CPSC 5042: Week 1

Ritika Kanade for Seattle University

This Class

- CPSC 5042: Computer Systems Principles II
- Computer system infrastructure, implementation, and design.
- Topics include concurrency, synchronization, reliability, input/output, networking, and security. Development of multithreaded concurrent programs and client-server networking programs.

Week 1: What is an Operating System

- Many of you have received an Introduction in prior class, but because Operating System is fairly complex, the feeling is that a re-introduction is in order.

- We look at Virtualization, Concurrency, and Persistence.

- We will take a high-level look at the various Managers that make up an OS.

Week 2: Main Memory

- We will step back in time and explore early Memory Management systems
- Concepts of First-fit, relocatable dynamic partitions, fragmentation will be explored

Week 3: Virtual Memory

- Exploring modern Memory Management schemes, and exploring concept of Virtual Memory

Week 4: Process Synchronization

- Multi-Processing environments will be analyzed.
- Deadlocks, Starvation, and Race Conditions will be explored
- Project Milestone 1 concepts will be discussed
- Midterm Review

Week 5: Midterm Week

- Midterm in class, closed book.

Week 6: Concurrency I

- How Multi-Core technology works
- Analyze different multi-processing systems
- Critical Section and Semaphores
- Threads are introduced

Week 7: Concurrency II

- A continuation of prior week, as Concurrency is fairly large topic
- We will be studying Client-Server Project

Week 8: Network Organization I

- DNS, Bus Topology, Ring and Star Topology
- TCPIP
- Introduction to Network Programming
- History of Networks/Internet

Week 9: Network Organization II

- How a Network Operating System works
- Comparison to Distributed Operating System
- If time permits we might do some comparisons of popular Operating Systems (Windows, MacOS, Linux, Mainframe)

Week 10: Revision

- Project Presentation Night
- Revision for the Finals
    - Practice finals!

Week 11: Finals

- To be held in class, closed book.

Introduction to OS

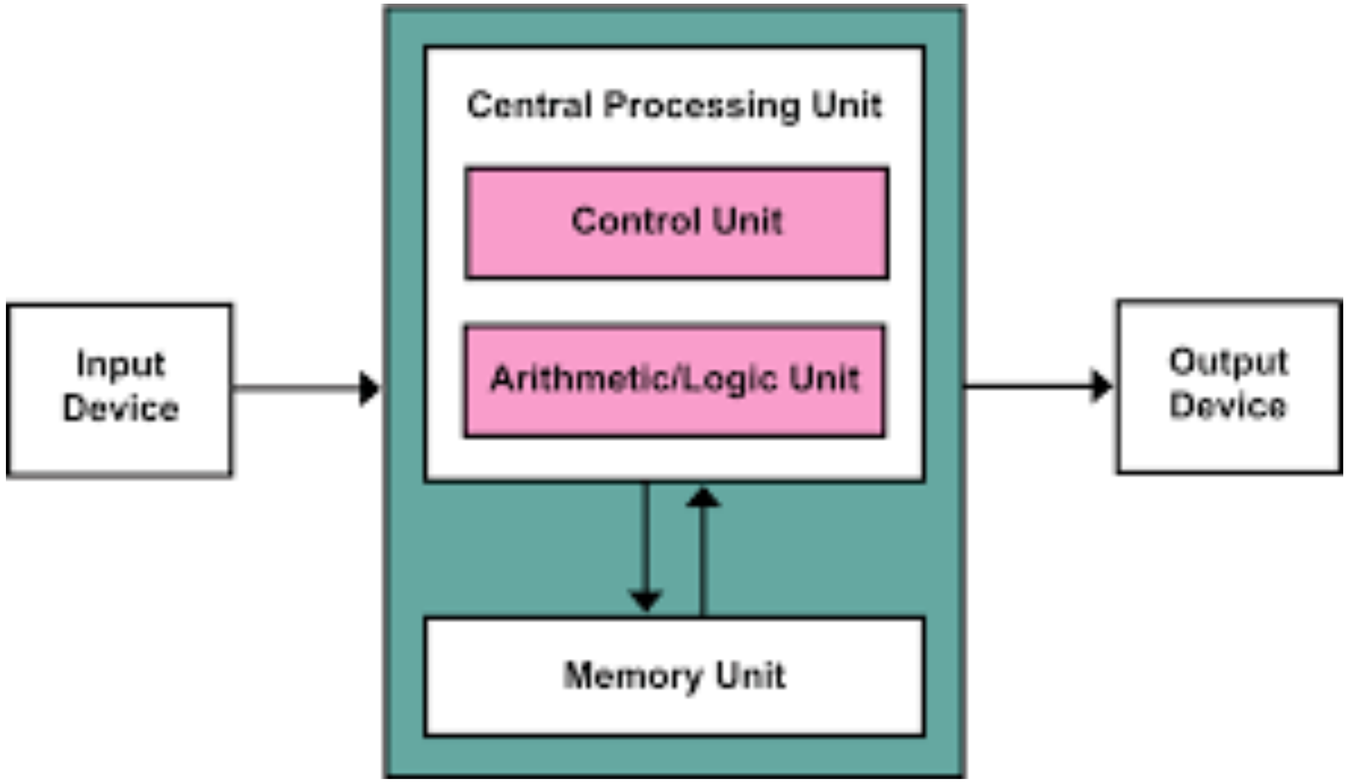
What Happens when a program runs?

Machine code	Assembly code	Description
001 1 000010	LOAD #2	Load the value 2 into the Accumulator
010 0 001101	STORE 13	Store the value of the Accumulator in memory location 13
001 1 000101	LOAD #5	Load the value 5 into the Accumulator
010 0 001110	STORE 14	Store the value of the Accumulator in memory location 14
001 0 001101	LOAD 13	Load the value of memory location 13 into the Accumulator
011 0 001110	ADD 14	Add the value of memory location 14 to the Accumulator
010 0 001111	STORE 15	Store the value of the Accumulator in memory location 15
111 0 000000	HALT	Stop execution

Is this a computer? Does it have an OS?

- No (Although it is debatable)
- Operating systems have several purposes, such as interfacing with the hardware and, managing multiple concurrent or sequential applications and providing protection between different users. A calculator has only user and only one application so there's nothing to manage there.

Von Neumann Model



What is an OS?

- It is the #1 important piece of software on computer.
- What many people often get confused with is that it is ALL Software. There is no hardware involved
- It manages memory, files, processes, …
- It manages people, devices, cpu time
- It says who can use the system and how it is used.
- What are some of the other management functions it has?

