

# Introduction

OPS102 Week 1 Class 1

---

Tiayyba Riaz/Chris Tyler

September 3, 2024

Seneca Polytechnic

# Outline

Introduction and How This Course Works

What is an Operating System?

Components of an Operating System

User Interfaces

A Bit of History

# Introduction and How This Course Works

---

# Welcome to OPS102

- OPS102 – Operating Systems for Programmers
- An introduction to Linux and Windows
- For software developers
- First offered Fall 2023, replacing ULI101 for SDDS students
- See the official OPS102 course outline

# Course and Mark Breakdown

- Labs – 20%
  - 10 labs throughout the term
  - You must submit all labs
- Quizzes – 25%
  - 7 quizzes throughout the term, top 5 results are used
- Mid-Term Test – 25%
  - Week 7, just before the study break
- Final Test – 30%
  - Final week of the term

# Swapping Weeks 1 and 2

- For OPS102 section(s) NEE in fall 2024
- In calendar week one, I'll present course week two material
- In calendar week two, I'll present course week one material
- Why? Because I think the week two topics will help you in your other courses (e.g. IPC C Programming)

# What is an Operating System?

---

# What is an Operating System?

- Software that does “what it says on the tin”: it **operates** the computer **system**.
- Simple computers don’t need much of an operating system, if any:
  - Early computers that ran only one program at a time
  - Embedded systems, such as the computer (microcontroller) in a basic microwave oven, because it also runs only one simple program, doesn’t manage any sensitive information, and has no communication capability
- But for most contemporary computer systems, the operating system is a crucial component.



# What Does an OS Do?

Four key things that an OS does:

1. Resource Management and Separation
2. Security Enforcement
3. Hardware Abstraction
4. Maintaining the Programming Model

# 1. Resource Management

- A computer has finite resources – a certain amount of CPU capability, memory, storage, network bandwidth, and peripheral devices.
- The operating system is responsible for allocating these resources in a standardized manner to prevent conflicts.
- Examples:
  - Ensuring that each program has its own unique area of memory so they don't crash each other
  - Preventing multiple programs from simultaneously sending output to the same peripheral, such as a printer, garbling the output

## 2. Security Enforcement

- Information must be kept private in some contexts, but shared in others.
- The OS is responsible for enforcing security rules for both privacy and sharing.
- Examples:
  - A social media app should not be able to access data from your banking app!
  - One customer of a cloud server shouldn't be able to access another customer's data
  - However, multiple smartphone apps might be permitted to access a single photo album

### 3. Hardware Abstraction

- There are many hardware details that vary from system to system, such as the brand of GPU, size of the screen, and whether the keyboard is connected via USB or Bluetooth connection.
- The OS hides the details of the hardware so that application software can deal with the hardware in a consistent manner.
- Examples:
  - A program can request input from the keyboard without knowing how the keyboard is connected
  - An application can send data to the network in the same way regardless of whether the connection uses wired ethernet, WiFi, or 5G.

## 4. Maintaining the Programming Model

- Programmers (Software Developers) need a single conceptual framework when creating software.
- The OS controls the hardware (and works with tools such as the compiler) to consistently maintain the illusion of the programmer's model.
- Example:
  - Multiple programs may be written to work in the same region of memory. The operating system will use the hardware virtual memory system to load those programs into different regions of physical memory but create the illusion that each one is in the same portion of memory, so that they can operate simultaneously without conflict.

# Components of an Operating System

---

# Components of an Operating System

The four key components of an operating system are:

1. The Kernel
2. System Libraries
3. Services
4. User Interface(s)
5. Utilities and Applications

# 1. The Kernel

- This is the heart of the Operating System – the main program
- Operates in a special *Privileged Mode*, which enables it to manage resources and security settings for all other software
- Loaded by the computer's *firmware* (built-in software)
- Starts by setting up (initializing) the computer's hardware and resources before starting the services and user interface(s) in non-privileged mode, programming the hardware to enforce the privilege level



## 2. System Libraries

- Many programs perform the same operations: drawing on the screen, accessing the network, playing sound
- Libraries provide a common set of software routines (aka methods, procedures, subroutines, or functions) which programs access to perform these common operations
- This eliminates the need for each program to contain duplicate code for these common operations
- Although other libraries may be installed on the computer, the *system libraries* provided as part of the OS provide the most broadly-used routines, required by nearly every program

## 3. Services

- These are programs that run continuously in the background, providing services such as WiFi authentication, print management, and file sharing.
- Unlike the kernel, these programs do not operate with full system privilege, and are subject to the same types of resource and security management as regular programs. They also use the system libraries.

