Basics of Operating System Part - II

Foundation Course on Operating Systems for GATE 2022

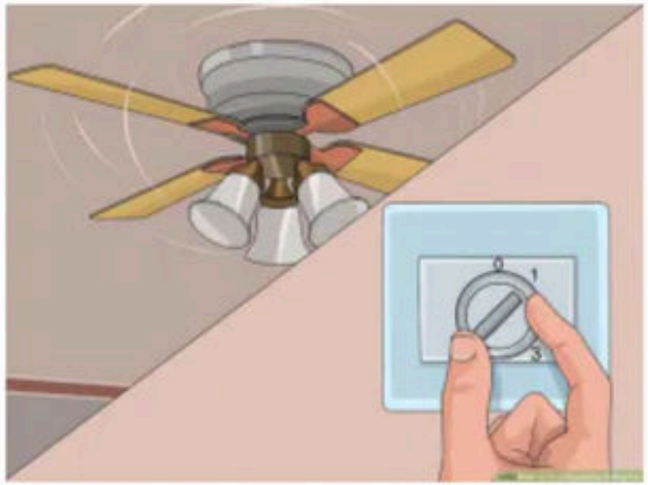
Sanchit Jain • Lesson 2 • July 22, 2021

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

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.



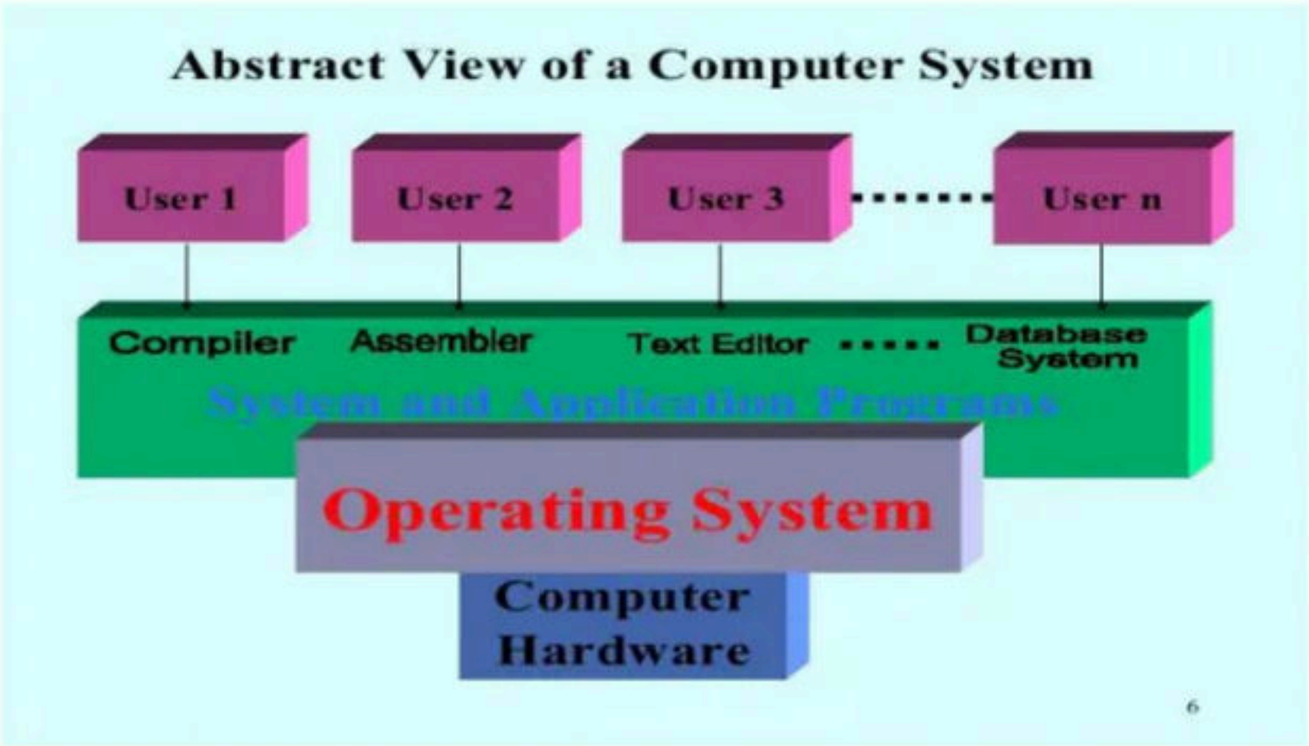
- Question comes why we need an operating system?
  - To enable everybody to use h/w In a convenient and efficient manner

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.



