APPLIED OPERATING SYSTEM

Er. Rudra Nepal

Chapter 1
Introduction

What Is An Operating System

A modern computer consists of:

- One or more processors
- Main memory
- Disks
- Printers
- Various input/output devices

Managing all these components requires a layer of software – the **operating system**

What is an Operating System?

- A modern computer is very complex.
  - Networking
  - Disks
  - Video/audio card
  - ….
- It is impossible for every application programmer to understand every detail
- A layer of computer software is introduced to provide a better, simpler, cleaner model of the resources and manage them

What Is An Operating System

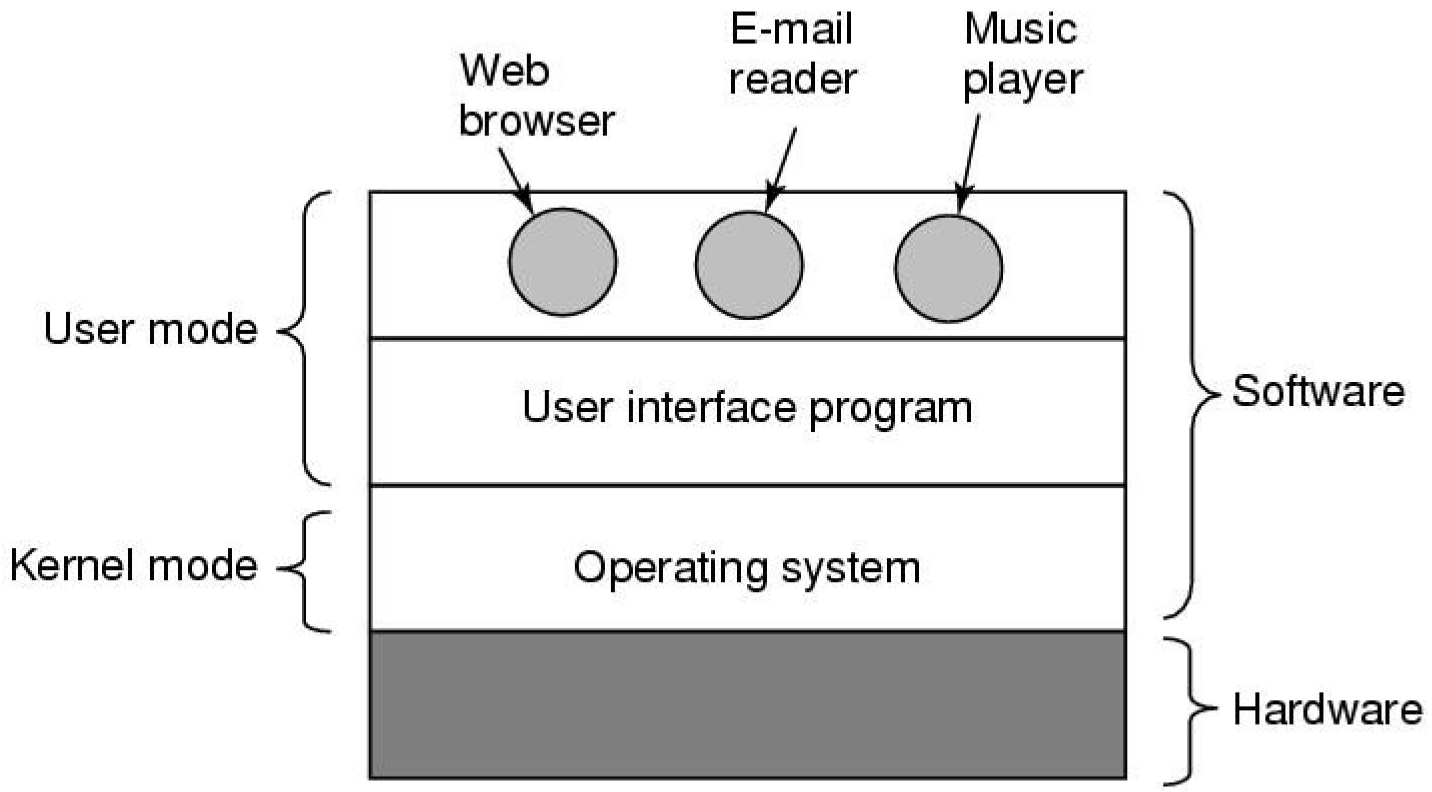


Figure 1-1. Where the operating system fits in.

What is an Operating System?

- Users use various OS
  - Windows, Linux, Mac OS etc.
- User interacts with shell or GUI
  - part of OS?
  - they use OS to get their work done
- Is device driver part of OS?

What is an Operating System?

- On top of hardware is OS
- Most computers have two modes of operation:
  - Kernel mode and user mode
  - OS runs in **kernel mode,** which has complete access to all hardware and can execute any instruction
  - Rest of software runs in user mode, which has limited capability
  - Shell or GUI is the lowest level of user mode software

What is an operating system?

