Basics of Operating System Part - I

Foundation Course on Operating Systems for GATE 2022

Sanchit Jain • Lesson 1 • July 21, 2021

Months Wise Calendar for GATE-2022 (Unacademy Plus) (Sanchit Jain)

Month	Morning Class (7:00am – 9:00am)	Evening Class (7:00pm – 9:00pm)
Feb-2021	DS & Programming (15-Feb to 23-Mar) (56 Hours)	DM (22-Feb to 27-Mar) (50 Hours)
Mar-2021	Algo (24-Mar to 26-April) (38 Hours)	DBMS (29-Mar to 30-April) (50 Hours)
Apr-2021	CN(29-April to 15-June) (54 Hours)	DE (7-May to 1-June) (40 Hours)
May-2021	TOC (26-May to 05-July) (58 Hours)	
Jun-2021		COA (23-June to 20-July) (38 Hours)
Jul-2021	Compiler (7-July to 5-Aug) (34 Hours)	OS (21-July to 24-Aug) (58 Hours)
Aug-2021	DM (11-Aug to 20-Sept) (50 Hours)	DS & Programming (30-Aug to 5-Oct) (56 Hours)
Sep-2021	DBMS (22-Sept to 29-Oct) (50 Hours)	Algo (6-Oct to 1-Nov) (38 Hours)
Oct-2021	DE (1-Nov to 26-Nov) (40 Hours)	CN(2-Nov to 6-Dec) (54 Hours)
Nov-2021	COA (29-Nov to 23-Dec) (38 Hours)	TOC (7-Dec to 14-Jan) (58 Hours)
Dec-2021	OS (23-Dec to 2-Feb) (58 Hours)	Compiler (17-Jan to 8-Feb) (34 Hours)
Jan-2022		
Feb-2022		

Date	Day	Subject-1	Topic	Time	Duration
07-07-2021	Wednesday	compiler	Basics of Compiler Part-1	7:00 am to 9:00 am	2
08-07-2021	Thursday	compiler	Basics of Compiler Part-2	7:00 am to 9:00 am	2
09-07-2021	Friday				
10-07-2021	Saturday				
11-07-2021	Sunday				
12-07-2021	Monday	compiler	Lexical Analysis	7:00 am to 9:00 am	2
13-07-2021	Tuesday	compiler	Doubt Clearing Session	7:00 am to 9:00 am	2
14-07-2021	Wednesday	compiler	Context Free Grammar Part-1	7:00 am to 9:00 am	2
15-07-2021	Thursday	compiler	Context Free Grammar Part-2	7:00 am to 9:00 am	2
16-07-2021	Friday				
17-07-2021	Saturday				
18-07-2021	Sunday				
19-07-2021	Monday	compiler	Parsing Part-1	7:00 am to 9:00 am	2
20-07-2021	Tuesday	compiler	Doubt Clearing Session	7:00 am to 9:00 am	2
21-07-2021	Wednesday	compiler	Parsing Part-2	7:00 am to 9:00 am	2
22-07-2021	Thursday	compiler	Parsing Part-3	7:00 am to 9:00 am	2
23-07-2021	Friday				
24-07-2021	Saturday				
25-07-2021	Sunday				
26-07-2021	Monday	compiler	Parsing Part-4	7:00 am to 9:00 am	2
27-07-2021	Tuesday	compiler	Doubt Clearing Session	7:00 am to 9:00 am	2
28-07-2021	Wednesday	compiler	Parsing Part-5	7:00 am to 9:00 am	2
29-07-2021	Thursday				
30-07-2021	Friday				
31-07-2021	Saturday				
01-08-2021	Sunday				
02-08-2021	Monday	compiler	Syntax Directed Translation Part-1	7:00 am to 9:00 am	2
03-08-2021	Tuesday	compiler	Syntax Directed Translation Part-2	7:00 am to 9:00 am	2
04-08-2021	Wednesday	compiler	Doubt Clearing Session	7:00 am to 9:00 am	2
05-08-2021	Thursday	compiler	Code Optimization	7:00 am to 9:00 am	2

Date	Day	Subject-2	Topic	Time	Duration
23-06-2021	Wednesday	COA	memory hierarchy	7:00 pm to 9:00 pm	2
24-06-2021	Thursday	COA	Cache Mapping Part-1	7:00 pm to 9:00 pm	2
25-06-2021	Friday				
26-06-2021	Saturday				
27-06-2021	Sunday				
28-06-2021	Monday	COA	Cache Mapping Part-2	7:00 pm to 9:00 pm	2
29-06-2021	Tuesday	COA	Doubt Clearing Session	7:00 pm to 9:00 pm	2
30-06-2021	Wednesday				
01-07-2021	Thursday	COA	Cache Management Part-1	7:00 pm to 9:00 pm	2
02-07-2021	Friday	COA	Cache Management Part-2	7:00 pm to 9:00 pm	2
03-07-2021	Saturday				
04-07-2021	Sunday				
05-07-2021	Monday	COA	Input-Output Organization Part-1	7:00 pm to 9:00 pm	2
06-07-2021	Tuesday	COA	Doubt Clearing Session	7:00 pm to 9:00 pm	2
07-07-2021	Wednesday	COA	Input-Output Organization Part-2	7:00 pm to 9:00 pm	2
08-07-2021	Thursday	COA	Input-Output Organization Part-3	7:00 pm to 9:00 pm	2
09-07-2021	Friday	COA	Input-Output Organization Part-4	7:00 pm to 9:00 pm	2
10-07-2021	Saturday				
11-07-2021	Sunday				
12-07-2021	Monday	COA	Doubt Clearing Session	7:00 pm to 9:00 pm	2
13-07-2021	Tuesday	COA	Pipelining Part-1	7:00 pm to 9:00 pm	2
14-07-2021	Wednesday	COA	Pipelining Part-2	7:00 pm to 9:00 pm	2
15-07-2021	Thursday	COA	Instruction Formate and Addressing mode Part-1	7:00 pm to 9:00 pm	2
16-07-2021	Friday	COA	Doubt Clearing Session	7:00 pm to 9:00 pm	2
17-07-2021	Saturday	COA	Instruction Formate and Addressing mode Part-2	7:00 pm to 9:00 pm	2
18-07-2021	Sunday				
19-07-2021	Monday	COA	Control Unit Part-1	7:00 pm to 9:00 pm	2
20-07-2021	Tuesday	COA	Control Unit Part-2	7:00 pm to 9:00 pm	2

Date	Day	Subject-2	Topic	Time	Duration
21-07-2021	Wednesday	Operating System	Basics of Operating System Part-1	7:00 pm to 9:00 pm	2
22-07-2021	Thursday	Operating System	Basics of Operating System Part-2	7:00 pm to 9:00 pm	2
23-07-2021	Friday	Operating System	Process Management Basics	7:00 pm to 9:00 pm	2
24-07-2021	Saturday	Operating System	Doubt Clearing Session	7:00 pm to 9:00 pm	2
25-07-2021	Sunday				
26-07-2021	Monday	Operating System	CPU Scheduling Part-1	7:00 pm to 9:00 pm	2
27-07-2021	Tuesday	Operating System	CPU Scheduling Part-2	7:00 pm to 9:00 pm	2
28-07-2021	Wednesday	Operating System	CPU Scheduling Part-3	7:00 pm to 9:00 pm	2
29-07-2021	Thursday	Operating System	Doubt Clearing Session	7:00 pm to 9:00 pm	2
30-07-2021	Friday	Operating System	CPU Scheduling Part-4	7:00 pm to 9:00 pm	2
31-07-2021	Saturday	Operating System	CPU Scheduling Part-5	7:00 pm to 9:00 pm	2
01-08-2021	Sunday				
02-08-2021	Monday	Operating System	Process Synchronization Part-1	7:00 pm to 9:00 pm	2
03-08-2021	Tuesday	Operating System	Doubt Clearing Session	7:00 pm to 9:00 pm	2
04-08-2021	Wednesday	Operating System	Process Synchronization Part-2	7:00 pm to 9:00 pm	2
05-08-2021	Thursday	Operating System	Process Synchronization Part-3	7:00 pm to 9:00 pm	2
06-08-2021	Friday	Operating System	Dead Lock Handling Part-1	7:00 pm to 9:00 pm	2
07-08-2021	Saturday	Operating System	Doubt Clearing Session	7:00 pm to 9:00 pm	2
08-08-2021	Sunday				
09-08-2021	Monday	Operating System	Dead Lock Handling Part-2	7:00 pm to 9:00 pm	2
10-08-2021	Tuesday				
11-08-2021	Wednesday	Operating System	Dead Lock Handling Part-3	7:00 pm to 9:00 pm	2
12-08-2021	Thursday				
13-08-2021	Friday	Operating System	Memory Management Part-1	7:00 pm to 9:00 pm	2
14-08-2021	Saturday	Operating System	Doubt Clearing Session	7:00 pm to 9:00 pm	2
15-08-2021	Sunday	Operating System	Memory Management Part-2	7:00 pm to 9:00 pm	2
16-08-2021	Monday	Operating System	Virtual Memory Part-1	7:00 pm to 9:00 pm	2
17-08-2021	Tuesday	Operating System	Virtual Memory Part-2	7:00 pm to 9:00 pm	2
18-08-2021	Wednesday	Operating System	Doubt Clearing Session	7:00 pm to 9:00 pm	2
19-08-2021	Thursday	Operating System	Disk Scheduling Part-1	7:00 pm to 9:00 pm	2
20-08-2021	Friday	Operating System	Disk Scheduling Part-2	7:00 pm to 9:00 pm	2
21-08-2021	Saturday	Operating System	File Management	7:00 pm to 9:00 pm	2
22-08-2021	Sunday				
23-08-2021	Monday	Operating System	Doubt Clearing Session	7:00 pm to 9:00 pm	2
24-08-2021	Tuesday	Operating System	Understanding Fork command	7:00 pm to 9:00 pm	2

Months Wise Calendar for GATE-2023 (Unacademy Plus) (Sanchit Jain)

Month	Morning Class (7:00am – 9:00am)	Evening Class (7:00pm – 9:00pm)
Feb-2022 Mar-2022 Apr-2022 May-2022 Jun-2022 Jul-2022 Aug-2022	3 DS & Programming Algo CN TOC Compiler	DM DBMS DE COA OS
Sep-2022 Oct-2022 Nov-2022 Dec-2022 Jan-2023 Feb-2023	4 DM DBMS DE COA OS	DS & Programming Algo CN TOC Compiler

Sanchit Jain

#4 Educator in CS & IT · GATE & ESE

I am a computer science faculty, having a teaching experience of more than 8 years. I have a very sound experience in teaching students.

31M Watch mins | **3M** Watch mins (last 30 days) | **18K** Followers | **2K** Dedications

Educator highlights

- Works at **Knowledge Gate**
- Studied at **Delhi Technological University**
- Have qualified gate Have experience of more than 8 years Have a YouTube channel with 5 lakh follower
- Lives in **Ghaziabad, Uttar Pradesh, India**
- Unacademy Educator since **19th July, 2019**
- **844,388 live minutes** taught in last 30 days
- Knows **Punjabi, Hinglish, Hindi and English**

← Sanchit Jain's courses

GATE & ESE | Plus | All courses | All languages | All topics

HINDI CS & IT
Complete Course on Compiler Design for GATE
Lesson 5 • Today, 7:00 AM
Sanchit Jain

HINDI CS & IT
Complete Course on Computer Architecture for GATE 2022
Lesson 13 • Started at 7:00 PM
Sanchit Jain

HINGLISH CS & IT
Foundation Course on Theory of Computation for GATE 2022
Ended on Jul 5, 2021 • 29 lessons
Sanchit Jain

HINDI CS & IT
Foundation Course on Digital Electronics
Ended on Jun 1, 2021 • 20 lessons
Sanchit Jain

HINDI CS & IT
Foundation Course on Computer Networks
Ended on Jun 22, 2021 • 27 lessons
Sanchit Jain

HINDI CS & IT
Foundation Course on DBMS
Ended on May 4, 2021 • 25 lessons
Sanchit Jain

HINDI CS & IT
Foundation Course on Algorithms
Ended on Apr 26, 2021 • 19 lessons
Sanchit Jain

HINDI CS & IT
Foundation Course on Discrete Mathematics
Ended on Mar 30, 2021 • 27 lessons
Sanchit Jain

HINDI CS & IT
Foundation Course on Data Structure with C Programming
Ended on Mar 23, 2021 • 28 lessons
Sanchit Jain

HINDI CS & IT
Revision and Doubt Clearing Course on CS & IT for GATE 2021
Ended on Feb 11, 2021 • 9 lessons
Sanchit Jain

HINDI CS & IT
Complete Course on Compiler Design for GATE 2021
Ended on Feb 8, 2021 • 21 lessons
Sanchit Jain

HINDI CS & IT
Complete Course on Operating System for GATE 2021
Ended on Jan 31, 2021 • 30 lessons
Sanchit Jain

Link is above in the comment Section

operating system

Objective of the course

- Provide overall knowledge of the subject for Students in B.Tech, M.Tech, BCA, MCA, BSC, MSC and other courses for university exam.
- Solid knowledge base for Competitive exam students like GATE, NET, KVS, NIC, Metro rail, BSNL, State PSU's etc.
- If you are a Educator in College, Coaching etc. or you want to be one.
- Looking for a high package jobs in Big MNC's

File Management
5-7%
Security Management
Process Management
50 %
Operating system Components
M.M
40-45%
Secondary Storage Management
I/O Device Management
Network Management

Syllabus(University Exams)

Unit – I Introduction : Operating system and functions, Classification of Operating systems- Batch, Interactive, Time sharing, Real Time System, Multiprocessor Systems, Multiuser Systems, Multiprocess Systems, Multithreaded Systems, Operating System Structure- Layered structure, System Components, Operating System services, Reentrant Kernels, Monolithic and Microkernel Systems.

Unit – II Concurrent Processes: Process Concept, Principle of Concurrency, Producer / Consumer Problem, Mutual Exclusion, Critical Section Problem, Dekker's solution, Peterson's solution, Semaphores, Test and Set operation; Classical Problem in Concurrency- Dining Philosopher Problem, Sleeping Barber Problem; Inter Process Communication models and Schemes, Process generation.

Unit – III CPU Scheduling: Scheduling Concepts, Performance Criteria, Process States, Process Transition Diagram, Schedulers, Process Control Block (PCB), Process address space, Process identification information, Threads and their management, Scheduling Algorithms, Multiprocessor Scheduling. Deadlock: System model, Deadlock characterization, Prevention, Avoidance and detection, Recovery from deadlock.

Unit – IV Memory Management: Basic bare machine, Resident monitor, Multiprogramming with fixed partitions, Multiprogramming with variable partitions, Protection schemes, Paging, Segmentation, Paged segmentation, Virtual memory concepts, Demand paging, Performance of demand paging, Page replacement algorithms, Thrashing, Cache memory organization, Locality of reference.

Unit – V I/O Management and Disk Scheduling: I/O devices, and I/O subsystems, I/O buffering, Disk storage and disk scheduling, RAID. File System: File concept, File organization and access mechanism, File directories, and File sharing, File system implementation issues, File system protection and security.

Syllabus (GATE)

- Processes and Threads
- CPU scheduling
- Inter-Process Communication, Concurrency and Synchronization
- Deadlock
- Memory management and virtual memory
- File systems

Syllabus (NTA NET)

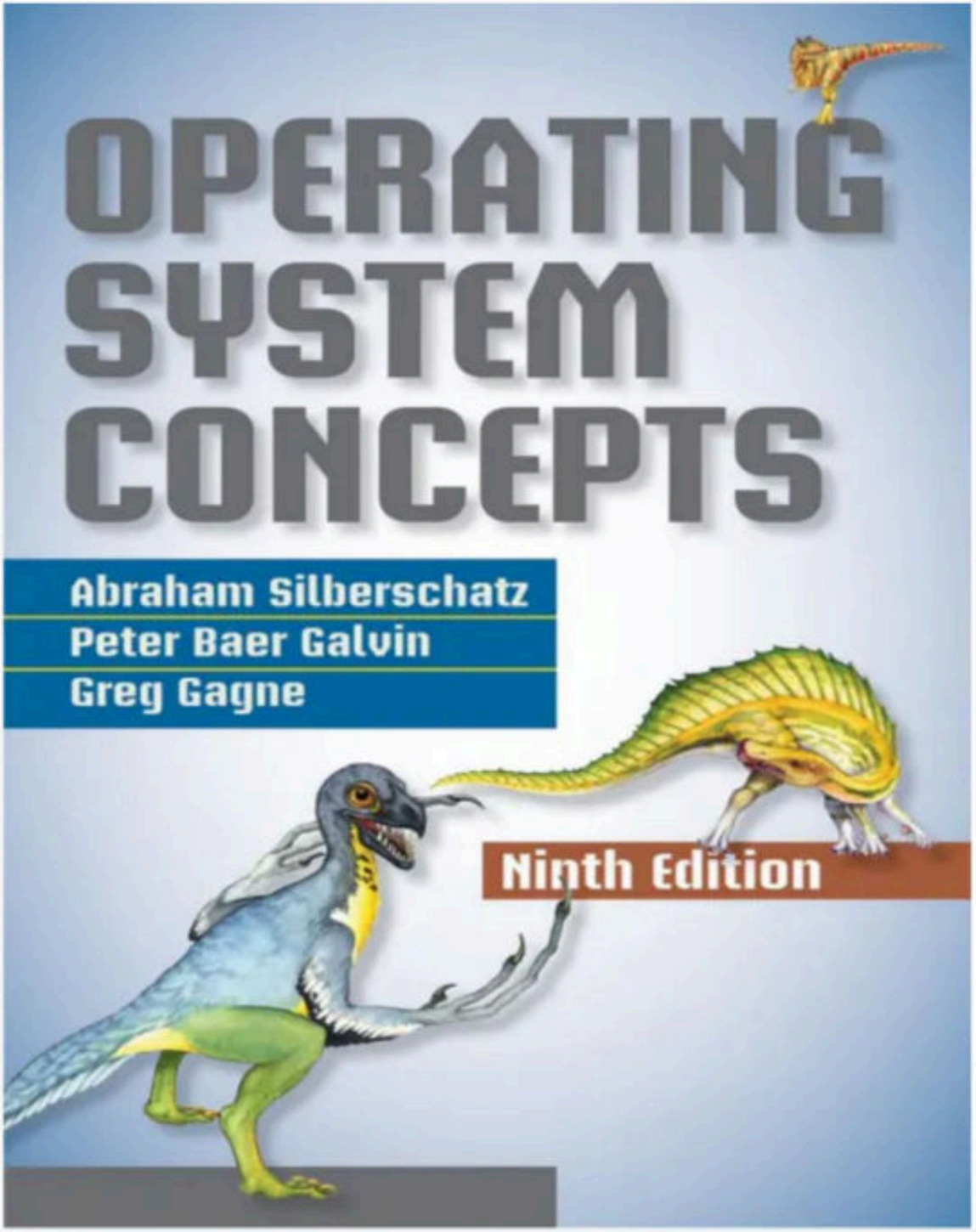
- System Software
- Basics of Operating Systems
- Process Management
- Threads
- CPU Scheduling
- Deadlocks
- Memory Management
- Storage Management
- Input/Output Systems
- Security
- Virtual Machines
- Linux Operating Systems
- Windows Operating Systems
- Distributed Systems