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

(B) Collection of input-output devices

(C) Collection of software routines

(D) All the above

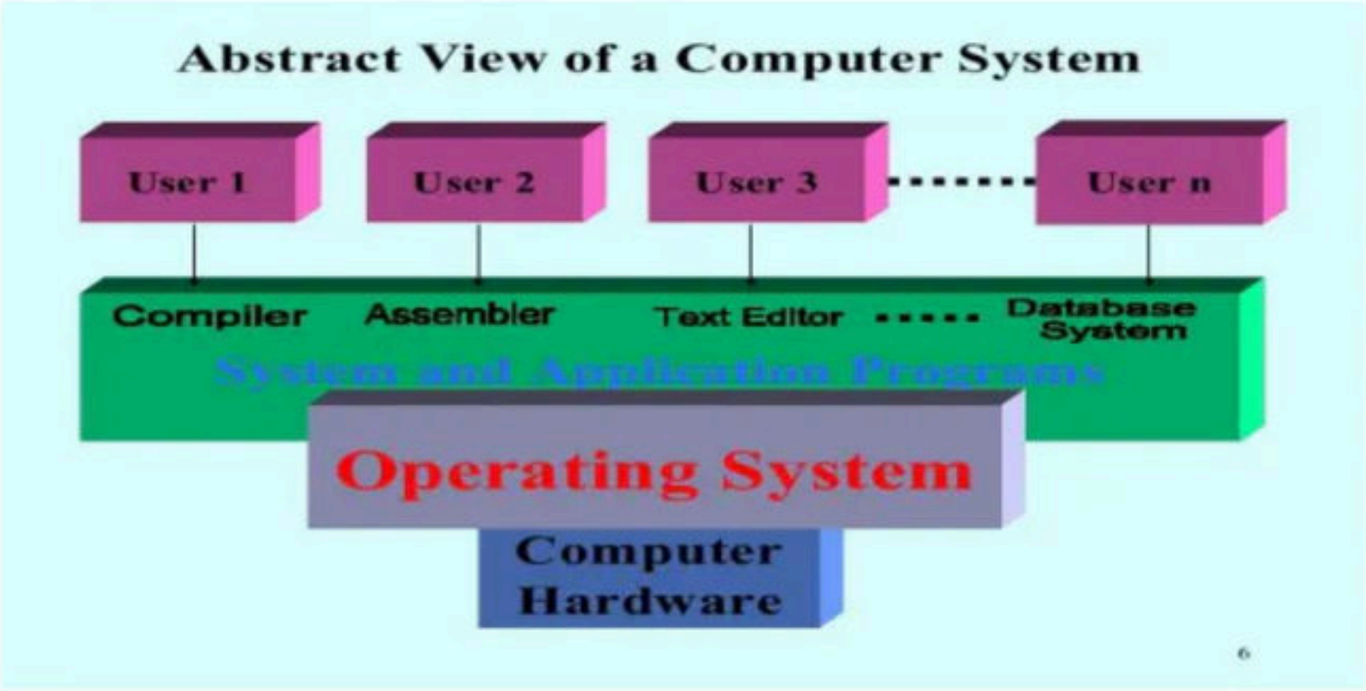
Break

- **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

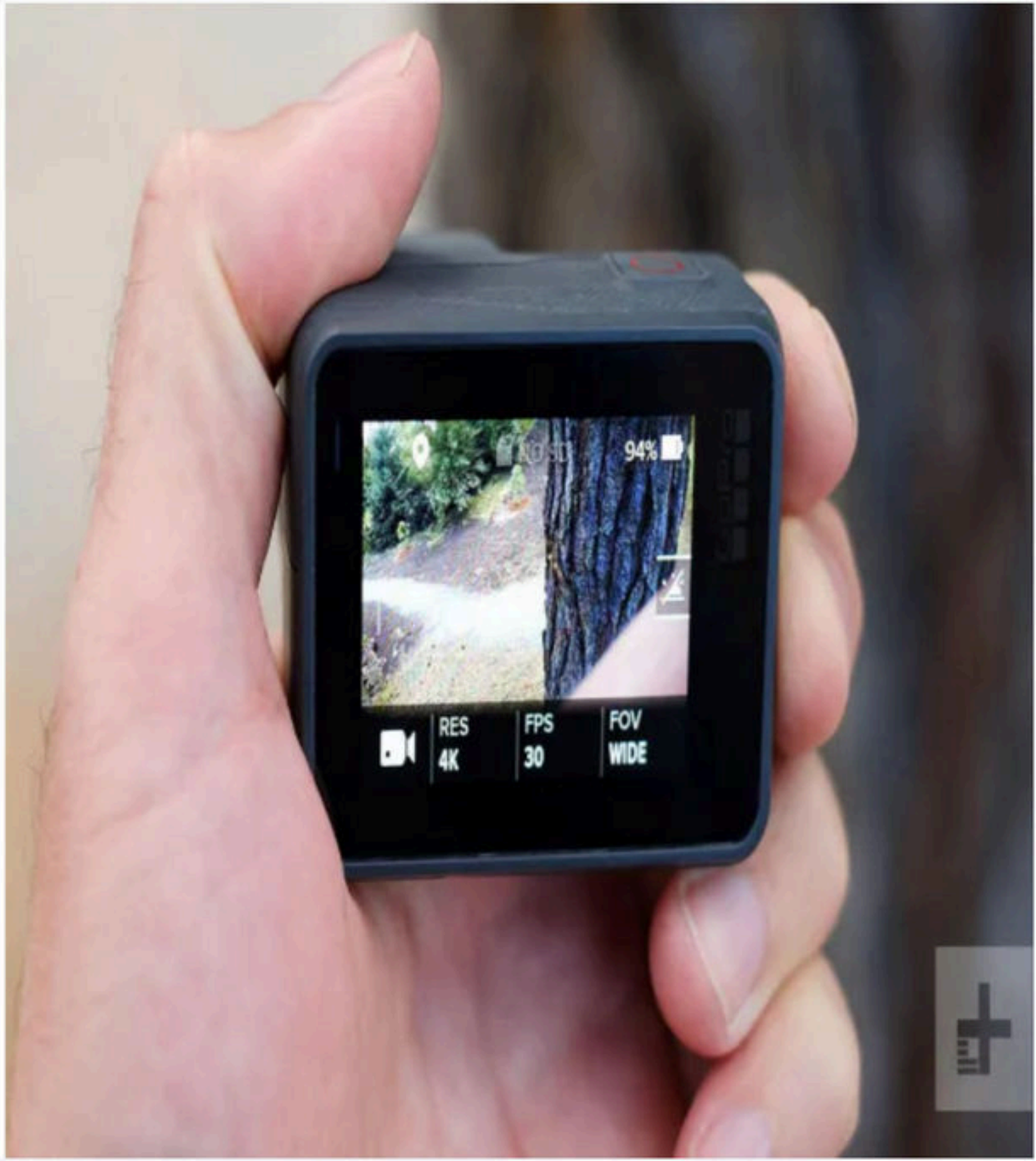
Goals and Functions of operating system

- 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)