- Two functions:
  - From top to down: provide application programmers a clean abstract set of resources instead of hardware ones
  - From down to top: Manage these hardware resources

The Operating System as an Extended Machine

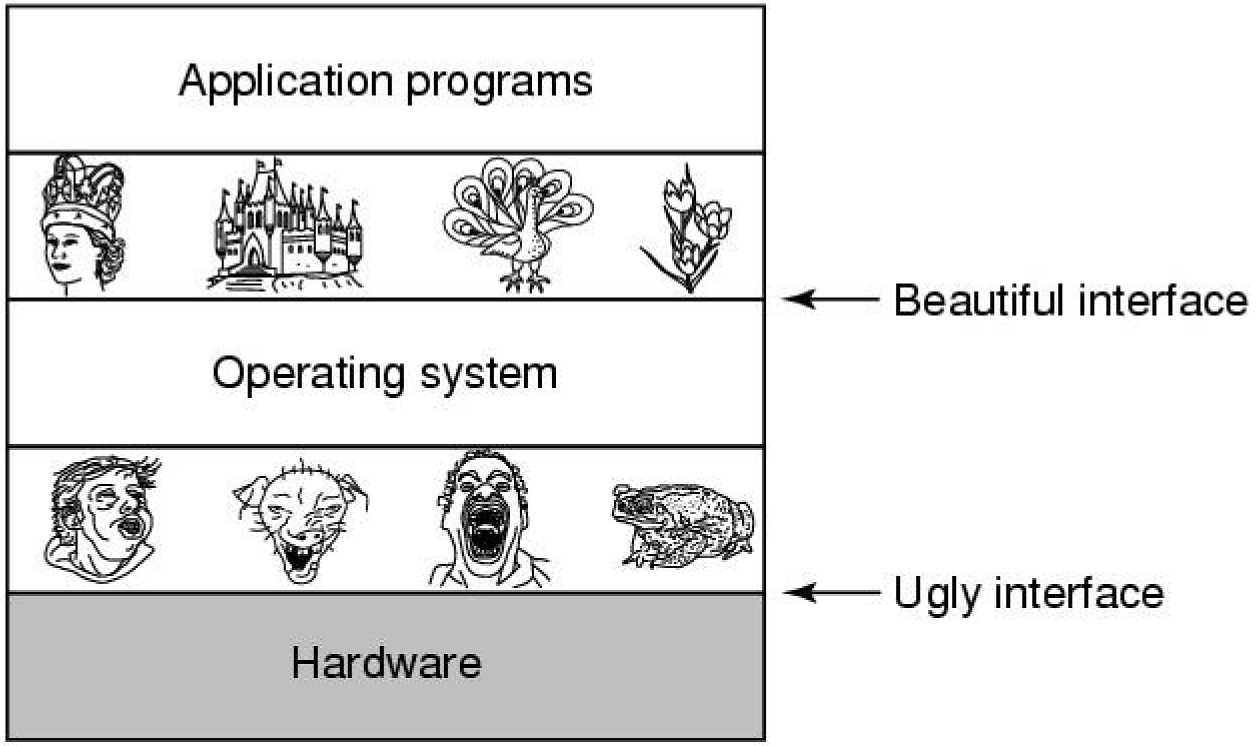


Figure 1-2. Operating systems turn ugly hardware into beautiful abstractions.

As an extended machine

- Abstraction:
  - CPU—process
  - Storage –- files
  - Memory– address space
- 4 types of people:
  - Industrial engineer: design hardware
  - Kernel designer
  - Application programmer: OS's user
  - End users

The Operating System as a Resource Manager

- Allow multiple programs to run at the same time
- Manage and protect memory, I/O devices, and other resources
- Includes multiplexing (sharing) resources in two different ways:
  - In time
  - In space

As a resource manager

- Modern OS runs multiple programs of multiple users at the same time
  - Imagine what would happen if several programs want to print at the same time?
  - How to account the resource usage of each process?
  - Resources can be multiplexed:
    - How to ensure fairness and efficiency?

summary

- Operating system is a software
  - Is a complex software
  - Runs in kernel mode
  - Manages hardware
  - Provide a friendly interface for application programmer

History of Operating Systems

Generations:

- (1945–55) Vacuum Tubes
- (1955–65) Transistors and Batch Systems
- (1965–1980) ICs and Multiprogramming
- (1980–Present) Personal Computers

1st: vacuum tubes

- Large and slow
- Engineers design, build, operate and maintain the computer
- All programming is done with machine language, or by wiring circuits using cables
- insert plugboards into the computer and operate
- The work is mainly numerical calculations

2nd: transistors and batch systems

- Also called mainframes
- Computers are managed by professional operators
- Programmers use punch card to run programs; operators operate (load compiler, etc) and collect output to the user
- Complains soon come:
  - Human Operation between computer operation
  - Lead to batch system
  - Collect a batch of jobs in the input room, then read them into a magnetic tape; the same for output

2nd: Transistors and Batch Systems (1)