Operating System

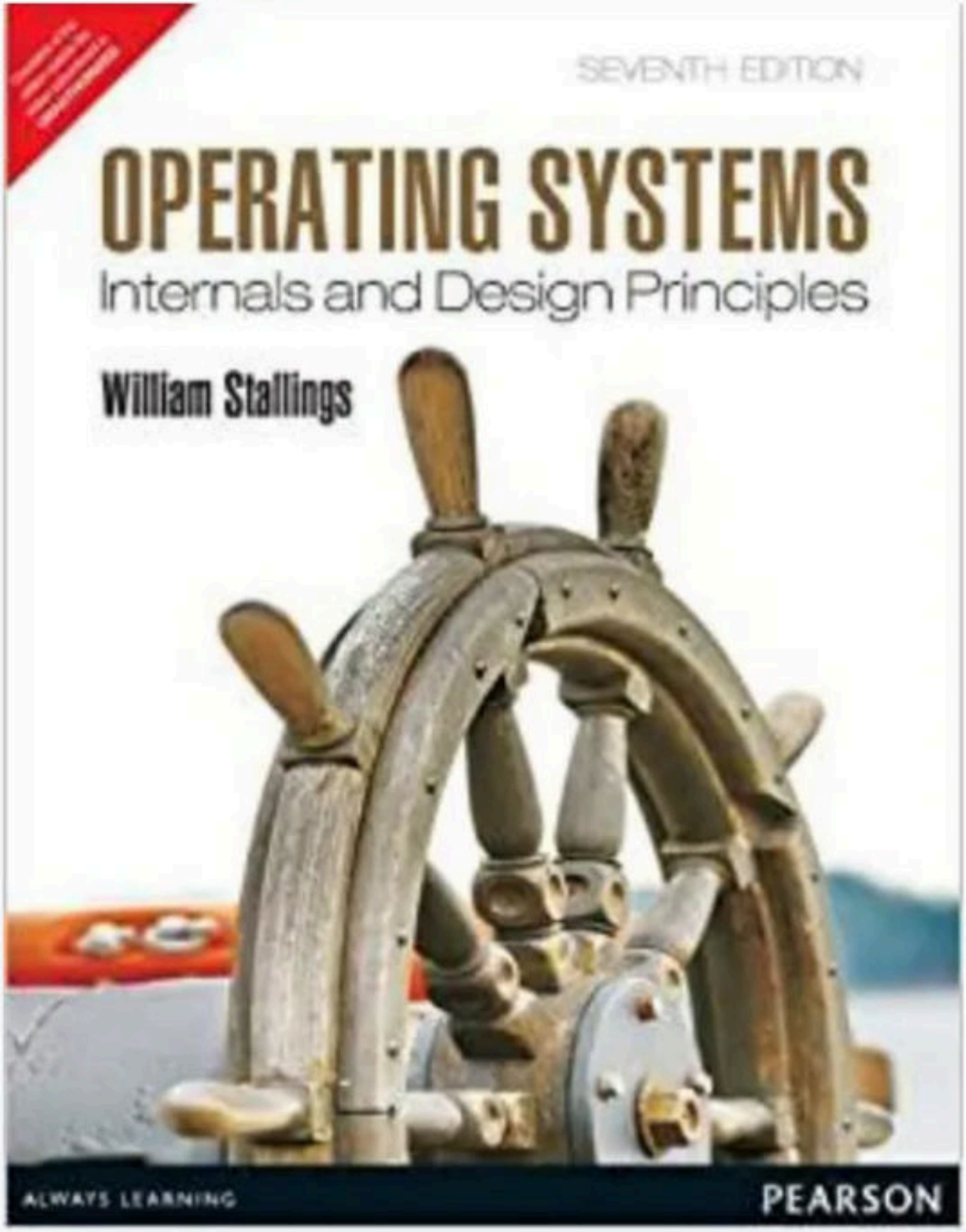
- In GATE 7-8 Marks, and 5-6 questions on an average
- In NET 20-22 Marks and 10-12 questions

Operating System

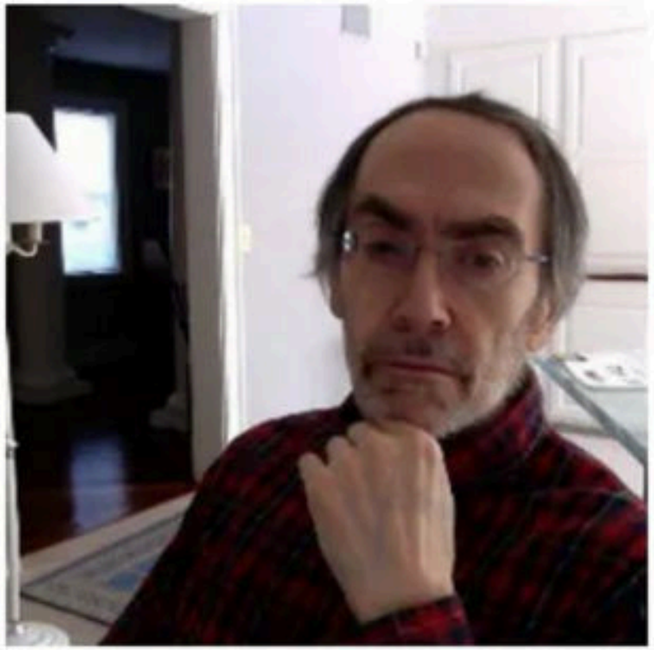
- Books to be referred
  - \ Galvin
  - Stalling
  - Tanenbaum

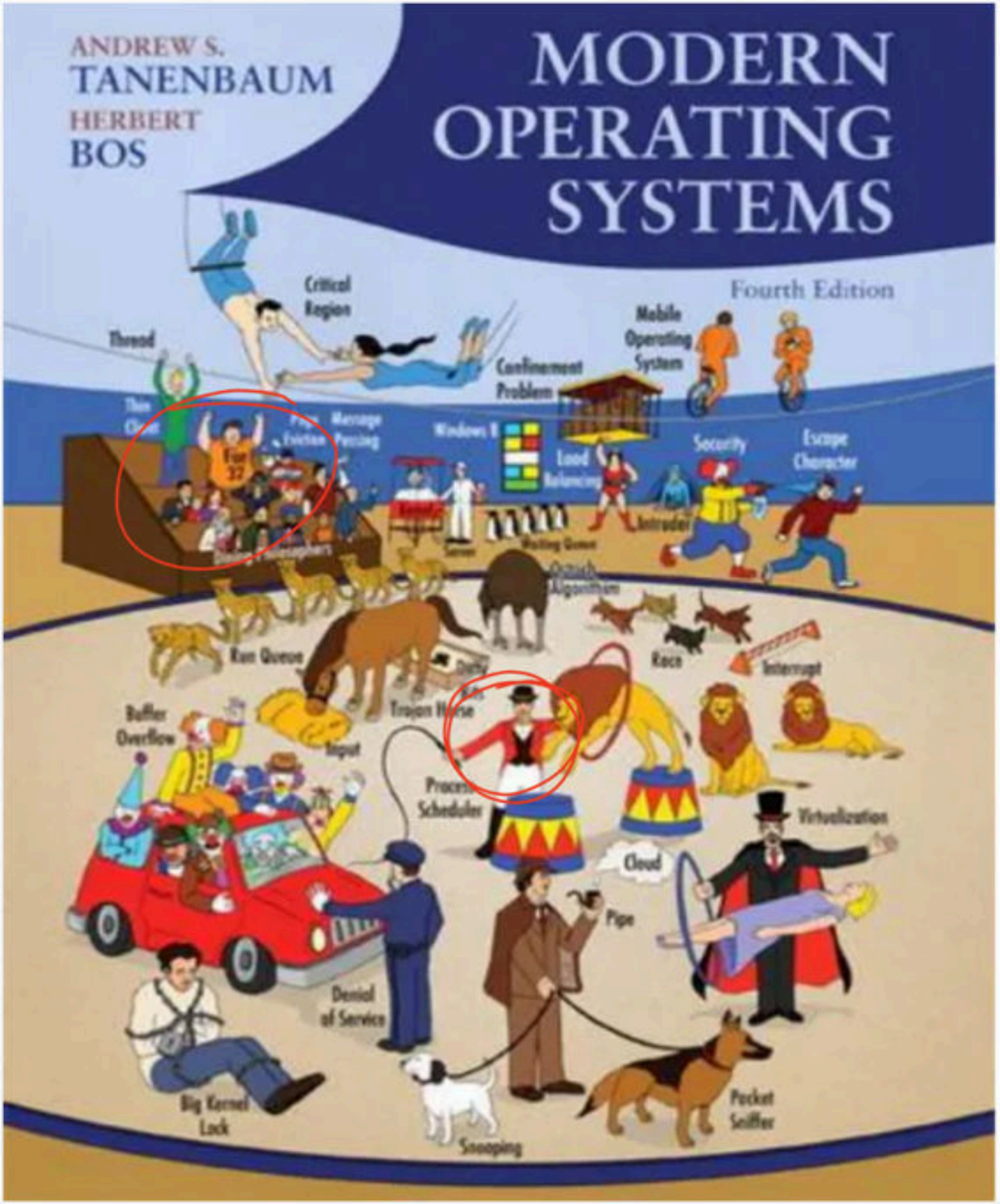


- Book Name - Operating System Concepts
- Writers -
  - Abraham Silberschatz
  - Peter B Galvin
  - Gerg Gane
- Publisher – Wiley India
- Edition – 9th



- Book Name - Operating System Internals and Design Principles
- Writers – William Stallings
- Publisher – Pearson
- Edition – 7th

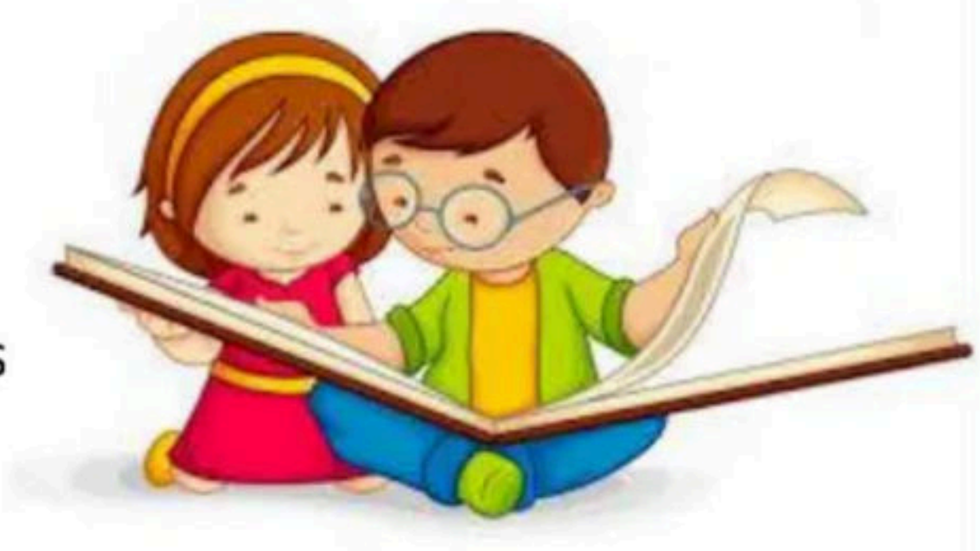




- Book Name – Modern Operating Systems
- Writers -
  - Andrew's Tanenbaum
  - Herbert Bos
- Publisher – Pearson
- Edition – 4th

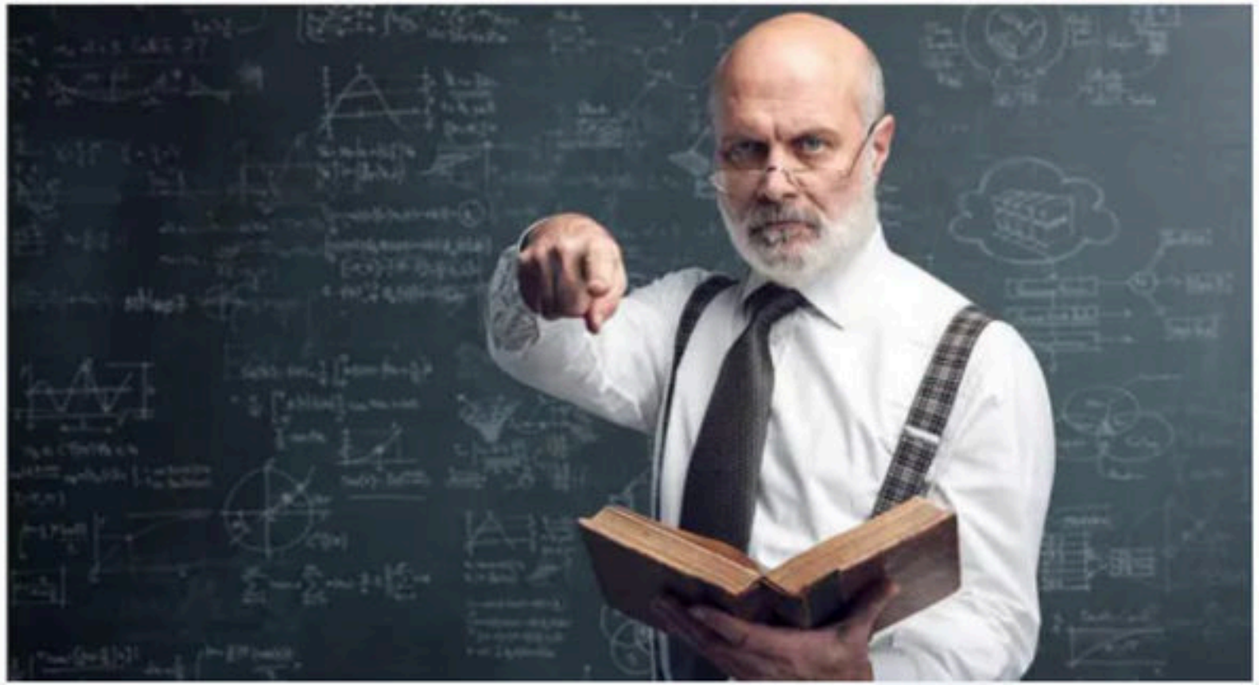
What you can expect from me

- Will take care of both types of questions theory and numerical.
- Will give more weightage to the topics that are asked more frequently in Competitive and University Exams
- Will provide PDF of related books
- Pls consider these lectures pdf as notes
- Will cover Maximum Questions from previous year papers
- Will Provide a good number of practice questions
- Will not emphasize more than required, on a topic



What i expect from you

- There is no hurry, first go through the entire topic, most of your doubts will be resolved automatically.
- Please revise the entire lecture before and after the class → PYQ
- Be regular, Consistency is most important
- More you practice, more clarity you will get
- If we follow all the above specified points Success in guaranteed



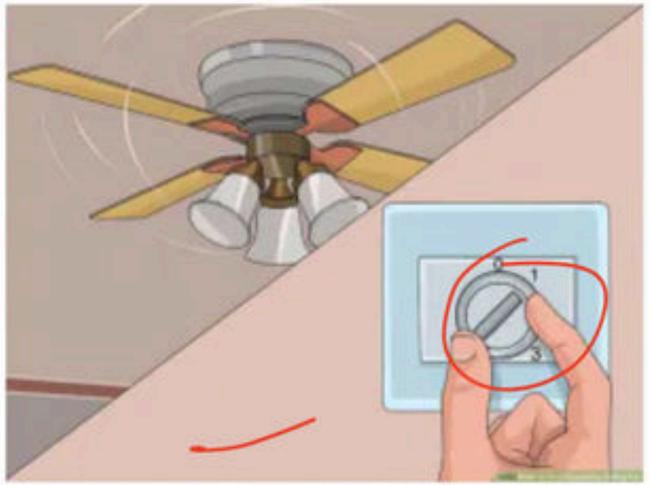
Basics of Operating system

1. What is Operating System
2. Abstract view of Operating System
3. Goals and functions of Operating System
4. Evolution of Operating System
5. Multiprogramming Operating System
6. Multitasking/Time Sharing Operating System
7. Multiprocessing Processing Operating System
8. Real Time Operating System
9. Distributed Operating System
10. CLI Vs GUI
11. Structure of Operating System
12. Micro-Kernel approach
13. System call

70-80%

What is Operating System

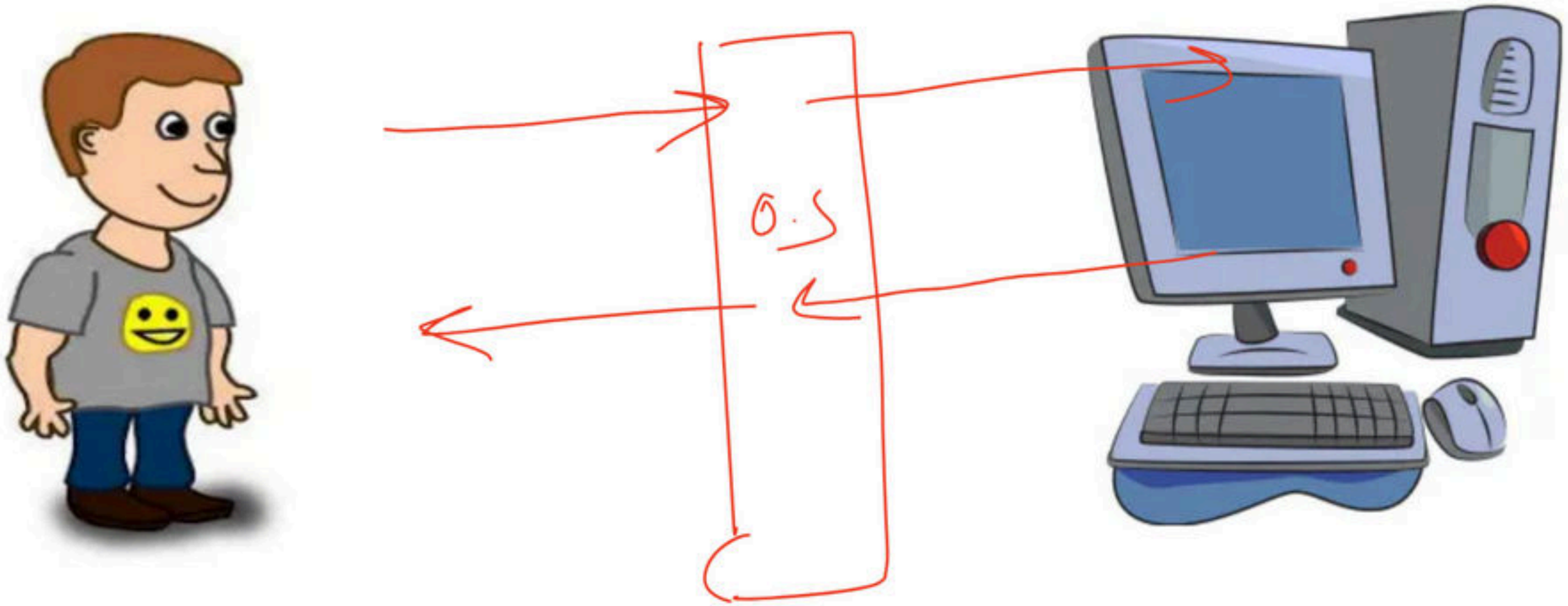
- Whatever used as an interface between the user and the core machine is OS.
  - E.g. steering of car, switch of the fan etc., Buttons over electronic devices.