Nikon
MENU
OK
Fn2


94%
RES
4K
FPS
30
FOV
WIDE

सत्यमेव जयते

Goverment Of India

सबका साथ सबका विकास

Ministries	Departments
58	93

Functions of operating system

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

Break

Evolution of Operating System

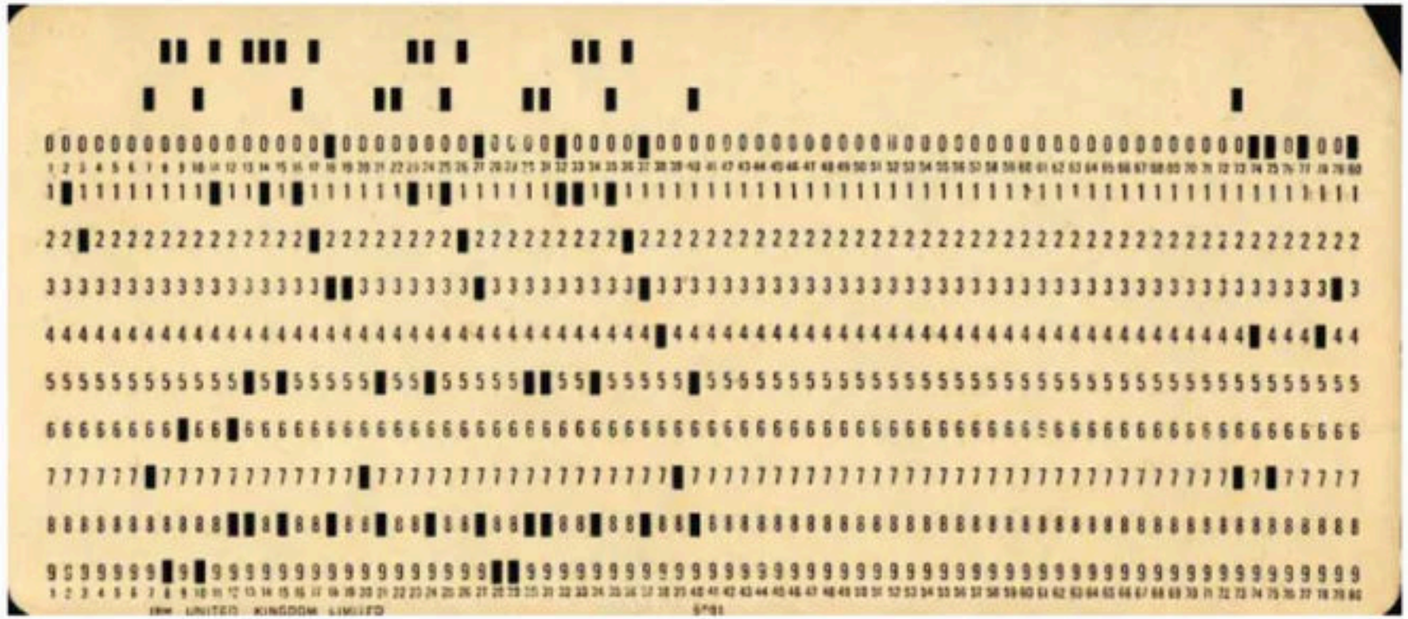
- Early computers were not interactive device, there user use to prepare a job which consist three parts
  - Program
  - Control information
  - Input data

- 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



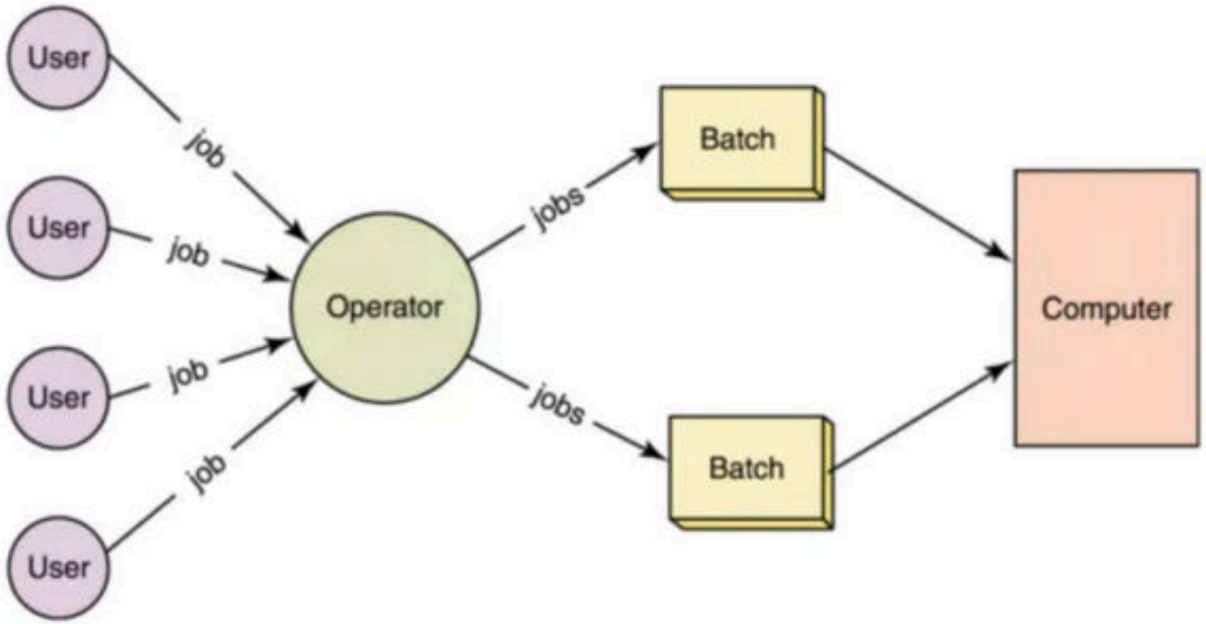
Batch Operating System

- 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.