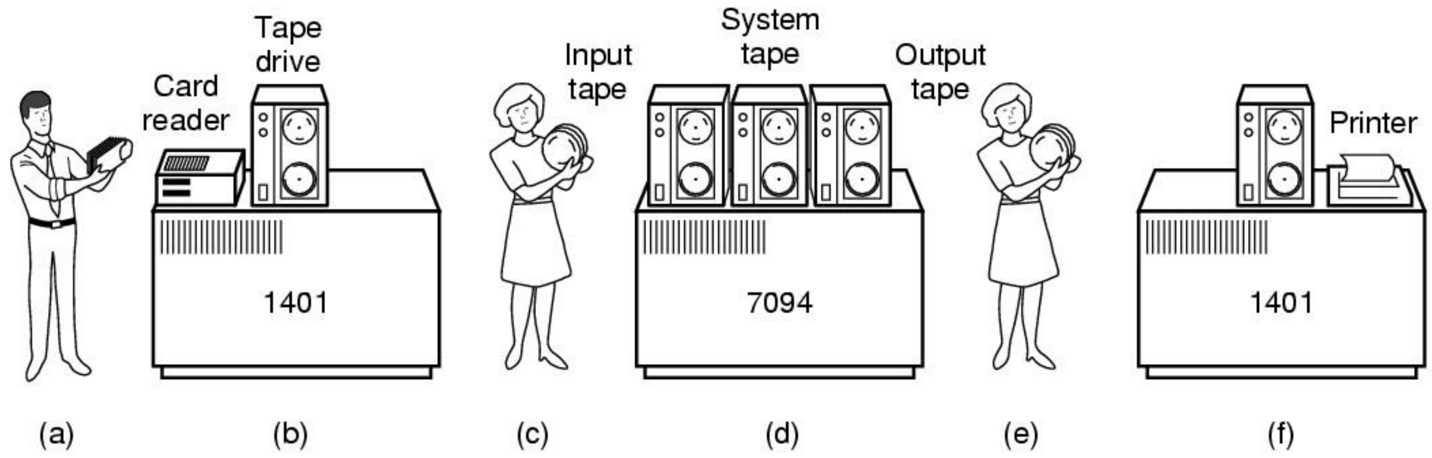


Figure 1-3. An early batch system.
(a) Programmers bring cards to 1401.
(b)1401 reads batch of jobs onto tape.

Transistors and Batch Systems (2)

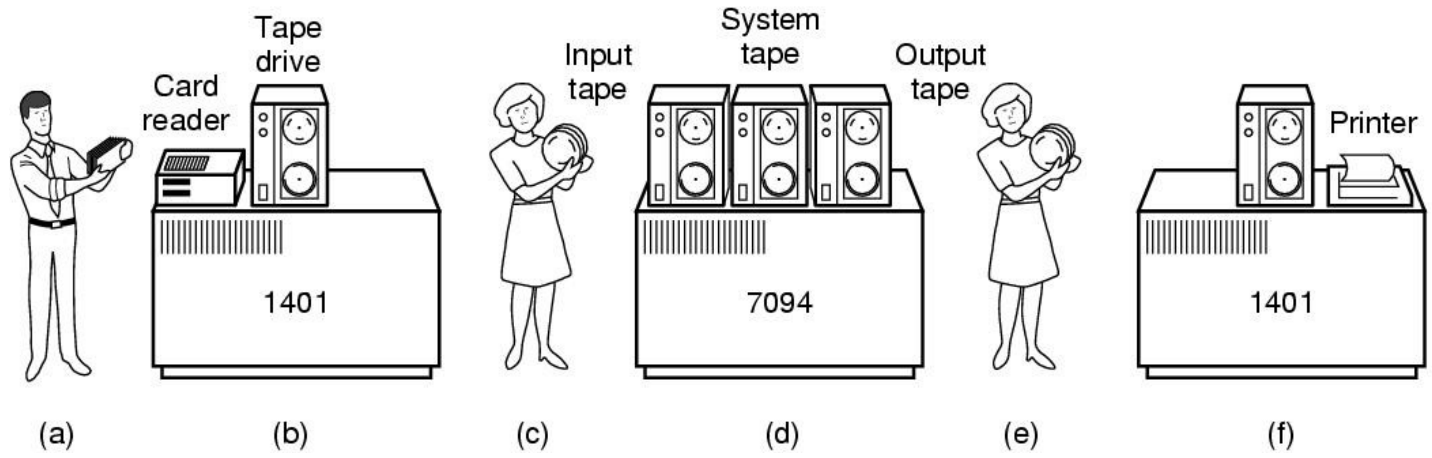


Figure 1-3. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

Transistors and Batch Systems (4)