- Question comes why we need an operating system?
  - To enable everybody to use h/w In a convenient and efficient manner

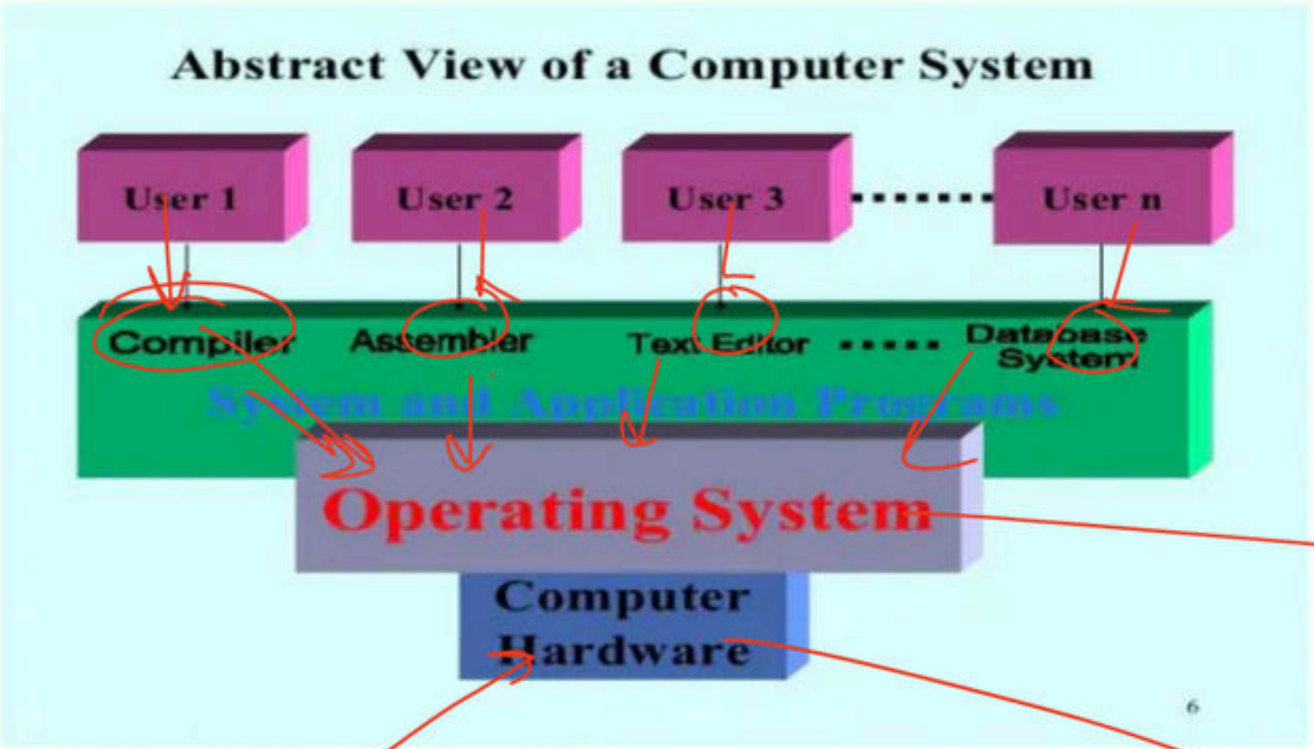
Definition of Operating System

- **There is no exact or precise definition for OS but we can say, "A program or System software"**
  - Which Acts as an **intermediary** between user & h/w



- **Resource Manager/Allocator** - Manage system resources in an unbiased fashion both h/w (mainly CPU time, memory, system buses) & s/w (access, authorization, semaphores) and provide functionality to application programs.
- OS controls and coordinates the use of resources among various application programs.

- OS provides platform on which other application programs can be installed, provides the environment within which programs are executed.



GOI

India

iOS
TIZEN
Operating Systems
Ubuntu
Mac
Windows
Android
ios
Linux
Tizen
Debian
Chrome

- **For personal computers** - Microsoft windows (82.7%), Mac (13.23), Linux (1.57)
- **For mobile phones** - Android (87.5), IOS (12.1) etc.

Q An operating system is: **(NET-DEC-2006)**

(A) Collection of hardware components → 4

(B) Collection of input-output devices — 1

(C) Collection of software routines → 5 %

(D) All the above ← 41 %

16 ~~15~~ 20 %

30 sec → 1 mark

force → 2 mark

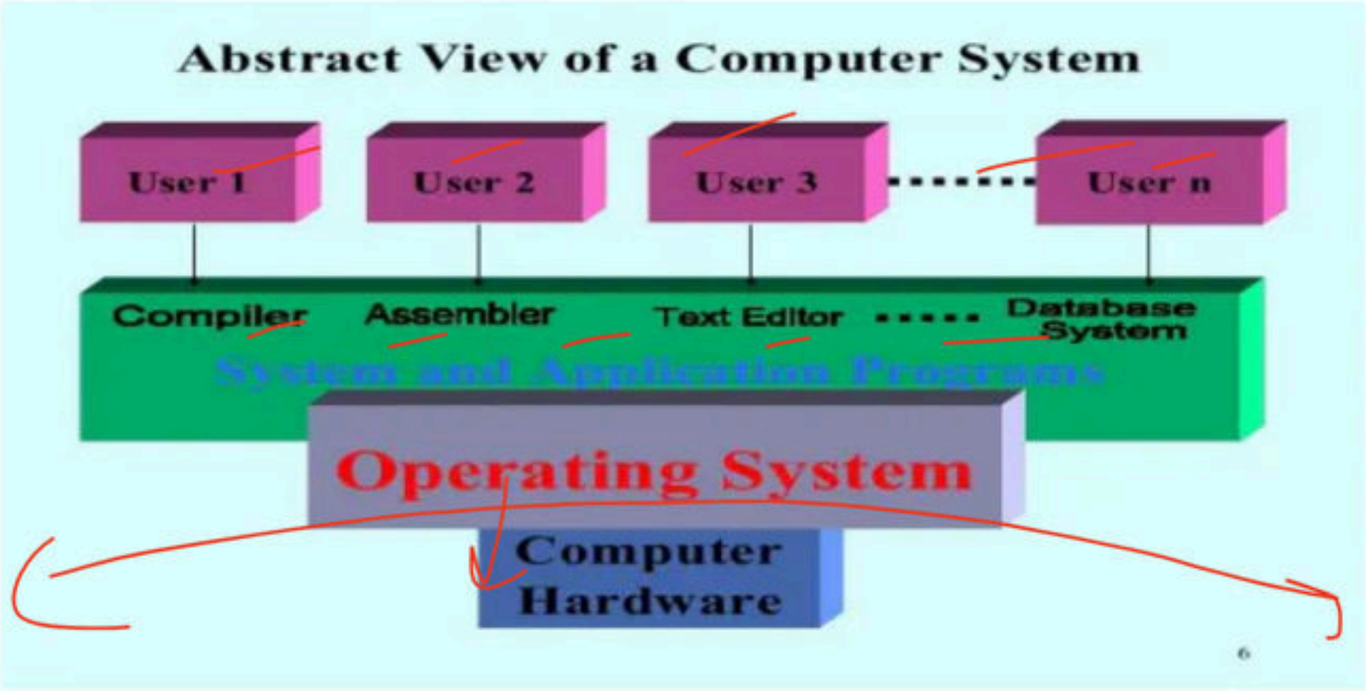
Break

- **Computer hardware** – CPU, memory units, i/o devices, system bus, registers etc. provides the basic computing resources.

- **OS** - Control and coordinates the use of the hardware among the various applications programs.

- **System and Applications programs** - Defines the way in which these resources are used to solve the computing problems of the user.

- **User**



Break

Goals and Functions of operating system

- Goals are the ultimate destination, but we follow functions to implement goals.

- Goals
  - Primary goals (Convenience / user friendly)
  - Secondary goals (Efficiency (Using resources in efficient manner) / Reliability / maintainability)

DSLR
auto
Nikon
94%
RES
4K
FPS
30
FOV
WIDE

सत्यमेव जयते

Goverment Of India

सबका साथ सबका विकास

Ministries	Departments
58	93

1

2

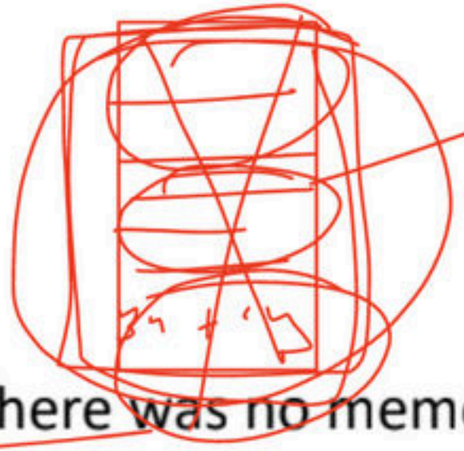
Functions of operating system

- Process management
- Memory management
- I\O device management
- File management
- Network management
- Security & Protection

Break

Evolution of Operating System

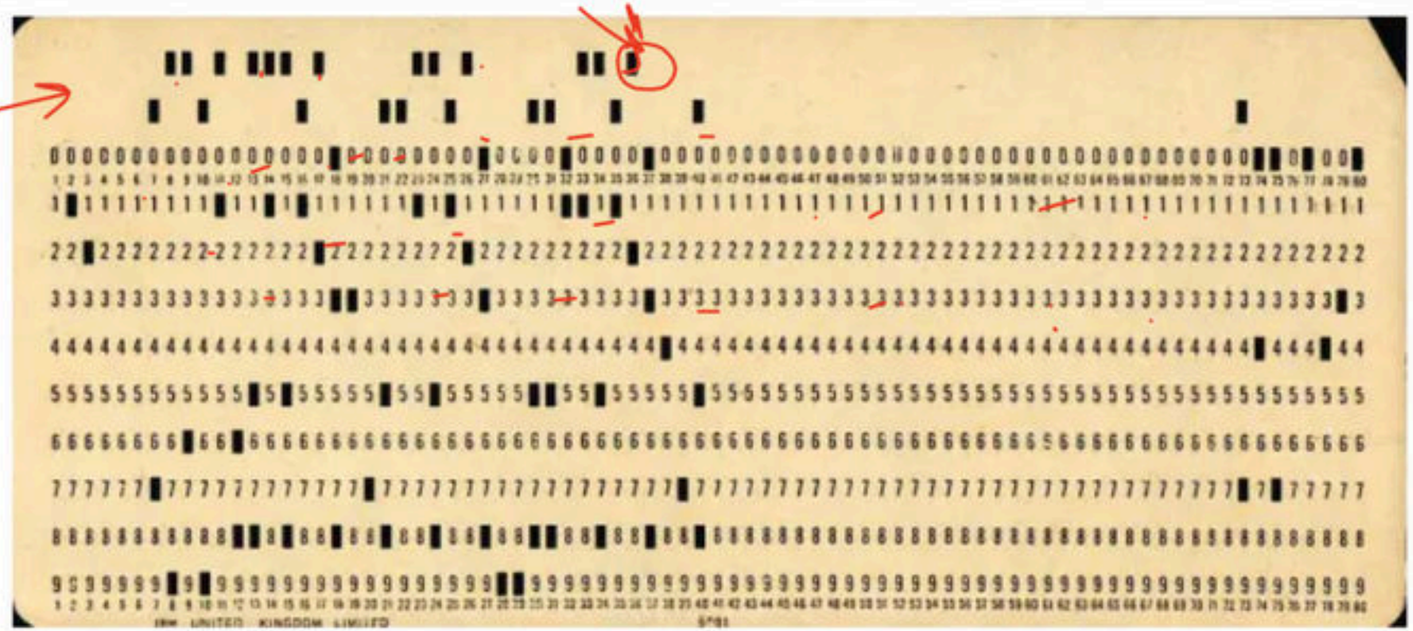
- Early computers were not interactive device, there user use to prepare a job which consist three parts
  - Program
  - Control information
  - Input data

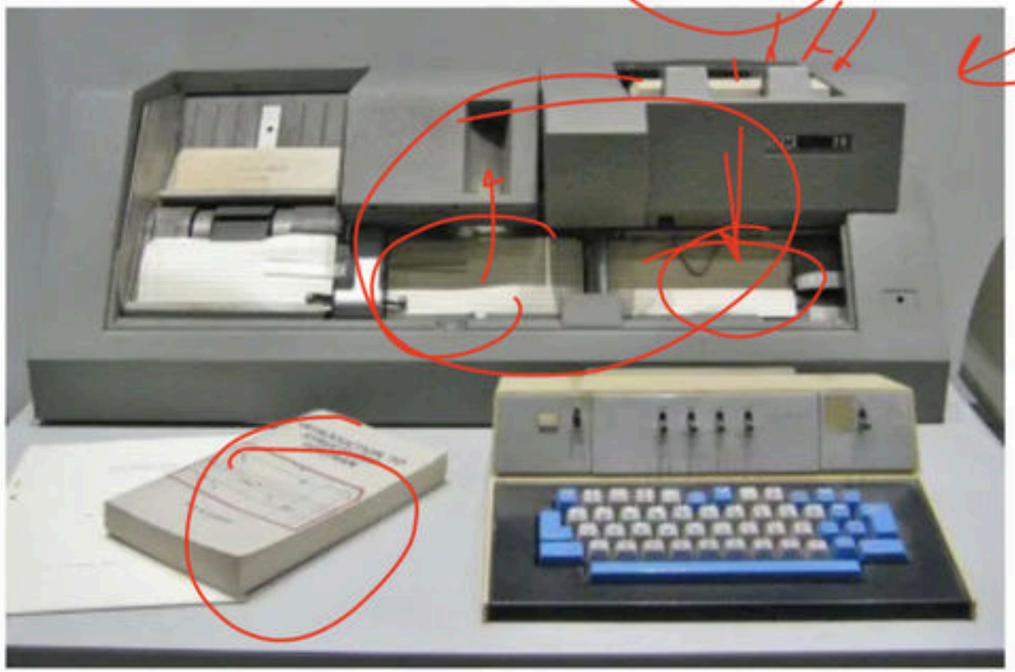


Job

- Only one job is given input at a time as there was no memory, computer will take the input then process it and then generate output.
- Common input/output device were punch card or tape drives.
- So these devices were very slow, and processor remain ideal most of the time.

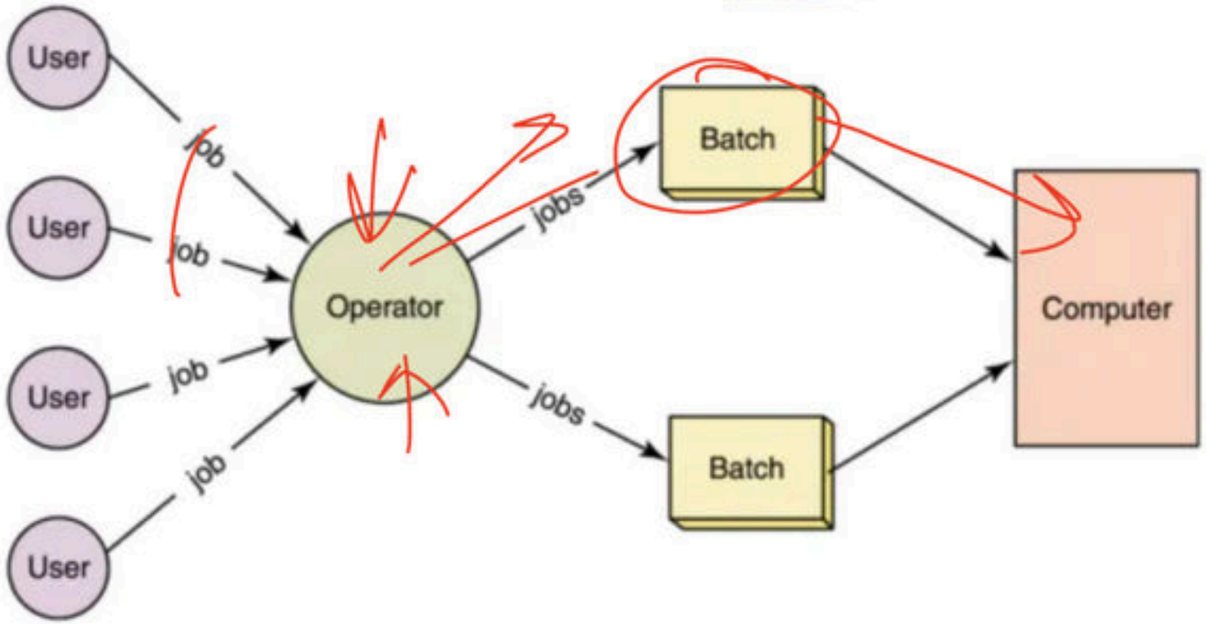
Punch Card in Punch Card Machine





Batch Operating System

- To speed up the processing job with similar types (programming language) were batched together and were run through the processor as a group (batch).
- In some system grouping is done by the operator while in some systems it is performed by the 'Batch Monitor' resided in the low end of main memory)
- Then jobs (as a deck of punched cards) are bundled into batches with similar requirement.
- Then the submitted jobs were 'grouped as FORTRAN jobs, COBOL jobs etc.



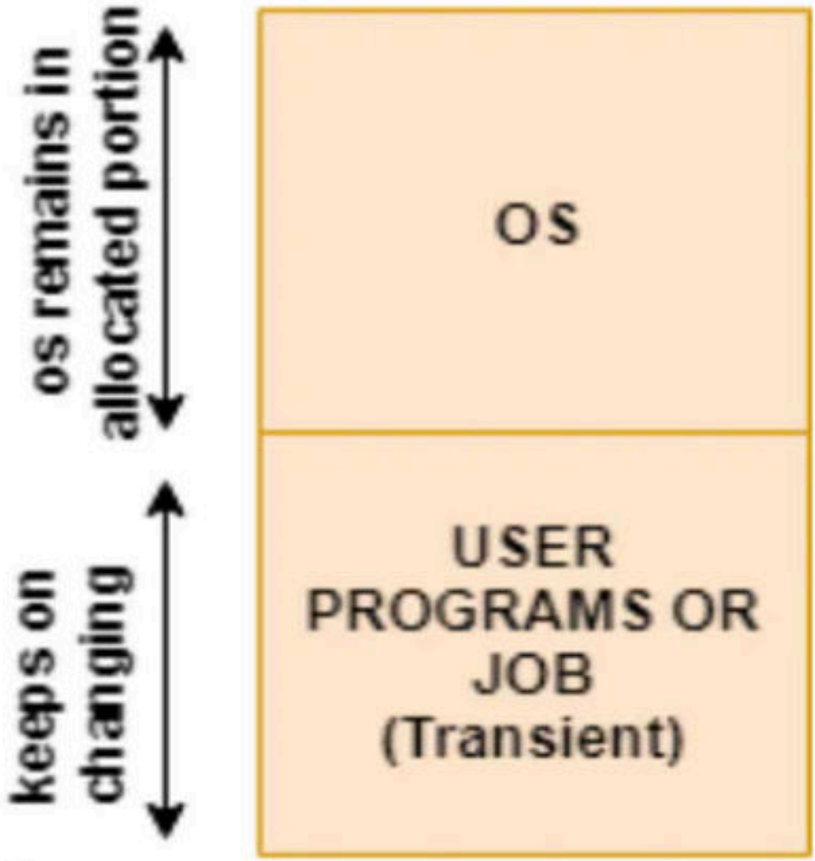
Advantage

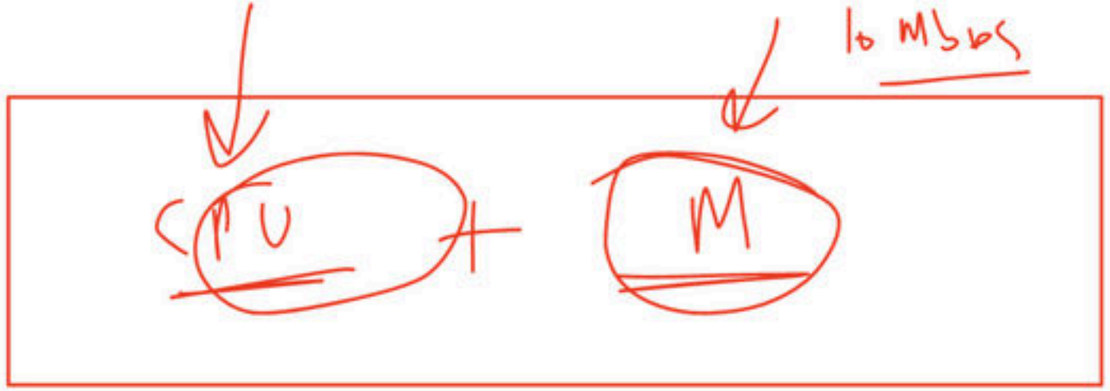
- The batched jobs were executed automatically one after another saving its time by performing the activities (like loading of compiler) only for once. It resulted in improved system utilization due to reduced turnaround time.