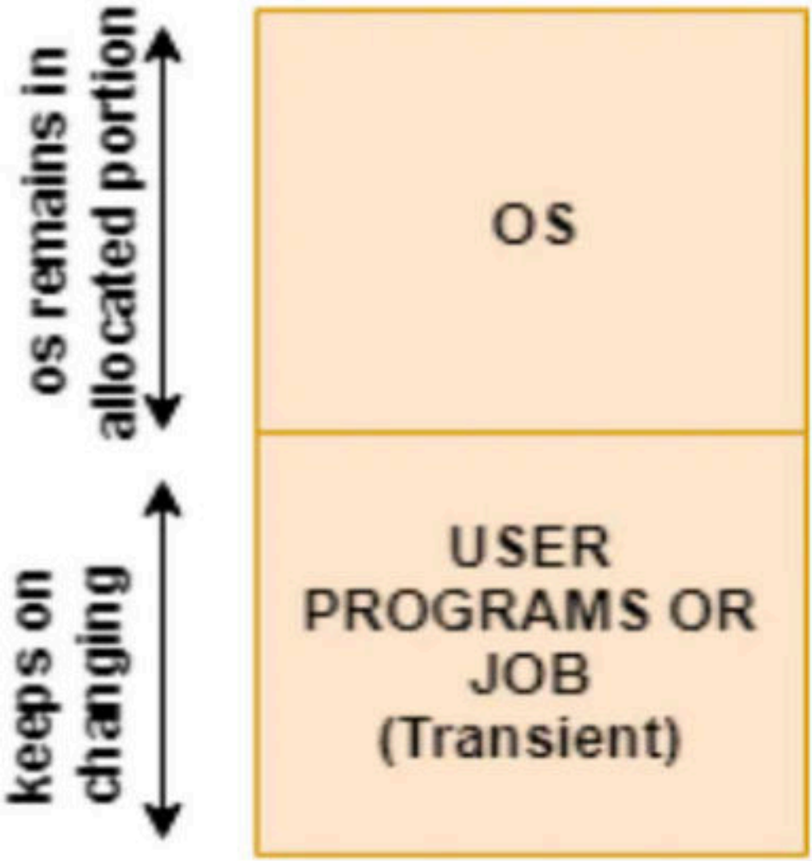
Advantage

- 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

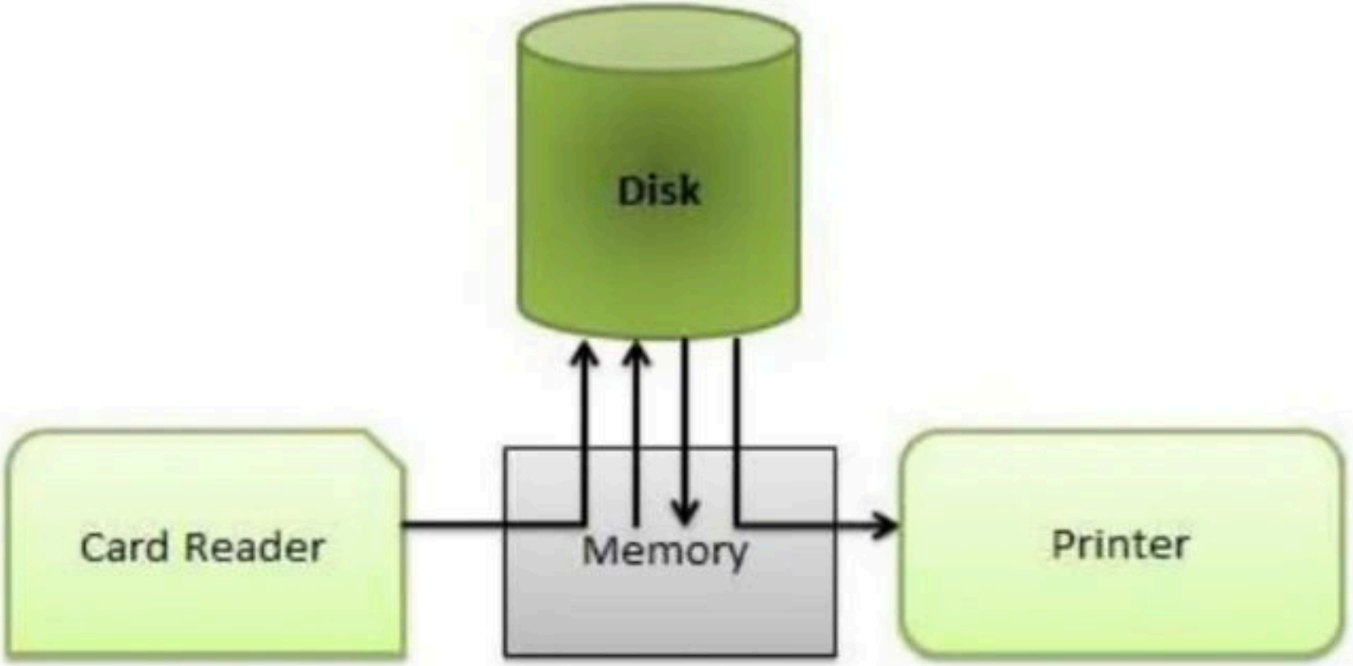
- Memory limitation – memory was very limited, because of which interactive process or multiprogramming was not possible