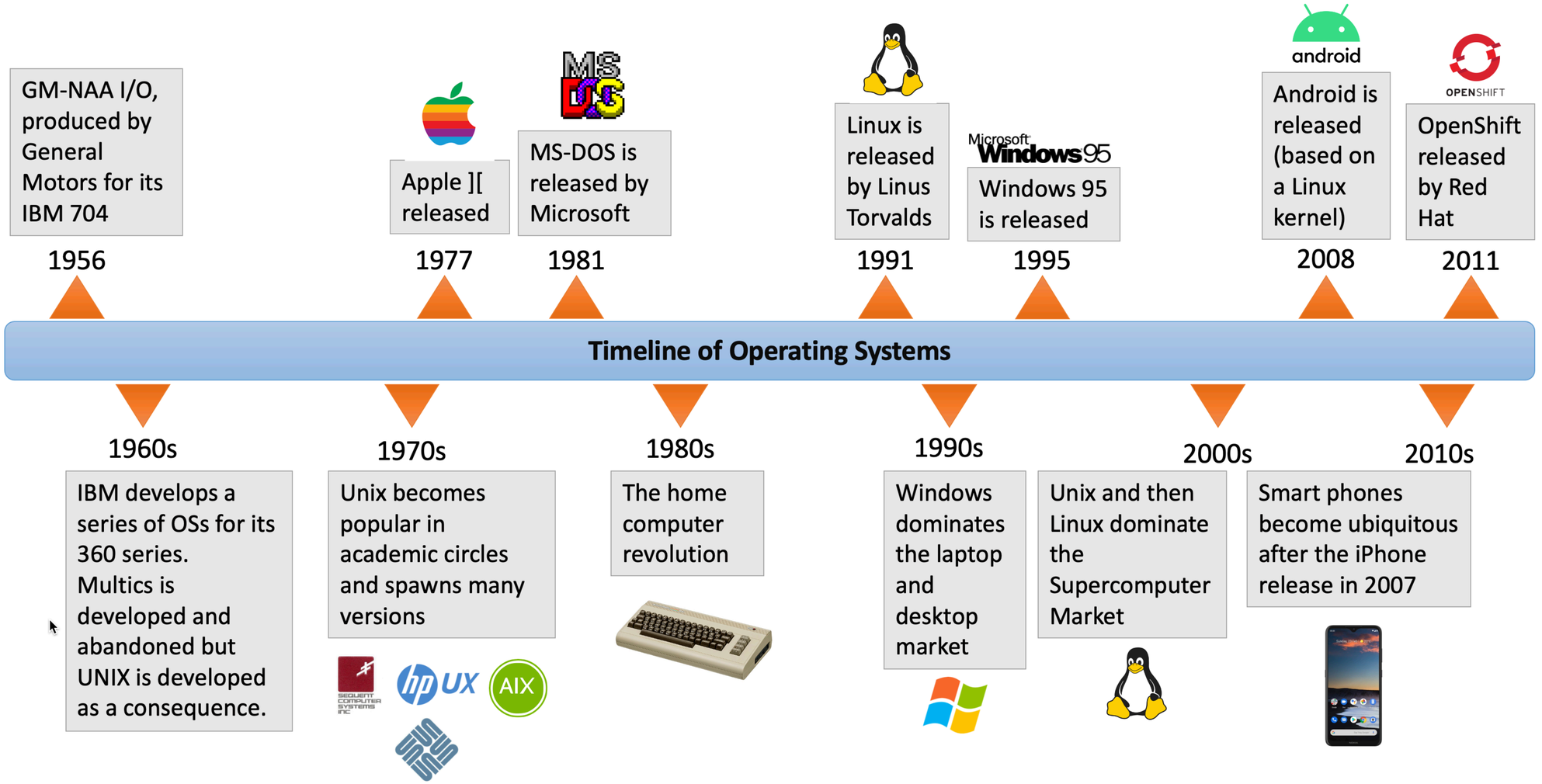
Who uses an OS?

- Programs, as they need a place to be hosted
- Anybody that has a terminal in which they interact with a computer
- Mainframe users/app’s
- Desktop users/app’s
- Phone users/apps?

Who writes an OS?

- Developers at Microsoft, Apple, Sun, and IBM could be part of various teams responsible.
- While many won’t necessarily write System Software, all of us at some time will likely use Operating System commands or functions for the various projects we are involved with.

Evolution


GM-NAA I/O, produced by General Motors for its IBM 704
1956
Apple][released
1977
MS-DOS is released by Microsoft
1981
Linux is released by Linus Torvalds
1991
Windows 95 is released
1995
Android is released (based on a Linux kernel)
2008
OpenShift released by Red Hat
2011
Timeline of Operating Systems
1960s
IBM develops a series of OSs for its 360 series. Multics is developed and abandoned but UNIX is developed as a consequence.
1970s
Unix becomes popular in academic circles and spawns many versions
1980s
The home computer revolution
1990s
Windows dominates the laptop and desktop market
2000s
Unix and then Linux dominate the Supercomputer Market
2010s
Smart phones become ubiquitous after the iPhone release in 2007

Evolutions in OS

- Run many programs at once
- Share Memory
- Interact with Other devices
- Hides the complexity behind it’s system software, enabling programs to be productive

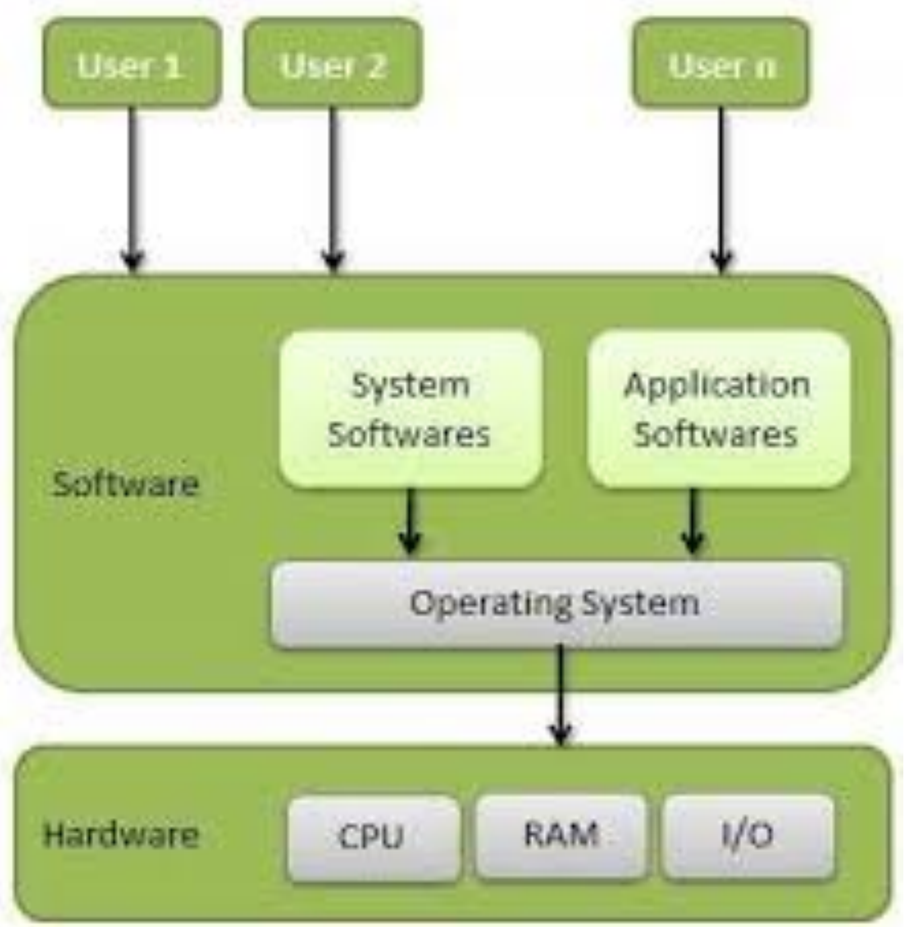
Why do we use an OS?

