Introduction to computing, architecture and the UNIX OS

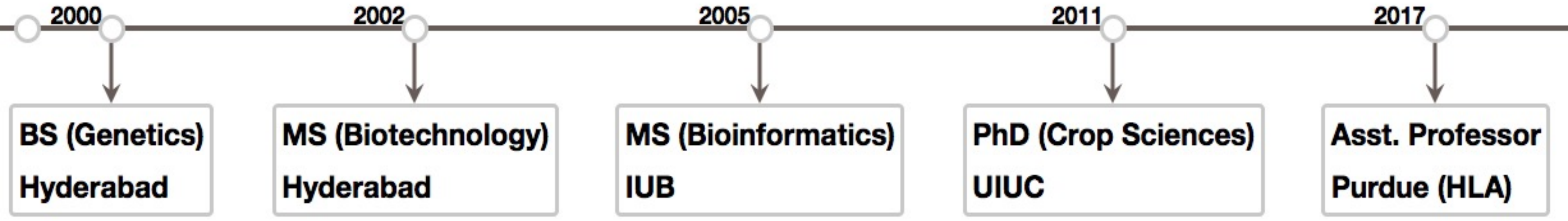
HORT 530 Lecture 1

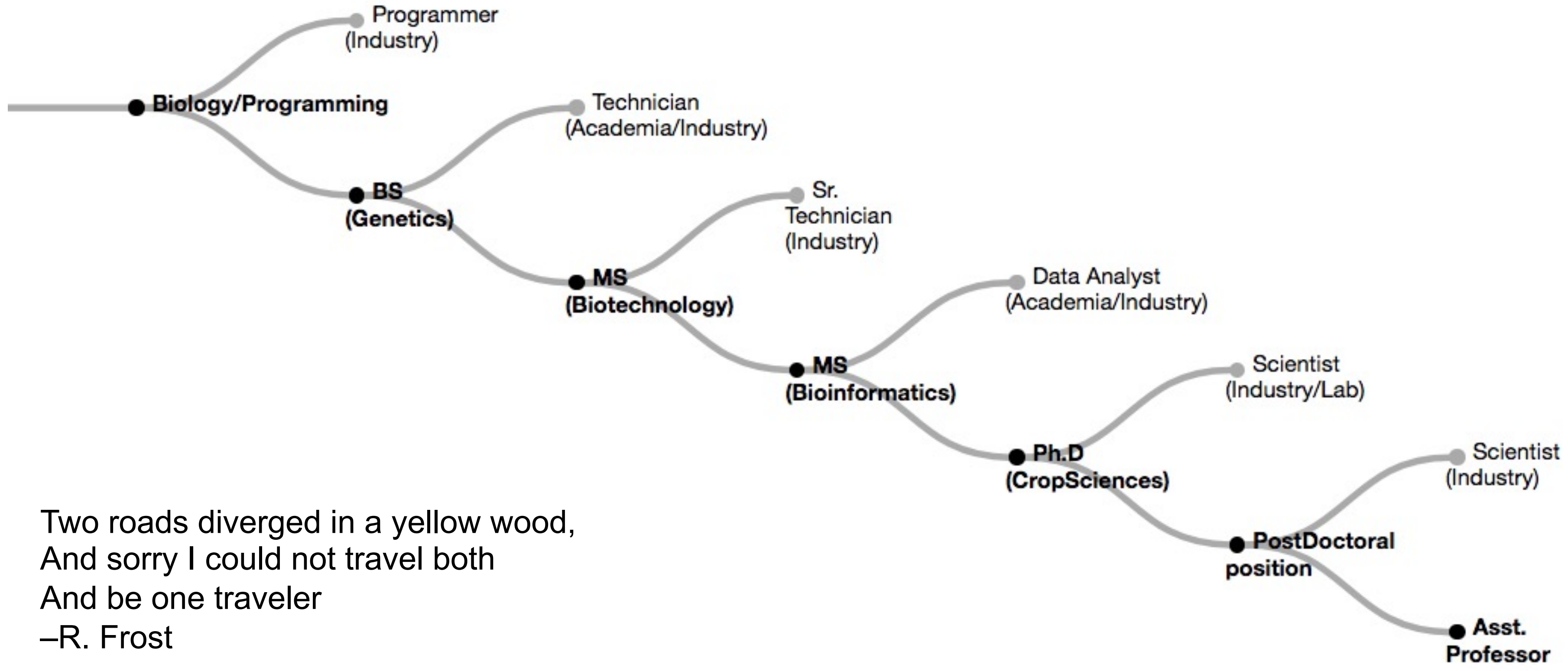
Jan 11th 2022

Instructor : Kranthi Varala

Email : kvarala@purdue.edu

Kranthi Varala





Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler
–R. Frost

Introduction to Computing for Biology

- Class Times:
  - ❖Lecture: Tuesday 3:00 pm - 4:15 pm BRNG B282
  - ❖Lab: Thursday 3:30 pm - 5:20 pm Hicks G959
- Lectures are in person, but also streamed on zoom
- Lab will be in person FOR NOW
- Zoom link sent via email
- 7 quizzes (70%), final project (30%)
- Website:

https://www.purdue.edu/hla/sites/varalalab/IDAB/

Course goals

- Learning to use remote servers
  - UNIX operating system
  - Command-line tools
  - Shell scripting
  - Clusters and job management
- Learning programming
  - Introduction to Python
  - Learn to:
    - Automate repetitive tasks
    - Handle large data sets
    - Link processes/tasks together (Serial & Parallel)

Lecture Plan

Week	Lecture	Lab	Topic	Quiz
1	11-Jan	13-Jan	Introduction to computing, architecture and the UNIX OS	
2	18-Jan	20-Jan	The UNIX Operating System	1
3	25-Jan	27-Jan	Doing more in UNIX: Command-line tools	2
4	1-Feb	3-Feb	Regular expressions: Text manipulation	3
5	8-Feb	10-Feb	Shell scripting and system variables	
6	15-Feb	17-Feb	Super Computers, Job management, PBS	4