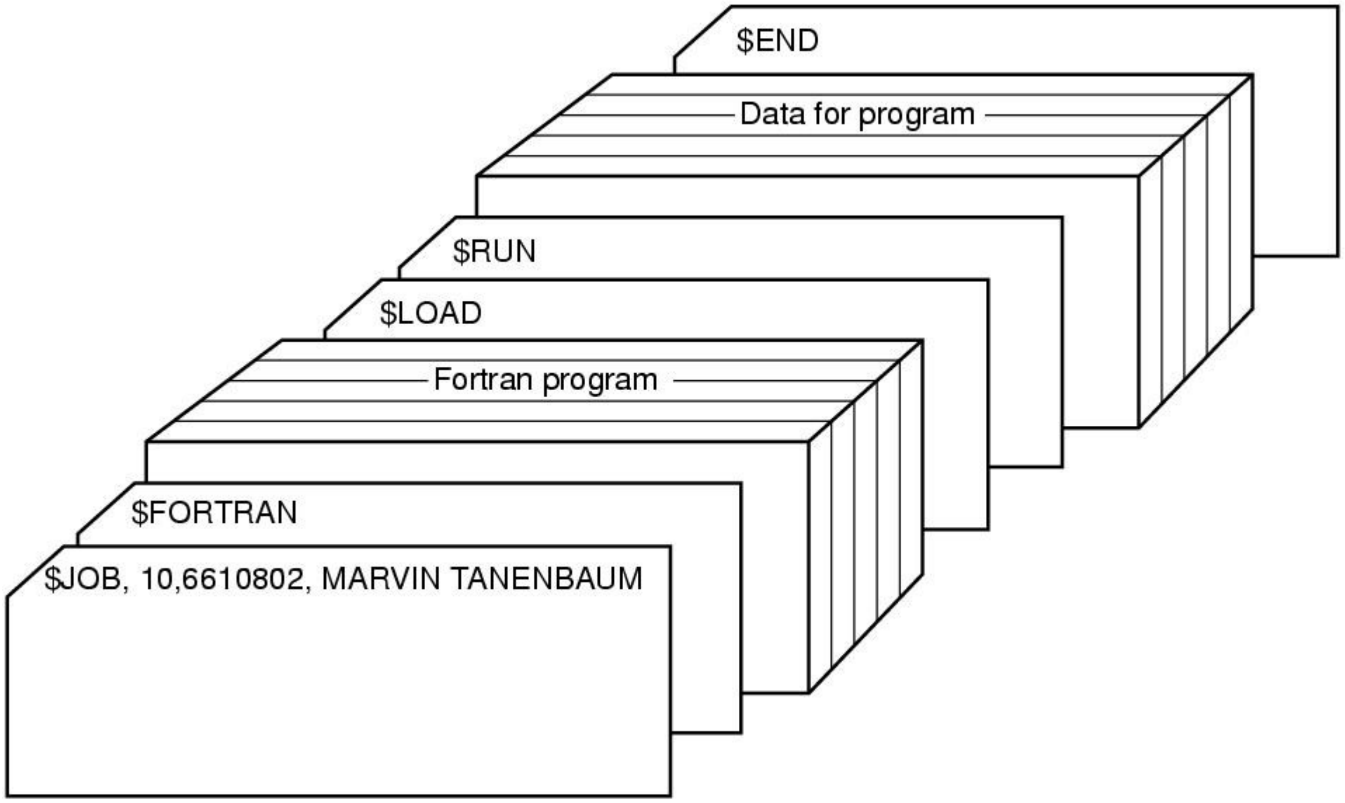


Figure 1-4. Structure of a typical FMS job.

3rd: IC and Multiprogramming

- OS/360: a dinosaur stuck in a tar pit
  - Aims to adapts 1401/7904, covers all trades of life
  - However, OS/360 introduces several key techniques
    - Multi-programming: solve the problem of CPU idling
    - Spooling: simultaneous peripheral operation on line
      - Whenever a job finishes, OS load a new job from disk to the empty-partition

3rd: ICs and Multiprogramming

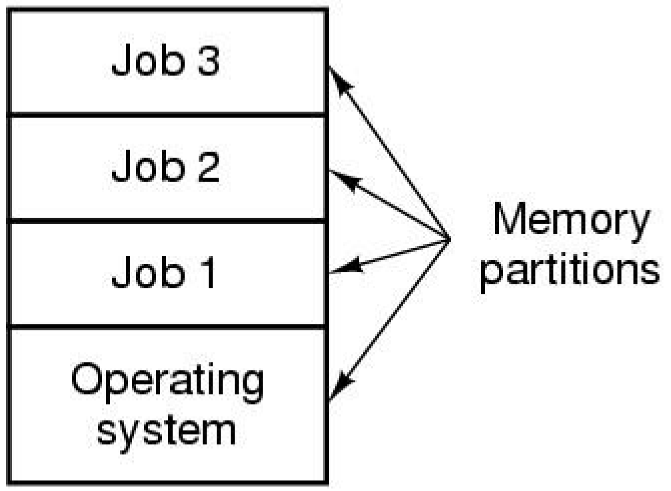


Figure 1-5. A multiprogramming system with three jobs in memory.

3rd: Ics and Multiprogramming

- Problems:
  - 3rd generation OS was well suited for big scientific calculations and massive data processing
  - But many programmers complain a lot… for not be able to debug quickly…. Why?
  - And the solution to this problem would be….?
  - Timesharing:
    - A variant of multiprogramming
    - Provide both fast interactive service but also fits big batch work

3rd: IC and Multiprogramming

- A system to be remembered: MULTICS
  - A machine that would support hundreds of simultaneous timesharing users– like the electricity system (like a web server nowadays)
  - Introduces many brilliant ideas but enjoys no commercial success
  - Its step-child is the well-known and time-honored UNIX
  - System V/ FreeBSD, MINIX, Linux

4th: personal computers

- Computers have performance similar to 3rd generation, but prices drastically different
- CP/M
  - First disk-based OS
- 1980, IBM PC, Basic Interpreter, DOS, MS-DOS
- GUI--Lisa—Apple: user friendly
- MS-DOS with GUI–Win95/98/me—winNT/xp…

Summary

- 4 generations OS
- Develops with hardware and user needs
- Multi-user, multi-programming, time-sharing