Break

Spooling

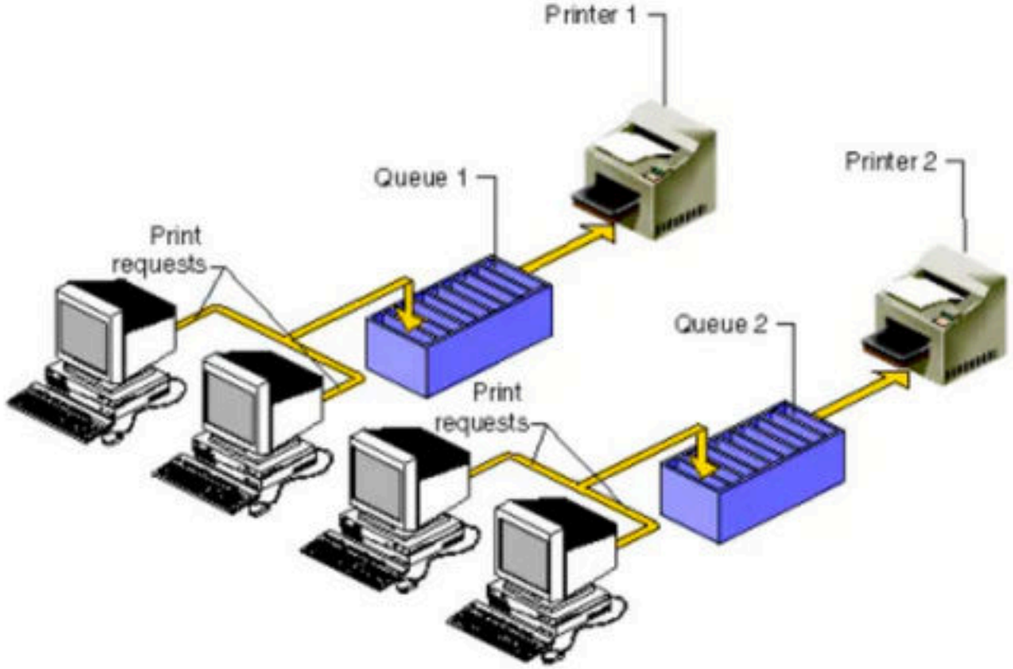
- ("Spool" is technically an acronym for simultaneous peripheral operations online.)
- In a computer system peripheral equipment, such as printers and punch card readers etc, are very slow relative to the performance of the rest of the system. Spooling is useful because devices access data at different 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

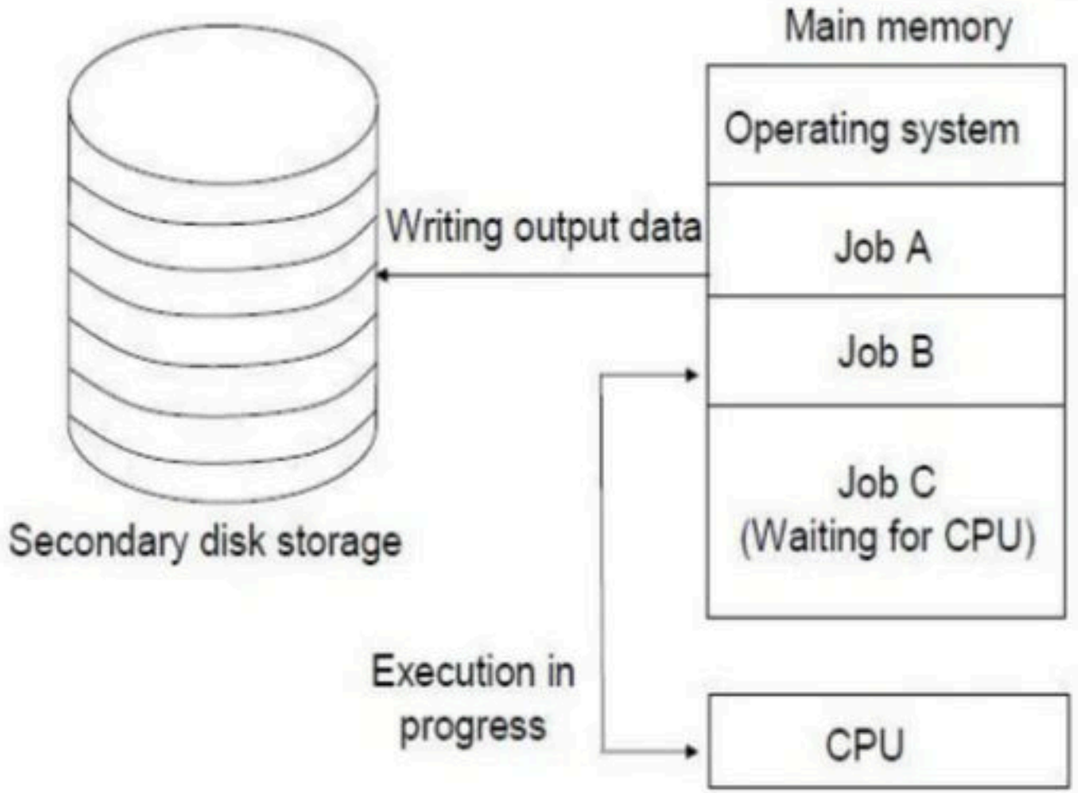
Break

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.

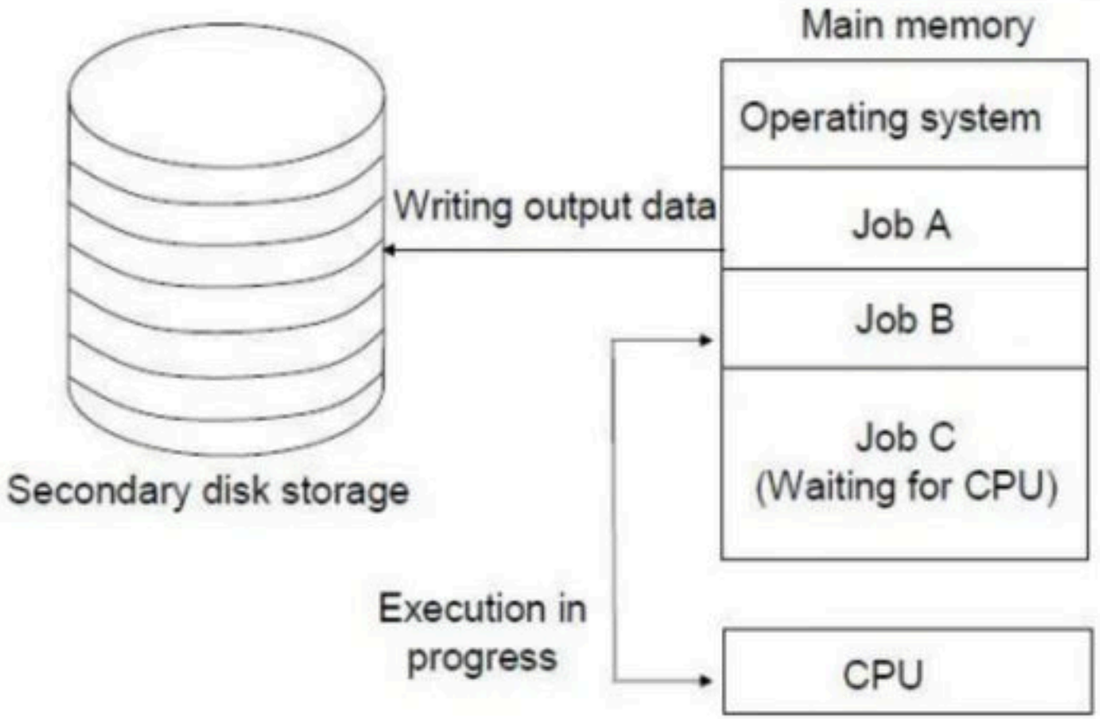
- The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation, to complete.



