Chapter: Introduction & Operating-System Structures

NARZU TARANNUM (NTR)
DEPT. OF CSE, BRAC UNIVERSITY

Information of Course Teachers

- Narzu Tarannum <narzu.tarannum@bracu.ac.bd
- UB80813

Recommended Text book

- Silberschatz, Peter B. Galvin, Greg Gagne, "Operating System Concepts", Wiley; 9th edition (2009) ISBN: 978-1-118-06333-0

Course Outline

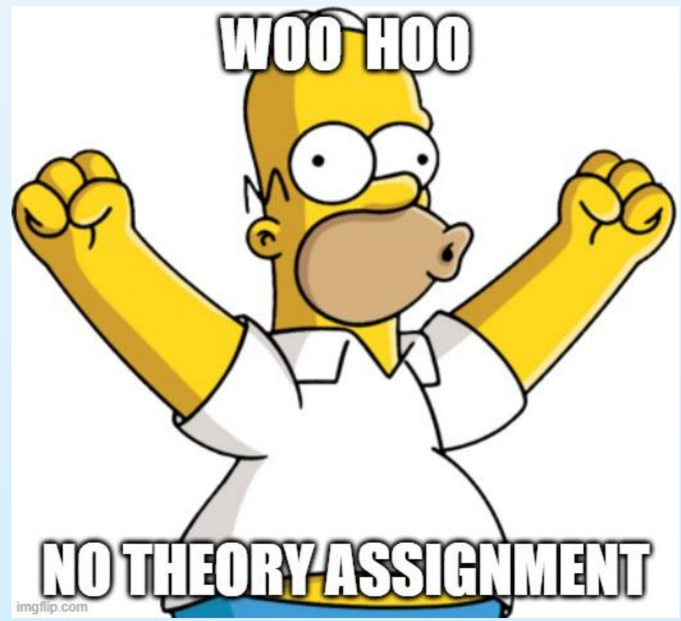
- Operating Systems Overview and Structures
- Processes Management
- CPU Scheduling
- Threads
- Process Synchronization
- Deadlocks
- Memory Management/ Main memory
- Virtual Memory
- File-System Interface

Objectives

- **To understand** the fundamental concepts of computer system organization and the structure of operating systems.
- **To explore** various aspects of process management in operating system
- **To know** how different CPU scheduling algorithm works and their respective importance
- **To develop practical knowledge on** the concept of threads
- **To inspect** process synchronization mechanisms and deadlocks
- **To be able to analyze** the management of main and virtual memory

Marks Distribution

- Theory – 80%
  - Class participation – 5%
  - Quiz – 15%
  - Mid – 25%
  - Final – 35%
- Lab – 20%



Content of this chapter

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Storage Structure
- System call
- Protection and Security
- Operating systems Services

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier and make the computer system convenient to use.
  - Use the computer hardware in an efficient manner.

What does an Operating Systems?

It works between users and computer hardware.

- Resource allocator – manages and allocates resources.
- Control program – controls the execution of user programs and operations of I/O devices .
- Kernel – the one program running at all times (all else being application programs).
  - The **kernel** is the central module of an **operating system** (**OS**). It is the part of the **operating system** that loads first, and it remains in main memory. The **kernel** code is usually loaded into a protected area of memory to prevent it from being overwritten by programs or other parts of the **operating system**.

Primary Functions of an OS

- Processes-management
- Storage-memory management
- Data-file management
- Input/output devices-i/o management
- Network management
- Protection& security

Software

The two most common types of software are :

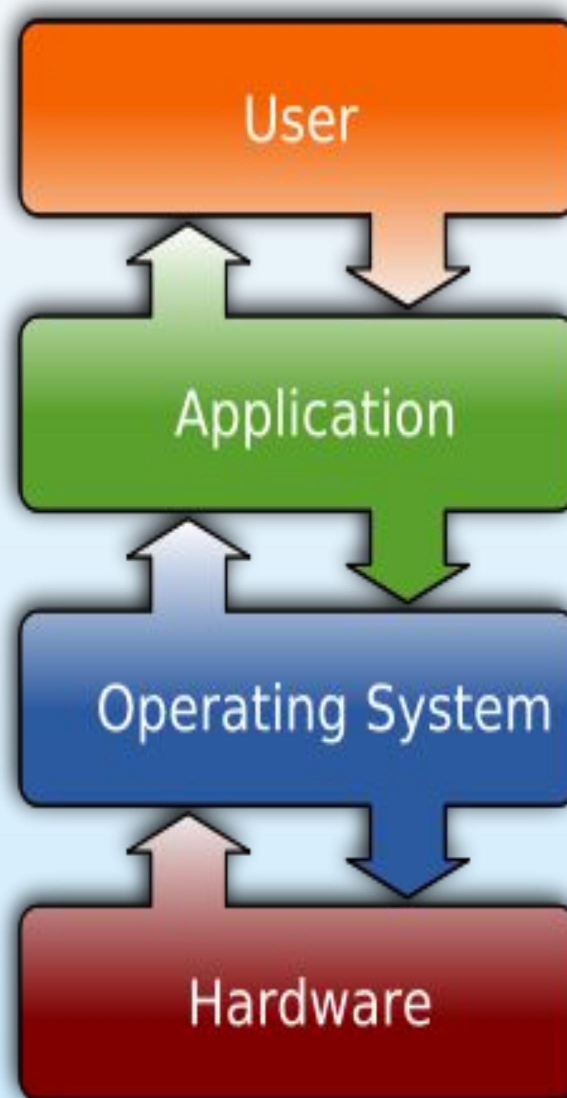
- System software
- Application software.

What is System Software?

System Software refers to the operating system and all utility programs that manage computer resources at a low level. Systems software includes compilers, loaders, linkers, and debuggers.

What is Application Software?

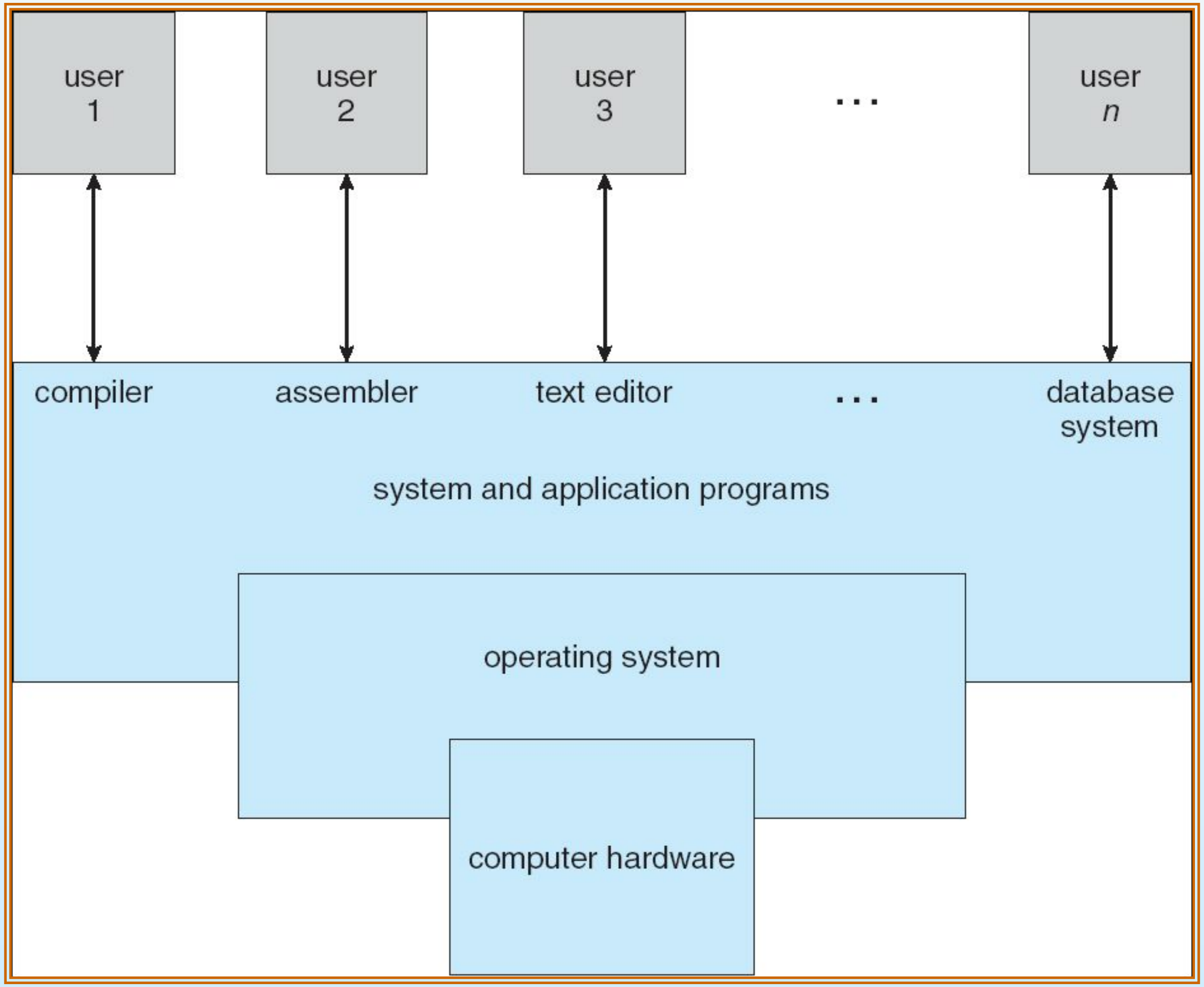
Applications software comprises programs designed for an end user, such as word processors, database systems, and spreadsheet programs.