7	22-Feb	24-Feb	Intro to Programming, Variables and Objects	Idea
8	1-Mar	3-Mar	Introduction to Python: Data types	5
9	8-Mar	10-Mar	Numbers, strings, and lists	
10	22-Mar	24-Mar	Lists, conditions and loops	Pseudocode
11	29-Mar	31-Mar	Dictionaries, tuples and sets	6
12	5-Apr	7-Apr	Functions, Scope, Arguments	7
13	12-Apr	14-Apr	Algorithms, Sorting	Version 1.0
14	19-Apr	21-Apr	Libraries	
15	26-Apr	28-Apr	No Lecture	

Final project stages

Computer architecture: “How stuff works©”

- Almost everything you do on a computer passes through a series of components and is eventually broken down to individual calculations/operations.

- The individual operations are performed on the CPU and the output is returned through another series of components to the user.

- For example, clicking on a web link goes as follows: Mouse->USB port->Motherboard->CPU->Browser. Browser responds to this click in reverse: CPU->Motherboard->Graphics card->Monitor.

- Much of this complexity is hidden from the average user, but becomes important when you are writing your own programs.

Hardware terminology

- CPU = Central Processing Unit
- Core = Complete subunit of CPU
- CPU => 1 to Many cores
- RAM = Memory = Fast, temporary storage
- Secondary Storage (Disk) = Slow, permanent storage
- Bus = Communication channel to move data
- Cache = Very small, but very fast storage
- I/O = Input and output devices or channels
- Bandwidth = Size of data moved in 1 operation