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. T

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

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

(A) (a) only 8

(B) (a) and (b) only 16

(C) (a) and (c) only 12

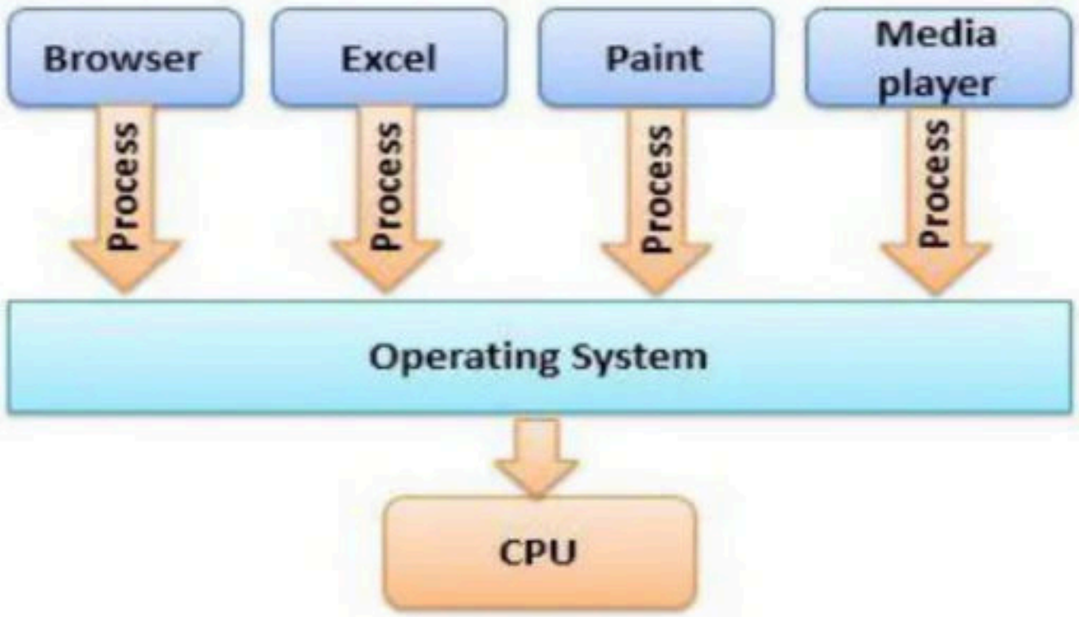
(D) (a), (b) and (c) only 64