Four Components of a Computer System

- Computer system can be divided into <u>four</u> components
  - **Hardware** – provides basic computing resources
    - CPU, memory, I/O devices
  - **Operating system**
    - Controls and coordinates use of hardware among various applications and users
  - **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers

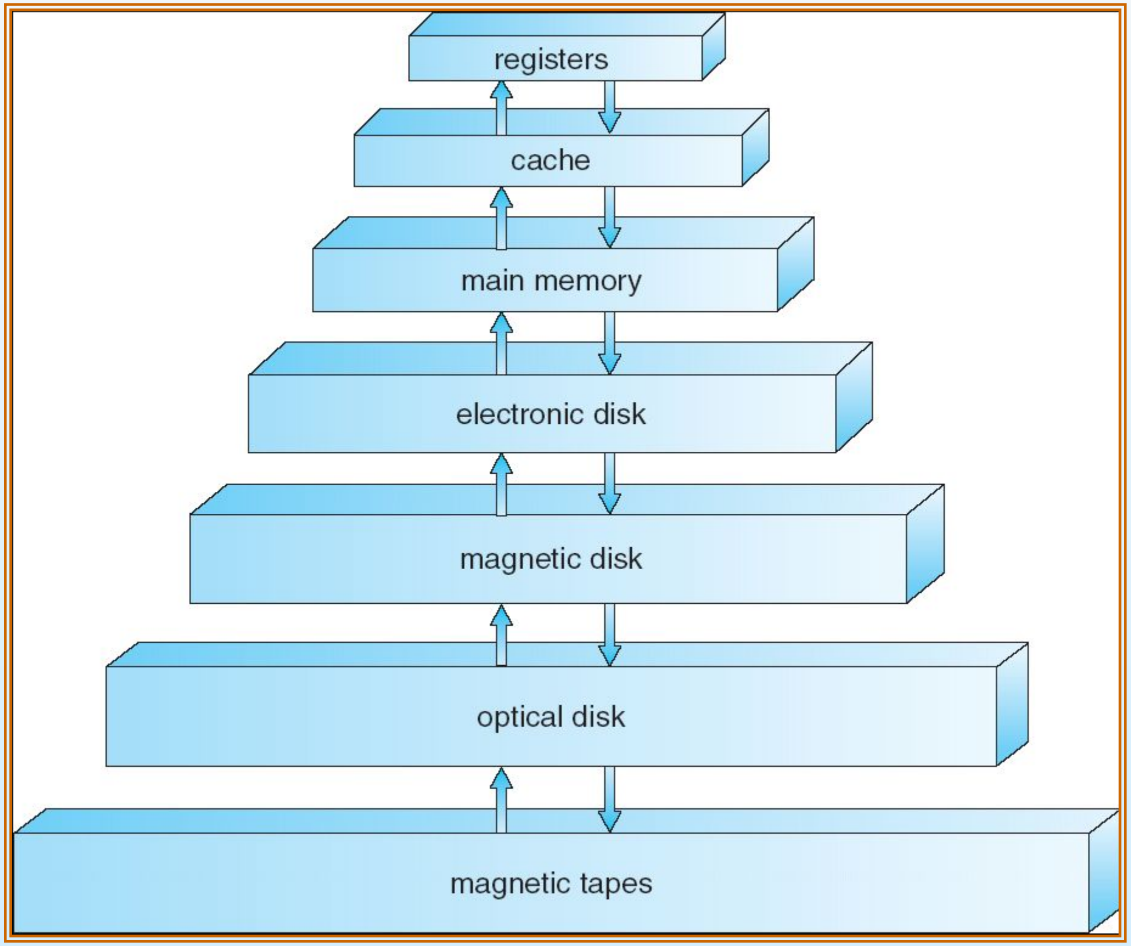
Four Components of a Computer System



Computer Startup

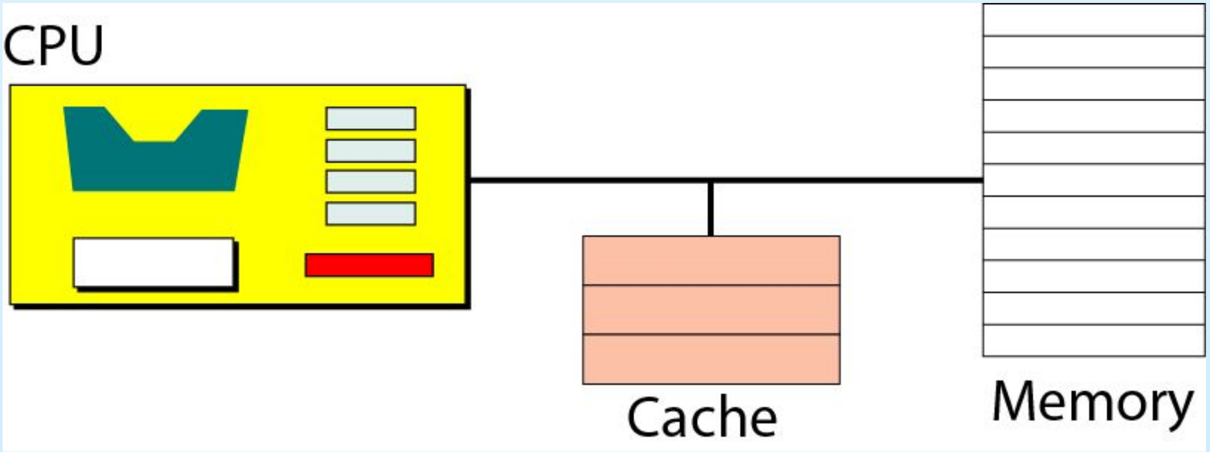
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Storage-Device Hierarchy



Storage Hierarchy

- Storage systems organized in **hierarchy**.
  - Size
  - Speed (Access time)
  - Cost
  - Volatility
- *Caching* – Information in use copied from slower to faster storage temporarily



Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

TO UNDERSTAND WHAT OS ARE, WE MUST FIRST UNDERSTAND HOW THEY HAVE DEVELOPED.