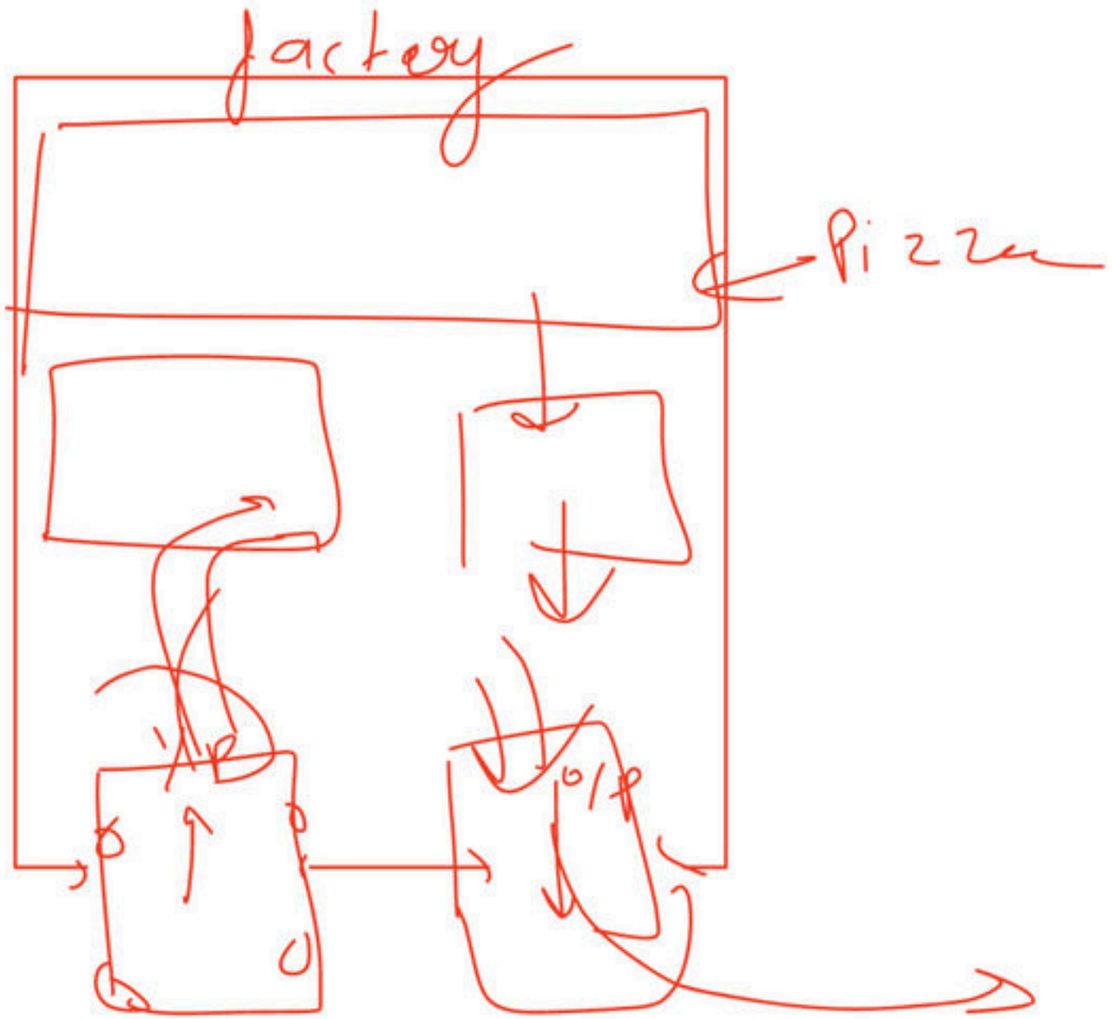
- Increased performance as a new job get started as soon as the previous job is finished, without any manual intervention

Disadvantage

- Memory limitation – memory was very limited, because of which interactive process or multiprogramming was not possible





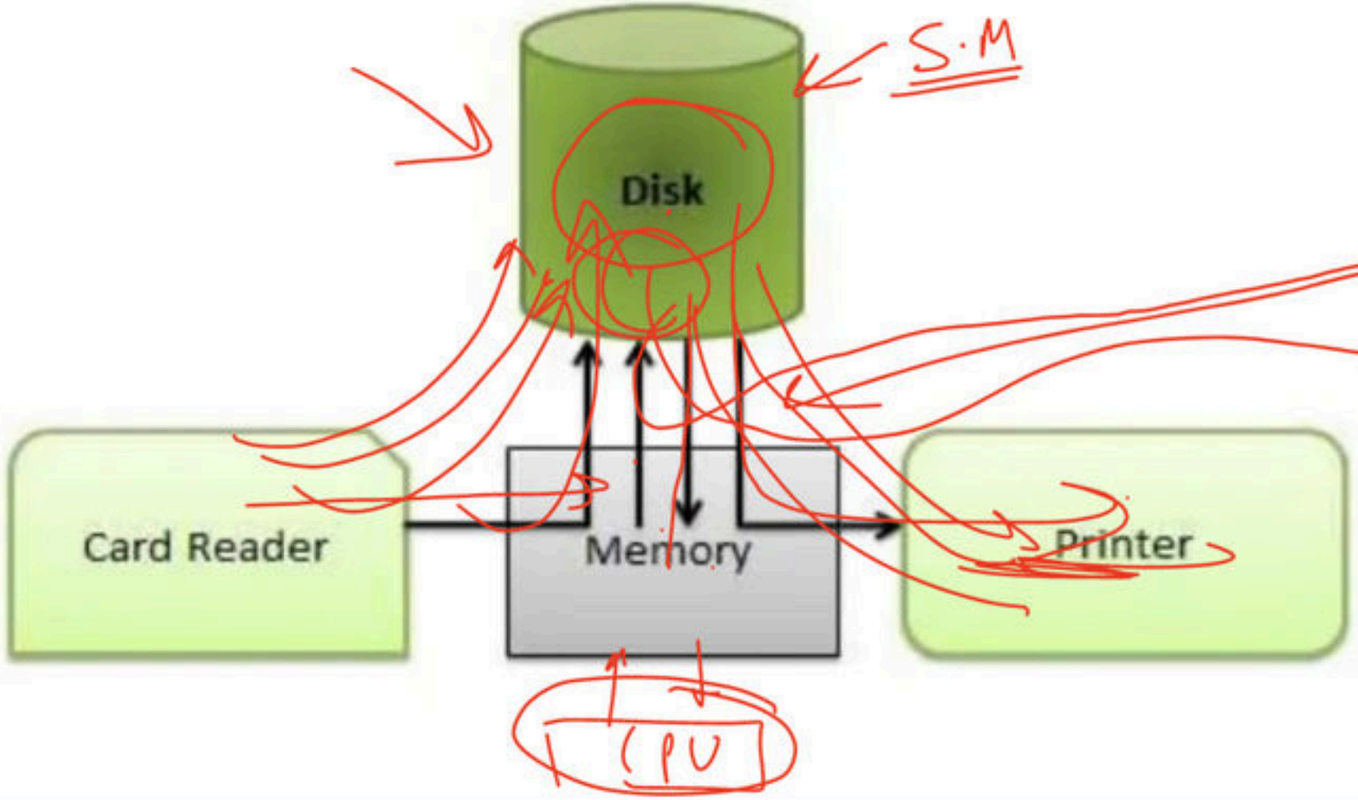


DY

Break

Spooling

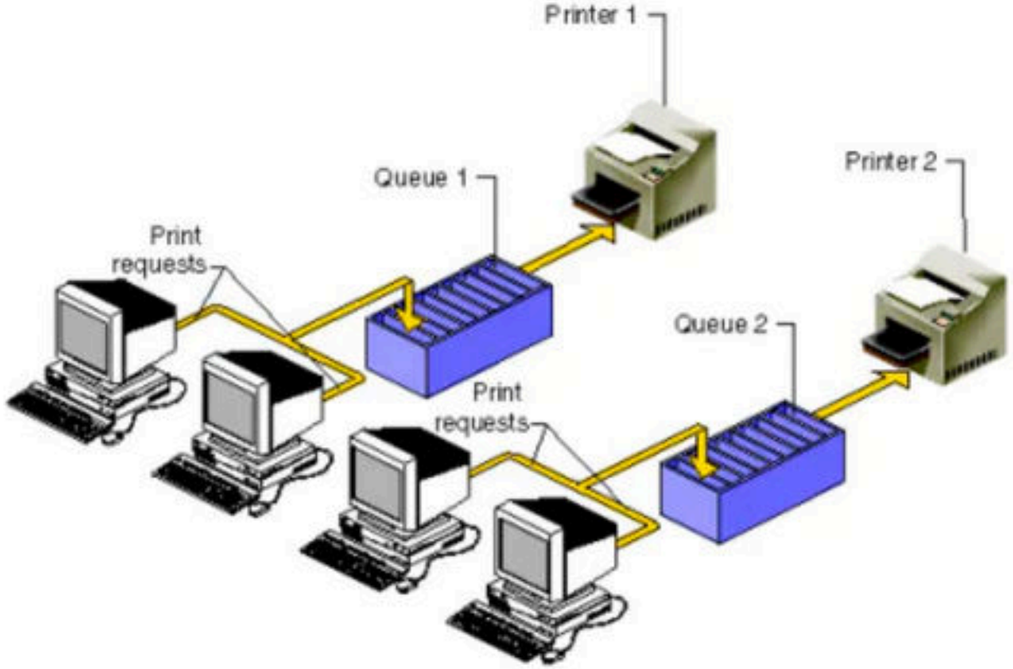
- ("Spool" is technically an acronym for simultaneous peripheral operations online.)
- In a computer system peripheral equipment, such as printers and punch card readers etc, are very slow relative to the performance of the rest of the system. Spooling is useful because devices access data at different rates.
- Spooling is a process in which data is temporarily held to be used and executed by a device, program or the system. Data is sent to and stored in memory or other volatile storage until the program or computer requests it for execution.
- Generally, the spool is maintained on the computer's physical memory, buffers or the I/O device-specific interrupts.



- The most common implementation of spooling can be found in typical input/output devices such as the keyboard, mouse and printer. For example, in printer spooling, the documents/files that are sent to the printer are first stored in the memory. Once the printer is ready, it fetches the data and prints it.

- A spooler works by intercepting the information going to the printer, parking it temporarily on disk or in memory. The computer can send the document information to the spooler at full speed, then immediately return control of the screen to you.

- The spooler, meanwhile, hangs onto the information and feeds it to the printer at the slow speed the printer needs to get it. So if your computer can **spool,** you can work while a document is being printed.



- Even experienced a situation when suddenly for some seconds your mouse or keyboard stops working? Meanwhile, we usually click again and again here and there on the screen to check if its working or not. When it actually starts working, what and wherever we pressed during its hang state gets executed very fast because all the instructions got stored in the respective device's spool.

- Spooling is capable of overlapping I/O operation for one job with processor operations for another job. i.e. multiple processes can write documents to a print queue without waiting and resume with their work.

Q Which of the following is an example of a spooled device? **(GATE-1996) (1 Marks)**

(A) a line printer used to print the output of a number of jobs

(B) a terminal used to enter input data to a running program

(C) a secondary storage device in a virtual memory system

(D) a graphic display device

Q which of the following is an example of a spooled device? **(GATE-1998) (1 Marks)**

(a) The terminal used to enter the input data for the C program being executed

(b) An output device used to print the output of a number of jobs

(c) The secondary memory device in a virtual storage system

(d) The swapping area on a disk used by the swapper

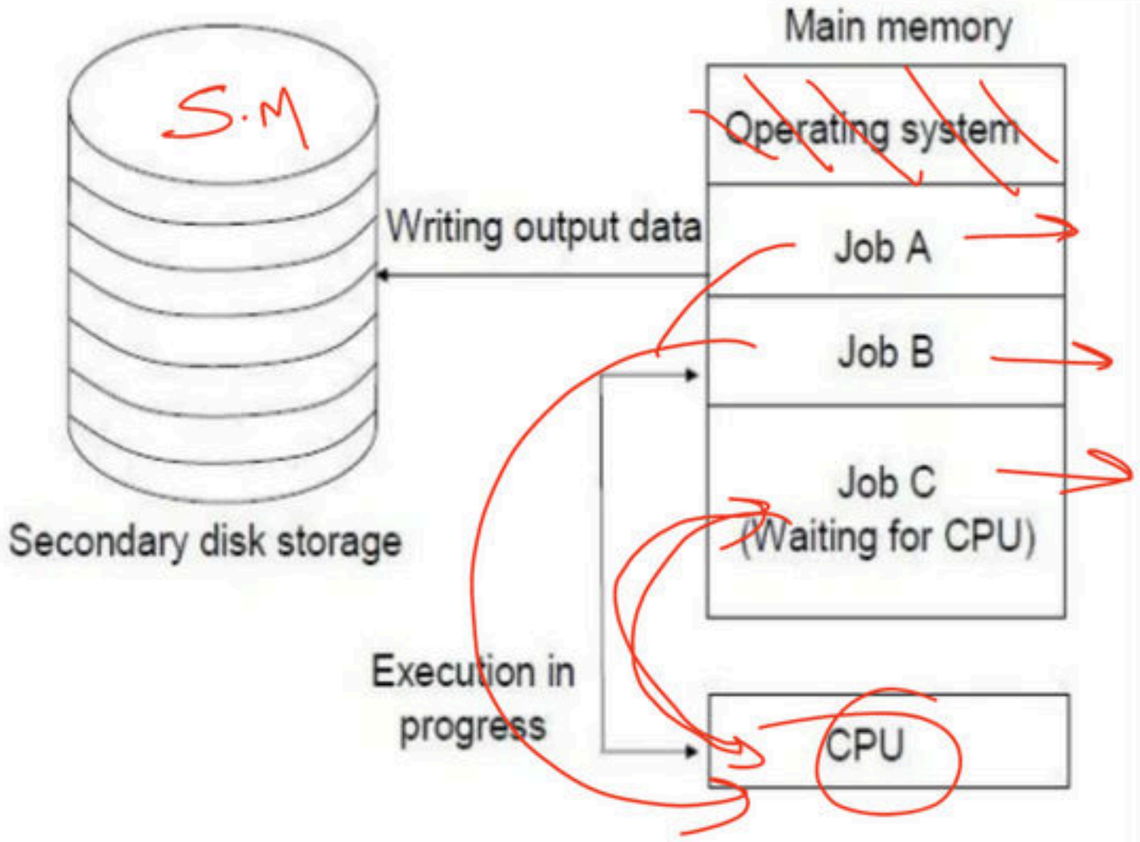
Break

Multiprogramming Operating System

- A single program cannot, in general, keep either the CPU or the I/O devices busy at all times. The basic idea of multiprogramming operating system is it keeps several jobs in main memory simultaneously.

- The jobs are kept initially on the disk in the **job pool**. This pool consists of all processes residing on disk awaiting allocation of main memory.

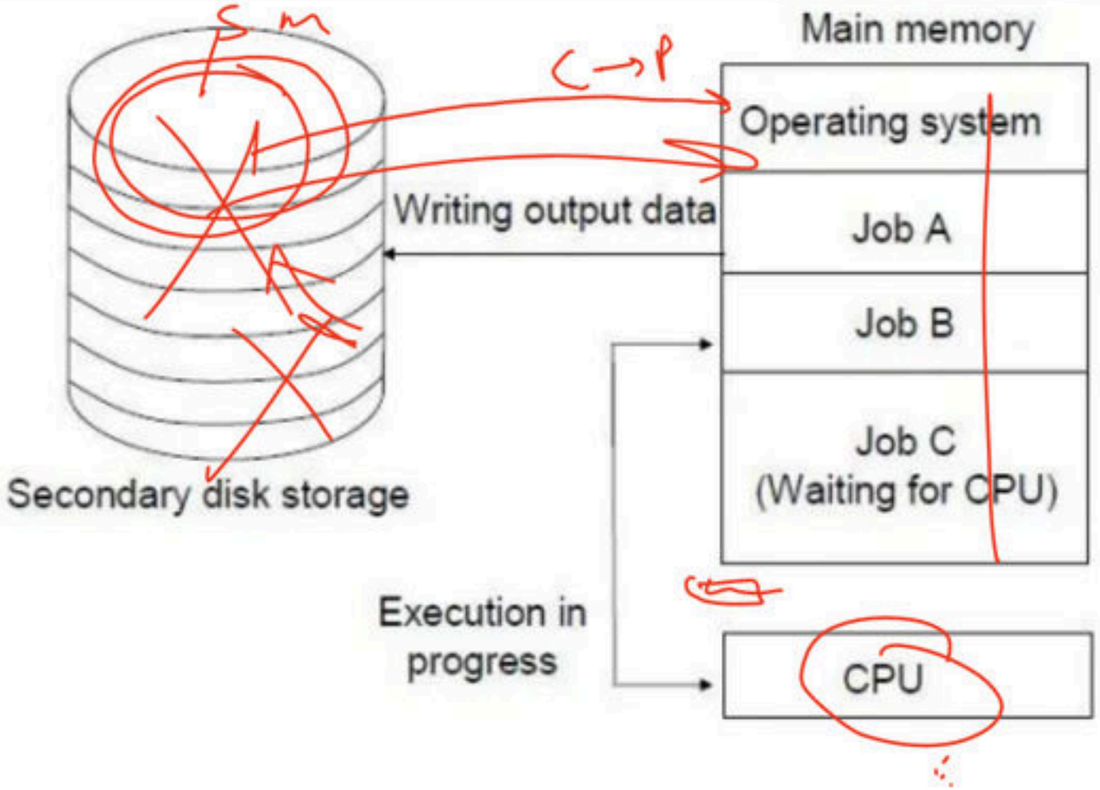
- The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation, to complete.



Show must go on

- In a non-multi-programmed system, the CPU would sit idle and wait but in a multi-programmed system, the operating system simply switches to, and executes, another job. When ***that*** job needs to wait the CPU switches to ***another*** job, and so on.

