii**, **ebcdic** or **binary**.

Syntax to use the ftp command:

```
$ftp hostname or ip-address
```

Network Communication Utilities

The telnet Utility

There are times when we are required to connect to a remote Unix machine and work on that machine remotely. **Telnet** is a utility that allows a computer user at one site to make a connection, login and then conduct work on a computer at another site.

Once you login using Telnet, you can perform all the activities on your remotely connected machine. The following is an example of Telnet session:

```
C:>telnet amrood.com
Trying...
Connected to amrood.com.
```

Network Communication Utilities

The finger Utility

The **finger** command displays information about users on a given host.

The host can be either local or remote.

Finger may be disabled on other systems for security reasons.

Check all the logged-in users on the local machine:

```
$ finger
Login    Name      Tty    Idle   Login Time    Office
amrood             pts/0         Jun 25 08:03  (62.61.164.115)
```

Processes

A **process** can be simply defined as an instance of a running program. It should be understood that a program is part of the file system that resides on a non-volatile media (such as disk), and a process is an entity that is being executed (with at least some portion, i.e. segment/page) in RAM. One may say that the file system has 'places' and that processes have 'life'."

Since we know that Unix is a multi-user, multi-tasking operating system, we know that multiple processes can be running on a Unix system at the same time. Not only can multiple processes be run at the same time, but on a typical multi-user Unix system, hundreds of processes are running at any given time.

Processes (cont.)

Each time a program is run, a process is created (typically) with the same name as the program itself. So, if the gimp application program is run, a process with the name of gimp is created. Most systems have only one gimp program, but if five users run the gimp program, there will be five processes created named gimp, one for each of the five users. If the processes have the same name, why doesn't the system get the processes confused, you ask?

Processes have the concept of a lifetime associated with them. Processes come to life, or are said to be **born** as a program is loaded and begins execution. Processes remain alive while the program continues execution (processes, however can have different states of life). Processes cease to exist and are said to **die** as the running program terminates (either normally or abnormally).

Processes (cont.)

All processes (except the very first one) have a **parent** which created them. Similarly, when a process is created, it is created as a **child** process, with the process responsible for its creation being its parent. When a process creates a child process, it is said to have **spawned** the child. Every process on a Unix system must have a parent (again, except the very first one), since "orphaned" processes are not (normally) allowed. Also, all processes on a Unix system can be linked to the one initial process. As you will see, processes have a similar hierarchical structure to that of the file system.

Each process will have many attributes associated with it. A processes attributes are stored in a structure memory which is called a Process Control Block, or **PCB**. Each individual process will have a PCB associated with it.

Processes (cont.)

Attributes relevant processes:

- each process will have a unique numeric identifier associated with it, referred to as its process identification number, or **PID**
- each process also has a reference to the PID of its parent, i.e. the parent process id, or PPID
- each process will have priorities associated with it, i.e. the priority of order of execution
- processes have ownership attributes associated with them, both from the user level and from the group level
- processes will have a **state** attributed associated with them, typically assigned and updated by the kernel

Processes (cont.)

Attributes relevant processes:

- each process will have a unique numeric identifier associated with it, referred to as its process identification number, or **PID**
- each process also has a reference to the PID of its parent, i.e. the parent process id, or PPID
- each process will have priorities associated with it, i.e. the priority of order of execution
- processes have ownership attributes associated with them, both from the user level and from the group level
- processes will have a **state** attributed associated with them, typically assigned and updated by the kernel

Starting a UNIX terminal

- Every general-purpose computer consists of:

1. **Hardware**, which consists of memory, CPU, ALU, and I/O devices, peripheral device, and storage device.
2. **System Programs,** which consists of compilers, loaders, editors, OS, etc.
3. **Application Programs**, which consists of business programs, database programs.

Then we have the **User(s)** who interacts with the computer.

The Directory Structure

If you're coming from Windows, the Linux file system structure can seem particularly alien. The C:\ drive and drive letters are gone, replaced by a "/" and cryptic-sounding directories, most of which have three letter names.

The Filesystem Hierarchy Standard (FHS) defines the structure of file systems on Linux and other UNIX-like operating systems.

/ – The Root Directory

Everything on your Linux system is located under the / directory, known as the root directory. You can think of the / directory as being similar to the C:\ directory on Windows – but this isn’t strictly true, as Linux doesn’t have drive letters. While another partition would be located at D:\ on Windows, this other partition would appear in another folder under / on Linux.

/bin – Essential User Binaries

The /bin directory contains the essential user binaries (programs) that must be present when the system is mounted in single-user mode. Applications such as Firefox are stored in /usr/bin, while important system programs and utilities such as the bash shell are located in /bin. The /usr directory may be stored on another partition – placing these files in the /bin directory ensures the system will have these important utilities even if no other file systems are mounted. The /sbin directory is similar – it contains essential system administration binaries.

/boot – Static Boot Files

The /boot directory contains the files needed to boot the system – for example, the GRUB boot loader's files and your Linux kernels are stored here. The boot loader's configuration files aren't located here, though – they're in /etc with the other configuration files.

/cdrom – Mount Point for CD-ROMs

The /cdrom directory isn't part of the FHS standard, but you'll still find it on Ubuntu and other operating systems. It's a temporary location for CD-ROMs inserted in the system. However, the standard location for temporary media is inside the /media directory.

/dev – Device Files

Linux exposes devices as files, and the /dev directory contains a number of special files that represent devices. These are not actual files as we know them, but they appear as files – for example, /dev/sda represents the first SATA drive in the system. If you wanted to partition it, you could start a partition editor and tell it to edit /dev/sda.

/etc – Configuration Files

The /etc directory contains configuration files, which can generally be edited by hand in a text editor. Note that the /etc/ directory contains system-wide configuration files – user-specific configuration files are located in each user's home directory.

/home – Home Folders

The /home directory contains a home folder for each user. For example, if your user name is bob, you have a home folder located at /home/bob. This home folder contains the user's data files and user-specific configuration files. Each user only has write access to their own home folder and must obtain elevated permissions (become the root user) to modify other files on the system.

/lib – Essential Shared Libraries

The /lib directory contains libraries needed by the essential binaries in the /bin and /sbin folder. Libraries needed by the binaries in the /usr/bin folder are located in /usr/lib.

Other folders

/lost+found – Recovered Files

(Any corrupted files found will be placed in the lost+found directory.)

/media – Removable Media

(contains subdirectories where removable media devices inserted into the computer are mounted.)

/mnt – Temporary Mount Points

(where system administrators mounted temporary file systems while using them.)

/opt – Optional Packages

(contains subdirectories for optional software packages.)

/proc – Kernel & Process Files

(contains special files that represent system and process information.)

/run – Application State Files

(store transient files they require like sockets and process IDs.)

Other folders

/sbin – System Administration Binaries

(contains essential binaries that are generally intended to be run by the root user for system administration.)

/selinux – SELinux Virtual File System

(contains special files used by SELinux on Fedora and RedHat Linux distributions.)

/srv – Service Data

(contains “data for services provided by the system.”)

/tmp – Temporary Files

(contains files that are generally deleted whenever the system is restarted and may be deleted by other utilities.)

/usr – User Binaries & Read-Only Data

(where system administrators mounted temporary file systems while using them.)

/var – Variable Data Files

(must be read-only in normal operation. Log files and everything else that would normally be written to /usr during normal operation are written to the /var directory. .)