History of OS / Types of OS

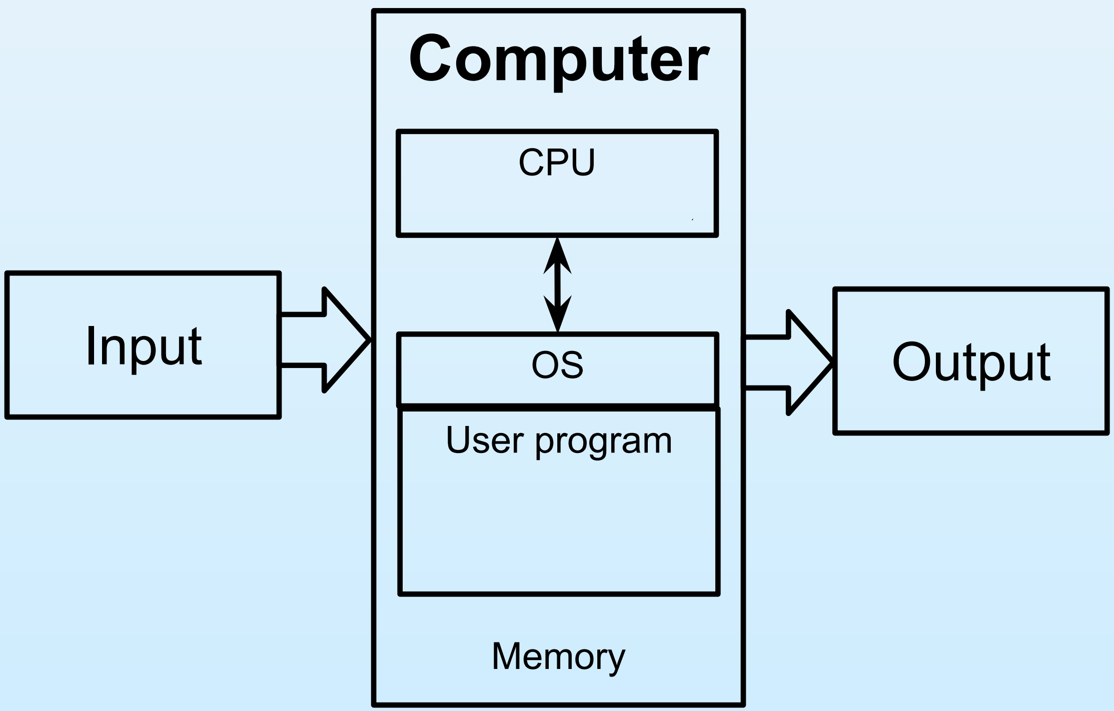
- Simple Batch Systems
- Multiprogramming Systems
- Time-Sharing Systems
- Parallel Systems
- Distributed Systems
- Real -Time Systems

History of OS/ Types of OS

- **Simple Batch Systems:** Automatically transfers control from one job to another.
- **Multi-programmed Batched Systems:** Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.
- **Time Sharing Systems:** Logical extension of multiprogramming.
- **Parallel Systems:** Multiprocessor systems with more than one CPU
- **Distributed Systems:** Distribute the computation among several physical processors.
  - **Loosely coupled system** – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- **Real-time Systems:** Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.

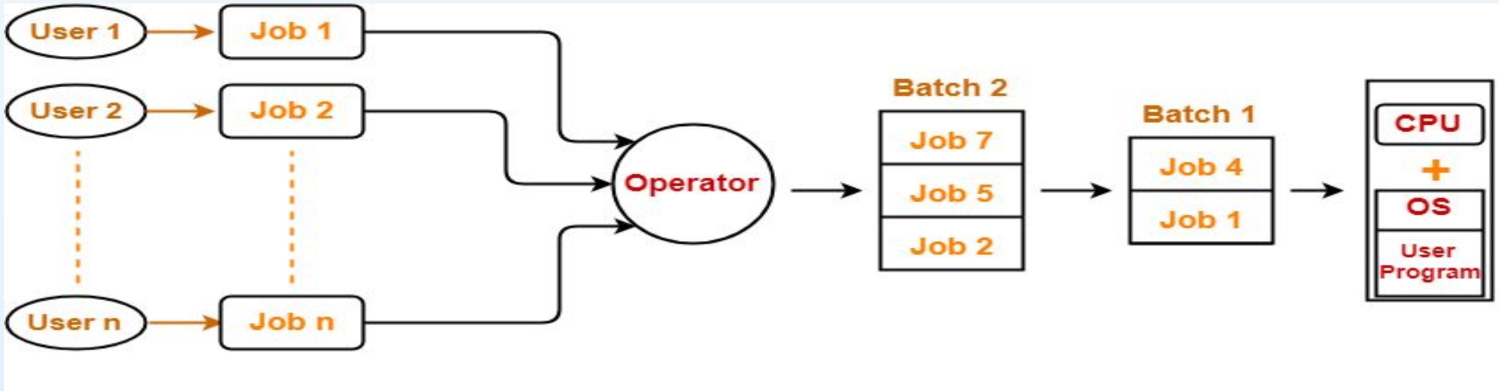
Simple Batch Systems

- Automatically transfers control from one job to another.
- Common input-output device are card readers and tape drivers.
- User prepare a job which consisted of program, input data and control instructions.



Simple Batch Systems

- User prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group.



Batch Operating System

<u>Advantages-</u>

- It saves the time that was being wasted earlier for each individual process in context switching from one environment to another environment.
- No manual intervention is needed.

<u>Disadvantages</u>

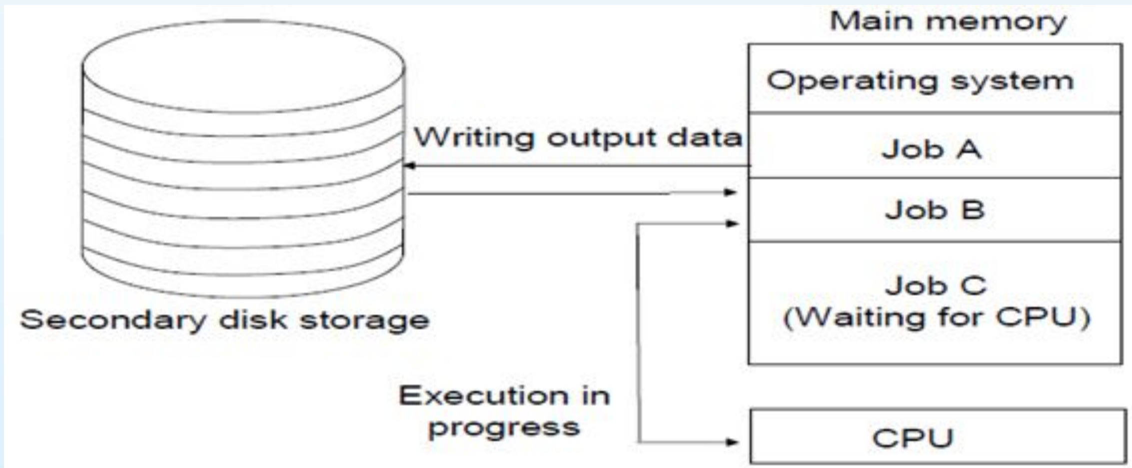
- Executing a series of non-interactive jobs all at one time.
- The output is obtained only after all the jobs are executed.
- Thus, priority can not be implemented if a certain job has to be executed on an urgent basis.

Limitations of Simple Batch Systems

- Not interactive
- Memory is limited
- CPU utilization is poor
- Speed mismatch between I/O device and CPU

Multi-programmed Operating Systems

- Multiprogramming needed for **efficiency** or maximize **CPU utilization**.
- Multiprogramming means more than one process in main memory which are ready to execute.



- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- Process generally require **CPU time** and **I/O time**, when CPU has to wait (for I/O for example), OS switches to another job and this idea will continue.

Multi-programmed Operating System

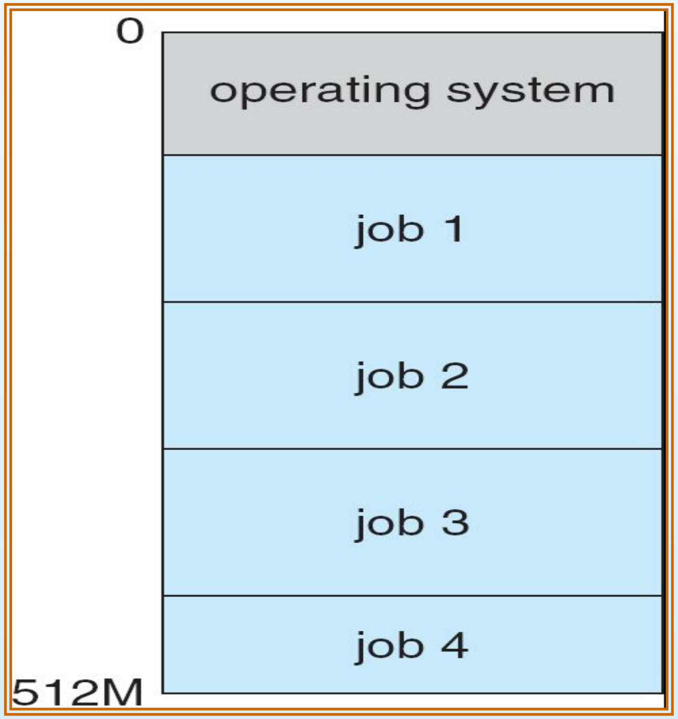
- Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

Advantages

- High CPU utilization.
- It appears that many programs are allotted CPU almost concurrently.
- Response time is shorter

Disadvantages

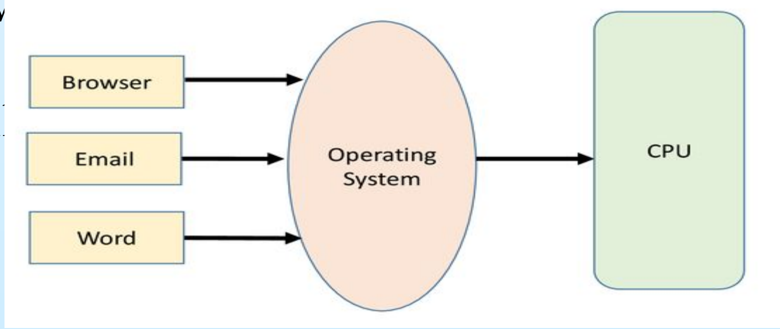
- CPU scheduling is required.
- To accommodate several jobs in memory, memory management is essential.