- Eventually, the first job finishes waiting and gets the CPU back. So, conclusion is as long as at least one job needs to execute, the CPU is never idle.



Processor किसी के लिए wait नहीं करेगा

Advantage

- High and efficient CPU utilization.
- Less response time or waiting time or turnaround time
- In most of the applications multiple tasks are running and multiprogramming systems better handle these type of applications
- Several processes share CPU time

Disadvantage

- It is difficult to program a system because of complicated schedule handling.
- To accommodate many jobs in main memory, complex memory management is required.

Q Which of the following features will characterize an OS as multi-programmed OS? **(NET-DEC-2012) (Gate-2002) (1 Marks)**

(a) More than one program may be loaded into main memory at the same time.

(b) If a program waits for certain event another program is immediately scheduled.

(c) If the execution of a program terminates, another program is immediately scheduled.

(A) (a) only

(B) (a) and (b) only

(C) (a) and (c) only

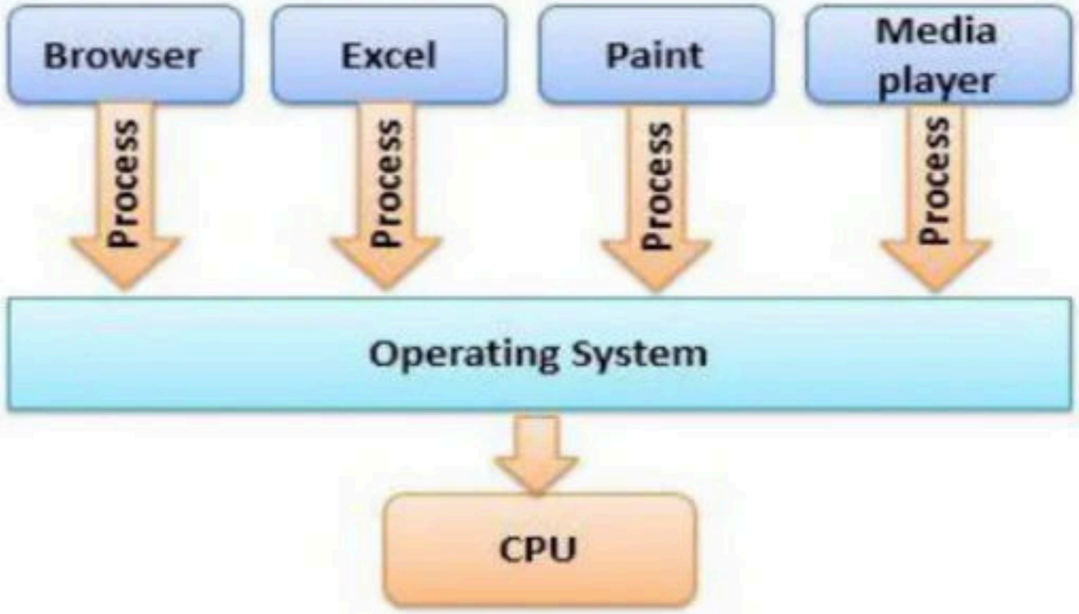
(D) (a), (b) and (c) only

Break

Multitasking Operating system/time sharing/Multiprogramming with Round Robin/ Fair Share

- In the modern operating systems, we are able to play MP3 music, edit documents in Microsoft Word, surf the Google Chrome all running at the same time. (by context switching, the illusion of parallelism is achieved)

- For multitasking to take place, firstly there should be multiprogramming i.e. presence of multiple programs ready for execution. And secondly the concept of time sharing.



Multitasking

- Time sharing (or multitasking) is a logical extension of multiprogramming, it allows many users to share the computer simultaneously. the CPU executes multiple jobs (May belong to different user) by switching among them, but the switches occur so frequently that, each user is given the impression that the entire computer system is dedicated to his/her use, even though it is being shared among many users.

Q If you want to execute more than one program at a time, the systems software that are used must be capable of: **(NET-DEC-2005)**

(A) word processing

(B) virtual memory

(C) compiling

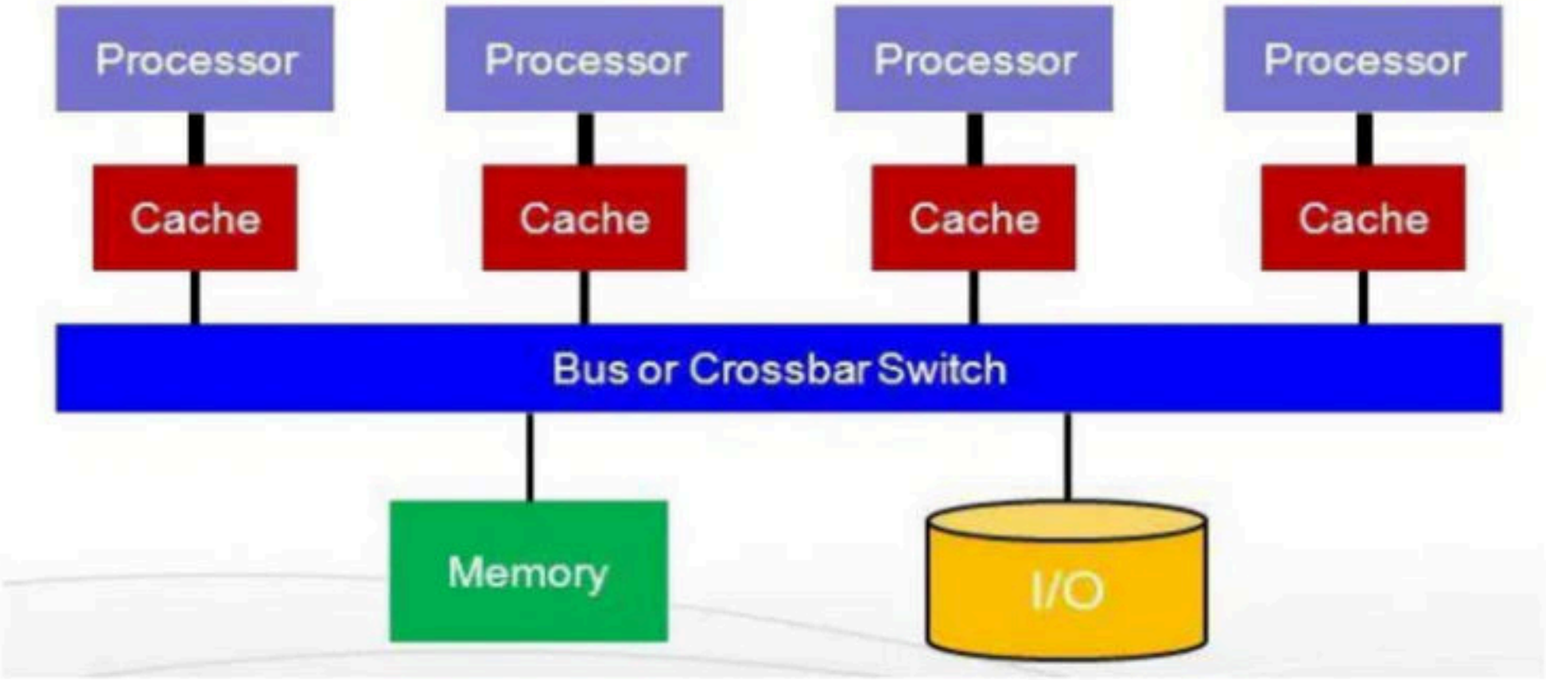
(D) multitasking

Break

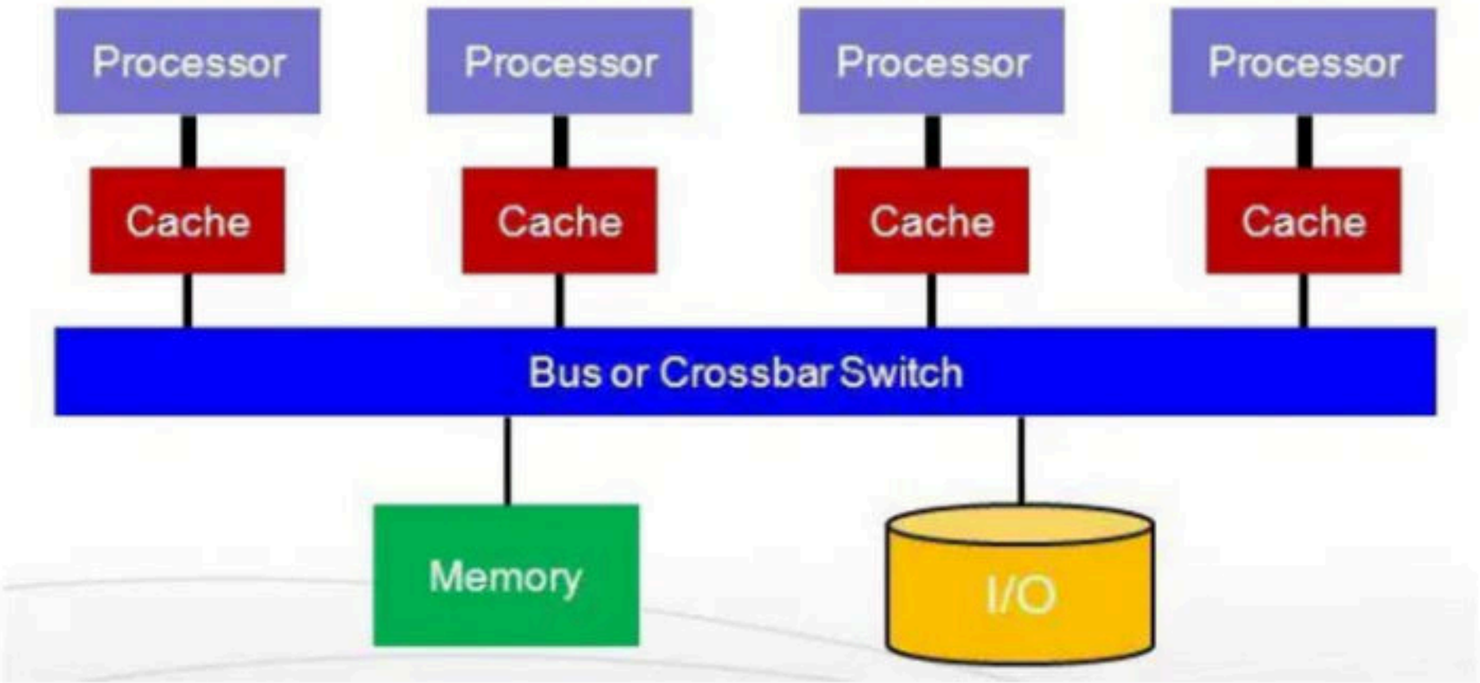
Multiprocessing Operating System/ tightly coupled system

- Multiprocessor Operating System refers to the use of two or more central processing units (CPU) within a single computer system. These multiple CPUs are in a close communication sharing the computer bus, memory and other peripheral devices. These systems are referred as tightly coupled systems

- So in multiprocessing, multiple concurrent processes each can run on a separate CPU, here we achieve a true parallel execution of processes. Parallel processing is the ability of the CPU to simultaneously process incoming jobs.



- Multiprocessing becomes most important in computer system, where the complexity of the job is more, and CPU divides and conquers the jobs. Generally, the parallel processing is used in the fields like artificial intelligence and expert system, image processing, weather forecasting etc.

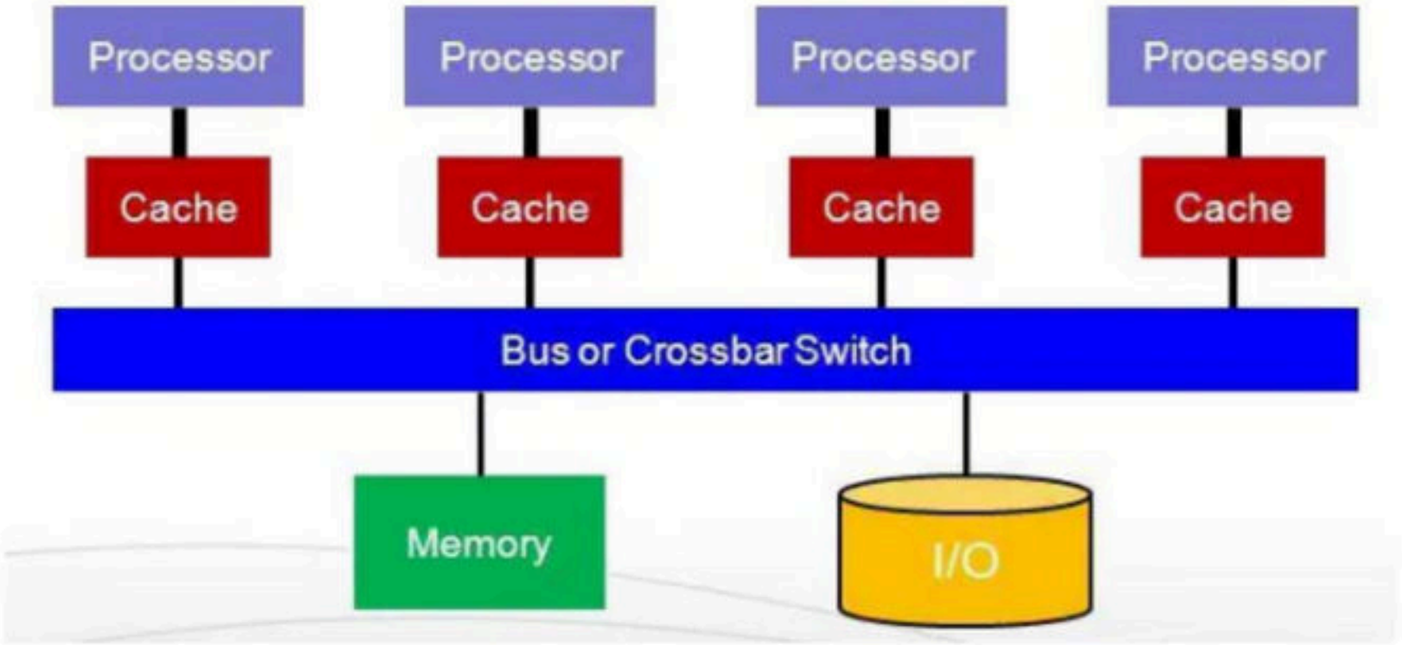


Multiprocessing can be of two types

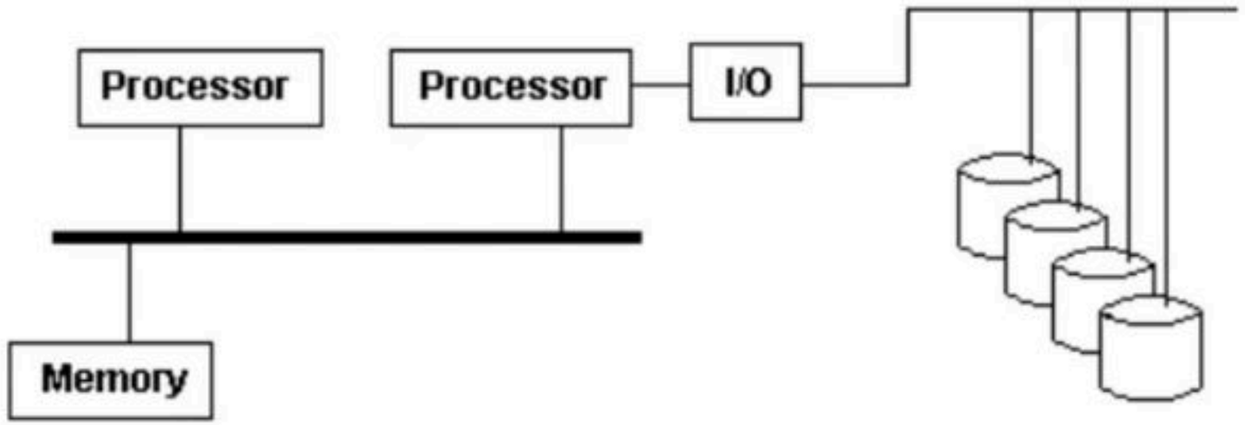
- **Symmetric multiprocessing** - In a symmetric multi-processing, a single OS instance controls two or more identical processors connected to a single shared main memory.

- Most of the multi-processing PC motherboards utilize symmetric multiprocessing. Here each processor runs an identical copy of operating system and these copies communicate with each other.

- SMP means that all processors are peers; no boss–worker relationship exists between processors. Windows, Mac OSX and Linux.



- **Asymmetric** - This scheme defines a master-slave relationship, where one processor behaves as a master and control other processor which behaves as slaves.
- For e.g. It may require that only one particular CPU respond to all hardware interrupts, whereas all other work in the system may be distributed equally among CPUs;
- Or execution of kernel-mode code may be restricted to only one particular CPU, whereas user-mode code may be executed in any combination of processors.
- Multiprocessing systems are often easier to design if such restrictions are imposed, but they tend to be less efficient than systems in which all CPUs are utilized.
- A boss processor controls the system, other processors either look to the boss for instruction or have predefined tasks. This scheme defines a boss–worker relationship. The boss processor schedules and allocates work to the worker processors.



Advantage of multiprocessing

•**Increased Throughput** - By increasing the number of processors, we expect to get more work done in less time. The speed-up ratio with *N* processors is not *N,* however; rather, it is less than *N.* When multiple processors cooperate on a task, a certain amount of overhead is incurred in keeping all the parts working correctly. This overhead, plus contention for shared resources, lowers the expected gain from additional processors. These types of systems are used when very high speed is required to process a large volume of data.

•**Economy of Scale** - Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies. If several programs operate on the same set of data, it is cheaper to store those data on one disk and to have all the processors share them than to have many computers with local disks and many copies of the data.

•**Increased Reliability (fault tolerance)** - If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down. If we have ten processors and one fail, then each of the remaining nine processors can pick up a share of the work of the failed processor. Thus, the entire system runs only 10 percent slower, rather than failing altogether

•**Less Battery consumption and heat generation**

Disadvantages

- It’s more complex than simple operating systems.
- It requires context switching which may impacts performance.
- If one processor fails then it will affect in the speed
- large main memory required

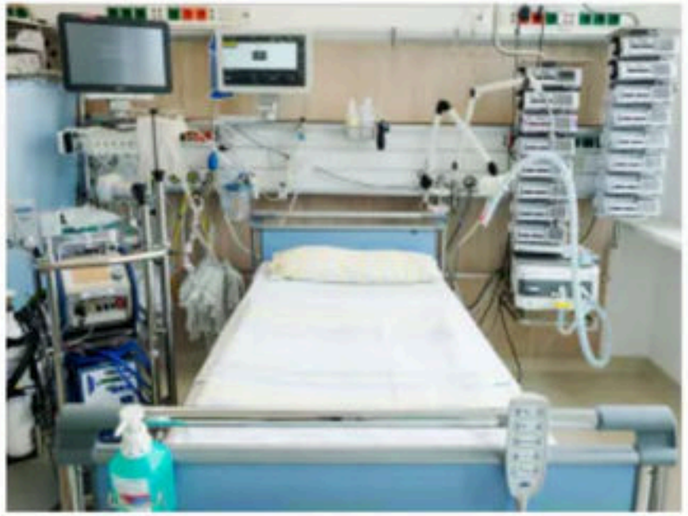
Break

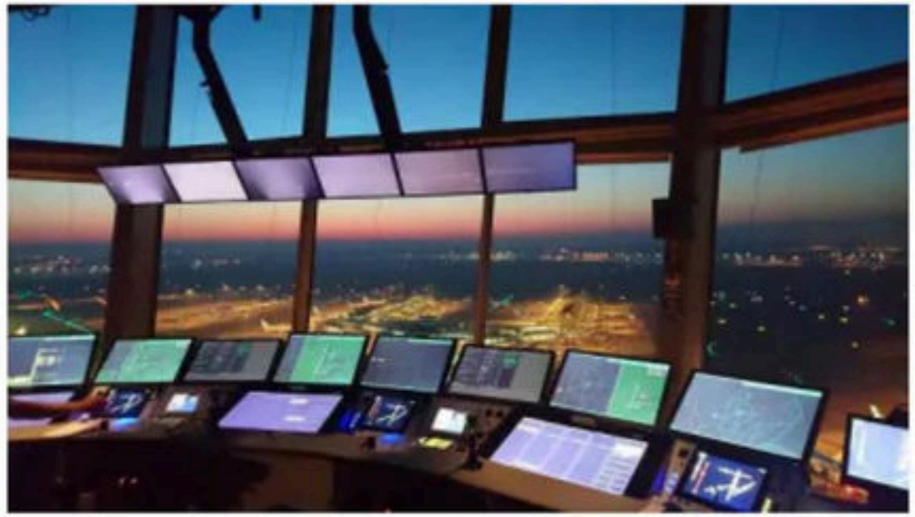
Real time Operating system

- A real time system is a time bound system which has well defined fixed time constraints. Processing must be done within the defined constraints or the system will fail. Very fast and quick respondent systems.