## 4. The User Interface(s)

- The user interface is the software enables the user to interact with the system.
- Most operating systems provide at least two types of user interfaces:
- A text-based user interface that enables the user to enter commands, view the output from those commands, and interact with full-screen text interfaces such as file editors and file managers. These may be referred to as *text user interfaces* (TUI) or *command-line interfaces* (CLI).
- A *graphical user interface* (GUI) that typically enables the user to interact with multiple application windows, typically using a keyboard and a pointing device such as a mouse/trackpad/touchscreen

## 5. Utilities and Applications

- Most operating systems provide a set of tools to enable users to perform setup, configuration, and maintenance tasks.
- Utilities may use the GUI or CLI.
- Most operating systems also provide a set of basic starter applications, such as a text editor, clock, a few games, and sometimes a web browser.

# User Interfaces

---

# Command Line Interfaces: Windows, Linux

```
chris@DESKTOP-GC3UWK C:\Users\chris>dir
Volume In drive C has no label.
Volume Serial Number is FC38-8615

Directory of C:\Users\chris

2023-09-04 10:51 AM <DIR>          .
2023-09-04 10:51 AM <DIR>          ..
2023-06-12 05:59 PM <DIR>          .ssh
2023-05-30 01:54 PM <DIR>          3D Objects
2023-05-30 01:54 PM <DIR>          Contacts
2023-05-30 01:54 PM <DIR>          Desktop
2023-06-12 03:03 AM <DIR>          Documents
2023-09-04 01:15 PM <DIR>          Downloads
2023-05-30 01:54 PM <DIR>          Favorites
2023-05-30 01:54 PM <DIR>          Links
2023-05-30 01:54 PM <DIR>          Music
2023-05-30 01:55 PM <DIR>          OneDrive
2023-05-30 01:55 PM <DIR>          Pictures
2023-09-04 10:52 AM <DIR>          Play
2023-06-14 01:33 PM <DIR>          play001
2023-06-12 00:32 AM <DIR>          Program Files
2023-05-30 01:54 PM <DIR>          Saved Games
2023-06-12 03:03 AM <DIR>          Scripts
2023-05-30 01:55 PM <DIR>          Searches
2023-06-02 12:51 AM <DIR>          Videos
           0 File(s)            0 bytes
           20 Dir(s) 107,972,431,872 bytes free

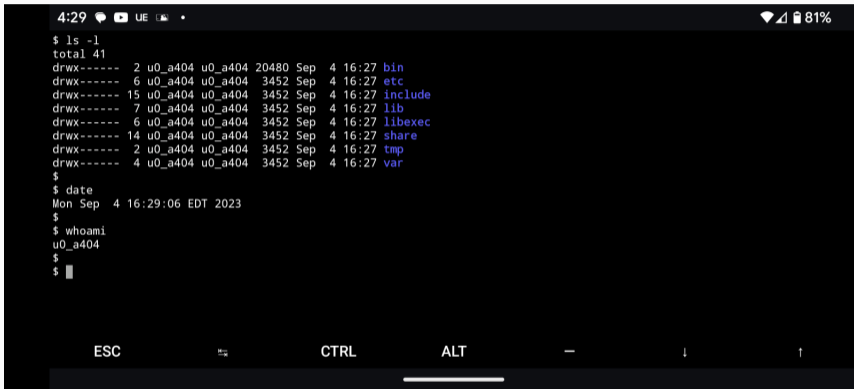
chris@DESKTOP-GC3UWK C:\Users\chris>
chris@DESKTOP-GC3UWK C:\Users\chris>
chris@DESKTOP-GC3UWK C:\Users\chris>date /t
2023-09-04

chris@DESKTOP-GC3UWK C:\Users\chris>
chris@DESKTOP-GC3UWK C:\Users\chris>
chris@DESKTOP-GC3UWK C:\Users\chris>whoami
desktop-gc3uwk\chris

chris@DESKTOP-GC3UWK C:\Users\chris>
```

```
[chris@fedora ~]$ ls -l
total 0
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Desktop
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Documents
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Downloads
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Music
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Pictures
drwxr-xr-x. 1 chris chris 0 Sep  4 13:52 Play
drwxr-xr-x. 1 chris chris 0 Sep  4 13:52 play001
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Public
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Templates
drwxr-xr-x. 1 chris chris 0 Jun 15 09:00 Videos
[chris@fedora ~]$
[chris@fedora ~]$
[chris@fedora ~]$ date
Mon 04 Sep 2023 01:58:58 PM EDT
[chris@fedora ~]$
[chris@fedora ~]$ whoami
chris
[chris@fedora ~]$
[chris@fedora ~]$
[chris@fedora ~]$
```