- **Multi-programmed systems provide an environment in which the various system resources(like CPU, memory and peripheral devices) are utilized effectively but they do not provide for user interaction with the computer system.**

Multitasking/Time-Sharing Systems–Interactive Computing

- Multitasking is multiprogramming with time sharing where CPU executes multiple jobs by switching among them.
- Timesharing (multitasking) is logical extension of multiprogramming, in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing.
- The **operating system** is able to keep track of where you are in these tasks and go from one to the other without losing information.
- Response time should be < 1 second
- Time sharing requires an interactive(or hands-on)computer system, which provides direct communication betw
- A time-shared OS al imultaneously.



Multitasking/Time-Sharing Systems–Interactive Computing

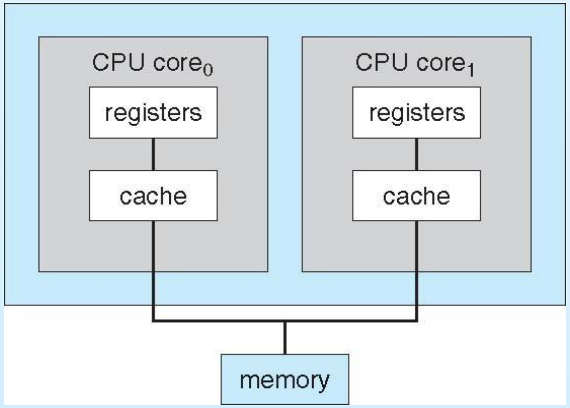
- Each user has at least one separate program for executing in memory.
- A program loaded into memory and executing is called a ⇨process
- ⇨ Job scheduling
- ⇨ CPU scheduling
- If processes don't fit in memory, **swapping** moves them in and out to run from main memory achieving this goal is virtual memory.

Computer-system Architecture

- **Single-processor Systems:** If there is only one general purpose CPU, then the system is a single-processor system.
  - Other special purpose processors are also present which perform device specific task.

Computer-system Architecture

- **Multiprocessor Systems/Parallel System:** Multiprocessor systems with more than one CPU in close communication.
  - **Tightly Coupled System:** processors share memory and a clock; communication usually takes place through the shared memory.
    - HAVE THREE MAIN ADVANTAGES
      - INCREASED THROUGHPUT
      - ECONOMY OF SCALE
      - INCREASED RELIABILITY

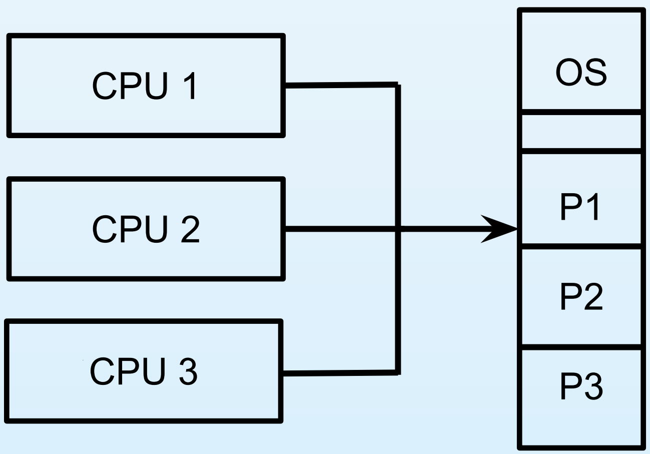


Computer-system Architecture

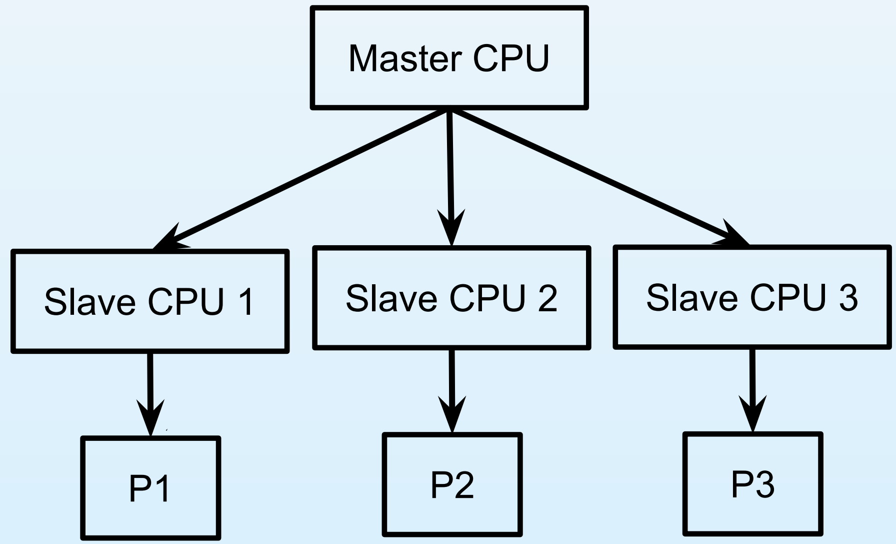
- **Multiprocessor Systems/Parallel System are of two types:**
  - Symmetric Multiprocessing(SMP):
    - Each processor runs an identical copy of the operating system.
    - Most modern operating systems support SMP.
  - Asymmetric Multiprocessing:
    - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors

Computer-system Architecture

- **Symmetric Multiprocessing Architecture Example**



- **Asymmetric Multiprocessing Architecture Example**



Computer-system Architecture

Clustered Systems:

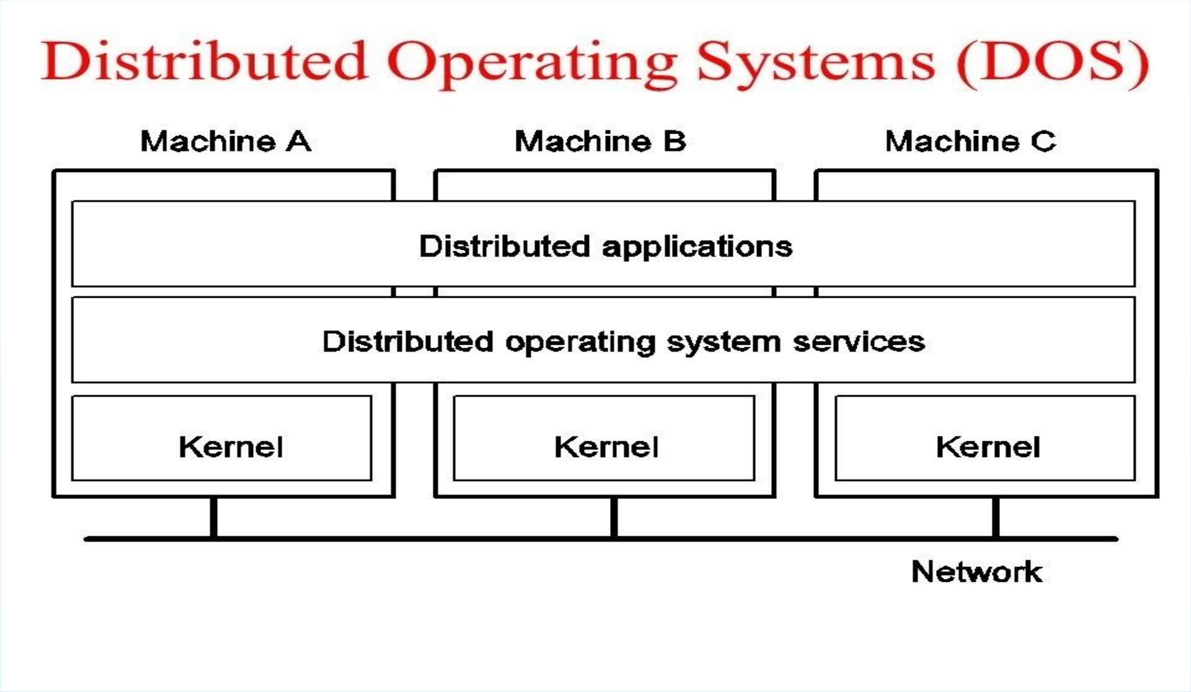
- Cluster computing is the process of sharing the computation tasks among multiple computers and those computers or machines form the cluster. It works on the distributed system with the networks. Several types of cluster computing are used based upon the business implementations, performance optimization, and architectural preference such as load balancing clusters, high availability (HA) clusters, high performance (HP) clusters. Some of the advantages are processing speed, cost efficiency, scalability, high availability of resources. Some of the popular implementations of cluster computing are Google search engine, Earthquake Simulation,
- They are composed of two or more individuals systems coupled together share common storage and are closely linked through local area network.
- Can be structured asymmetric or symmetric.