- RTOS intended to serve real-time applications that process data as it comes in, typically without buffer delays. Are used in an environment where a large number of events (generally external) must be accepted and processed in a short time.

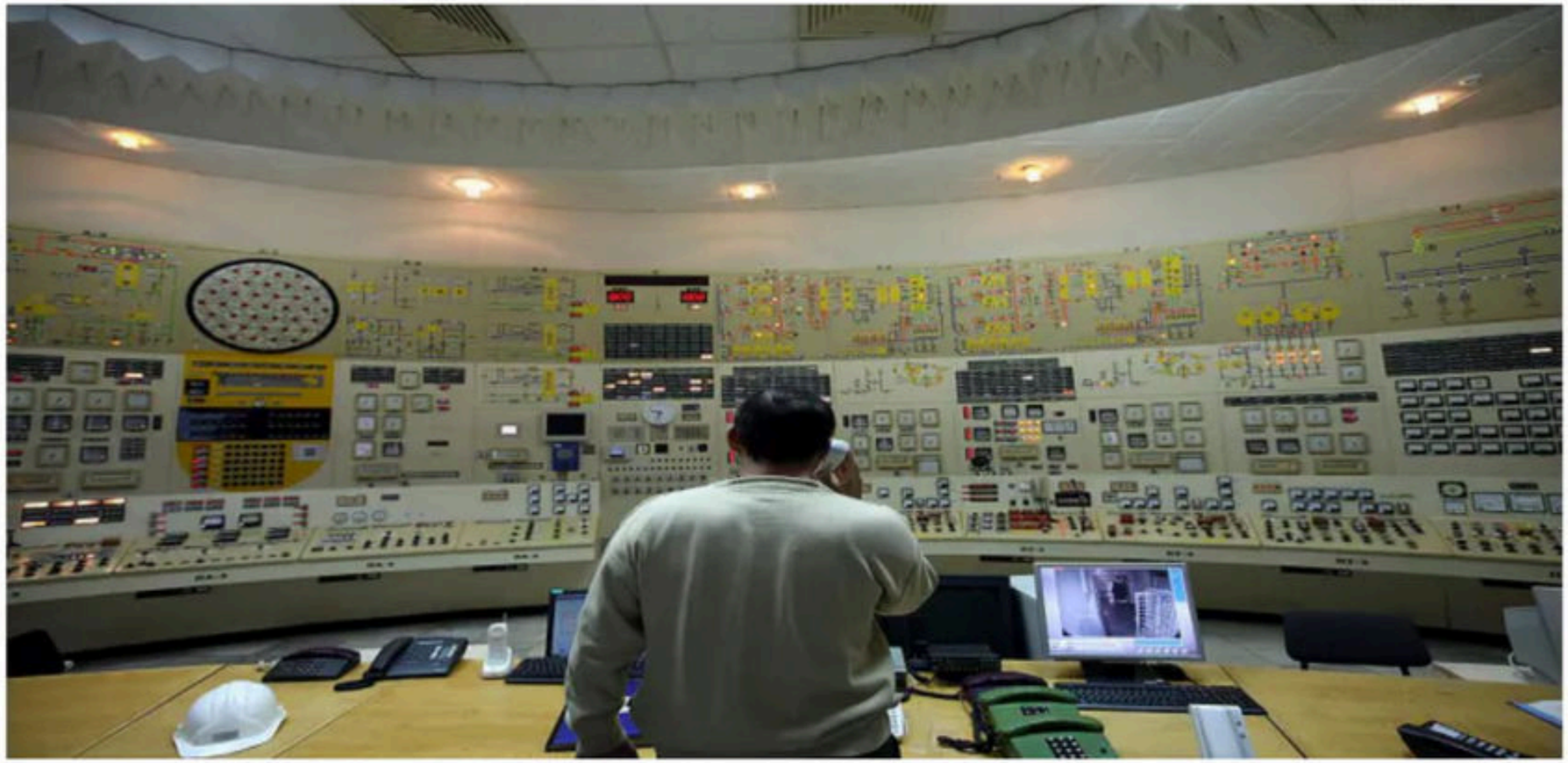
- For example, a measurement from a petroleum refinery indicating that temperature is getting too high and might demand for immediate attention to avoid an explosion. Airlines reservation system, Air traffic control system, Systems that provide immediate updating, Systems that provide up to the minute information on stock prices, Defense application systems like as RADAR.



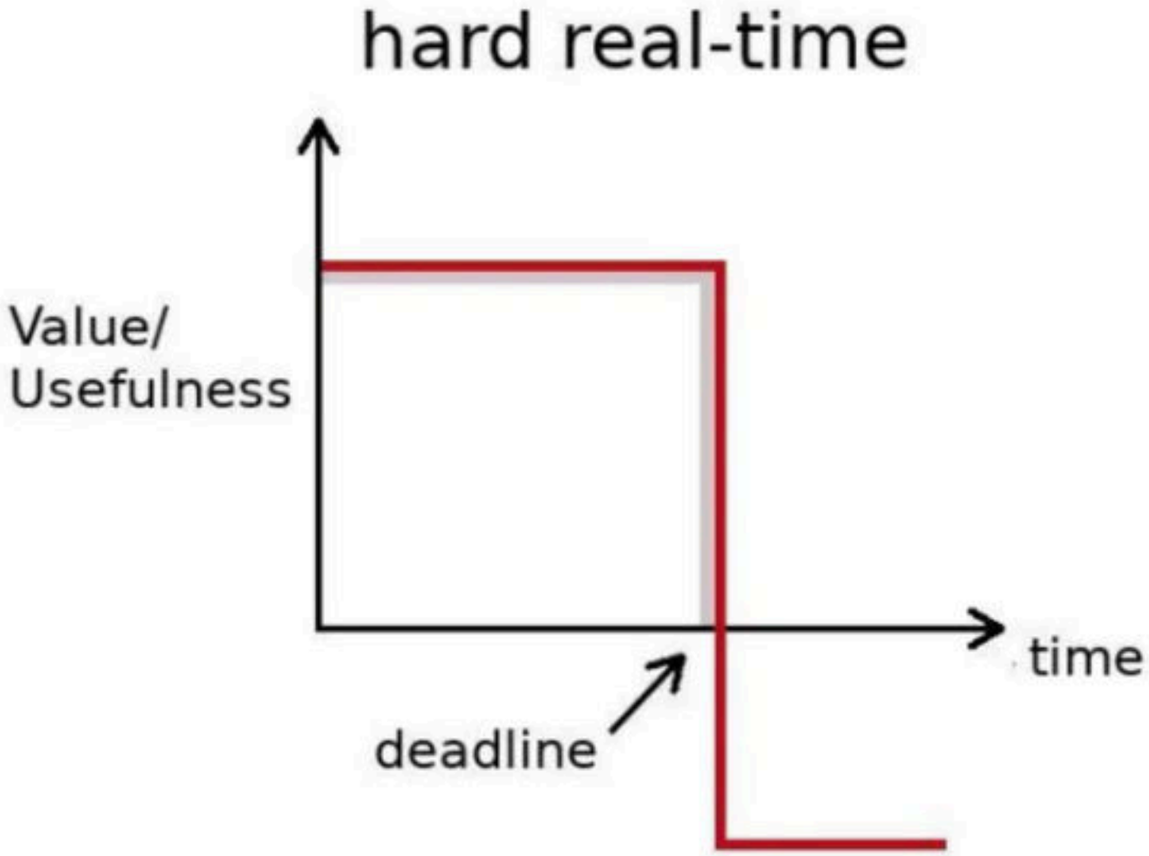


- Valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.

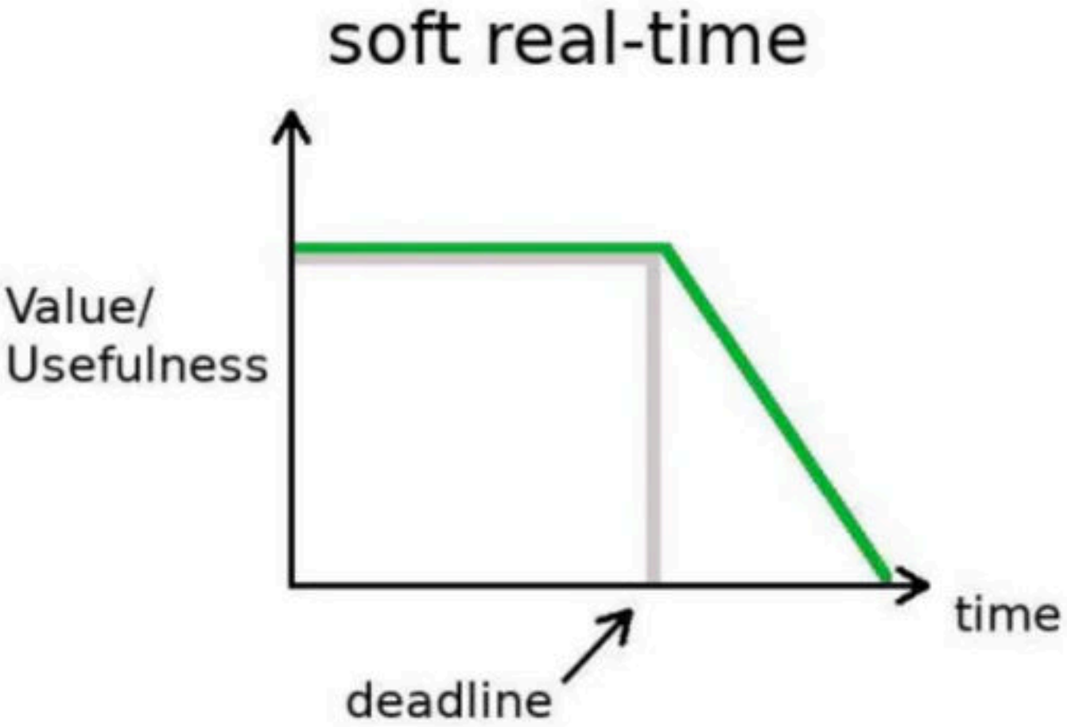
- Little swapping of programs between primary and secondary memory. Most of the time, processes remain in primary memory in order to provide quick response, therefore, memory management in real time system is less demanding compared to other systems.



- **Hard real-time operating system** - This is also a type of OS and it is predicted by a deadline. The predicted deadlines will react at a time t = 0. Some examples of this operating system are air bag control in cars, anti-lock brake, and engine control system etc.



- There are three different types of RTOS which are following
  - **Soft real-time operating system** - The soft real-time operating system has certain deadlines, may be missed and they will take the action at a time t=0+. The soft real-time operating system is a type of OS and it does not contain constrained to extreme rules. The critical time of this operating system is delayed to some extent. The examples of this operating system are the digital camera, mobile phones and online data etc.



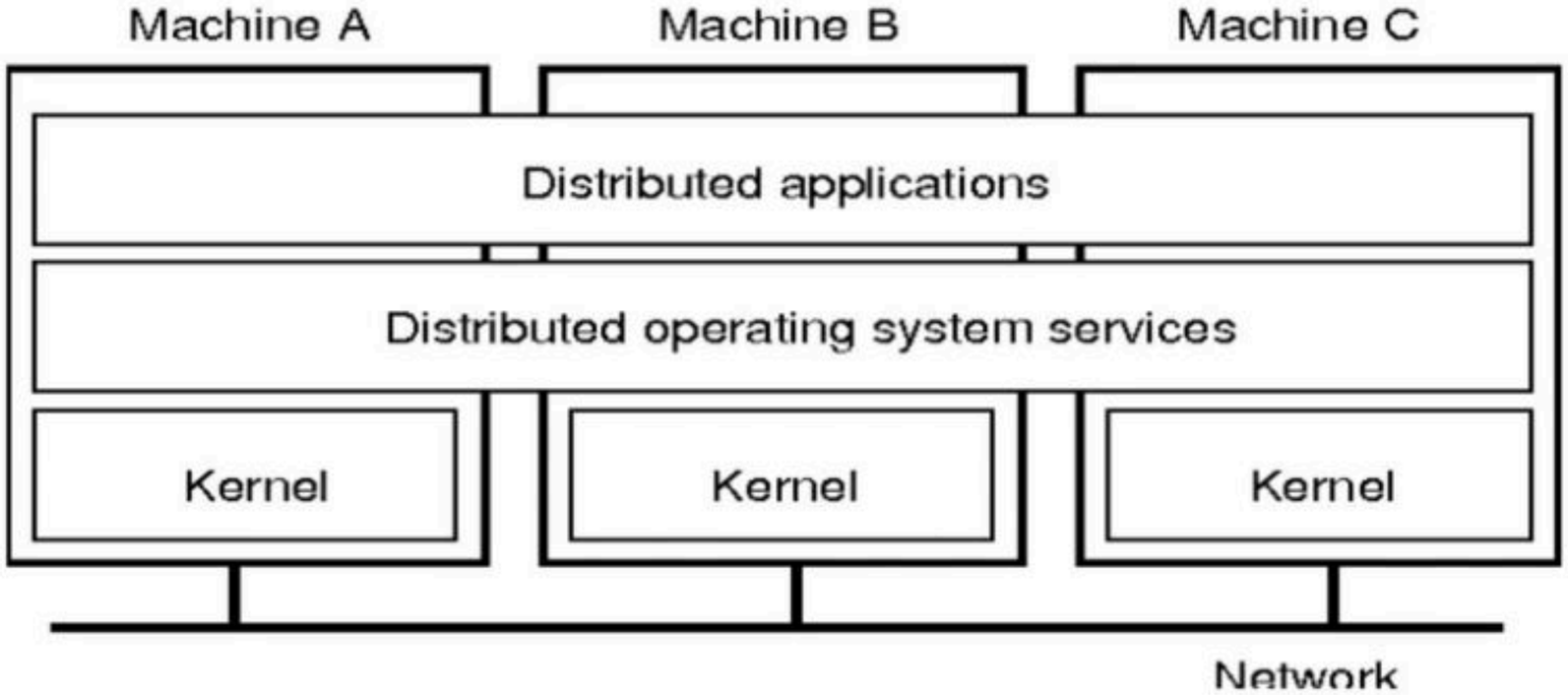
Break

Distributed OS

- A **distributed operating system** is a software over a collection of independent, networked, communicating, loosely coupled nodes and physically separate computational nodes.

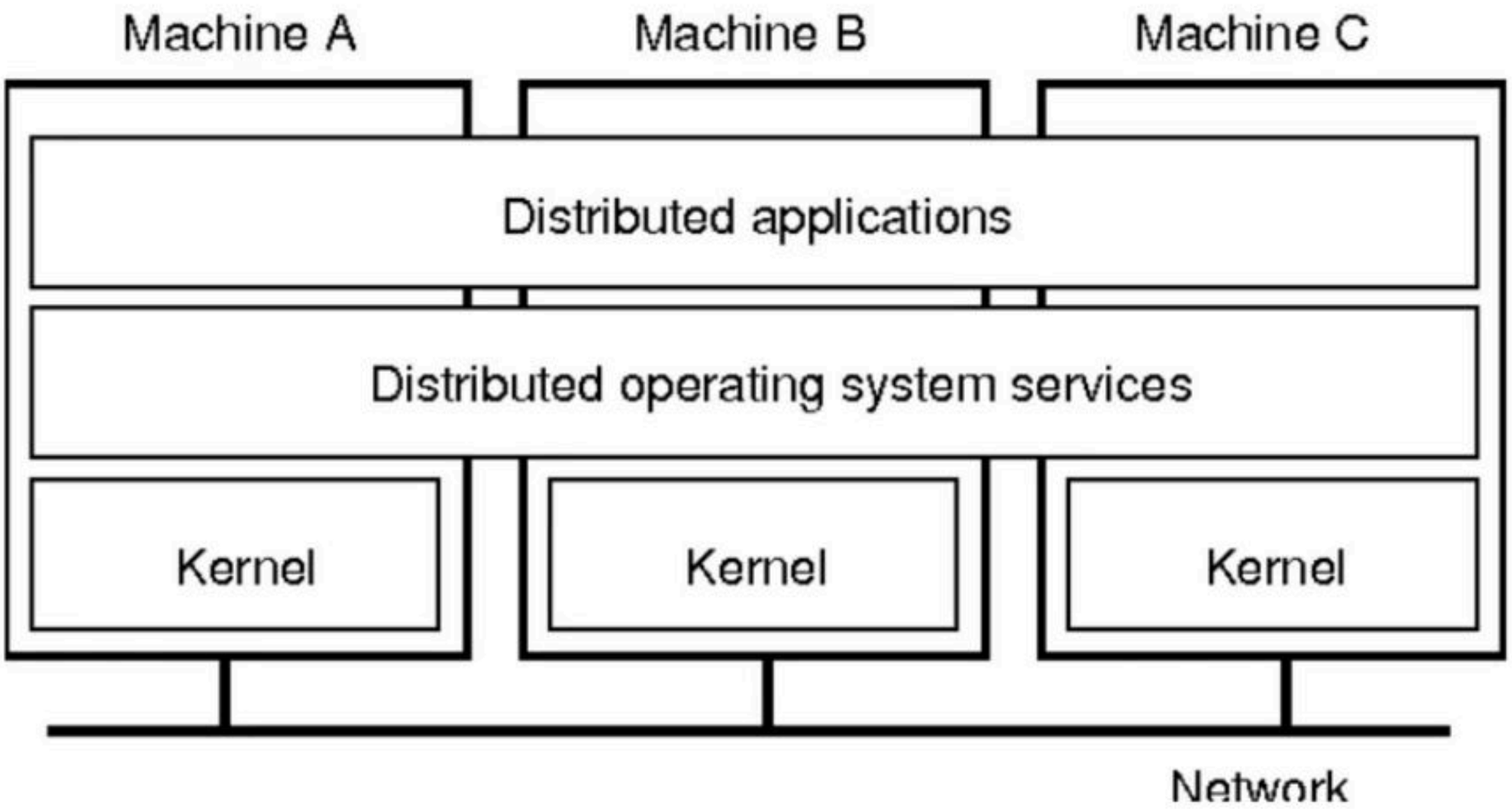
- These modes do not share memory or a clock. Instead, each node has its own local memory. The nodes communicate with one another through various networks, such as high-speed buses and the Internet.