Computer Hardware Review

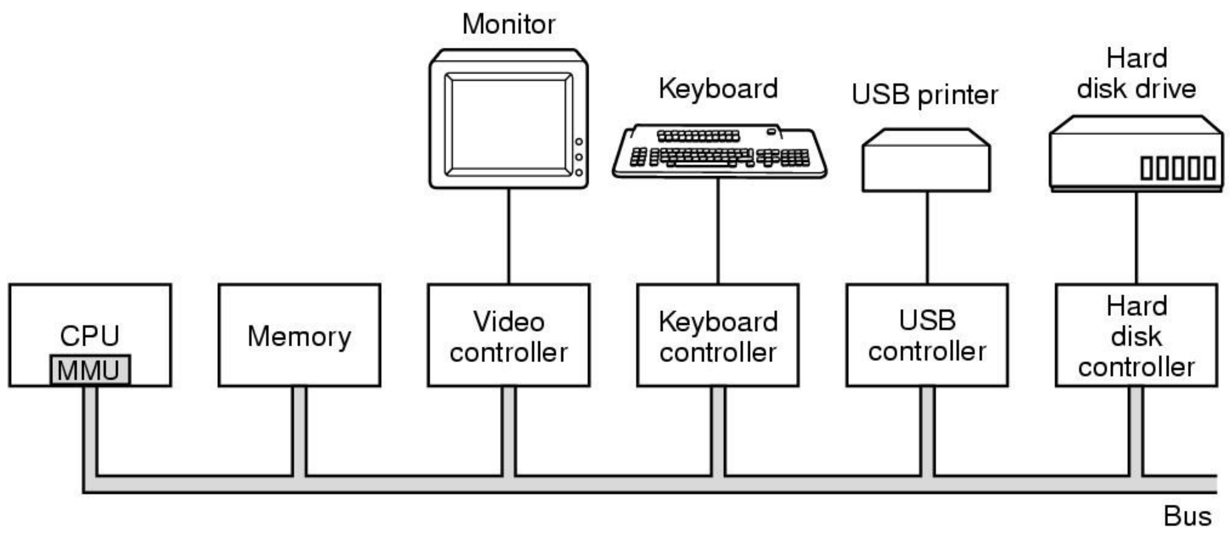


Figure 1-6. Some of the components of a simple personal computer.

Hardware: processor

- Brain of computer
  - Fetches instruction from memory and execute
  - Cycle of CPU:
    - fetch, decode, execute
  - CPU has registers to store variable and temporary result: load from memory to register; store from register to memory
  - Program counter: next instruction to fetch
  - Stack pointer: the top of the current stack
  - PSW: program status word, priority, mode…

TYPES OF OPERATING SYSTEMS

Batch Operating System:

Definition:

Batch operating systems process tasks in batches without any user interaction. Users submit jobs to the system, which are then executed in sequence without further input. These systems are efficient for processing large volumes of data without manual intervention.

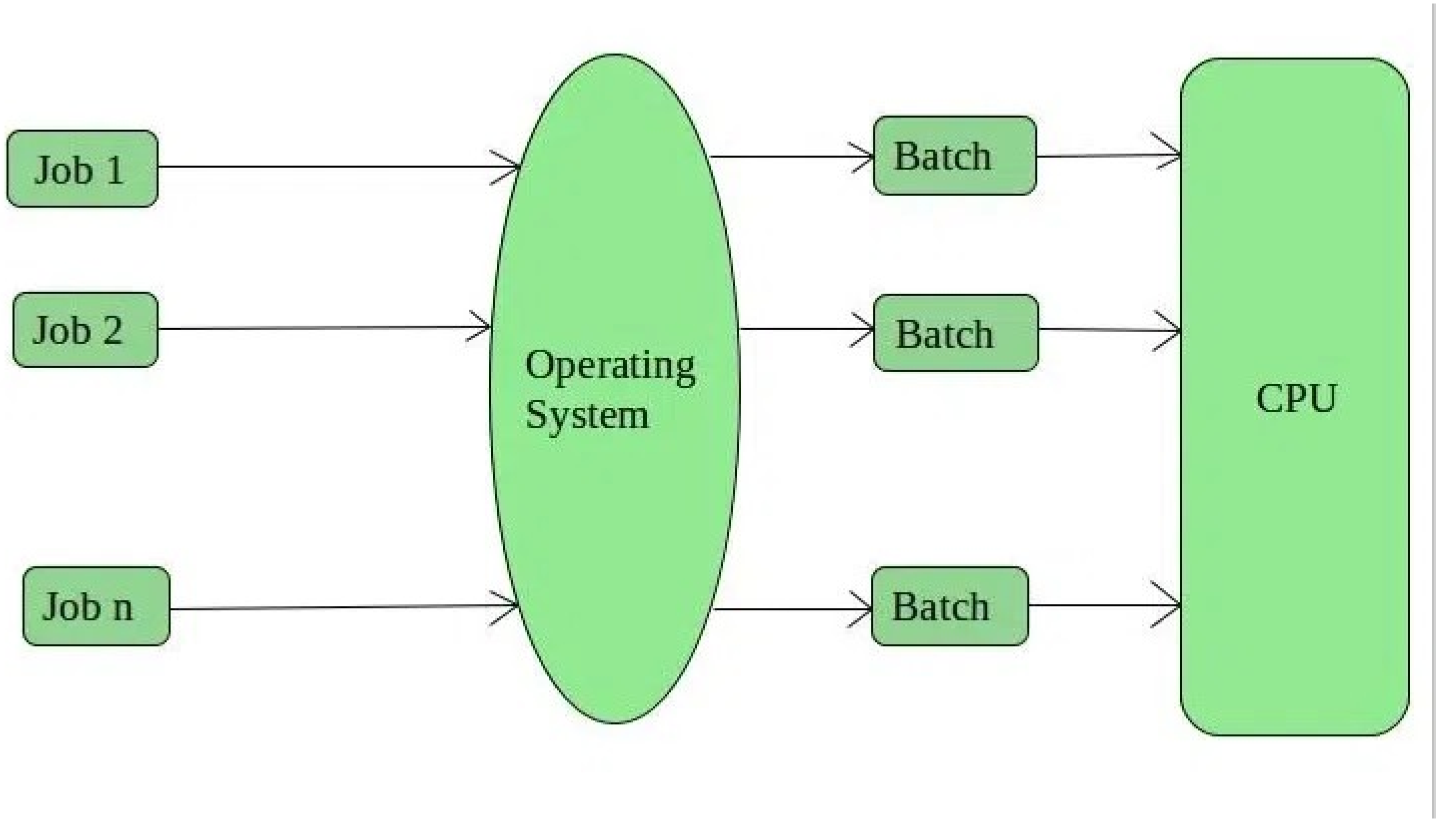
Characteristics:

Tasks are grouped into batches and processed sequentially.

