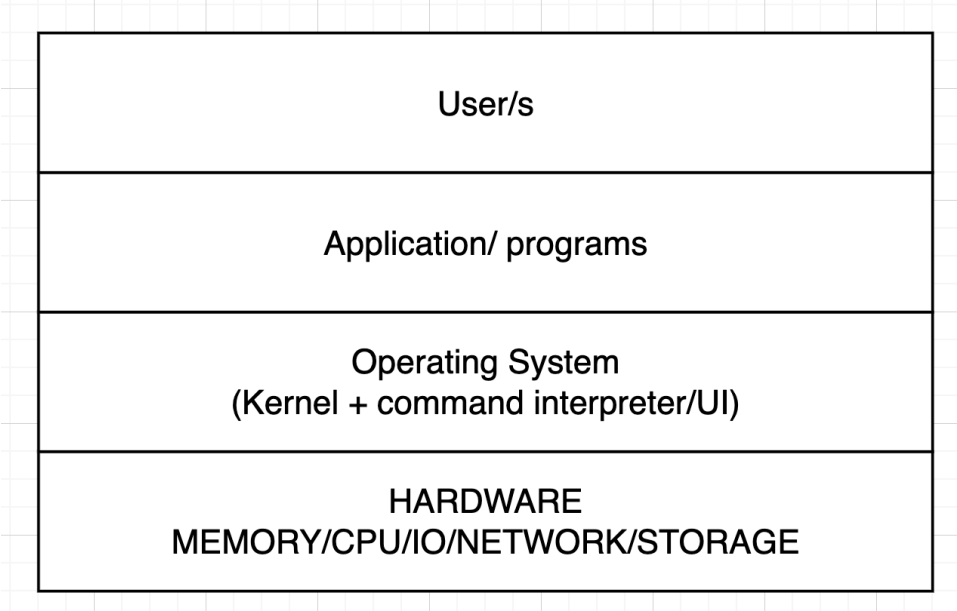
Operating System

- A Software to manage computer hardware
- Consists of
    - Kernel - communicates with and controls the hardware
    - Command interpreter / UI - Deals with the end user



Some of the most common operating systems for computers:

- macOS
- Linux (multitasking + PC or centralized operating system)
- MS Windows
- Unix (multitasking/ centralized operating system)
- MS DOS (single task os)

MS DOS

- command.com → command interpreter (processes internal commands)
- io.sys
- config.sys
- autoexec.bat (executes a bunch of commands to initialize your setup)
- A bunch of executable programs called external commands