- Because machines/computers have evolved to the point where we can’t survive without them.
- I always love hearing stories about people that “disconnect” from society, and wonder how they function
    - Maybe Richard Proenneke did. He was a survivalist in Alaska for many years.

Where are OSs located?

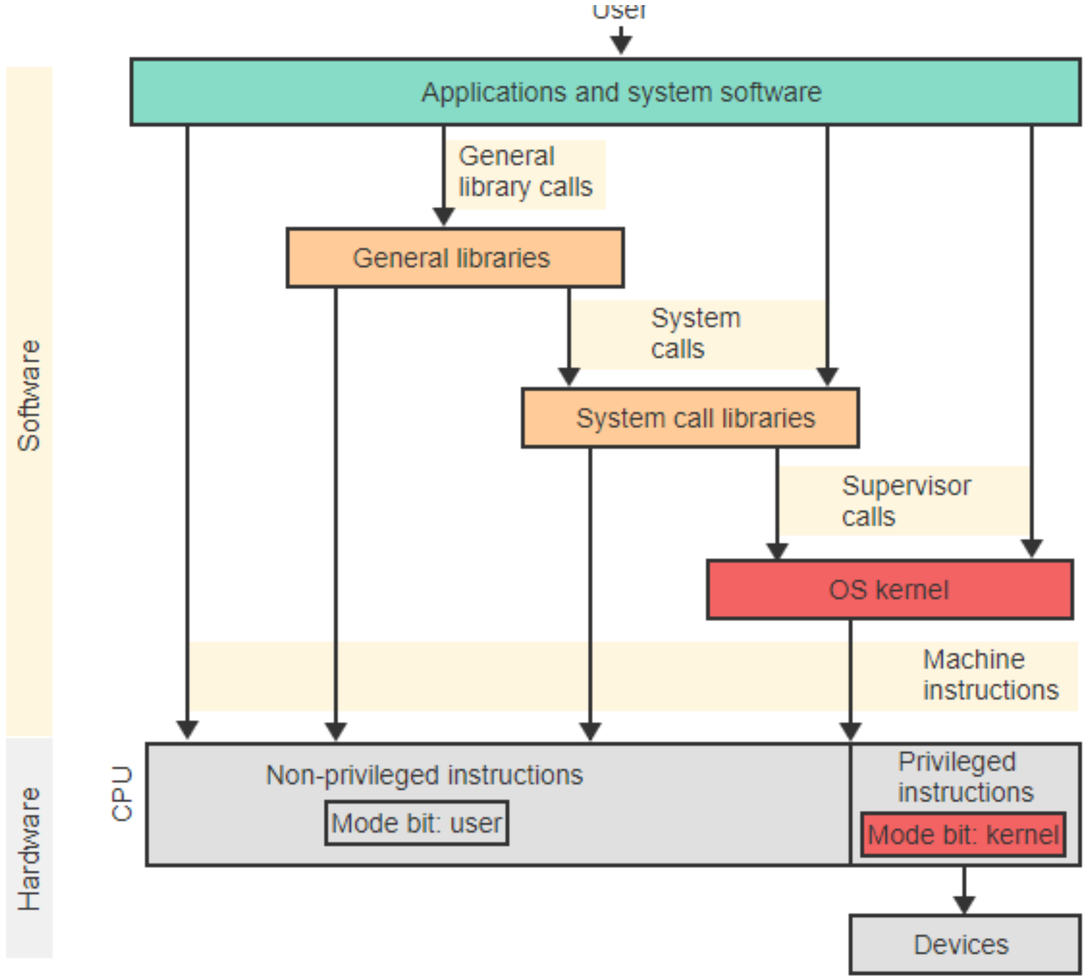
- Mobile (Phones)
- Embedded Operating System (ATM, POS, IOT)
- Network Operating System (Specialized OS handling communication on a LAN)
- Real Time (RTOS). Works with many devices, and where latency is avoided at all costs.

Formal Definition



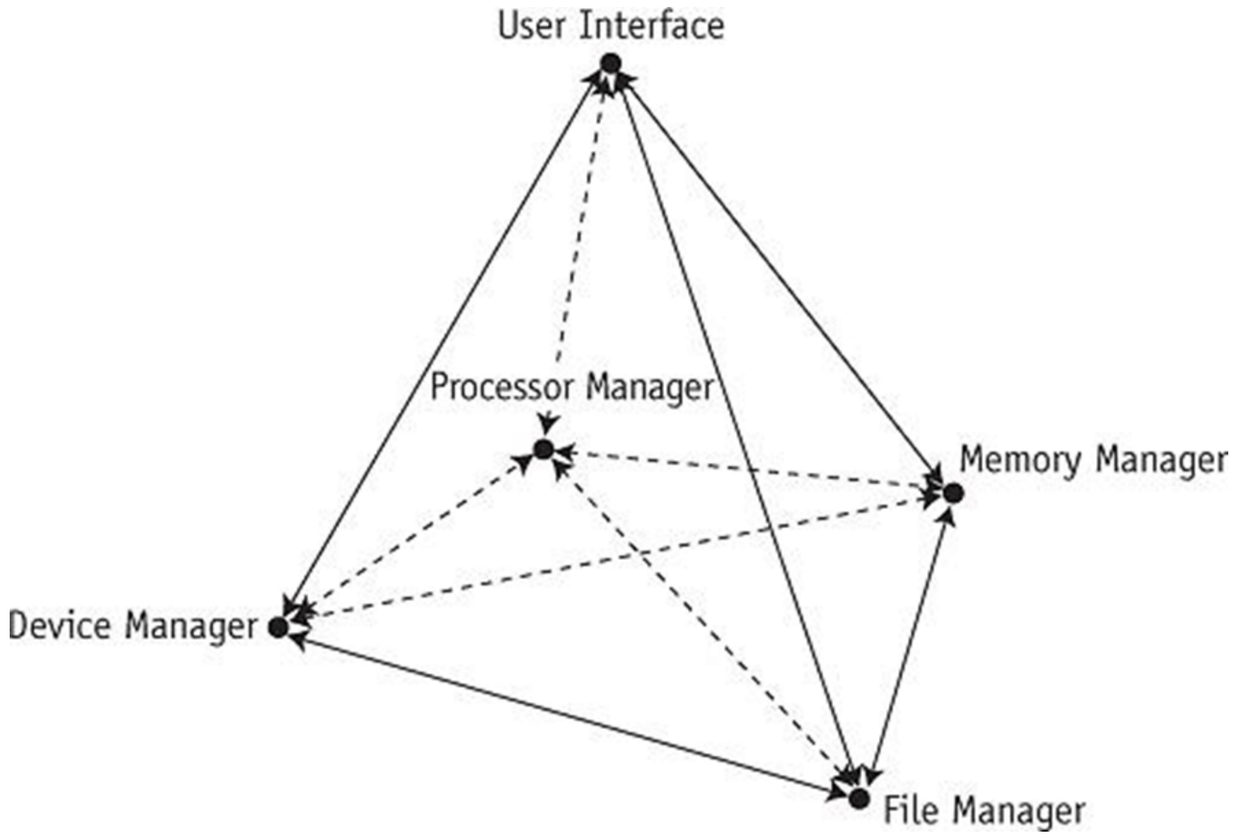
- The software that supports a computer's basic functions, such as scheduling tasks, executing applications, and controlling peripherals.