Minimal user interaction; jobs run without user intervention.

Efficient for processing repetitive tasks and bulk data.

Examples include early mainframe systems like IBM OS/360.


Job 1
Job 2
Job n
Operating System
Batch
Batch
Batch
CPU

Advantages:

Efficiency: Optimizes CPU usage by processing tasks in batches, reducing idle time.

Resource Utilization: Maximizes resource utilization by running jobs without interruptions.

Automation: Allows for the automation of repetitive tasks without user intervention

Disadvantages:

Lack of Interactivity: Lacks user interaction during job execution, making it unsuitable for tasks requiring immediate feedback.

Turnaround Time: Jobs might experience delays before execution, leading to longer turnaround times.

Debugging Challenges: Debugging batch jobs can be complex due to limited real-time feedback.

Time-Sharing Operating System

Definition:

Time-sharing operating systems allow multiple users to share a computer simultaneously. The operating system divides the CPU time into small slices and allocates them to different users, providing the illusion of concurrent execution for each user.

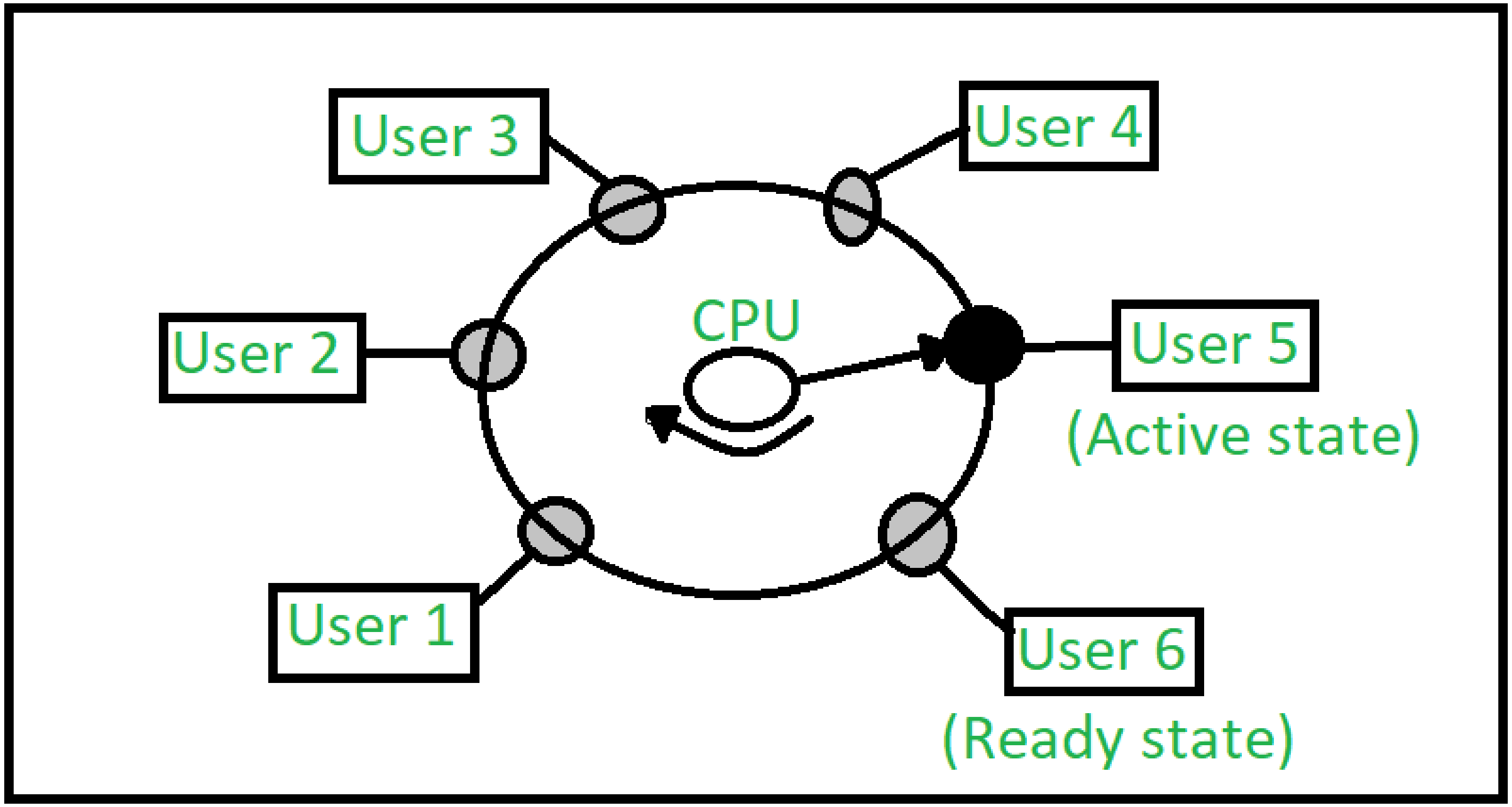
Characteristics:

Supports multi-user and multi-tasking environments.