Von Neumann Architecture

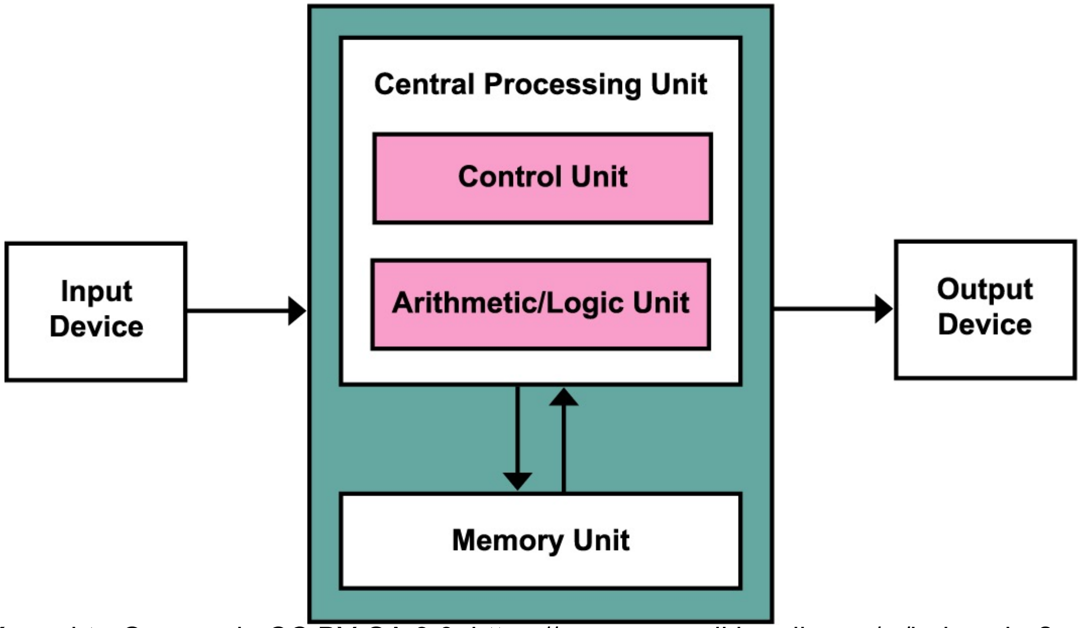
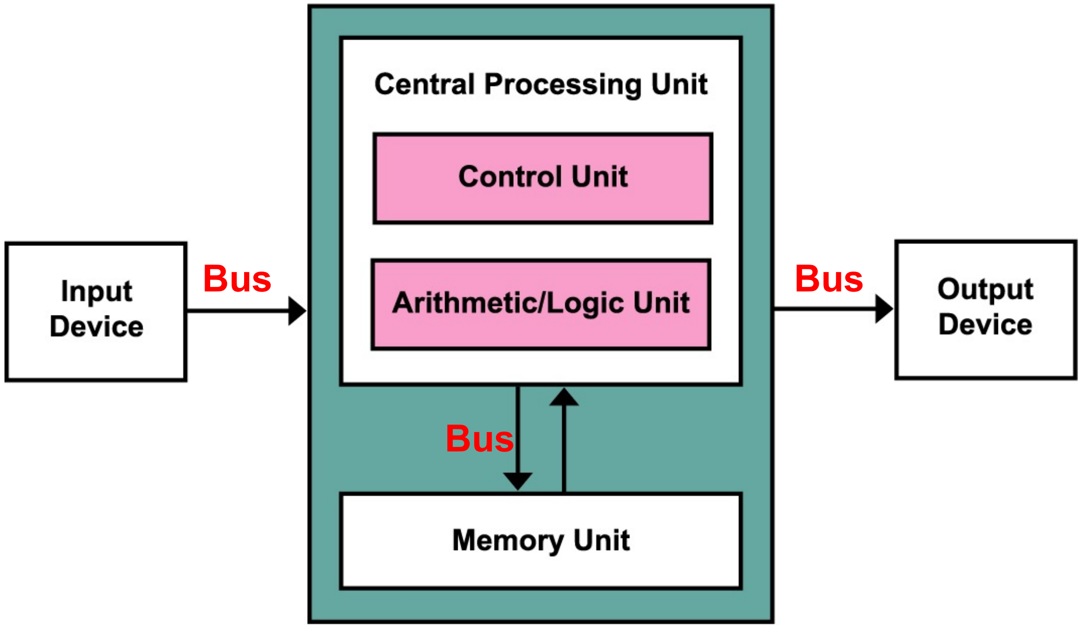


Image Credit: Kapooht - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=25789639

- This basic structure was first proposed by John Von Neumann in 1945.
- The Arithmetic/Logic Unit (ALU) does all the calculations.
- The Memory Unit (RAM) holds the program to be run and the data.
- The Control unit manages the flow of data and the execution of the programs.

Data flows through busses

Eg:
Keyboard,
Mouse, Hard
drive,
CD/DVD etc.



Eg:
Monitor, Hard
disk, Printer etc.