- They handle jobs which are serviced by multiple CPUs. Each individual node holds a specific software subset of the global aggregate operating system.



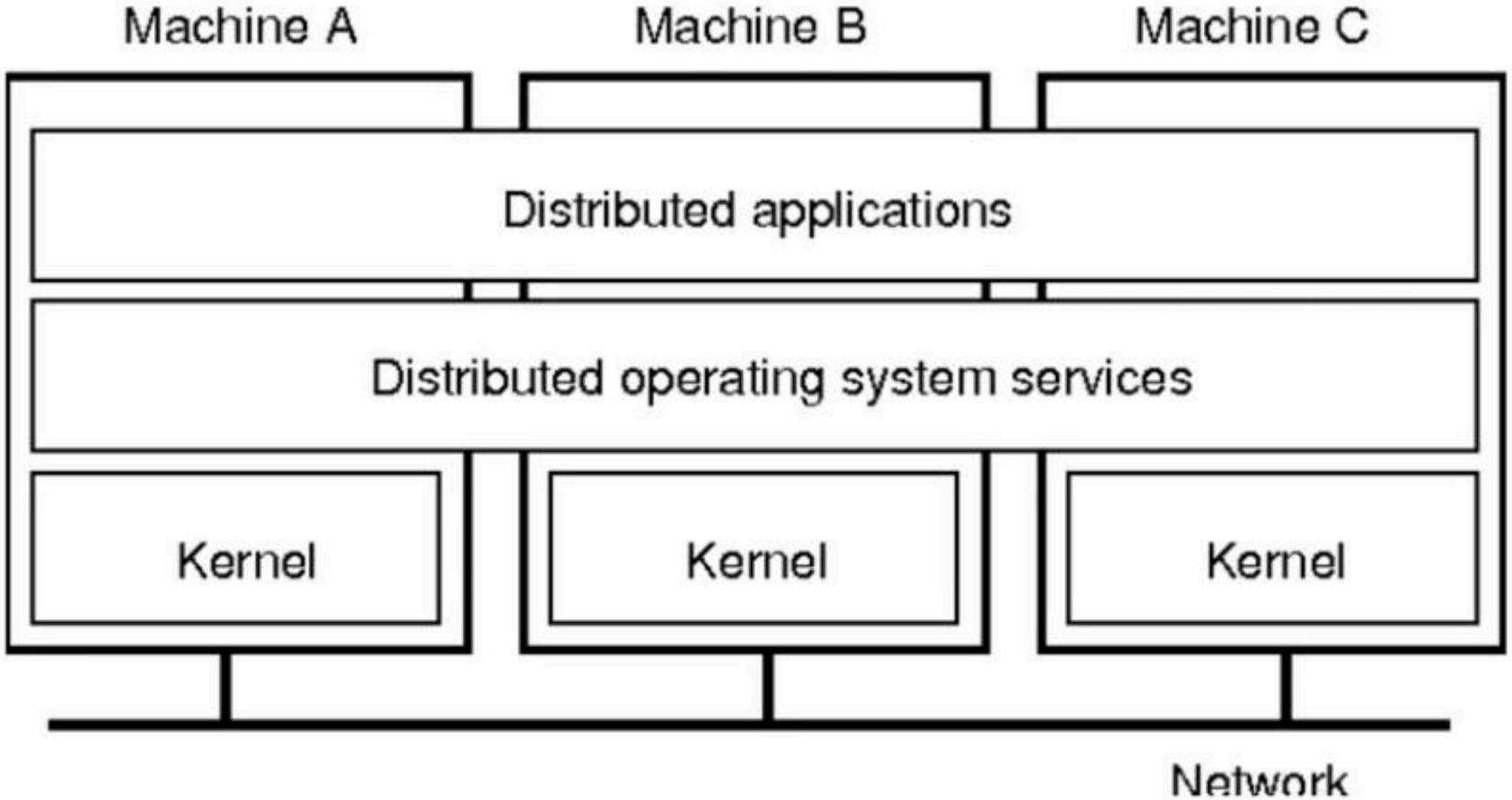
- Each subset is a composite of two distinct service provisioners. The first is a ubiquitous minimal kernel, or microkernel, that directly controls that node's hardware.

- Second is a higher-level collection of *system management components* that coordinate the node's individual and collaborative activities. These components abstract microkernel functions and support user applications.



- The microkernel and the management components collection work together. They support the system's goal of integrating multiple resources and processing functionality into an efficient and stable system.

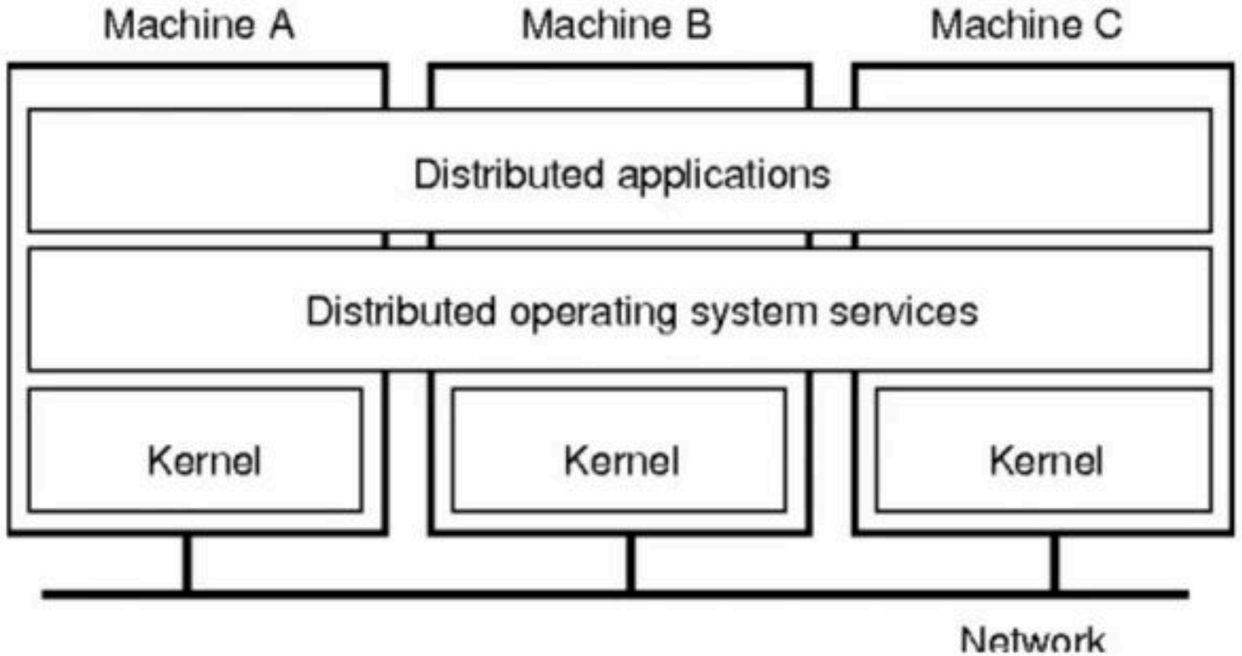
- This seamless integration of individual nodes into a global system is referred to as *transparency*, or *single system image*; describing the illusion provided to users of the global system's appearance as a single computational entity.



- To a user, a distributed OS works in a manner similar to a single-node, monolithic operating system. That is, although it consists of multiple nodes, it appears to users and applications as a single-node.

- There are four major reasons for building distributed systems: resource sharing, computation speedup, reliability, and communication.

- E.g. Plan 9 from Bell Labs, **Inferno**



Q Match the following: **(NET-June-2015)**

List - I	List - II
(1) Spooling	(i) Allows several jobs in memory to improve CPU utilization
(2) Multiprogramming	(ii)Access to shared resources among geographically dispersed computers in a transparent way
(3)Time sharing	(iii) Overlapping I/O and computations
(4)Distributed computing	(iv)Allows many users to share a computer simultaneously by switching processor frequently

	(1)	(2)	(3)	(4)
a)	iii	i	ii	iv
b)	iii	i	iv	ii
c)	iv	iii	ii	i
d)	ii	iii	iv	i

Break

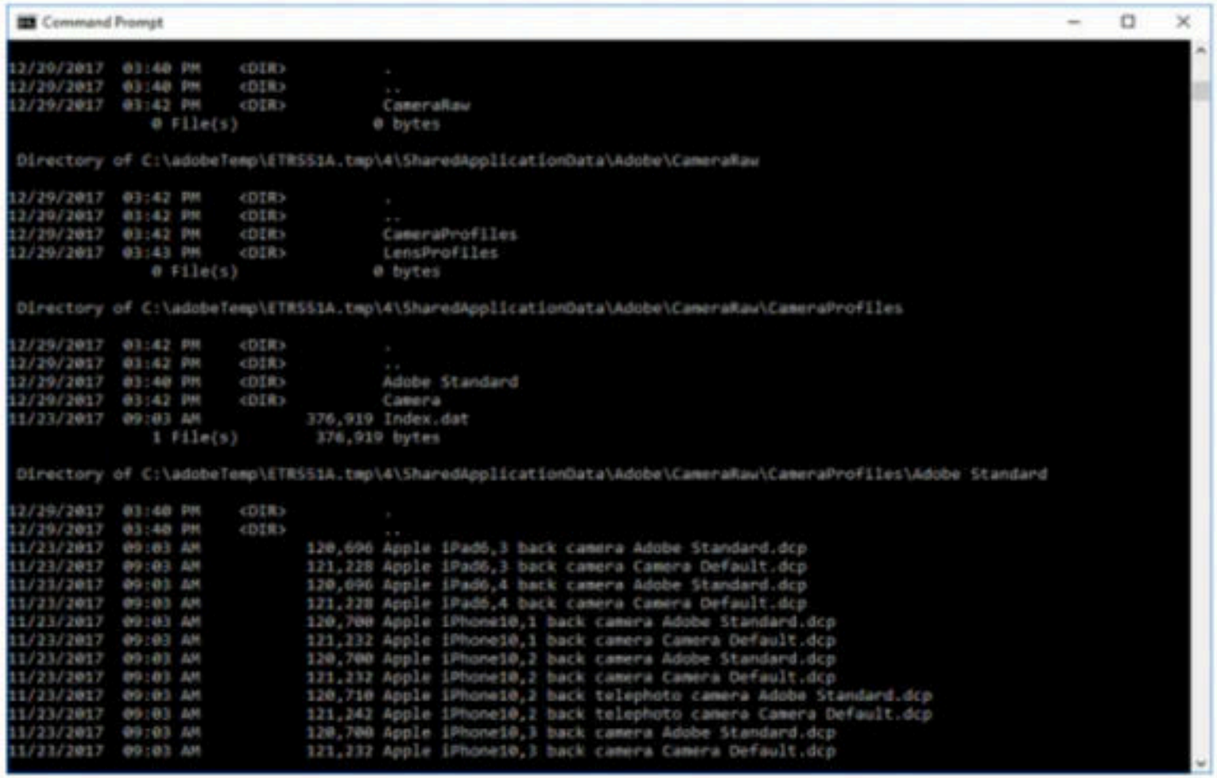
User and Operating-System Interface

- There are several ways for users to interface with the operating system. Here, we discuss two fundamental approaches.
- One provides a command-line interface, or command interpreter, that allows users to directly enter commands to be performed by the operating system.
- The other allows users to interface with the operating system via a graphical user interface, or GUI.

- **Command Interpreters** - Some operating systems include the command interpreter in the kernel.

- Others, such as Windows and UNIX, treat the command interpreter as a special program that is running when a job is initiated or when a user first logs on (on interactive systems).

- On systems with multiple command interpreters to choose from, the interpreters are known as shells. For example, on UNIX and Linux systems, a user may choose among several different shells, including the *Bourne shell*, *C shell*, *Bourne-Again shell*, *Korn shell*, and others.



- **Graphical User Interfaces** - A second strategy for interfacing with the operating system is through a user- friendly graphical user interface, or GUI.

- Here, users employ a mouse-based window- and-menu system characterized by a desktop.

- The user moves the mouse to position its pointer on images, or icons, on the screen (the desktop) that represent programs, files, directories, and system functions.

- Depending on the mouse pointer's location, clicking a button on the mouse can invoke a program, select a file or directory—known as a folder —or pull down a menu that contains commands.

- Because a mouse is impractical for most mobile systems, smartphones and handheld tablet computers typically use a touchscreen interface.

- Here, users interact by making gestures on the touchscreen—for example, pressing and swiping fingers across the screen.

- The choice of whether to use a command-line or GUI interface is mostly one of personal preference.

- System administrators who manage computers and power users who have deep knowledge of a system frequently use the command-line interface. For them, it is more efficient, giving them faster access to the activities they need to perform.

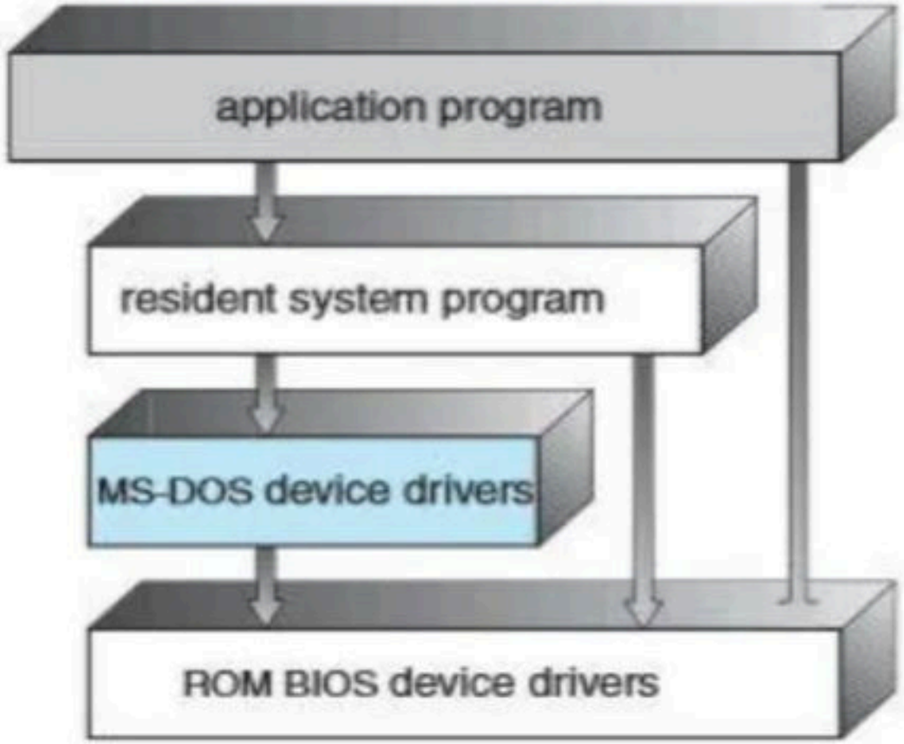
- Indeed, on some systems, only a subset of system functions is available via the GUI, leaving the less common tasks to those who are command-line knowledgeable.

Break

Structure of Operating System

- A common approach is to partition the task into small components, or modules, rather than have one monolithic system. Each of these modules should be a well-defined portion of the system, with carefully defined inputs, outputs, and functions.

- **Simple Structure** - Many operating systems do not have well-defined structures. Frequently, such systems started as small, simple, and limited systems and then grew beyond their original scope. MS-DOS is an example of such a system.



MS-DOS layer structure.

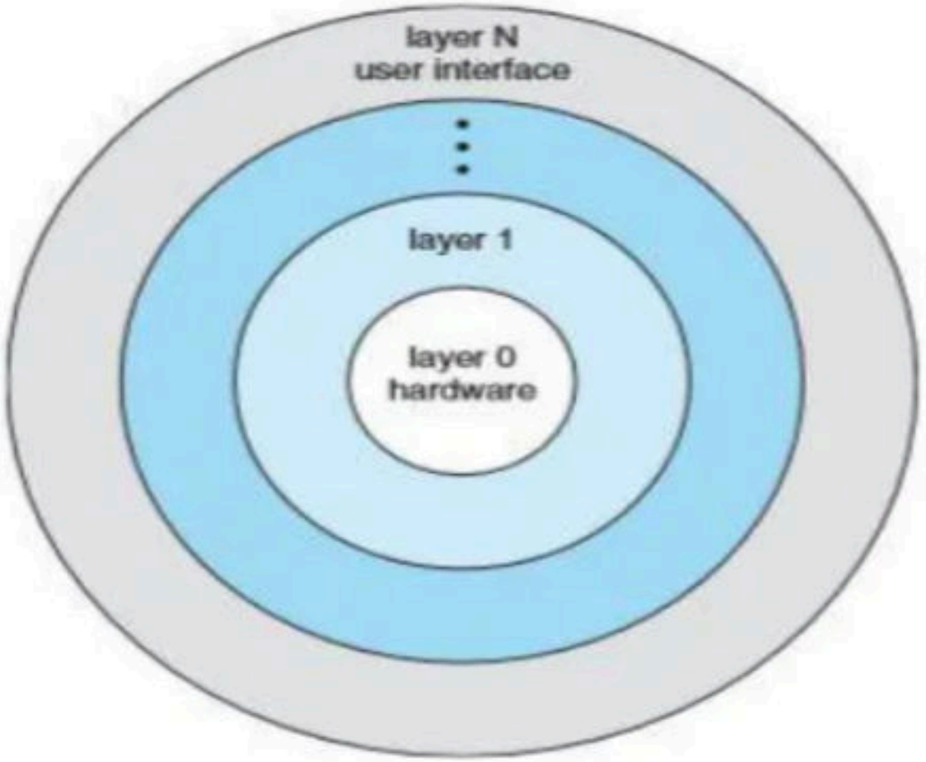
- **Layered Approach** - With proper hardware support, operating systems can be broken into pieces. The operating system can then retain much greater control over the computer and over the applications that make use of that computer.

- Implementers have more freedom in changing the inner workings of the system and in creating modular operating systems.

- Under a top-down approach, the overall functionality and features are determined and are separated into components.

- Information hiding is also important, because it leaves programmers free to implement the low-level routines as they see fit.

- A system can be made modular in many ways. One method is the layered approach, in which the operating system is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer *N*) is the user interface.