Users interact with the system through terminals or network connections.

Provides fair and equal distribution of CPU time among users.

Examples include UNIX, Linux, and modern versions of Windows.


User 3
User 4
CPU
User 2
User 5
(Active state)
User 1
User 6
(Ready state)

Advantages:

Multi-User Support: Supports multiple users concurrently, enhancing collaboration and resource sharing.

Task Management: Efficiently manages tasks, providing quick response times to user inputs.

Fair Resource Allocation: Ensures fair allocation of system resources among users and tasks.

Disadvantages:

Security Concerns: Users sharing resources raise security challenges, requiring robust access control mechanisms.

Complexity: Managing simultaneous user interactions can lead to system complexity and potential slowdowns.

Dependency on Server: Relies on server stability for continuous operation, making it vulnerable to server failures.

Personal Operating System:

Definition:

Personal operating systems, also known as single-user operating systems, are designed for individual users on personal computers, laptops, and workstations. They provide a user-friendly interface and support a wide range of applications for personal and professional use.

Characteristics:

Tailored for single-user environments.

Provides graphical user interfaces (GUIs) for user interaction.

Supports various applications such as word processing, web browsing, and gaming.

Examples include Microsoft Windows, macOS, and popular Linux distributions (like Ubuntu) used on personal computers.

Advantages:

User-Friendly Interface: Provides intuitive graphical interfaces, enhancing user experience.

Application Support: Wide range of applications available for personal and professional use.

Customization: Users can customize the system according to their preferences and requirements.

Disadvantages:

Limited Scalability: Designed for individual users, making it challenging to scale for large-scale enterprise applications.

Compatibility Issues: Compatibility problems can arise with certain software and hardware configurations.

Security Vulnerabilities: Popular operating systems are often targeted by malware and viruses.

Real-Time Operating System (RTOS)

Definition:

Real-time operating systems are specifically designed to process data and events within strict timing constraints. They are critical in applications where precise and predictable timing is essential, such as in embedded systems, robotics, and industrial automation.

Characteristics:

Guarantees timely and deterministic responses to events.

Supports both hard real-time (strict deadlines) and soft real-time (soft deadlines) requirements.

Used in applications where immediate and accurate responses to stimuli are crucial.

Examples include VxWorks, FreeRTOS, and RTLinux.

Advantages:

Deterministic Responses: Provides precise and predictable response times, crucial for time-sensitive applications.

Reliability: Ensures tasks are executed within specified timeframes, enhancing reliability in critical systems.

Optimized Performance: Optimized for specific hardware, maximizing performance for dedicated tasks.

Disadvantages:

Complex Development: Designing real-time systems can be complex and requires meticulous planning.

Limited Flexibility: Not easily adaptable to changing tasks or requirements due to stringent timing constraints.

Cost: Can be more expensive due to specialized hardware and software requirements.

Parallel Operating System:

Parallel Operating System:

Definition: Parallel operating systems are designed to manage parallel processing, where multiple processors work together to solve a problem. They are common in high-performance computing environments and supercomputers.

Characteristics:

Harnesses the power of multiple processors for simultaneous processing.

Efficient for tasks that can be divided into smaller sub-tasks and processed independently.

Used in scientific simulations, weather forecasting, and other computationally intensive tasks.

Examples include Parallel versions of UNIX, Linux clusters, and specialized parallel computing systems.

Advantages:

High Performance: Achieves high processing speeds by dividing tasks among multiple processors.

Scalability: Scales effectively with the addition of more processors, increasing computational power.

Parallel Processing: Efficiently handles tasks that can be divided into smaller parallel sub-tasks.

Disadvantages:

Complex Programming: Parallel programming can be intricate and challenging, requiring synchronization and communication between processors.

Limited Applicability: Not all tasks can be parallelized, limiting the scope of its application.

Cost: Setting up and maintaining parallel systems can be costly due to specialized hardware requirements.

Distributed Operating System

Definition:

Distributed operating systems run on multiple machines and allow them to work together as a single system. They manage resources across a network, providing features like transparency, reliability, and scalability.

Characteristics:

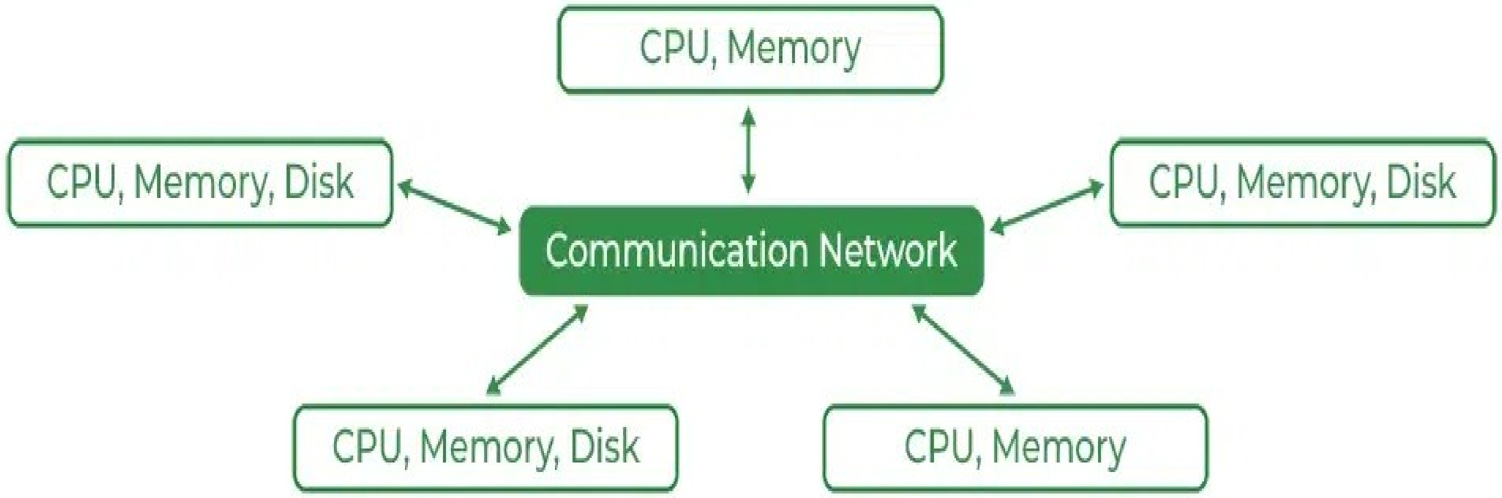
Resources such as files, printers, and processing power are shared across a network.