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

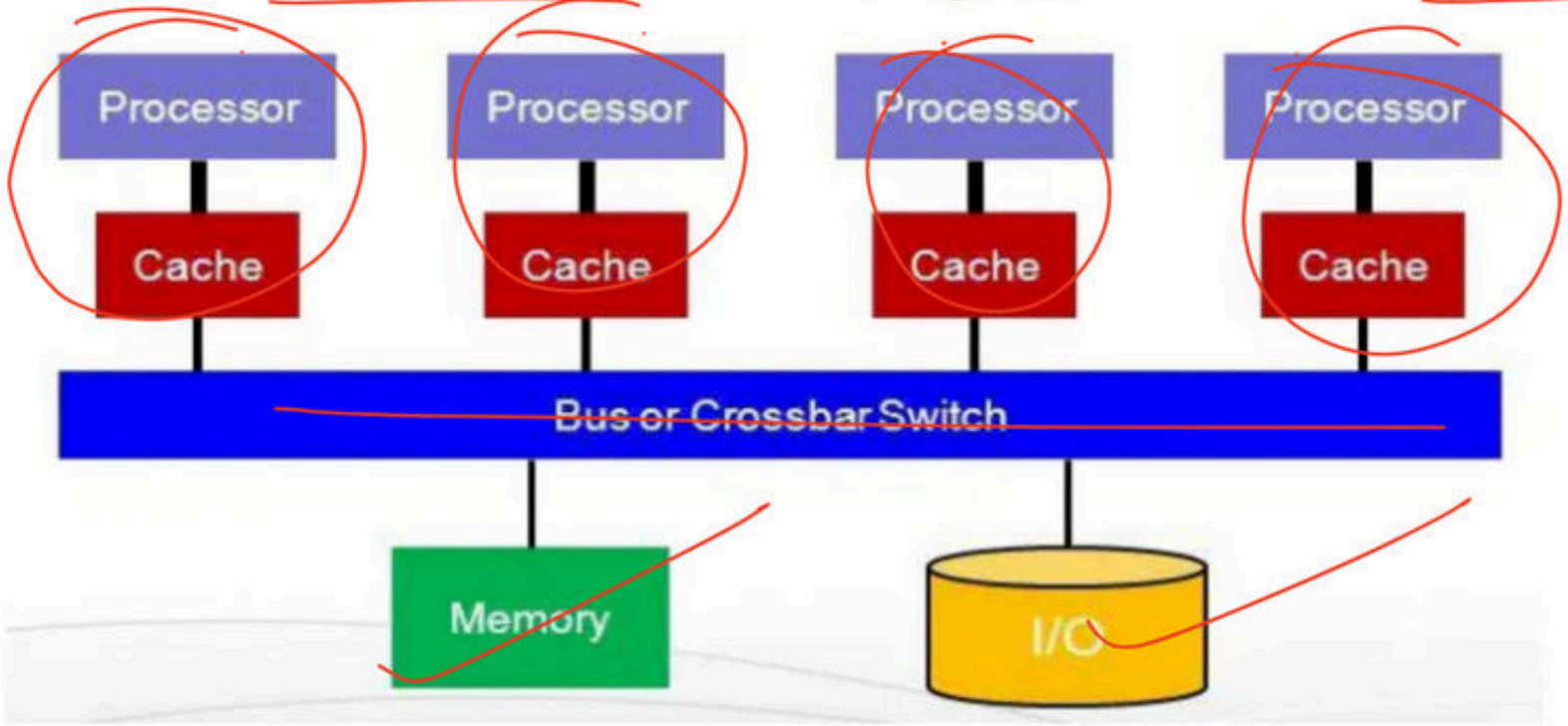
(D) multitasking

Break

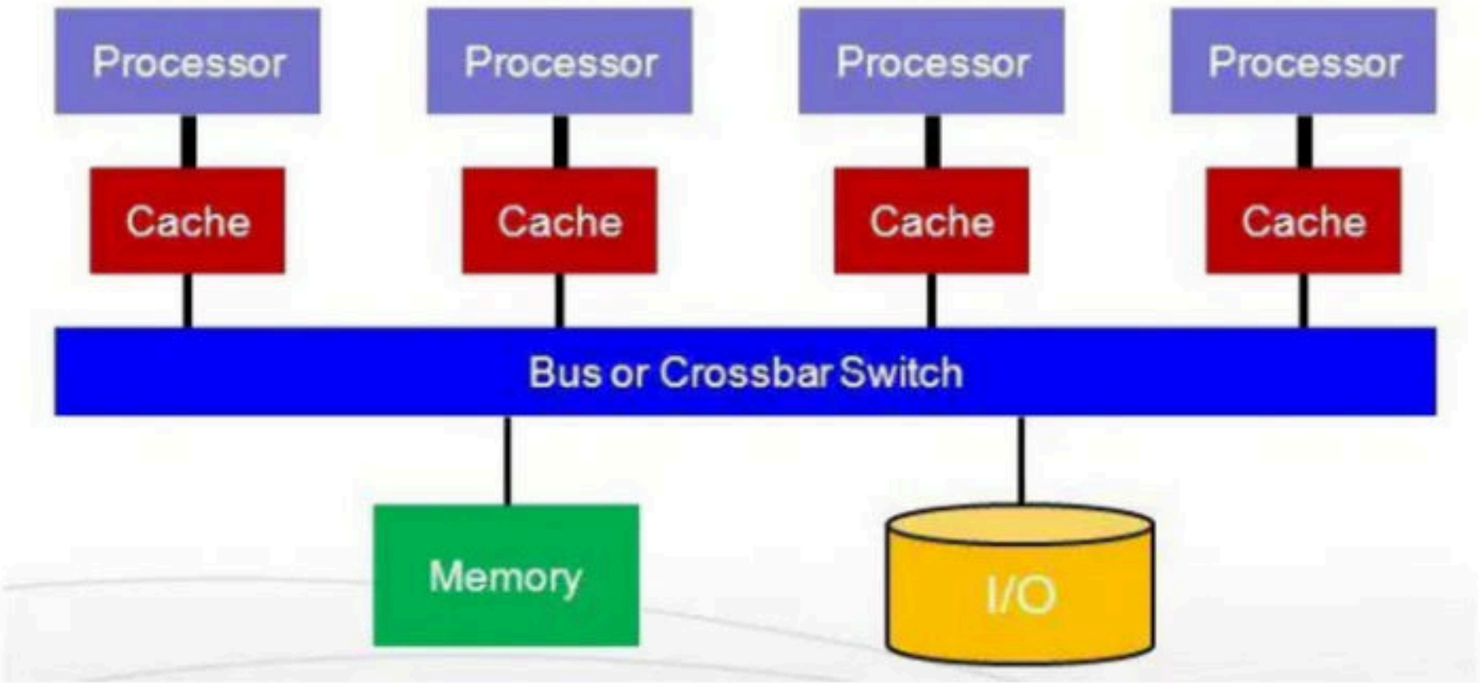
Multiprocessing Operating System/ tightly coupled system

- 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.