OS Structure

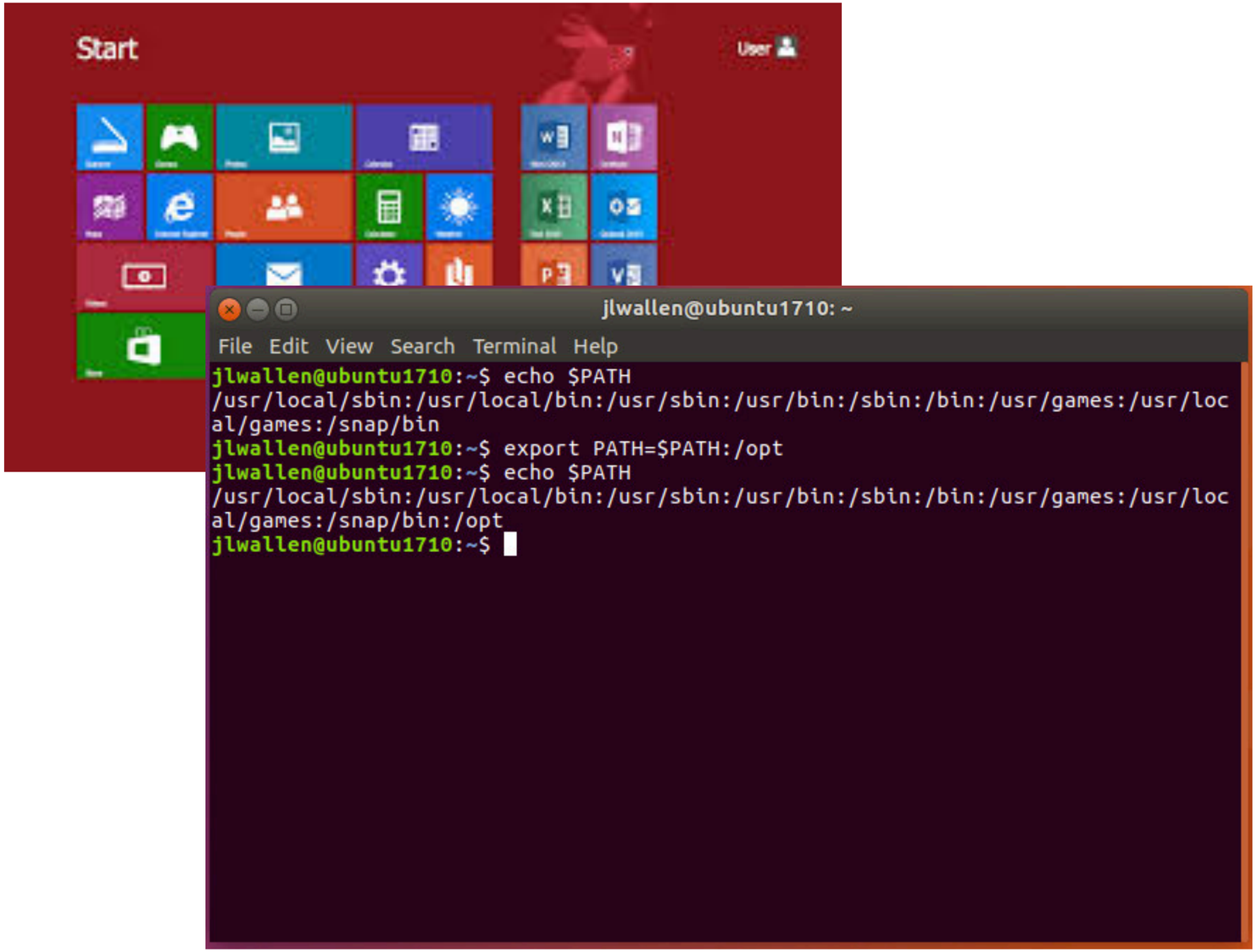


- Abstraction is the important concept, as these various levels of abstractions build on each other
- You as developer may never directly use any layer lower than the Application Layer, but you will be “using” these various layers as you run the program.

Intercommunication



User Interface

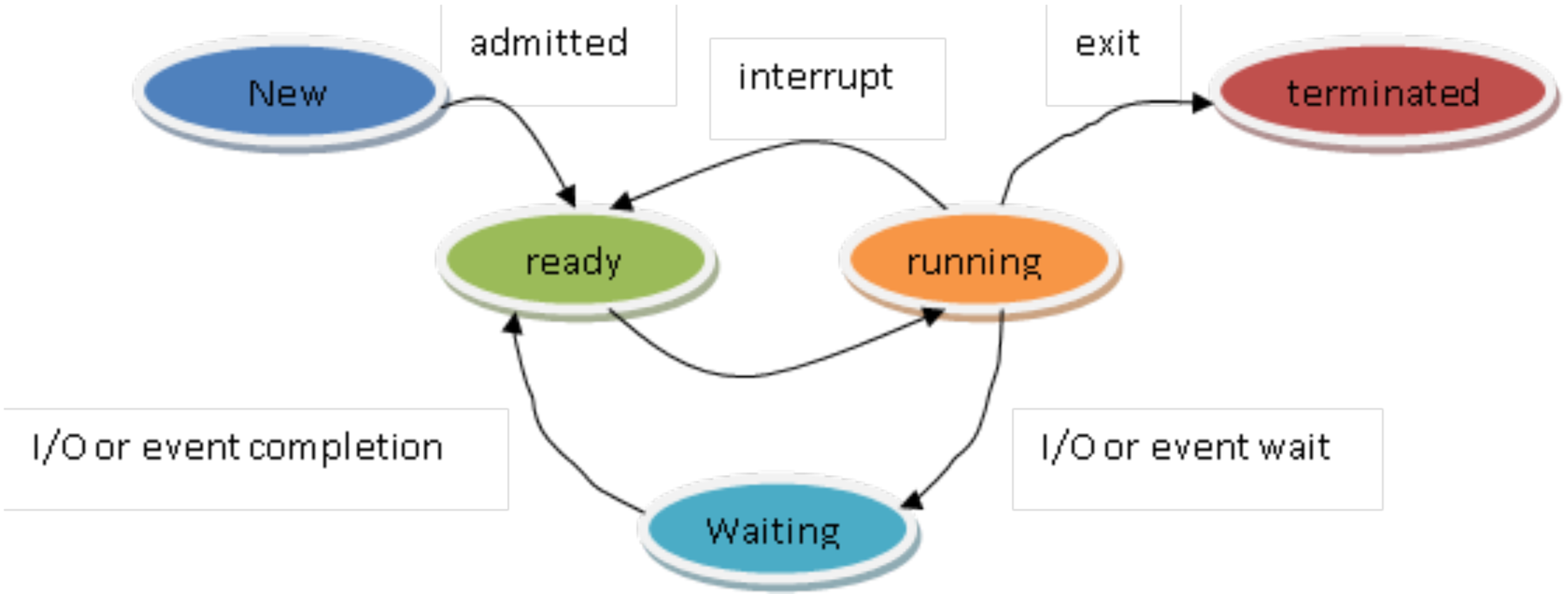


- User Interface is portion of OS that users interact with. Modern GUI OS's are really no different then command-line OS's in that they all access the Operating System directly.

Processor Manager

- Main function is to:
  - How to allocate the CPU Unit
- On a single computer/single process system this is fairly simple coordination.
- On a single Computer/many process’s system, there is much coordination.