Distributed Systems and

- **A Distributed Operating Systems** is the software over a collection of independent, networked, communicating and physically separate **computational nodes.** Distributed operating system is a model where the distributed applications running on networked computers communicate and coordinate their action by simply passing the messages. The application on this networked computer work towards achieving common goals.

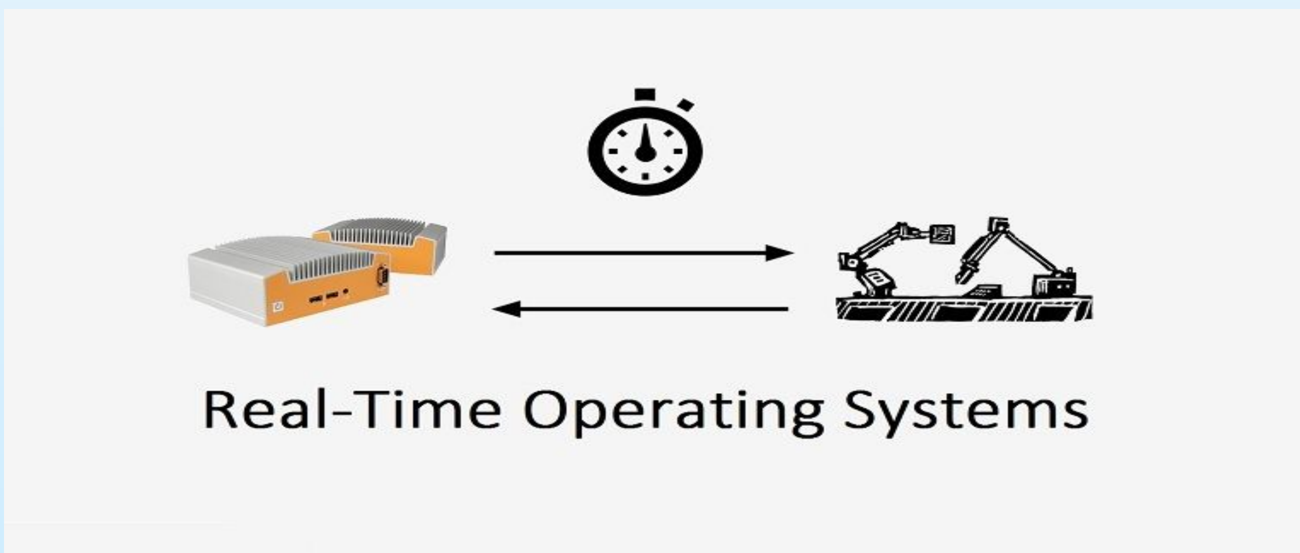
- Load balancing, computation speed up, Remote data access possible.

- Goals are: connecting user and resources, transparency, reliability.

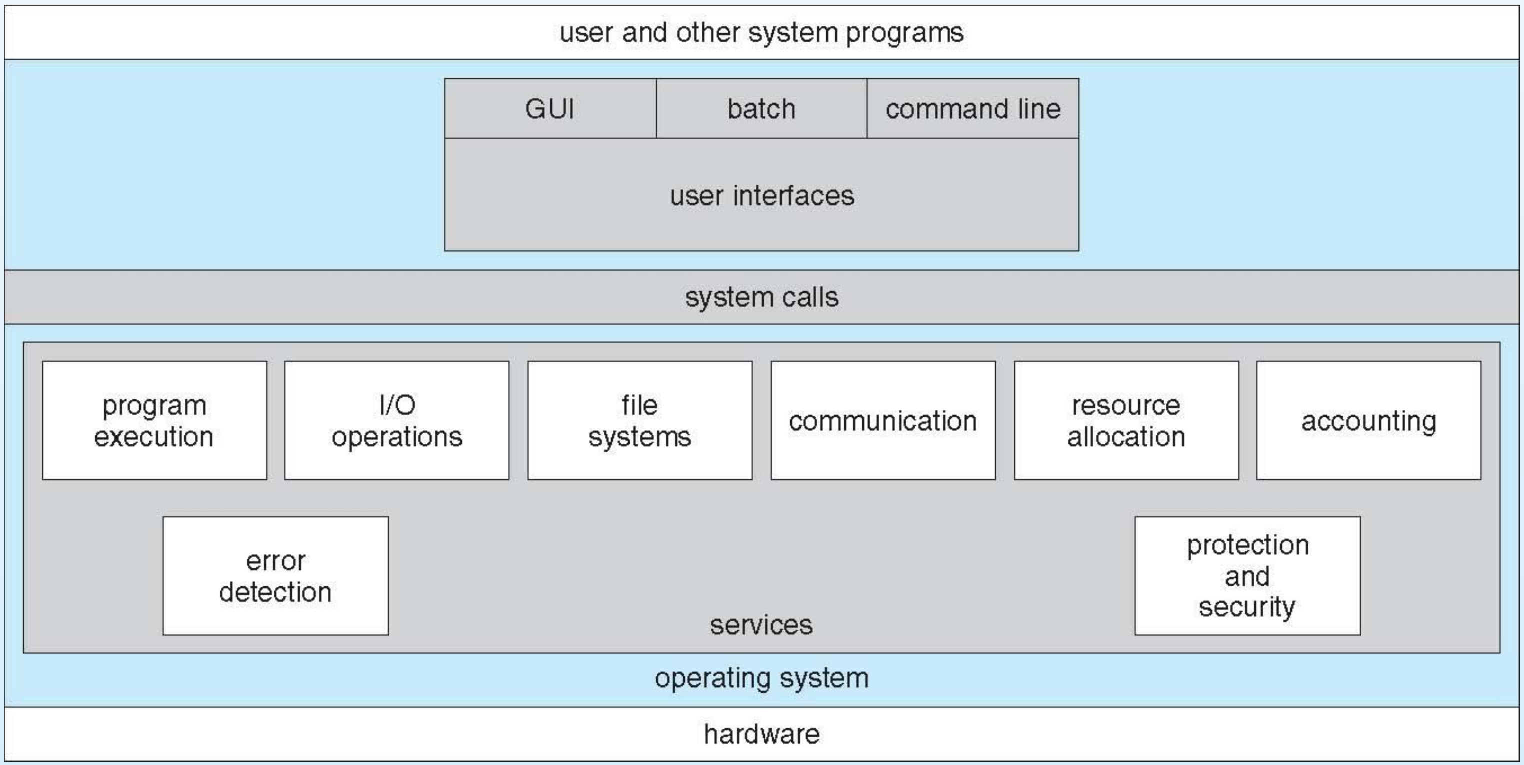


Real-time Systems

- The RTOS is an operating system, it is a brain of the real-time system and its response to inputs immediately. In the RTOS, the task will be completed by the specified time and its responses in a predictable way to unpredictable events.
- **Real-time Systems:** Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems. Mostly used in real time application.



A View of Operating System Services



Operating System Services

- Operating systems provide an environment for execution of programs and services to programs and users
- One set of operating-system services provides functions that are helpful to the user:
  - **User interface** - Almost all operating systems have a user interface (**UI**).
    - Varies between **Command-Line** (**CLI**), **Graphics User Interface** (**GUI**), **Batch interface-** in which commands and directives to control those commands are entered into files, and those files are executed.
  - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device

Operating System Services (Cont.)

- One set of operating-system services provides functions that are helpful to the user (Cont.):
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
  - **Communications** – Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - **Error detection** – OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Operating System Services (Cont.)