Image Credit: Kapooht - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=25789639

- A Bus is a dedicated channel for transmitting data from one component to another. Busses vary in size and speed.
- The speed of information transfer through the various busses is a key limitation on computational speed.

Harvard Architecture

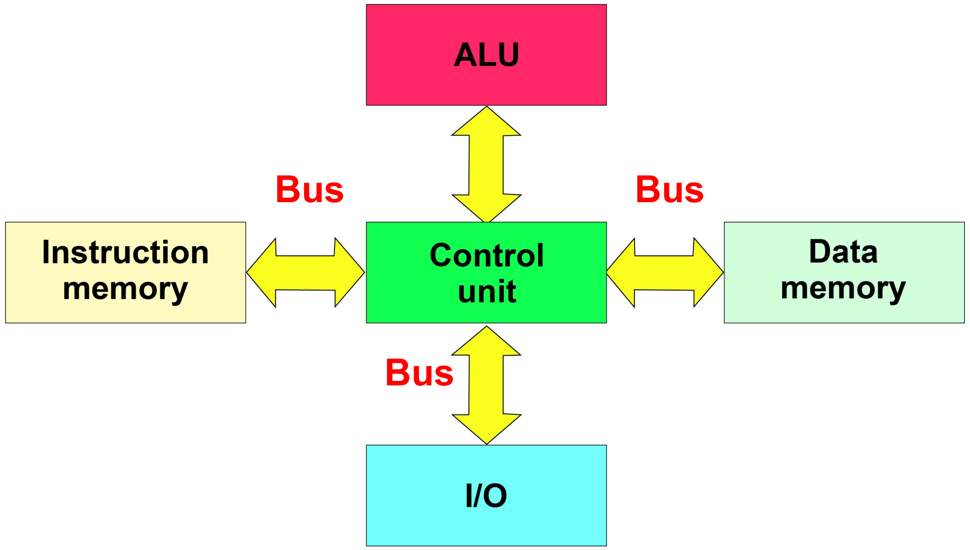


Image Credit: Nessa los - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10303637

- The main difference is that there are two separate busses for the data and instructions (i.e., programs).
- These independent busses can then be different in size. For example, the data bus can be bigger than the instructions bus.

Caches improve CPU performance

- A cache is a small, temporary storage location directly attached to the CPU.
- Modern CPUs are built with extremely fast caches to reduce the limitation caused by the need to move data through the busses.
- A cache can hold the data/instructions needed by the CPU for the next calculation(s) which reduces the time the CPU has to wait for the next instruction AND data.
- The CPU, Memory and Storage devices have separate caches.
- Multiple cores in a CPU share caches.