Processor Manager will coordinate



Main Memory Manager

Comparison chart

	RAM	ROM
Definition	Random Access Memory or RAM is a form of data storage that can be accessed randomly at any time, in any order and from any physical location., allowing quick access and manipulation.	Read-only memory or ROM is also a form of data storage that can not be easily altered or reprogrammed.Stores instuctions that are not nescesary for re-booting up to make the computer operate when it is switched off.They are hardwired.
Stands for	Random Access Memory	Read-only memory
Use	RAM allows the computer to read data quickly to run applications. It allows reading and writing.	ROM stores the program required to initially boot the computer. It only allows reading.
Volatility	RAM is volatile i.e. its contents are lost when the device is powered off.	It is non-volatile i.e. its contents are retained even when the device is powered off.
Types	The two main types of RAM are static RAM and dynamic RAM.	The types of ROM include PROM, EPROM and EEPROM.

- Memory Manager responsible for managing RAM/ROM chips. The RAM/ ROM chips are the “physical” part , while the Memory Manager checks “requests” for memory, and if valid will allocate it to the process.

Memory Manager

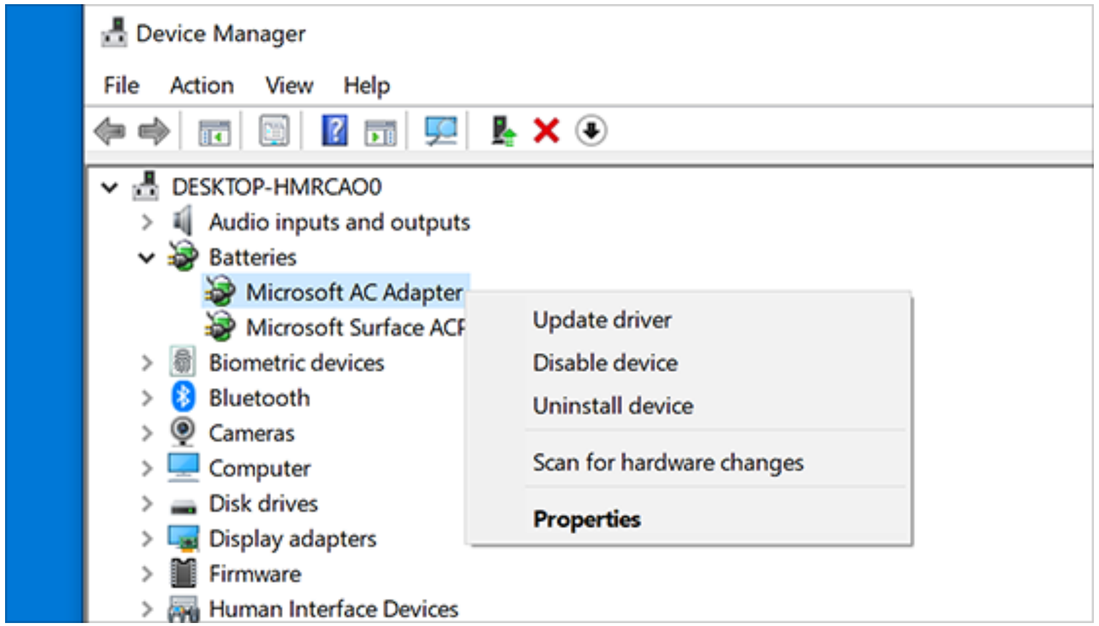
- Four main functions of the Memory Manager is to:
  - Validate Request
  - Allocate Request
  - Track Request
  - Deallocate Request

Device Manager

FIGURE 1-7
Typical computer hardware.

- Computer would be “boring” and pretty useless if no devices were attached to it

Softwareneeded to make devices accessible

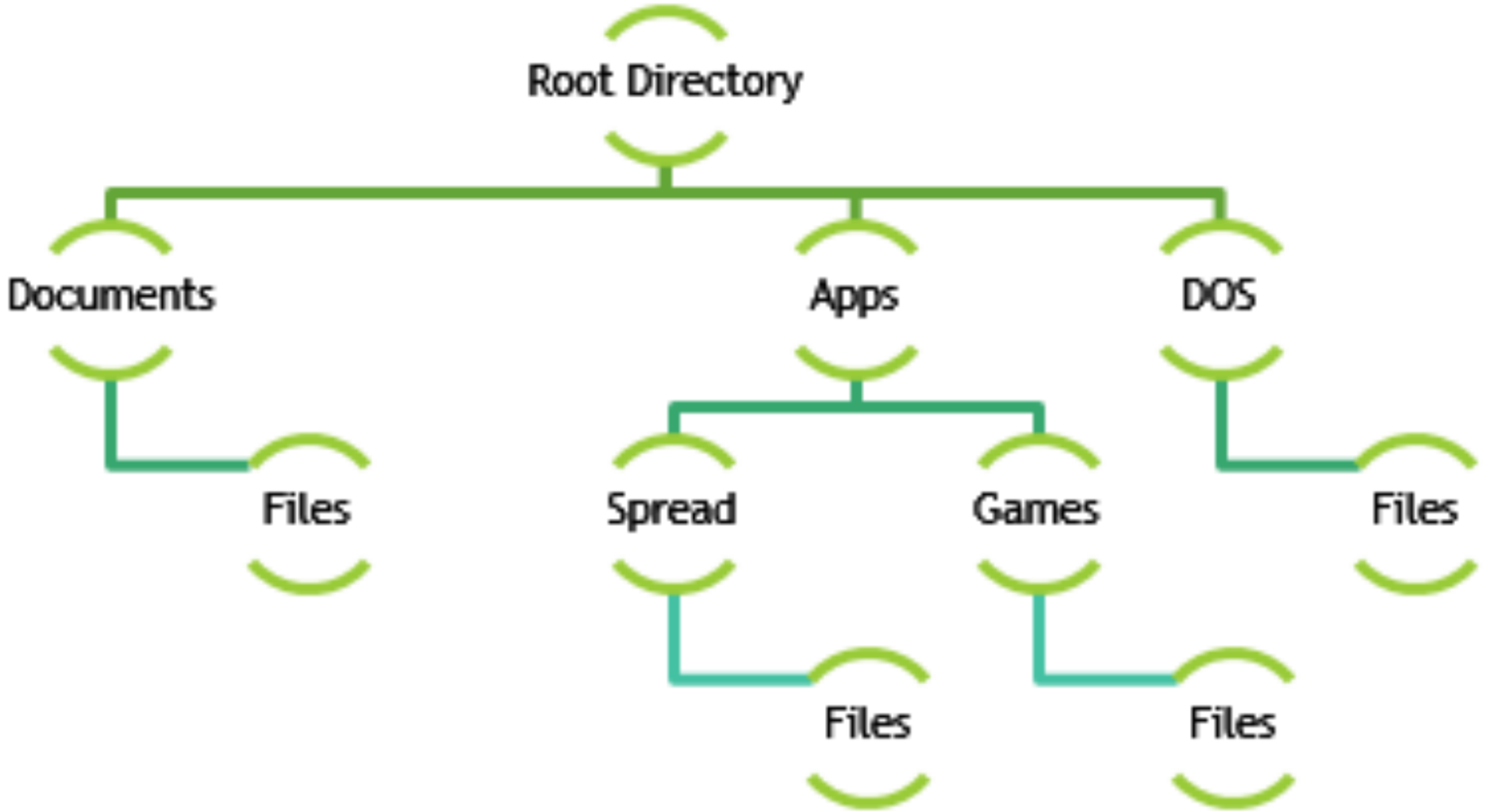


- Usually an Operating System will have “device drivers” that will be the software that “speaks” to the device.
- Device Manager responsible for managing the wide assortments of devices that your computer is attached to