# Command Line Interfaces: Android

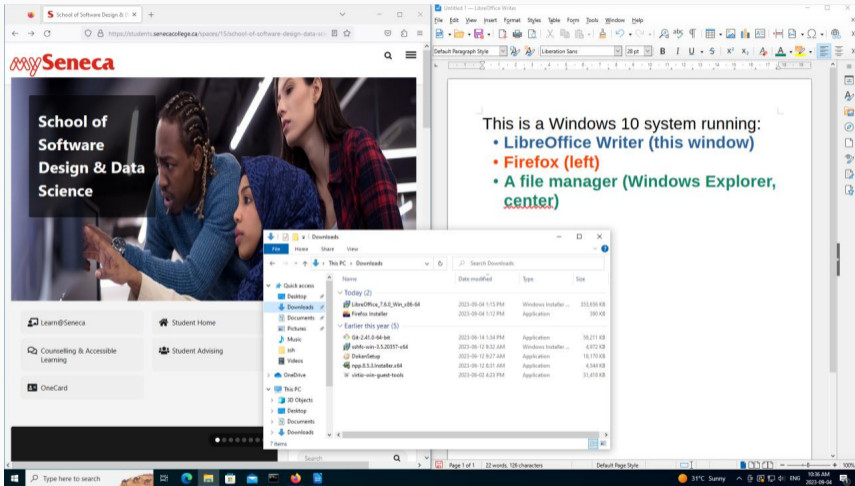


The screenshot shows an Android terminal window with a black background and white text. At the top, the status bar displays the time 4:29, signal strength, Wi-Fi, and battery level at 81%. The terminal content is as follows:

```
$ ls -l
total 41
drwx----- 2 u0_a404 u0_a404 20480 Sep  4 16:27 bin
drwx----- 6 u0_a404 u0_a404  3452 Sep  4 16:27 etc
drwx----- 15 u0_a404 u0_a404  3452 Sep  4 16:27 include
drwx----- 7 u0_a404 u0_a404  3452 Sep  4 16:27 lib
drwx----- 6 u0_a404 u0_a404  3452 Sep  4 16:27 libexec
drwx----- 14 u0_a404 u0_a404  3452 Sep  4 16:27 share
drwx----- 2 u0_a404 u0_a404  3452 Sep  4 16:27 tmp
drwx----- 4 u0_a404 u0_a404  3452 Sep  4 16:27 var
$
$ date
Mon Sep  4 16:29:06 EDT 2023
$
$ whoami
u0_a404
$
$ █
```