Modern CPU architecture

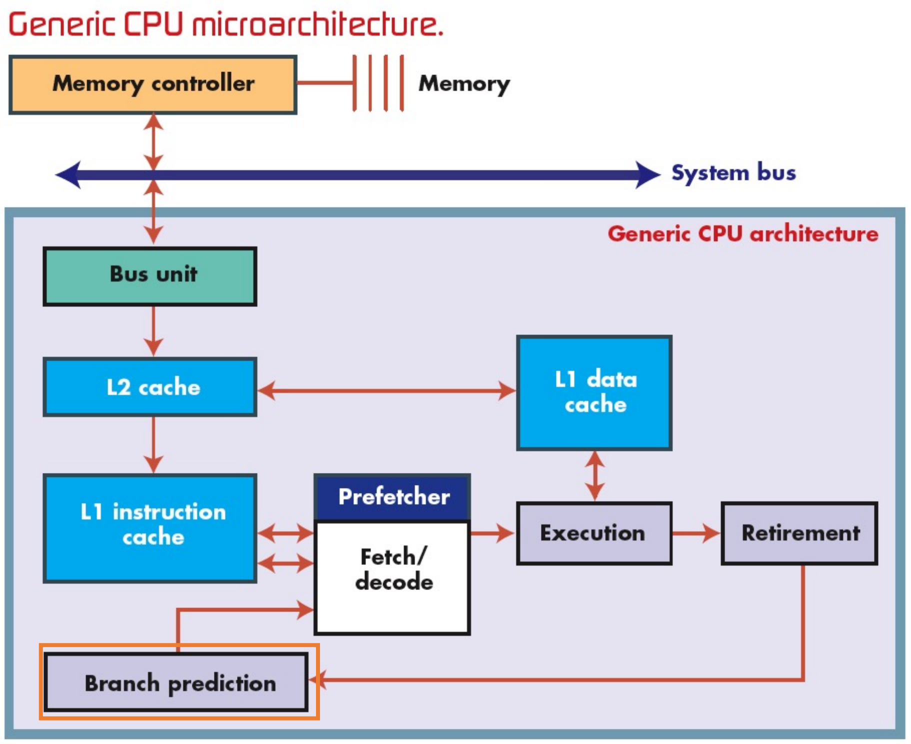
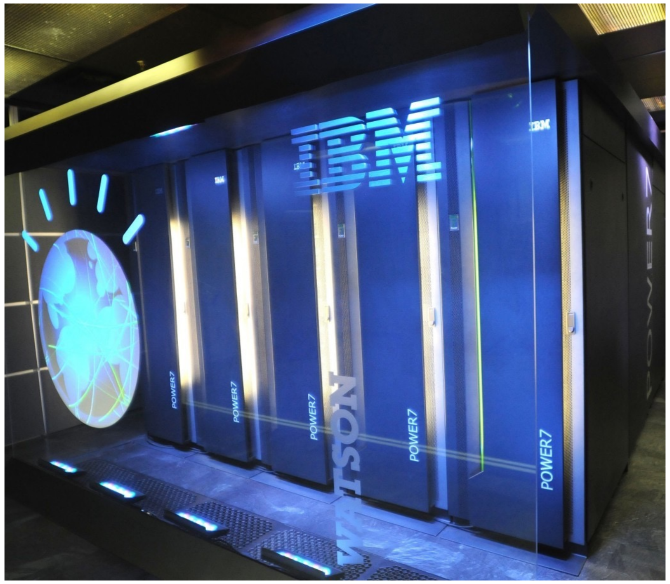


Image Credit: https://www.embedded.com/print/4007065

- Most modern computers use a hybrid architecture that is sometimes called a modified Harvard architecture.
- Many of the modern gains in computing speed are a result of cache optimization and prediction of how data/instructions will be reused.

Examples of Hybrid architecture



Mainframe

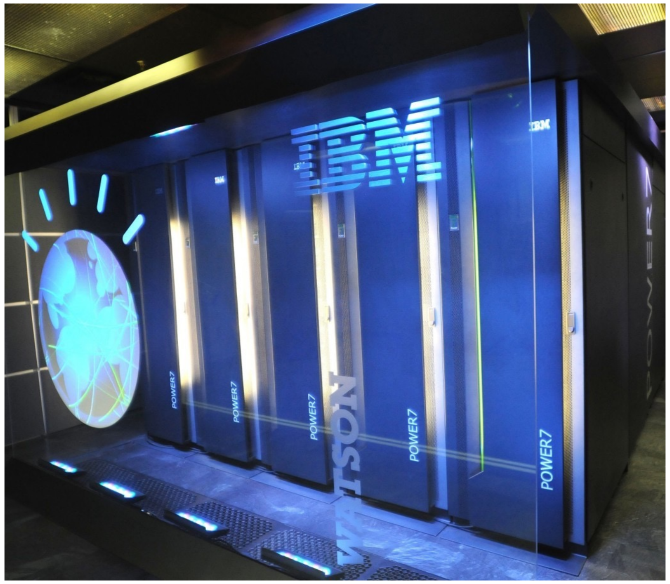
Workstation

Personal Computer

- Computers come in a wide range of shapes and sizes.
- They still follow the same basic architecture.
- These computers differ vastly in their computational ability, portability and user experience.

Image Credit: 1. IBM, 2. Fujitsu 3. By Ashley Pomeroy - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=62729054

Examples of Hybrid architecture



Mainframe

Workstation

Personal Computer

Decreasing computational power

Increasing portability

Increasing ease-of-use

Image sources: 1. IBM
2. https://pixabay.com/en/imac-pc-personal-computer-computer-2493287/
3. http://www.flickr.com/photos/intelfreepress/6675870523/sizes/o/in/photostream/

Hardware vs. Software

- Hardware is the physical component of a computer and includes the CPU, Storage and I/O devices. Hard to change and typically remains constant for the life of the machine.
- Software is the set of instructions that allows you to use the hardware. It includes the Operating System (OS), Device drivers, Applications etc. Easier to update, and is often patched to improve function and/or security.
- Firmware is a specialized kind of software that is specific to and resides on the individual hardware components. May be updated, but rarely does.