Most common applications of olden days

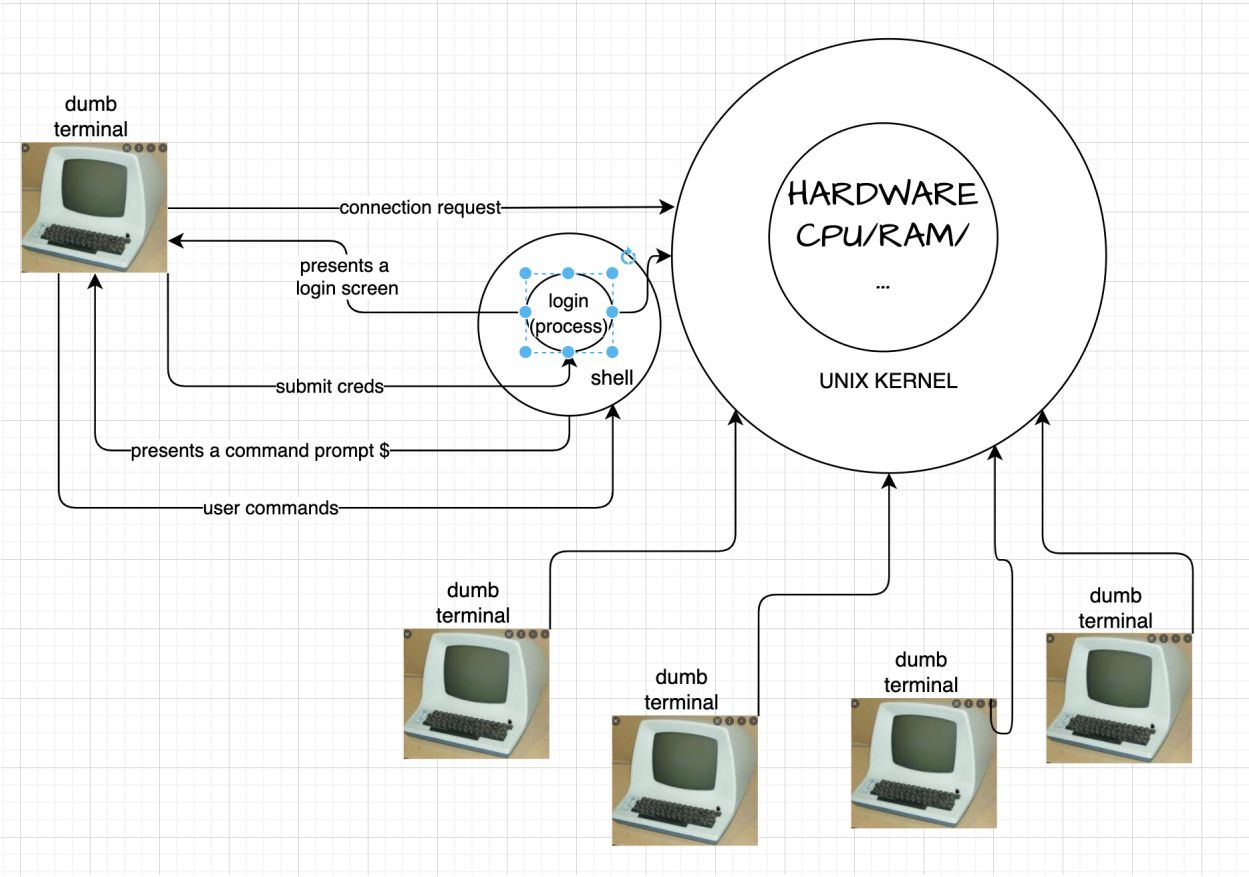
- Wordstar → word processor software (like MS-Word)
- Lotus 123 → spreadsheet software (like MS-Excel)
- Windows 3.11 → a GUI for managing your system/ applications

Responsibilities of an OS:

- Memory management
- I/O Management
- Network management
- CPU / Process management
- Disk/ Storage management
- User/Security management

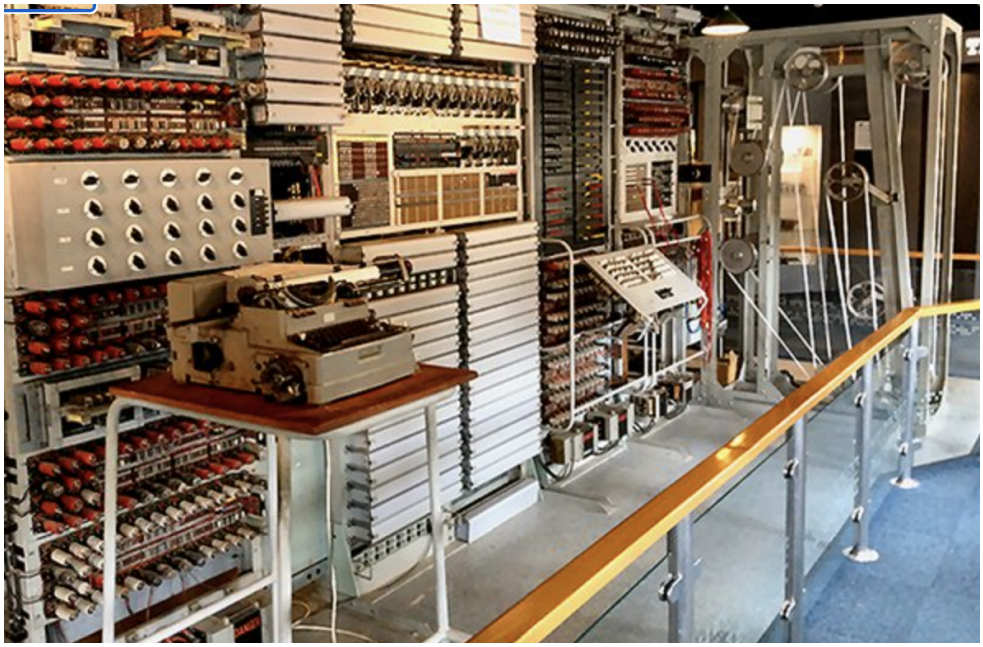
Unix Operating System

- Multi user operating system
- Multitasking operating system
- Centralized operating system