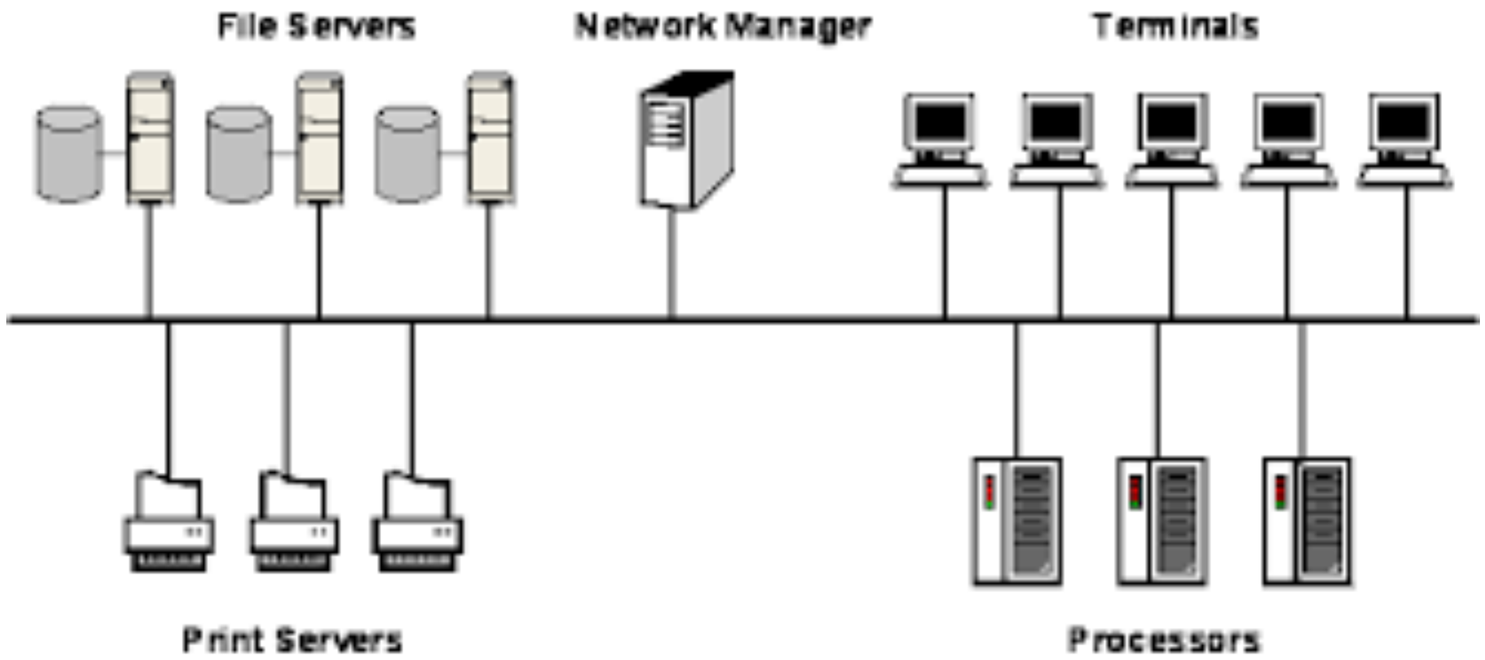
File Manager

- We won’t be studying the File Manager in this class. :(
- However, the main objectives are to keep track of
  - Files
    - Data Files
    - Program Files
    - Utilities
  - Allocating/Deallocating Space on a storage device
    - SSD (Hard Drive, Flash Drive, Floppy Disk (couldn’t resist)

Simple File System



Network Manager



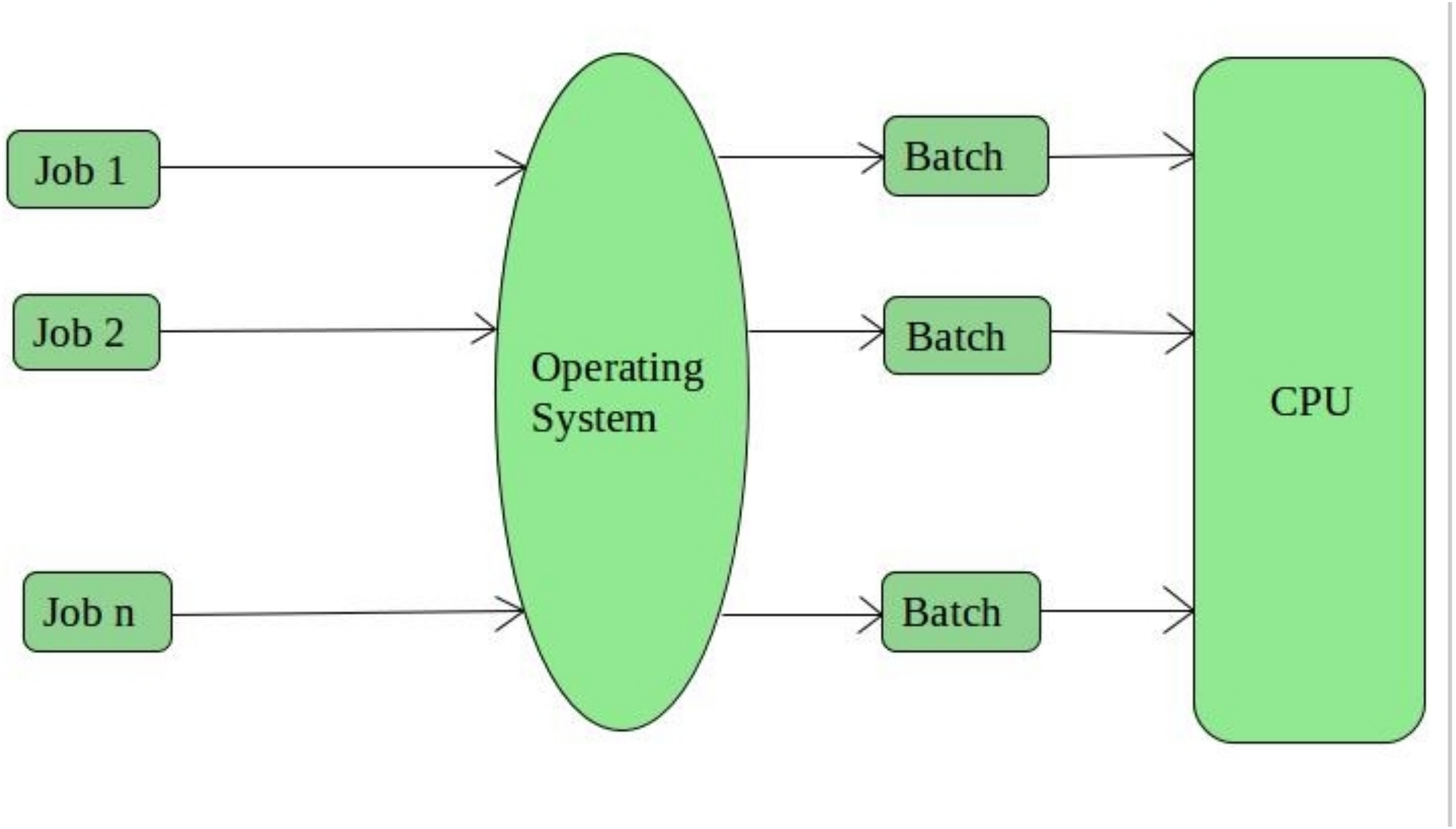
- Controls access and usage of network resources

When a mouse clicks..