Enables collaboration and resource sharing among geographically dispersed users.

Provides fault tolerance and load balancing.

Examples include Google's Chrome OS, various cloud computing platforms, and distributed versions of UNIX and Linux.

Architecture of Distributed OS



Advantages:

Resource Sharing: Enables efficient sharing of resources such as files, printers, and processing power across a network.

Reliability: Offers fault tolerance as tasks can be redistributed if one node fails, ensuring system stability.

Scalability: Scales seamlessly with network expansion, accommodating growing demands.

Disadvantages:

Complex Management: Managing distributed resources and ensuring consistency can be complex.

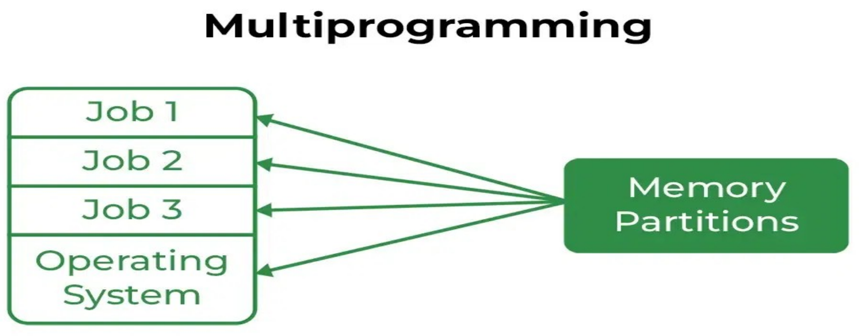
Network Dependency: Relies heavily on network stability; network failures can disrupt operations.

Security Challenges: Ensuring secure communication and data integrity across distributed nodes is challenging.

Multi-Programming Operating System:

Definition:

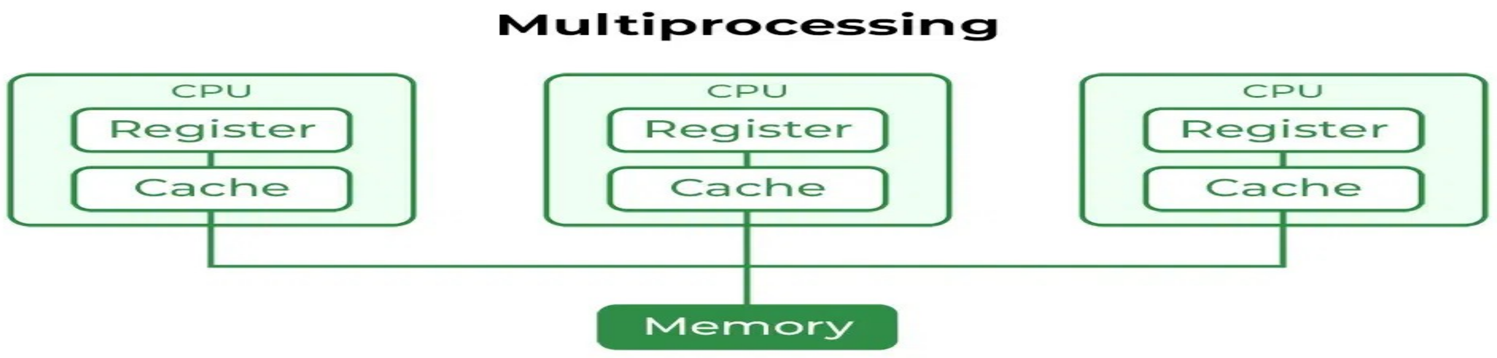
A multi-programming operating system allows multiple programs to run simultaneously on a computer. It achieves this by keeping several programs in memory at once, and the central processing unit (CPU) is multiplexed among them. This type of operating system aims to maximize CPU utilization and increase overall system efficiency by overlapping the execution of programs.



Multi-Processing Operating System:

Definition:

A multi-processing operating system is designed to support the execution of multiple processes or tasks on multiple central processing units (CPUs) within a single computer system. Unlike multi-programming systems that manage multiple tasks, multi-processing systems specifically focus on distributing tasks across multiple processors, aiming to enhance overall performance and handle complex computations.


Multiprocessing
CPU
Register
Cache
CPU
Register
Cache
CPU
Register
Cache
Memory