Generations of computer and OS

- 1st generation
  - 1945-55
  - Vacuum tubes as electronic parts
  - OS was implemented using plug boards / punch cards

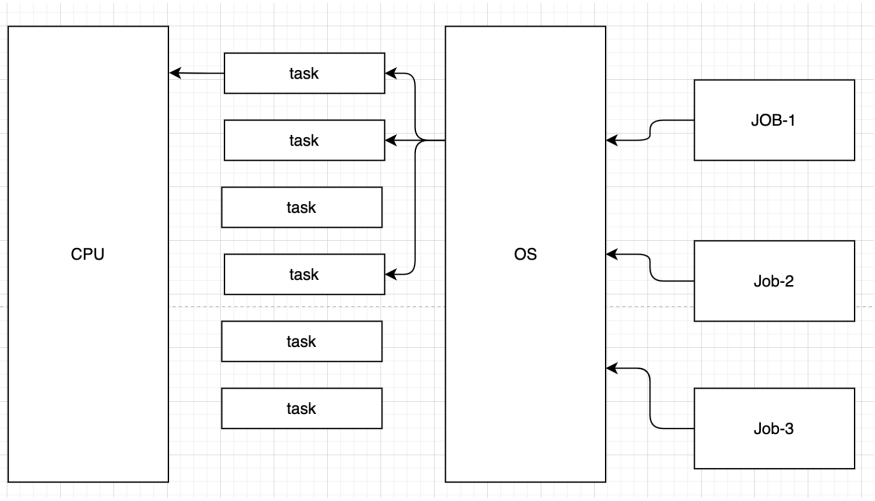


- 2nd generation
  - 1955-65
  - Transistors for electronic components
  - OS was implemented as batch system

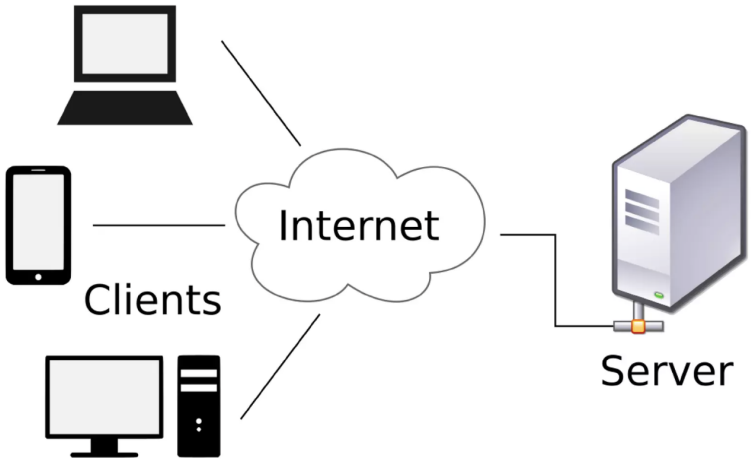
- 3rd generation
    - 1965-80
    - Integrated circuit chips for electronic components
    - Multiprogramming
- 4th generation
    - 1980 onwards
    - LSI
    - Different operating systems