Operating Systems

- An Operating System (OS) serves as the middle layer between the hardware and the programs the user needs.
- Examples of common operating systems are: UNIX, Windows, MacOS, Linux, Android, iOS etc.
- The common tasks of an OS include communication with hardware, memory management, storage management, process/task management, networking, security, etc…

Operating Systems

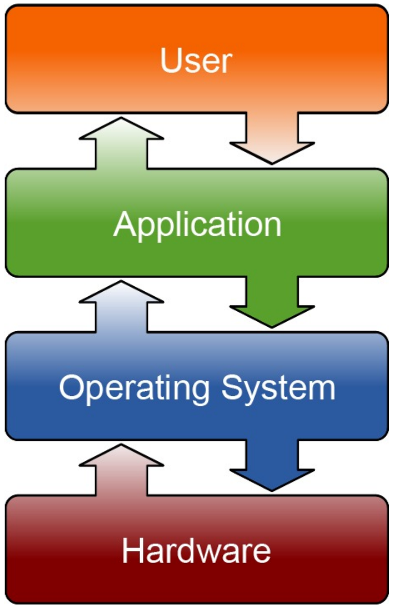


Image By Golftheman - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4558519

- Operating systems allow the separation of hardware management from applications/programs.
- For example, when an application reads or writes a file, the file access and the writing functions are handled by the OS.
- This allows the applications to work across different hardware platforms, although the applications are still specific to the OS.