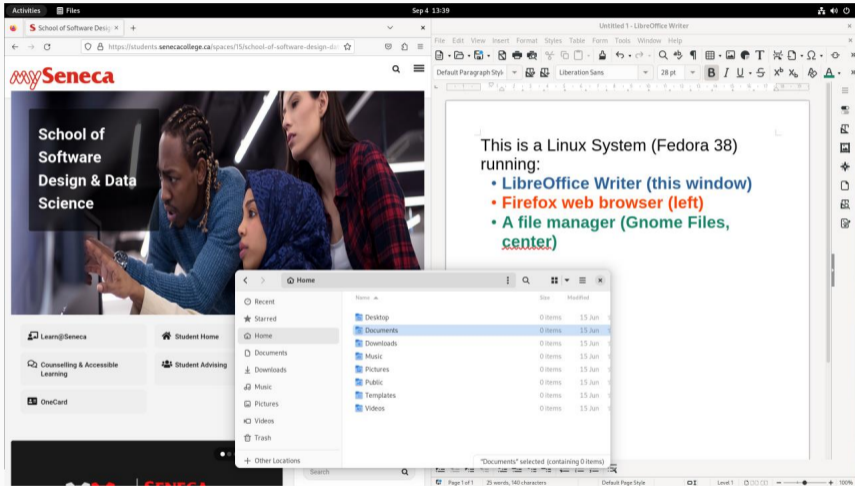
At the bottom of the terminal, there is a navigation bar with several icons: ESC, a keyboard icon, CTRL, ALT, a minus sign, a downward arrow, and an upward arrow.

# Graphical User Interfaces: Windows





# Graphical User Interfaces: Linux



## GUI vs CLI: Strength and Weaknesses

- GUIs are well-suited to graphical tasks, such as editing images and documents. However, they may require excessive repetitive actions in some situations.
- CLIs are well-suited to automation, such as mass-conversion of thousands of images. However, they're not well suited to occasional tasks.
- GUIs require far more data than CLIs. A typical CLI display contains about 2 kilobytes of data; a typical HD GUI display contains about 6 megabytes (6000 kilobytes) of data. Therefore, CLIs are often used over remote connections.

- A text user interface employs the same display technology as a CLI, but presents a full- screen interface rather than the scrolling command-and-response output used in a CLI.
- Before Microsoft Windows and the Apple Macintosh existed, this was the dominant form of interface on personal computers.
- Very common (still) in traditional business applications (e.g. your bank).



## A Bit of History

---

## A Bit of History

- In the very early days of computing, nothing was standardized – in fact, standardization was impossible, because many of the computers were one-of-a-kind!
- Many concepts in modern operating systems trace their roots to Multics, an OS developed at MIT starting around 1965.
- Bell Labs was involved with the Multics project, but withdrew early in its development. Two Bell employees who participated in Multics, Dennis Ritchie and Ken Thompson, went on to develop a smaller operating system using some of the same concepts, which they named *Unix*.

- Unix was originally written for just one type of computer (as were most early operating systems) but was soon altered to be *portable* (useful on multiple, different types of computers)
- Unix was popular with the many different hardware vendors that existed at that time, because it eliminated the need for them to each develop their own operating system. Unix was also very popular in academia because the source code was widely available for study.
- Microsoft licensed the Unix operating system from Bell Labs and created a derivative called *Xenix* for small computer systems.

- Another popular operating system was CP/M (originally the *Control Program/Monitor*, later *Control Program for Microcomputers*)
- When IBM entered the microcomputer market in 1981, it contracted Microsoft to provide an operating system. Microsoft in turn licensed and later purchased 86-DOS from another company, renaming it *PC-DOS* (the IBM version) and *MS-DOS* (the version directly available from Microsoft). The design and internal structure of these operating systems was largely based on CP/M.
- Unlike Unix, DOS was a single-tasking system and did not have a hierarchical filesystem (nested folders or directories to organize large numbers of files).



## MS-DOS takes on Xenix Features

- As the IBM PC models – and compatible systems from other manufacturers – grew in their capabilities, it became necessary to extend DOS to take advantage of these new features.
- Microsoft introduced features from Xenix (their version of Unix) into DOS starting with version 2.0 in 1983.
- Thus, DOS had a unique combination of Unix-like features, with some differences originating from its original CP/M-like heritage. For example, DOS couldn't use the forward-slash in filenames, because that character was already being used for another purpose, so DOS pathnames used the backward-slash character (which was basically unknown before that time!).

# End of Class One!

- Lots of new ideas and concepts to digest
- Lots more to come!