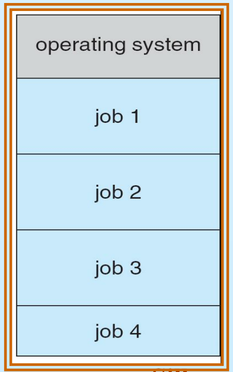
- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources -   CPU cycles, main memory, file storage, I/O devices.
  - **Accounting -** To keep track of which users use how much and what kinds of computer resources
  - **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

Operating-System Operations

- **An Operating system is Interrupt driven**
  - **Hardware interrupt**
    - Occurs by one of the devices
  - **Software interrupt** (exception or trap)
    - Request for operating system service
    - Software error (e.g., division by zero)
    - A *trap* is a software-generated interrupt caused either by an error or a user request.

Operating-System operations-protection

- We know modern operating systems are interrupt driven.
- **Hardware** generates interrupt.
  - Many errors detected by hardware can be handled by OS.
- **Software** error handled by **exception** or **trap.**
  - A trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service need to be performed.
- With sharing many processes could be adversely affected by a bug in one program. Since the **operating system** and the **users program** share the **hardware** and **software** resources of the computer system, a properly designed OS must ensure that an incorrect program can not run and also can not cause other programs to execute incorrectly.



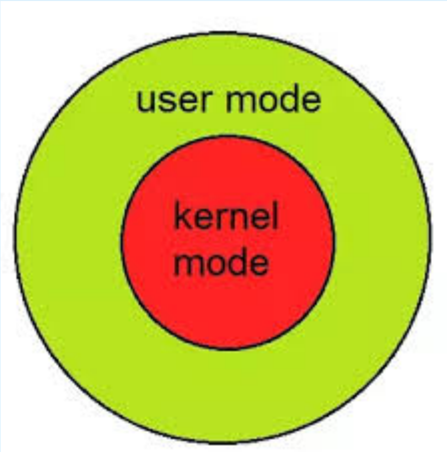
Operating-System operations-protection

- In order to ensure the proper execution of operating system, we must be able to distinguish between the execution of the operating system code and user defined code.

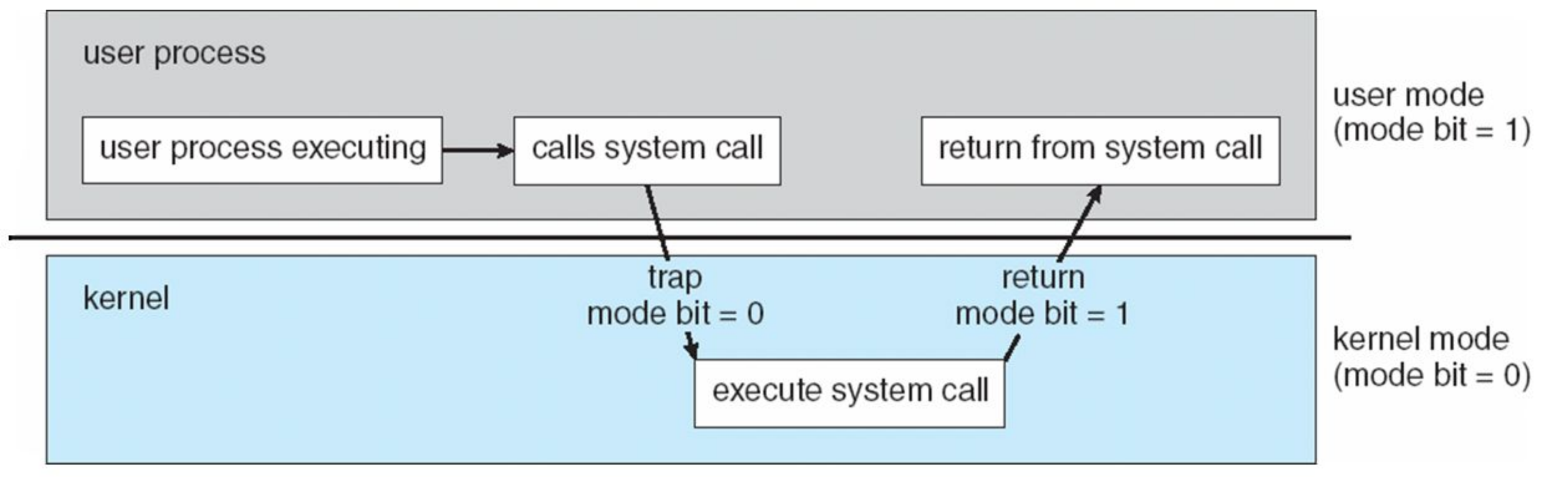
Hardware Protection

- The approach taken by most computer system is to provide hardware support that allows us to differentiate among various modes of execution.

- **Dual-mode operation** allows OS to protect itself and other system components.
  - **User mode (1)** and **Kernel/Monitor/System mode (0)**
  - **Mode bit** provided by hardware.
    - Provides ability to distinguish when system is running user code or system code
    - Some instructions designated as **privileged**, only executable in system mode
    - System call changes mode to kernel, return from call resets it to user



Transition from User to Kernel Mode



- With the mode bit, we can distinguish between a task that is executed on behalf of the operating system that is ‘zero’ and ‘one’ is executed on behalf of the user. When the computer system is executing on behalf of a user application , the system is in user mode .

- when a user application requests a service from the operating system (via a system call), the system must transition from user to kernel mode to fulfill the request. This is shown in his figure.

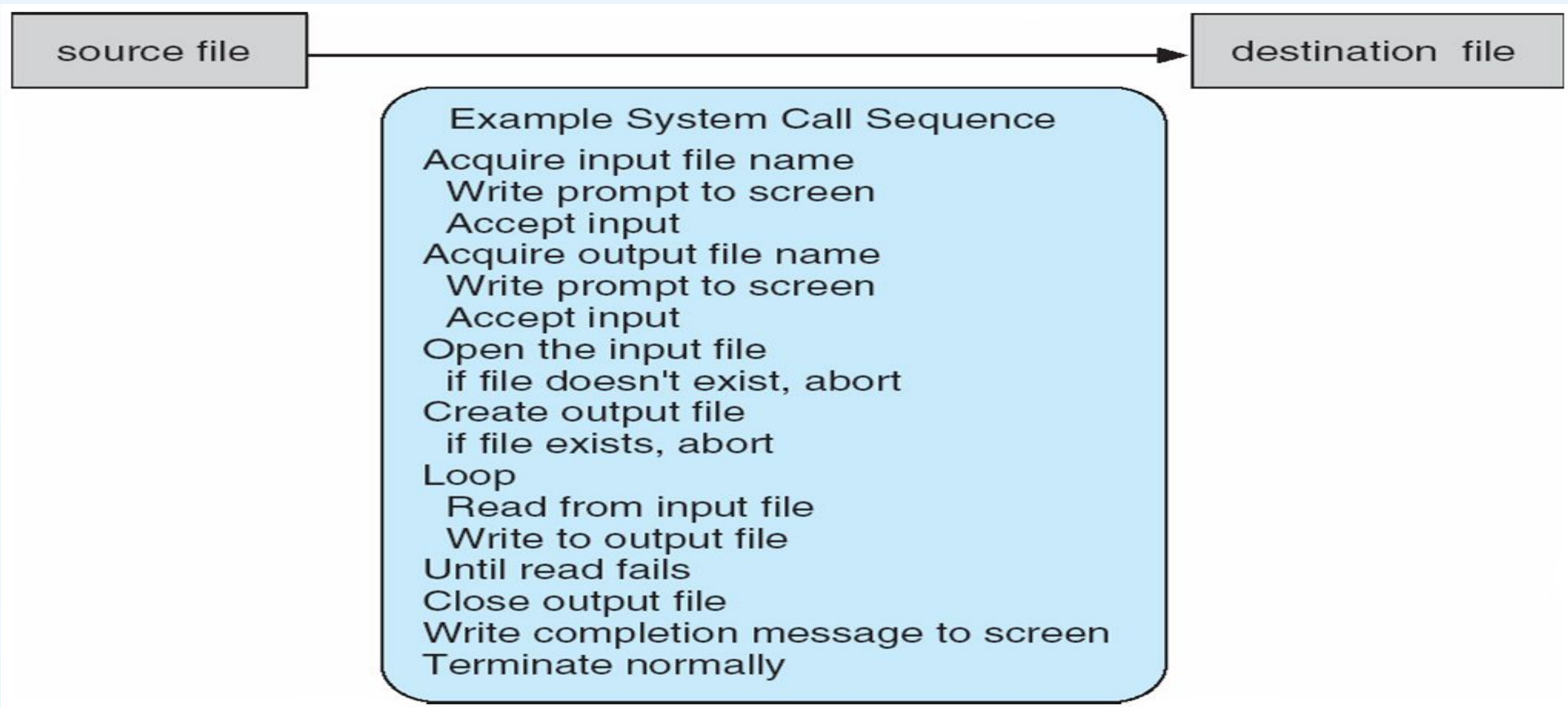
System Calls

- A **system call** is a way for programs to interact with the **operating system**.
- A computer program makes a **system call** when it makes a request to the **operating system's** kernel. **System call** provides the services of the **operating system** to the user programs via Application Program Interface(API).
- Programming interface to the services provided by the OS
- Routines typically written in a high-level language (C or C++)