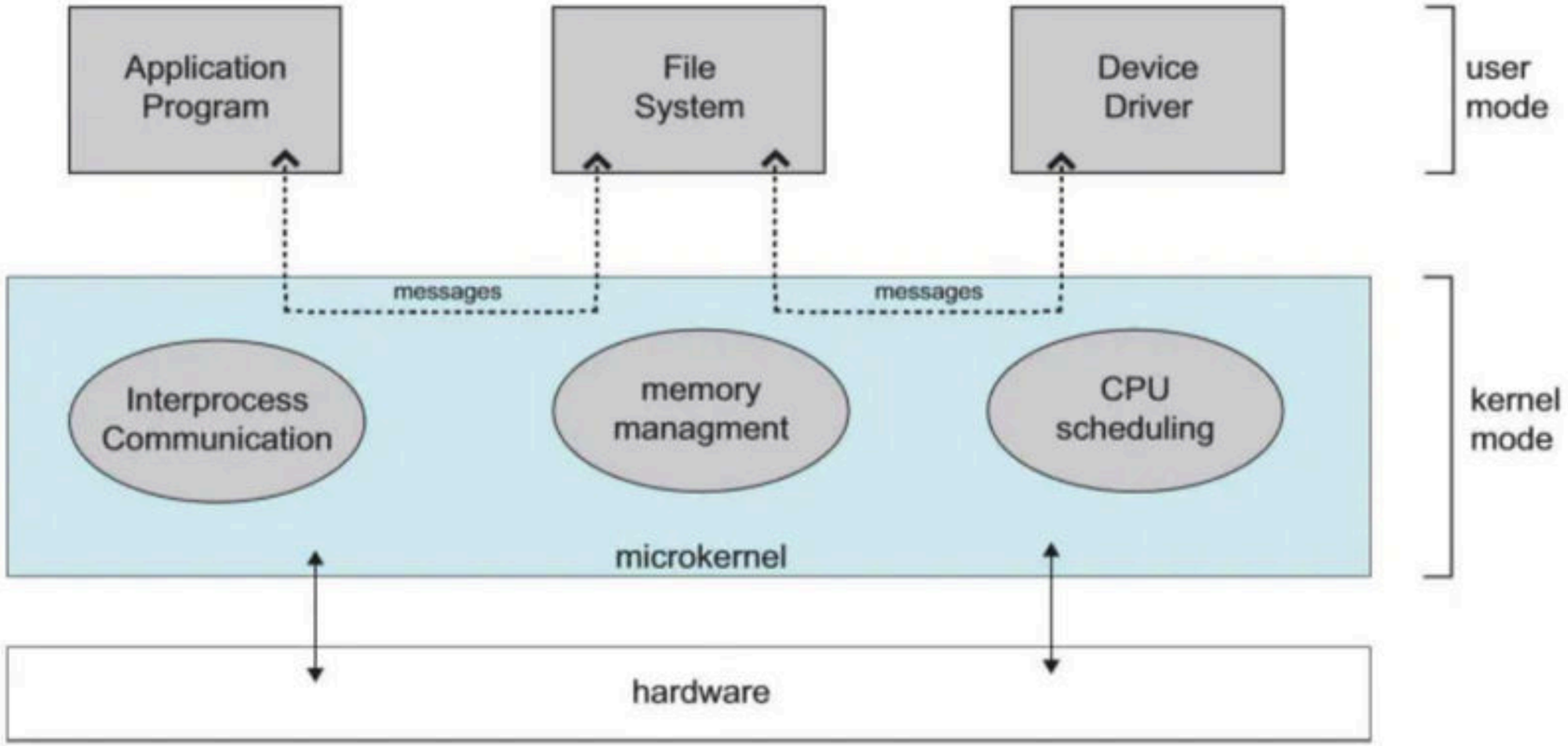


A layered operating system.

Break

Micro-Kernel approach

- In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called **Mach** that modularized the kernel using the **microkernel** approach.
- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel.



- One benefit of the microkernel approach is that it makes extending the operating system easier. All new services are added to user space and consequently do not require modification of the kernel.

- When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel.

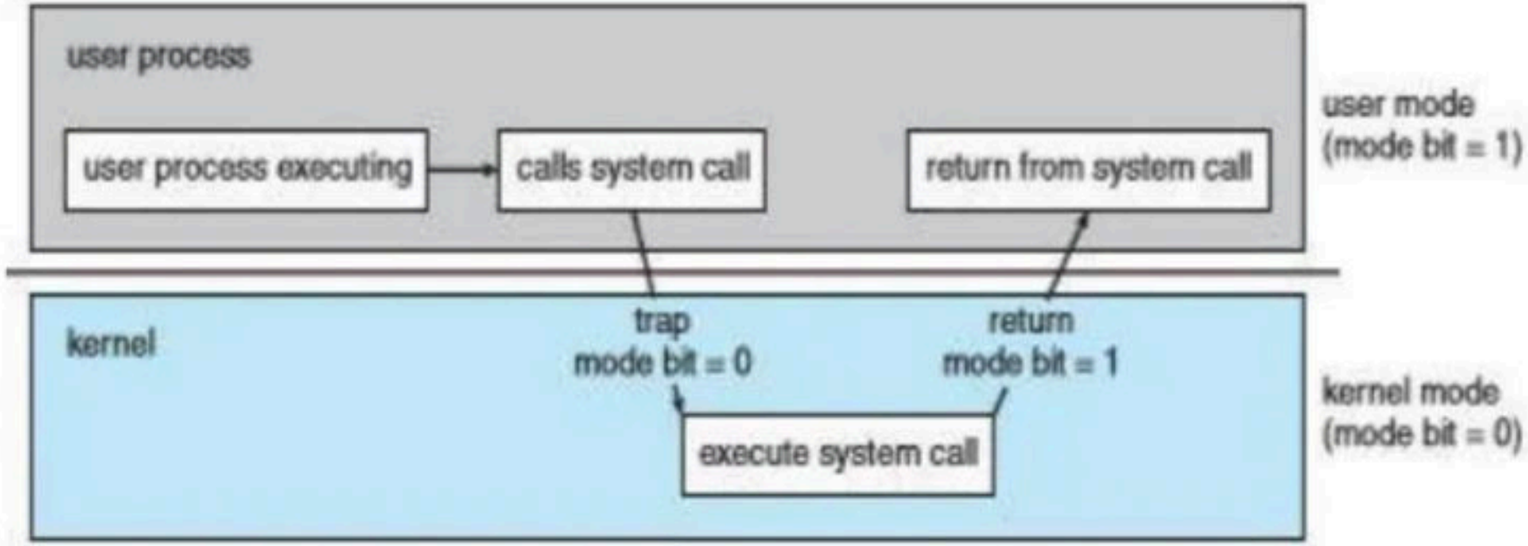
- The MINIX 3 microkernel, for example, has only approximately 12,000 lines of code.

- Developer Andrew S. Tanenbaum

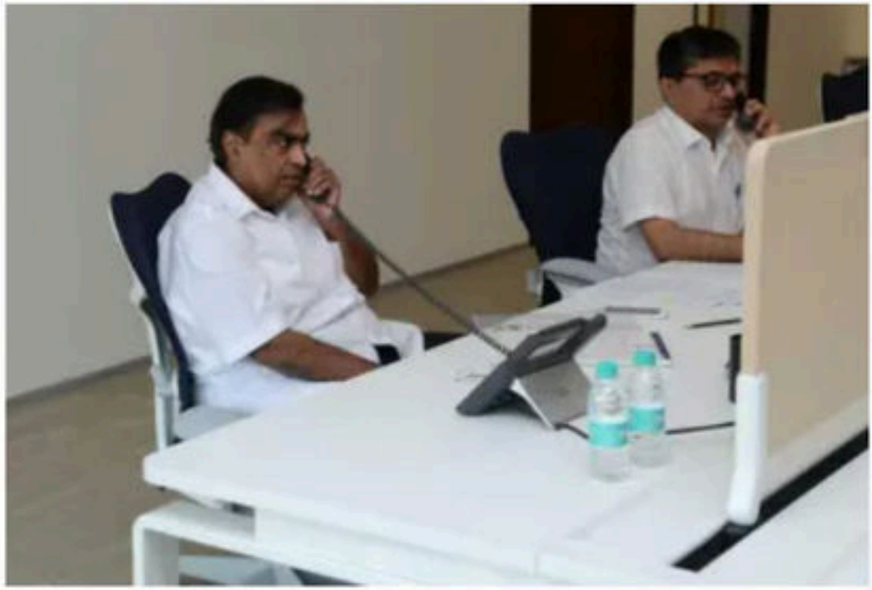
Break

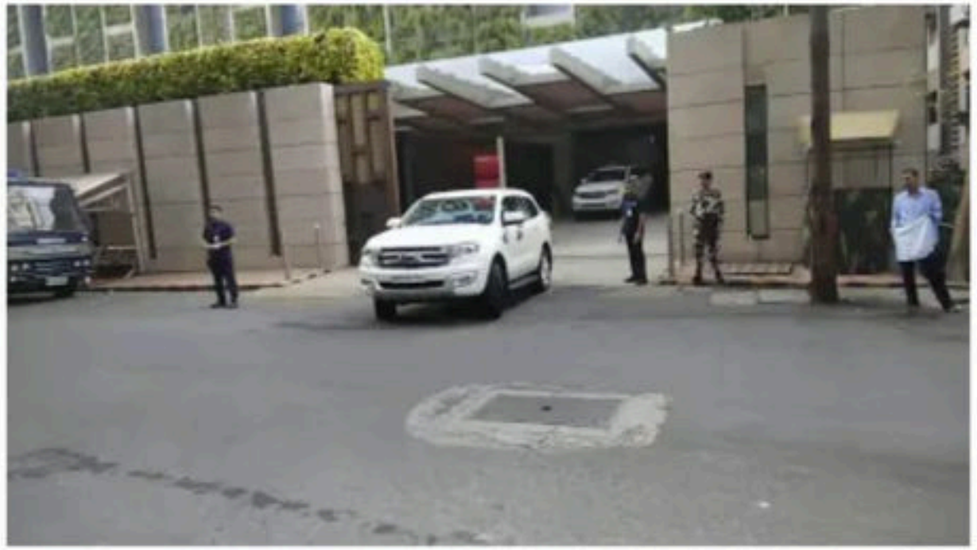
Mode bit

- In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user- defined code.
- At the very least, we need two separate *modes* of operation: user mode and kernel mode (also called supervisor mode, system mode, or privileged mode). A bit, called the mode bit, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).
- When the computer system is executing on behalf of a user application, the system is in user mode. However, 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.



User Mode





Kernel Mode

- At system boot time, the hardware starts in kernel mode. The operating system is then loaded and starts user applications in user mode. Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (that is, changes the state of the mode bit to 0).

- The hardware allows privileged instructions to be executed only in kernel mode. If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

- The instruction to switch to kernel mode is an example of a privileged instruction. Some other examples include I/O control, timer management, and interrupt management.

Break

Farmer

Instruction set programmer

Industry(Gov / Private)

OS designer(API)

Chef

Programmer

Instruction set programmer

OS designer(API)

Programmer

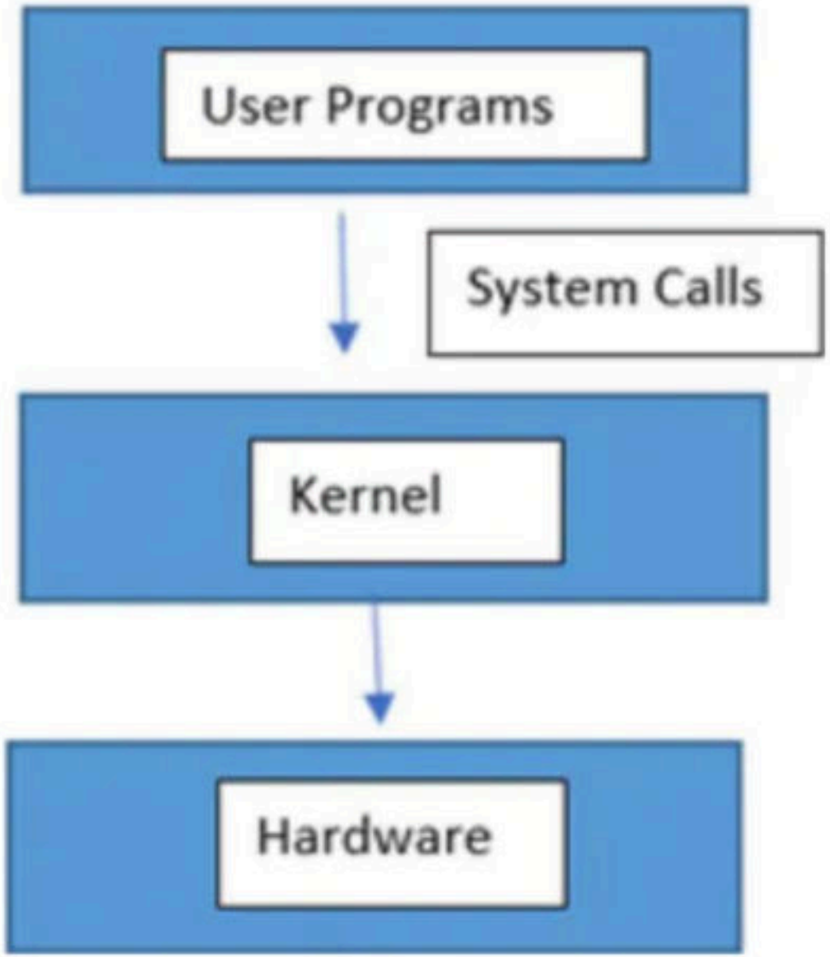
Break

System call

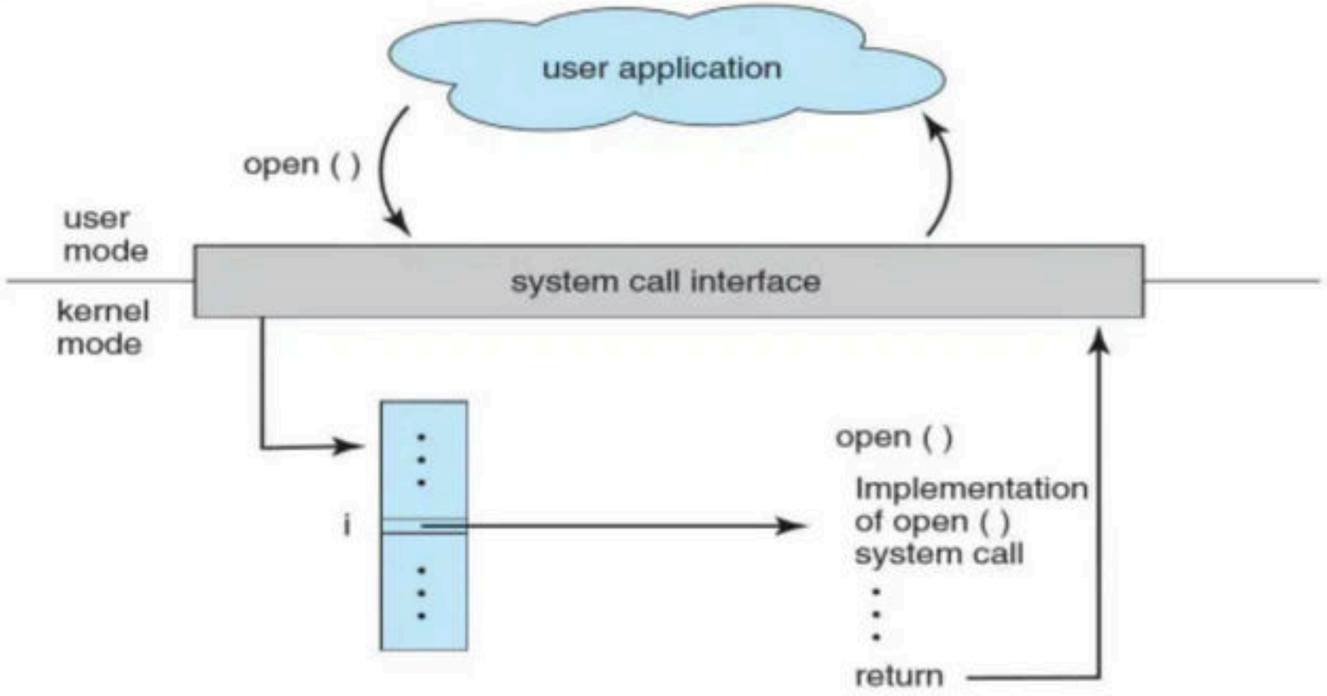
- System calls provide the means for a user program to ask the operating system to perform tasks reserved for the operating system on the user program's behalf.

- System calls provide an interface to the services made available by an operating system. These calls are generally available as routines written in C and C++.

- The API specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expect.

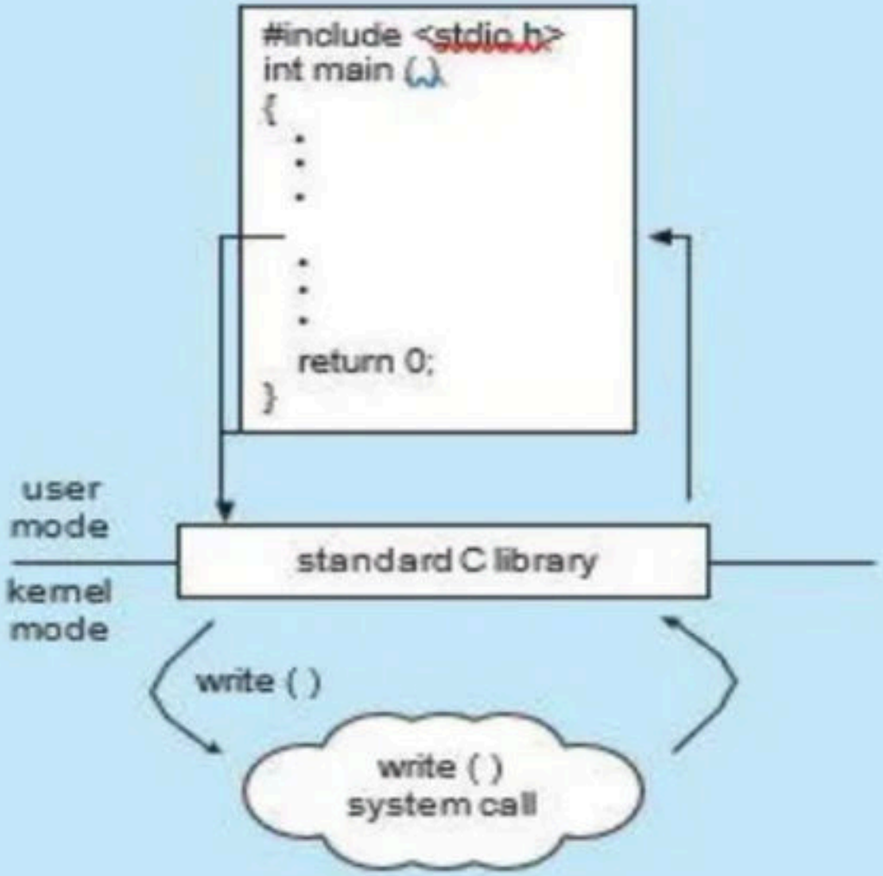


- Three of the most common APIs available to application programmers are the
  - Windows API for Windows systems
  - POSIX API for POSIX-based systems (which include virtually all versions of UNIX, Linux, and Mac OS X)
  - The Java API for programs that run on the Java virtual machine.
- The caller need know nothing about how the system call is implemented or what it does during execution. Rather, the caller need only obey the API and understand what the operating system will do as a result of the execution of that system call.



EXAMPLE OF STANDARD C LIBRARY

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the printf() statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system— in this instance, the write() system call. The C library takes the value returned by write() and passes it back to the user program. This is shown below:



- **Types of System Calls** - System calls can be grouped roughly into six major categories: process control, file manipulation, device manipulation, information maintenance, communications, and protection.
- **Process control**
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory

- **File management**
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes

- **Device management**
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices

- **Information maintenance**
  - get time or date, set time or date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes

- **Communications**
  - create, delete communication connection
  - send, receive messages transfer status information