- Somebody wanted to open a file
- Device Manager receives electrical impulse, figures out XY location
- Processor Manager dispatches control to a User process Mouse Handler, where the user program wants the file to be opened
- File Manager finds file on disk and grants access
- …. Many other Manager’s get involved

Types of OS

- **Batch / Mainframe**



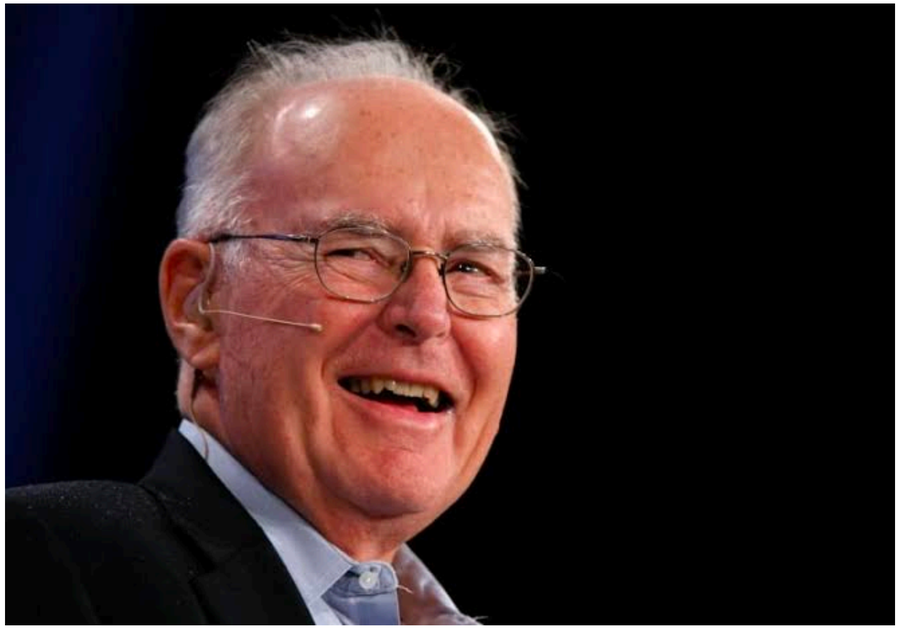
Types of OS

- **Interactive Systems**
- Windows, Mac, Linux are examples
  - They can run multiple jobs
  - Usually have a GUI which can delay jobs running in background

Types of OS

- **Hybrid/ Real-time & Embedded**
- Hybrid similar to interactive, but a combination of batch and interactive
- Real-time are time-critical OS’s with minimal GUI
- Embedded System’s are mini-computers stored in your DVR, Refrigerator. Minimal in complexity, and highly specialized to delivering fast response

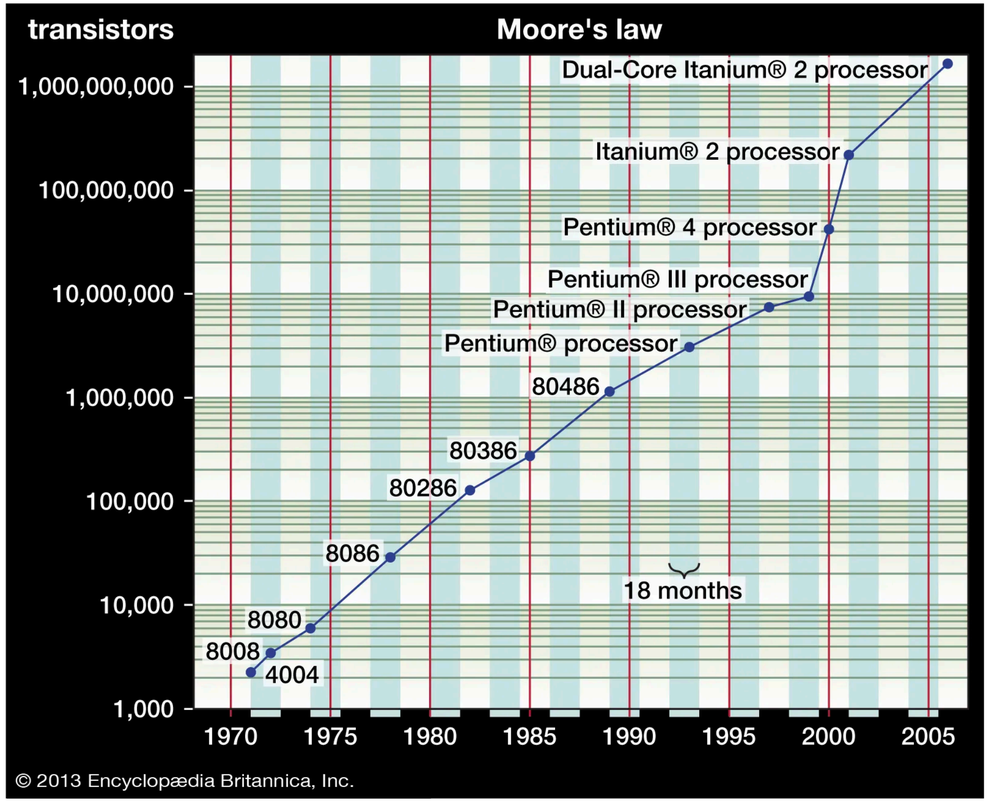
Moore’s Law



- “The number of transistors on a microchip will double every two years, while the price of the device will be cut in half.” - *Gordon Moore, 1965*