Operating Systems

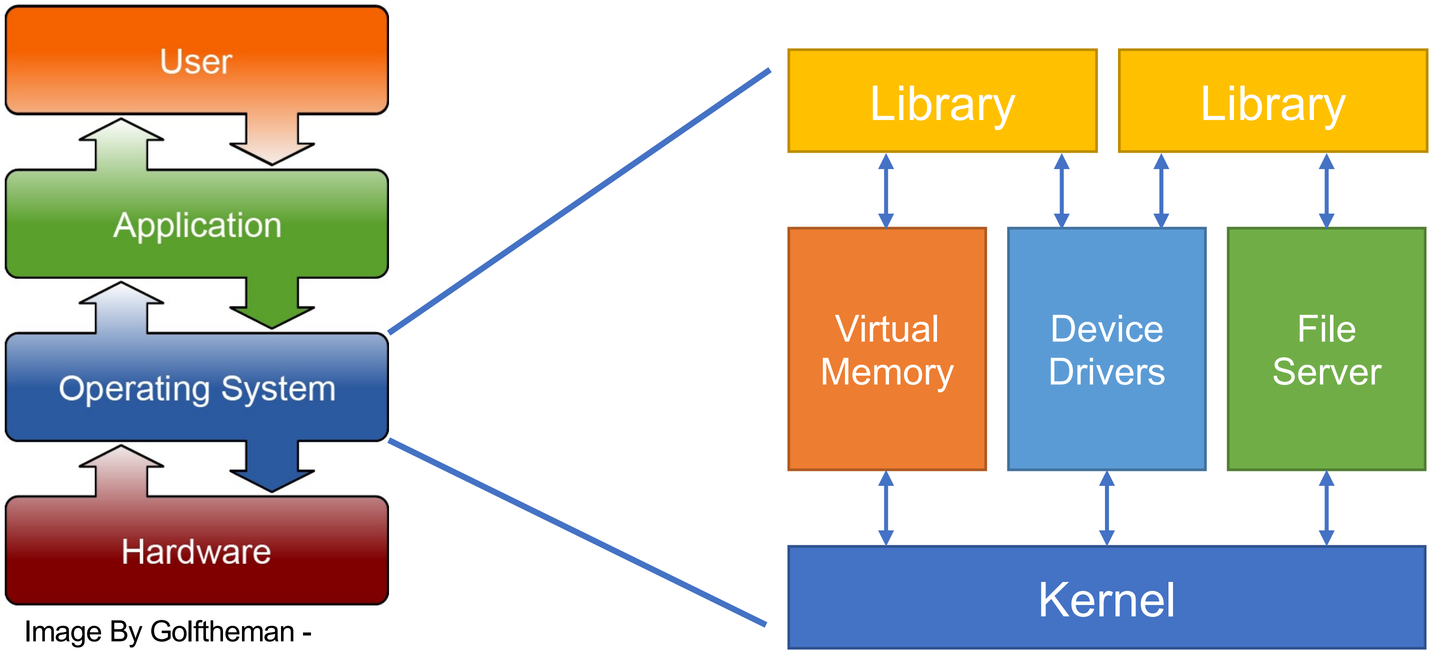


Image By Golftheman - https://commons.wikimedia.org/w/index.php?curid=4558519

- The Kernel is the core function of the OS and handles basic-level communication between the various processes and the hardware.
- Specific modules such as the memory manager and device drivers allow easier ways to update the OS as required.
- Libraries provide applications with standardized access to kernel functions.

Virtual Memory

- An abstract layer created and managed by the kernel.
- All memory requests from applications are sent to the Virtual memory management process.
- Allows the applications to use physically separate memory locations as if they are continuous.
- Allows combining Memory chips (RAM) and Disk space into a single memory space.
- Disk space configured for memory use is called “swap”. It is sloooow.

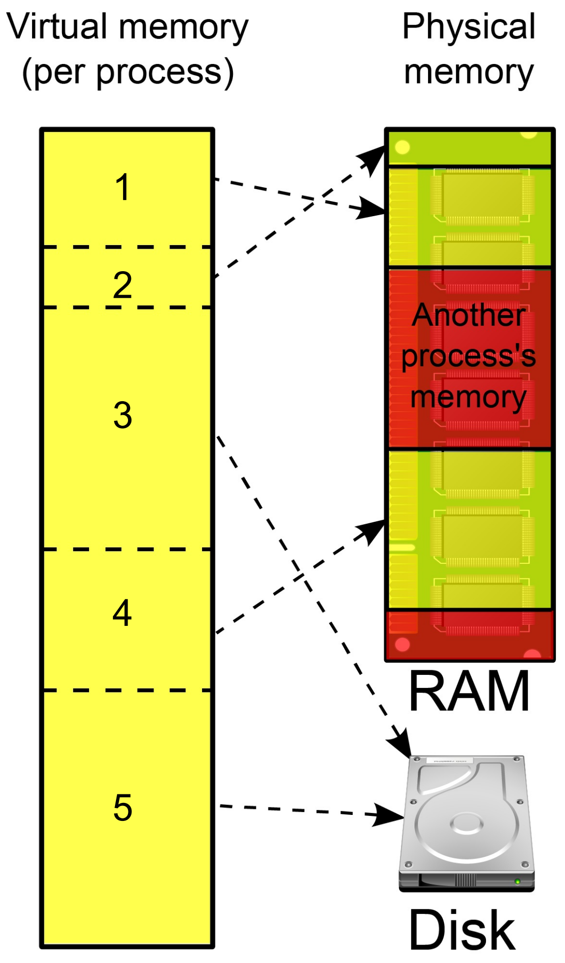


Image Credit: Ehamberg - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=8352077

OS paradigms for user interaction

- Multiuser, shared access
- Examples: Clusters, Web clients, Browsers etc.
- Computational resources are managed by the OS and users may have different levels of priority.
- User tasks are scheduled by the OS based on priority.

- Single user, exclusive access
- Examples: PCs, Smartphones etc.
- All resources are dedicated to the single active user.

Client/Server paradigm

- A Server is a central, powerful machine that typically has LOTS of computational power, memory and disk space.
- A client is a relatively smaller machine that connects to the server and uses its computational power for specific tasks.
- Servers are optimized for speed and stability, but have very limited User Interface (UI).
- Clients are optimized for ease-of-user and typically have a graphical UI.
- Clients and servers may have the same or different OS. They communicate through standardized communication protocols.
- Examples of communication protocols are: HTTP, FTP, SSH etc.

UNIX operating system

- First developed in 1970s, it is a multitasking OS that supports simultaneous use by multiple users.
- User interface is typically limited to text-only interactions, thus avoids wasting resources on generating graphics.
- It was built to simultaneously run thousands of programs and allow linking different programs together.
- Follows the Client/Server architecture where the UNIX server supports multiple clients/users using a communication protocol (eg., SSH).