❖ Note that the system-call names used throughout this text are generic

System Calls

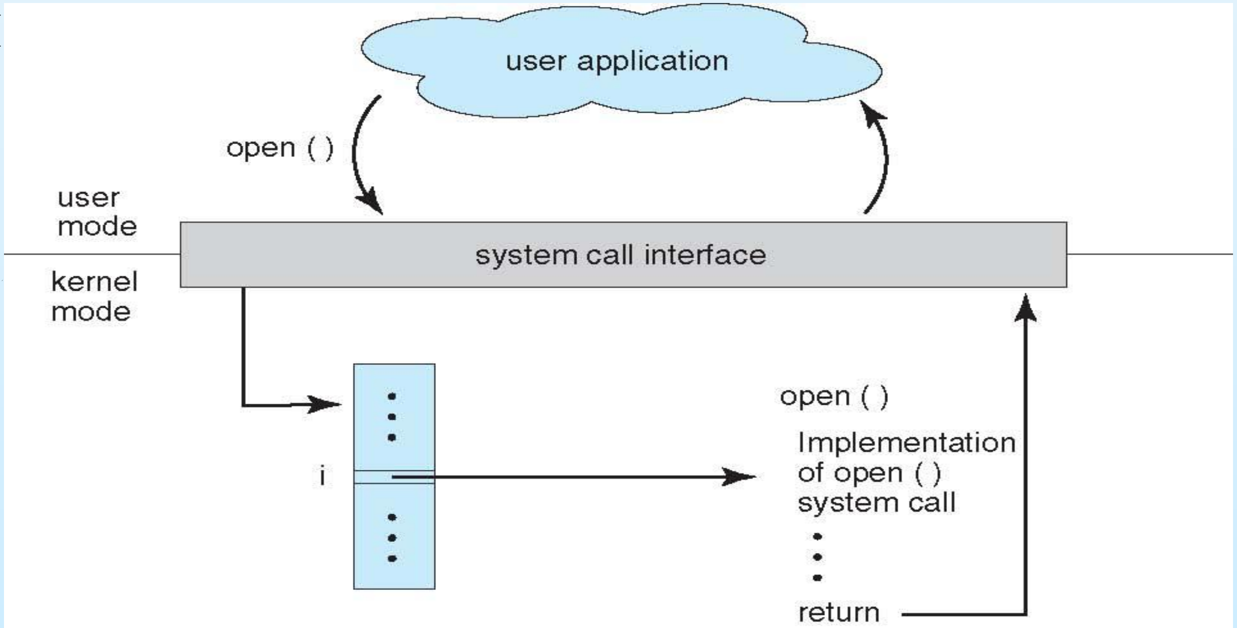
- System call sequence to copy the contents of one file to another file



System Call Implementation

- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented

- API



Types of System Call

Type	Windows OS	Linux OS
Process Control	**CreateProcess()** **ExitProcess()** **WaitForSingleObject()**	**fork()** **exit()** **wait()**
File Manipulation	**CreateFile()** **ReadFile()** **WriteFile()** **CloseHandle()**	**open()** **read()** **write()** **close()**
Device Manipulation	**SetConsoleMode()** **ReadConsole()** **WriteConsole()**	**ioctl()** **read()** **write()**

Types of System Call

Type	Windows OS	Linux OS
Information Maintenance	**GetCurrentProcessID()** **SetTimer()** **Sleep()**	**getpid()** **alarm()** **sleep()**
Communication	**CreatePipe()** **CreateFileMapping()** **MapViewOfFile()**	**pipe()** **shm_open()** **mmap()**
Protection	**SetFileSecurity() InitlializeSecurityDescriptor()** **SetSecurityDescriptorGroup()**	**chmod()** **umask()** **chown()**

OS Structure

- An operating system is a construct that allows the user application programs to interact with the system hardware. Since the operating system is such a complex structure, it should be created with utmost care so it can be used and modified easily. An easy way to do this is to create the operating system in parts. Each of these parts should be well defined with clear inputs, outputs and functions.

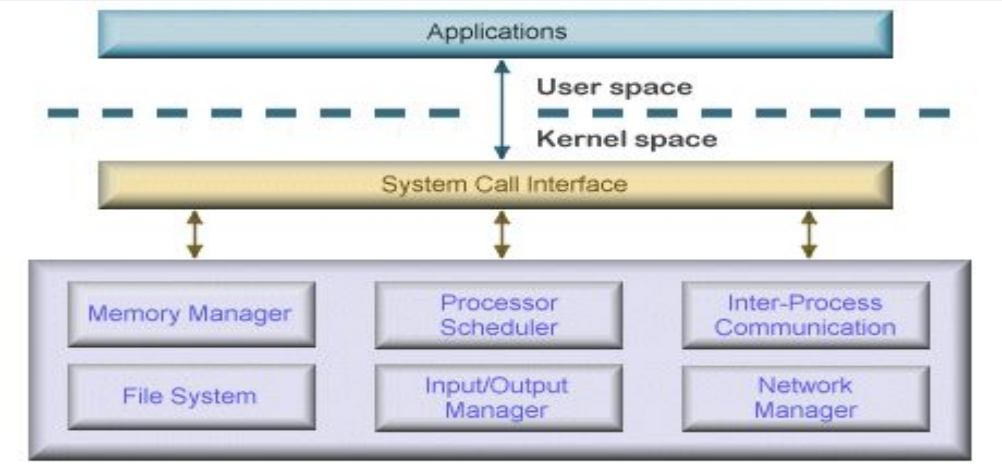
- General-purpose OS is very large program

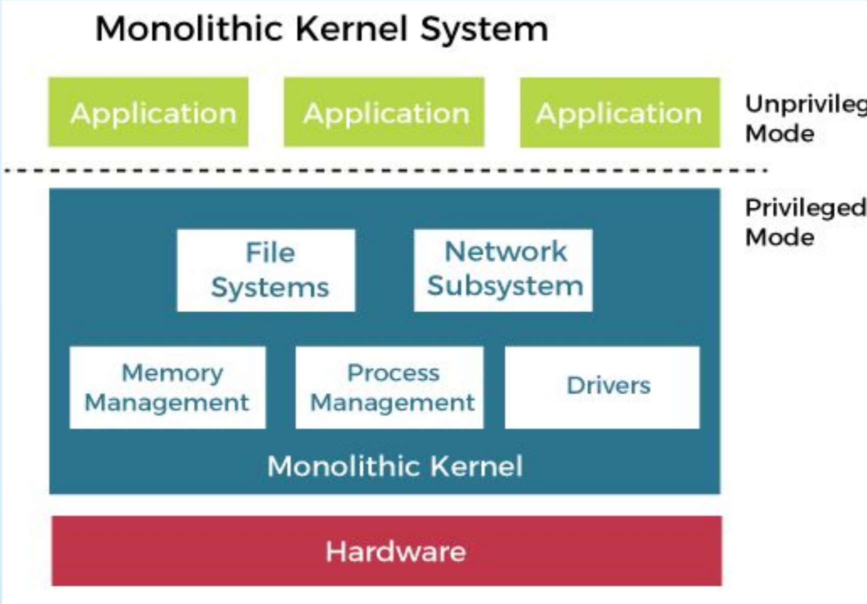
- Various ways to structure ones
  - Simple structure/ Monolithic kernel – MS-DOS
  - Layered – an abstraction
  - Microkernel -Mach

OS Structure: Monolithic

Simple/Monolithic structure:

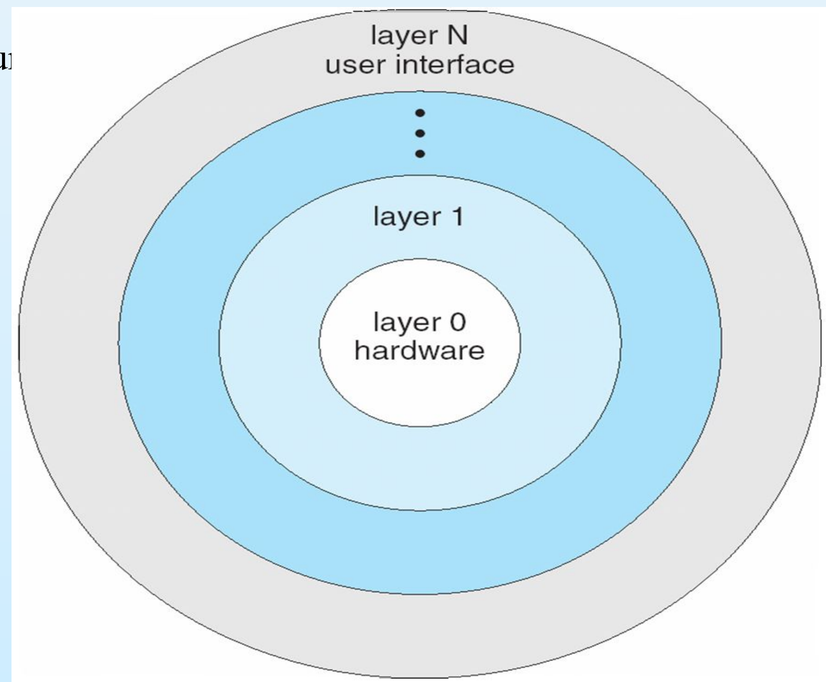
- Earliest and most common architecture
- Every component of OS is in the kernel and can communicate with each other directly
- Complex and Large (millions line of code ,) hard to maintain





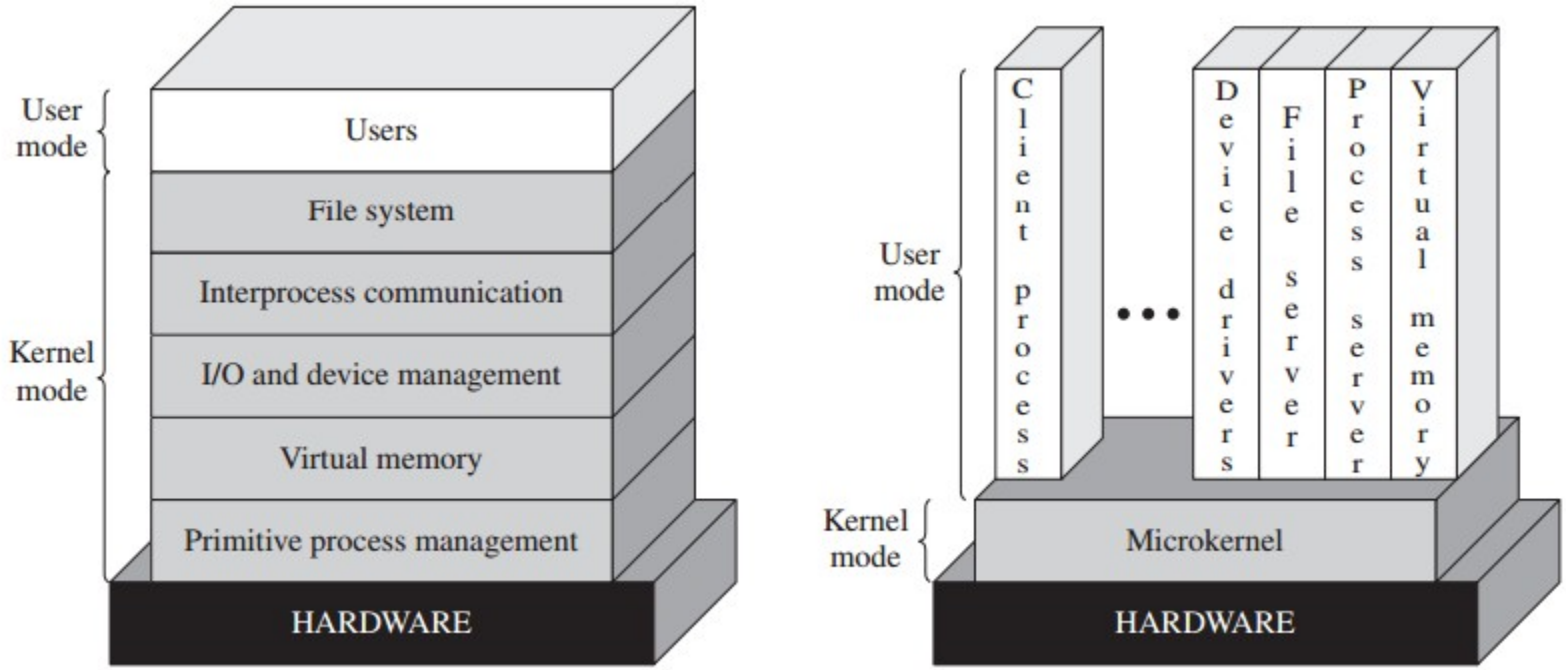
OS Structure: Layered

- In this, the bottom layer is the hardware and the topmost layer is the user interface.
- Each upper layer is built on the bottom layer. All the layers hide some structures, operations etc. from their upper layers.
- Each layer can use services of its lower layers.
- All the layers can be defined separately and interact with each other as required.
- One problem with the layered structu
- Easy to debug and develop.



OS Structure: Layered

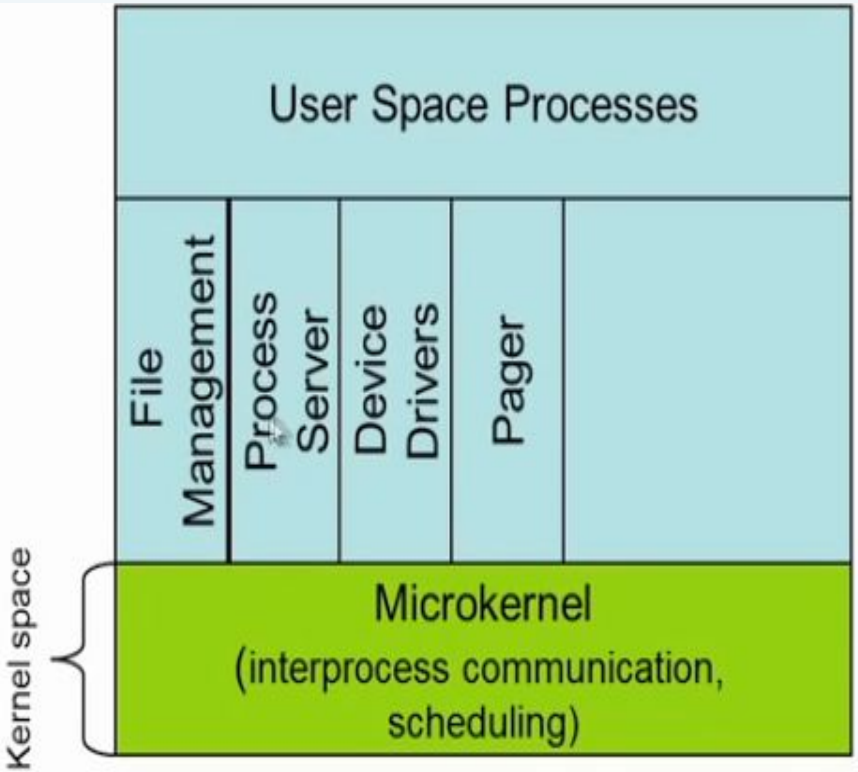
- Layering provides a distinct advantage in an operating system. Also, it is easier to create, maintain and update the system if it is done in the form of layers. Change in one layer specification does not affect the rest of the layers.
- A major disadvantage, however, is that is can be difficult to precisely assign the functions to the right, and the appropriate, layers. Moreover, given that there are too many layers, performance of the system can be degraded.



(a) Layered kernel

(b) Microkernel

OS Structure: Microkernel



Eg. QNX and L4

- Highly modular.
  - Every component has its own space.
  - Interactions between components strictly through well defined interfaces (no backdoors)
- Kernel has basic inter process communication and scheduling
  - Everything else in user space.
  - Ideally kernel is so small that it fits in the first level cache

OS Structure: Microkernel

- Benefits typically include the following: (a) adding a new service does not require modifying the kernel, (b) it is more secure as more operations are done in user mode than in kernel mode, and (c) a simpler kernel design and functionality typically results in a more reliable operating system.

- The primary disadvantages of the microkernel architecture are the overheads associated with interprocess communication and the frequent use of the operating system's messaging functions in order to enable the user process and the system service to interact with each other

- Microkernels are modular, and the various modules may be swapped, reloaded, and modified without affecting the kernel. It It is expensive as compared to the monolithic system architecture.

End