- i.e. Computers will get smaller, faster, cheaper

Moore’s Law



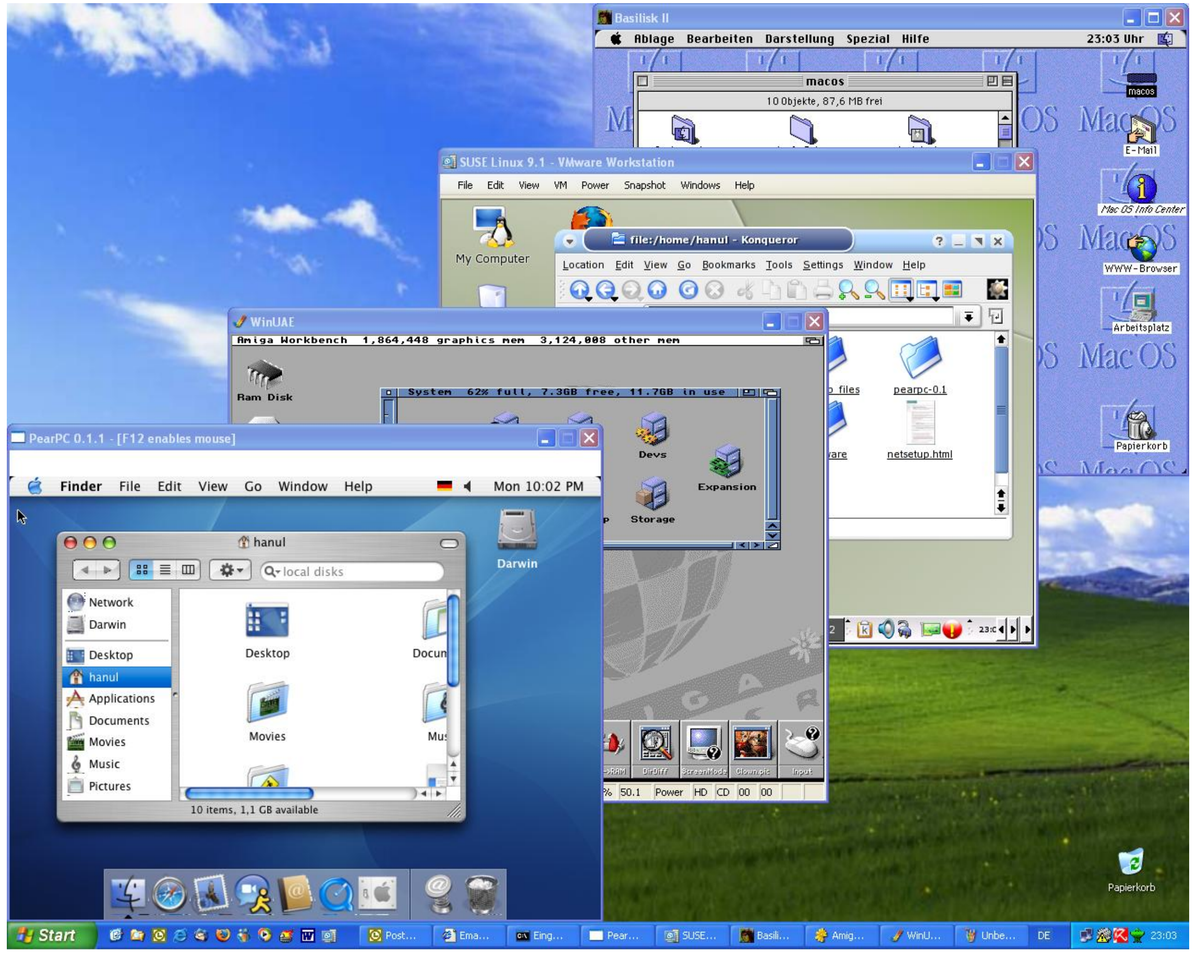
- “One of these days we’re going to have to stop making things smaller” - *Gordon Moore, 2007*
- i.e. at some point we expect this to stop being true.
- Computers are projected to reach their limits because transistors will be unable to operate within smaller circuits at increasingly higher temperatures. This is due to the fact that cooling the transistors will require more energy than the energy that passes through the transistor itself.

Three Easy Pieces

Virtualization, Concurrency, Persistence

If you had to define virtual in one word (or short sentence), it would be?

Virtualization Is Key to OS

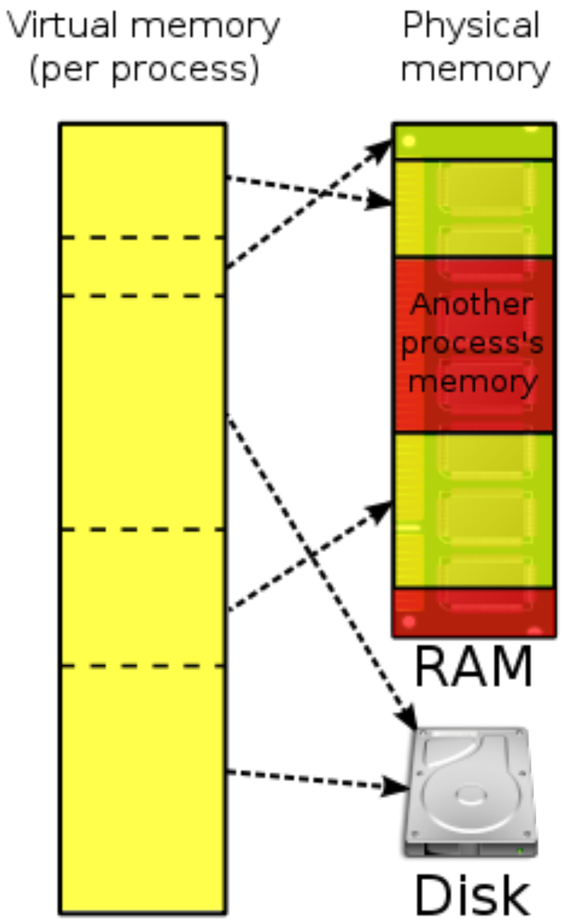


- In computing, “virtualization refers to the act of creating a virtual version of something, including virtual computer hardware platforms, storage devices, and computer network resources”.

Virtualization with CPU

- This is an example of a CPU. It’s just one physical device (sometimes more if is multi-cpu). It’s a single hardware device that has a set number of registers. Yet all programs act if they are the only one who uses it. Why?

Virtualization with Memory

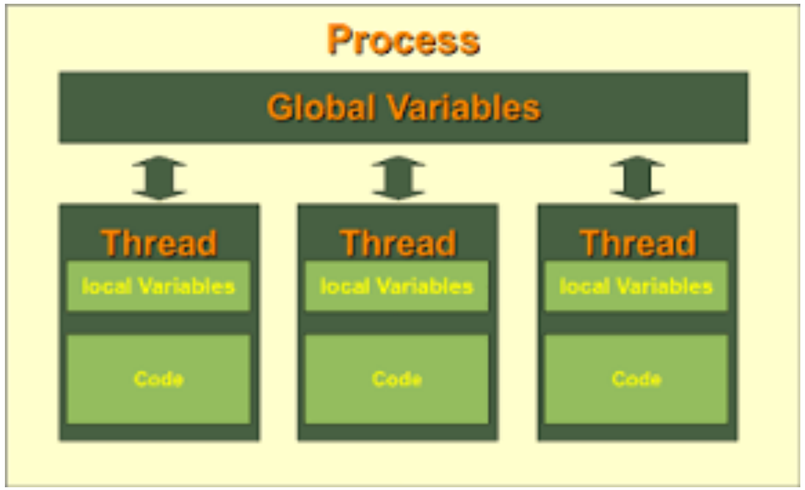


- Physical Memory is array of bytes and is accessed when your programs are run.
- Do you actually access a distinct memory location or is it virtual?

Concurrency

- Keeping multiple programs coordinated is truly a juggling act
- Back in the “old days” with a single process on computer was tough, but what about 100 users, 100 processes?

A Thread is like a Program



- Smallest sequence of programmed instructions that can be managed independently
- Multiple threads can exist within a program (running asynchronously)
- How do we coordinate them?

Persistence

- Another major aspect of OS is persistence.
- Memory can easily be lost (e.g. System Crash, Power loss)
- Are their mechanisms in place to handle dynamic problems
- For Operating Systems, this generally is File System
- In this course, we won’t touch on this as much, and instead will focus on Network System (which is actually similar to File System)

Design Goals

- Let’s say your were tasked with creating an Operating System for a new Computer that your company just created. What would be some of your design goals?
  - Performance
  - Protection between Applications
  - Reliability