Different types of OS

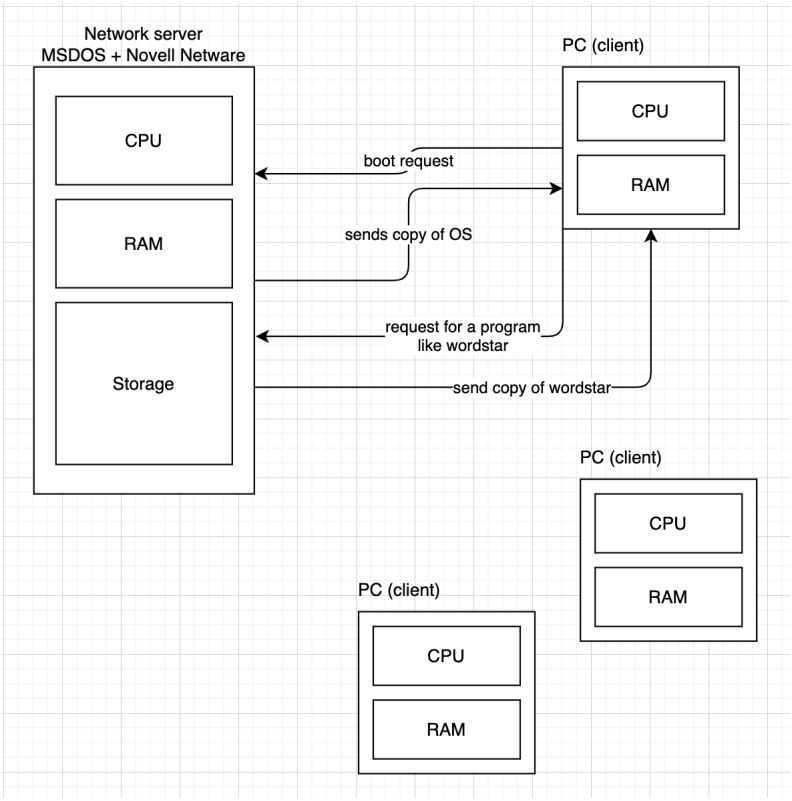
- Batch OS
    - Examples: Payroll system, banking application, controlling of huge machines in industries



- Time-shared operating system
    - Examples: Unix, Linux, Windows, Multics
    - Every task is given a fixed small amount of time (usually in milliseconds or nanoseconds)
    - At a given point in time, only one task is handled by the CPU
    - The task under execution by the CPU is processed for the given fixed time, and then will be suspended to give time to another task
- Distributed OS
    - Lots and lots of computers are interconnected in the internet world!
    - Think of the amount of unused CPU/RAM/Network etc collectively
    - What if we can aggregate all of the unused resources of these distributed computers and make them work like a single computer?
    - Leverages the computational power possessed by computers in the internet world



- Network OS



  -
  - Specialized network server is required
  - Client computers do not have Storage
  - OS and other software are loaded from the remote storage of the network server
-

Linux OS

- Developed by Linus Towalds in 1991
- 
- Free
- Open Source
- Over 600 distributions of Linux exists
- Has two parts:
    - Kernel
        - Talks to and controls the hardware
        - Core part of OS
        - Manages resources using **drivers**
    - Bins and libs (programs and commands)
        - Special programs that help accessing the kernel (from other programs)
        - Provide System calls (functions/apis that can be called from other programs)
            - open() close() read() write()

Process Management

- A process is a program under execution
- A Program is a set of instructions
    - Developers write in high level language like Java, C or C++, etc
    - OS requires these instructions in *machine* (OS) language
- A Process is an active entity
- A Program is a passive entity
- A Process has:
    - PID - Process ID
    - Status (new, ready, run, wait, complete, …)
    - CPU Registers (memory inside of CPU)
    - I/O Status
    - Schedule information (such as priority information)
    - Account information (such as user, group etc)

Different states of a process

- New
    - You just executed the command or the program
    - OS creates a running process for the same (allocates for example ID)
    - Added to a queue of a bunch of other already existing processes, and the process waits for its turn of the CPU time
    - Being prepared for execution
- Ready
    - Once the process is completely ready with all information like ID etc, the “Ready” is given to the process
    - As if the process “IS READY FOR EXECUTION”
    - Waiting…
        - Since one CPU handles one process at a time
- Running
    - CPU is currently executing/ interpreting/ handling the process
    - There will be a time cap as how long a process is handled by the CPU
        - If the task of the process is not finished with in the given time, the process state will be changed to “Ready”
        - If the task is done with in the given time, then the state of the process is changed to “Complete”
- Wait / Blocked
    - A Process becomes blocked when the process demands I/O access
        - For example, *ls -l* | *more*
            - The more command waits for user to enter RETURN or SPACE or ‘q’
    - Once the process is blocked, it can not be resumed back to the “Run” state. Instead, it gets the “Ready” state when the user input is complete.