UNIX operating system contd..

- UNIX OS has evolved to have numerous variants over the past 50 years.
- The most commonly used variant of UNIX is called Linux (Named after its inventor Linus Torvalds).
- Linux itself comes in hundreds of varieties, called distributions, that all share the same kernel and differ in the libraries and UI built on top of the kernel.
- MacOS and iOS is also based on a UNIX variant called BSD.
- Android is also derived from Linux.
- Windows is the only major OS that is not based on UNIX.

UNIX access via Terminals

- Earliest clients, called terminals, that connected to UNIX servers were teletypewriters (TTY).
- Video capable terminals included a video screen and a keyboard (1970’s onward).
- Modern clients (i.e, your PC) uses a terminal emulator to mimic the behavior of a terminal.
- This terminal emulator establishes a connection to the server to create a “shell”.
- A shell is a text interface that the users enter their commands on. The server returns the output to the terminal.
- We will learn more about terminals and shells in the next lecture..

Compute clusters

- Set of individual machines that are combined to work together and can be accessed as a single server.
- Each individual machine is called a node in the cluster. A specialized node called ”Head” node acts as the interaction point for users.
- A cluster OS manages the communication between nodes and the submission of user jobs to the appropriate nodes.
- Purdue hosts multiple clusters through the Research Computing facility (https://www.rcac.purdue.edu)
- You will all have accounts on the Scholar cluster for this course.

Distributed Computing

- A grid of individual machines that are configured to contribute their idle time to run jobs for the central job server.
- Typically used when a large job can be broken down into a set of small tasks that can be run independently of each other.
- For example: Near-Earth Asteroid search, Protein folding, Rendering animation etc.

Virtual Machines

- VMs simulate a physical computer with its “hardware” and OS within another OS.
- Allows one server to provide multiple virtual machines.
- Allows one OS to emulate and run applications from another OS.
- Examples: VirtualBox, Vmware, Xen etc…

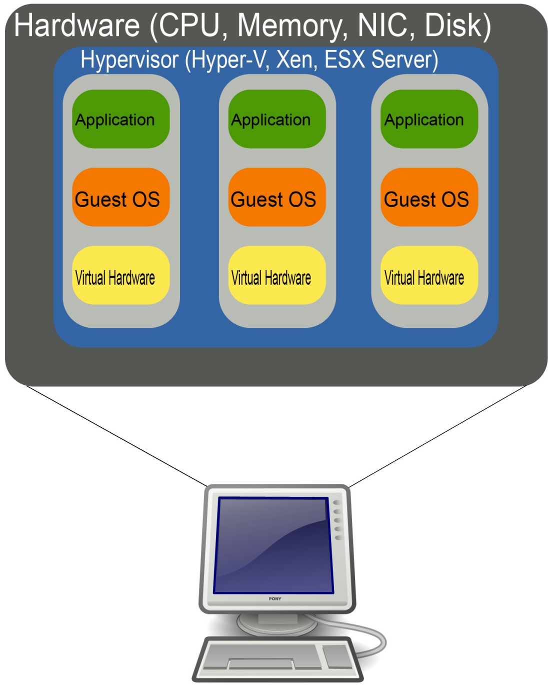


Image Credit: John Aplessed, https://commons.wikimedia.org/w/index.php?curid=12351968

Cloud computing

- An extension of the virtualization concept, where the VM is created on-demand, on a remote server.
- Extensive customization of the VM is possible by specifying the “hardware” and the OS+applications package on the VM.
- Numerous vendors offer cloud computing now, including Amazon, Microsoft, Oracle and Google.
- For example, I can request a VM instance with the following specs: “12 cores, 128GB RAM, 1 TB Disk, Linux OS with a specific list of libraries and applications”.

Architecture and OS Summary

- Be aware of the computer hardware you are working on.
- Pick the computing model best suited for your task, i.e., not every task is well suited for a cluster.
- Moving data around, especially large biological datasets, can be ”expensive”.
- When you write programs, try to minimize the slow parts, such as reading/writing from disk.
- Optimize your programs to make best use of the server architecture.
- Servers are meant to be shared. Be ”nice” by requesting only the amount of resources you can use.