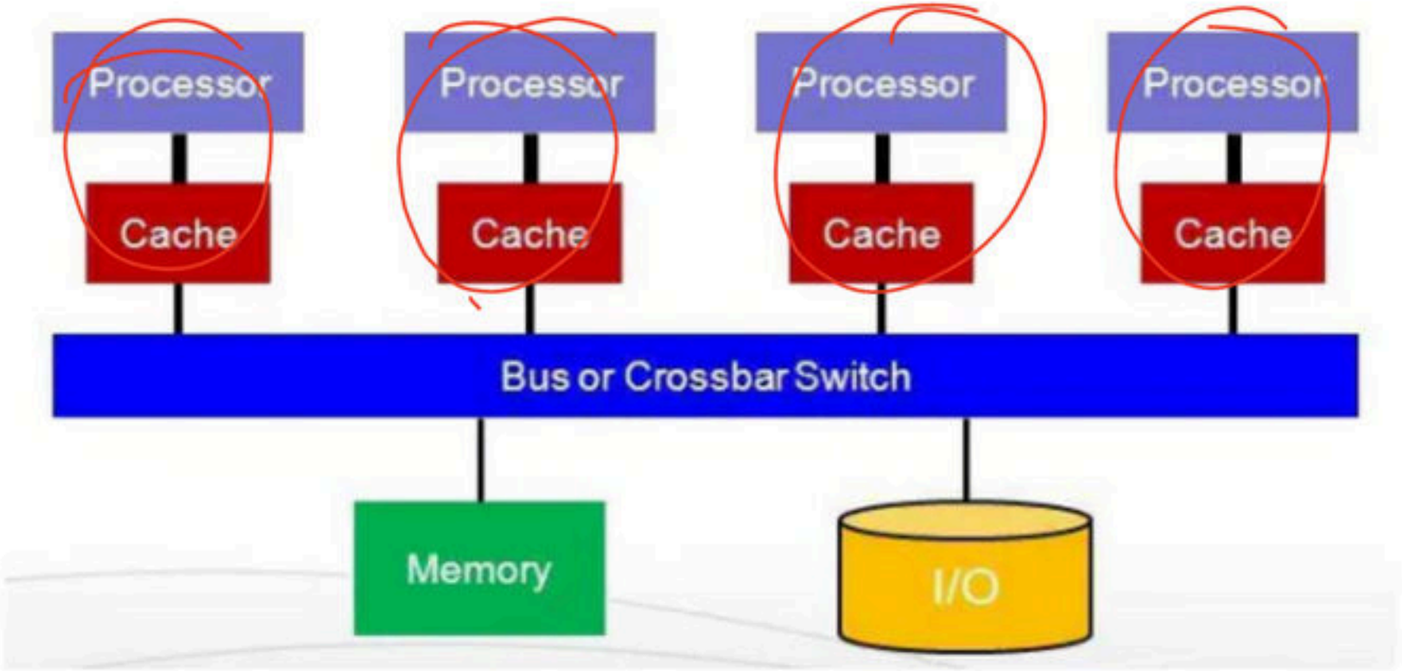


Multiprocessing can be of two types

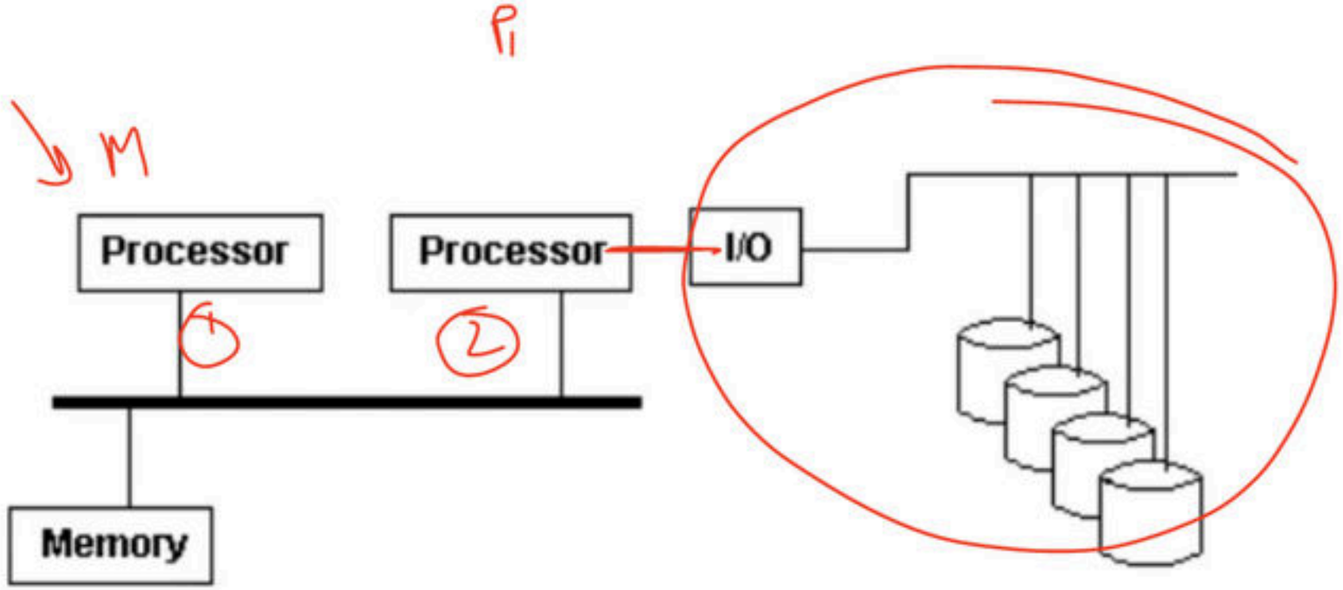
- **Symmetric multiprocessing** - In a symmetric multi-processing, a single OS instance controls two or more identical processors connected to a single shared main memory.

- 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.

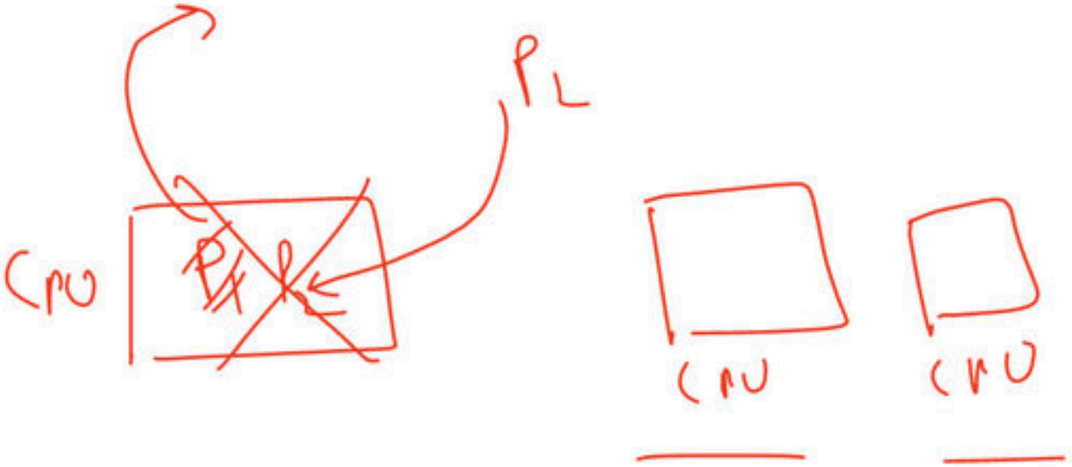
•**Economy of Scale** - Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies. If several programs operate on the same set of data, it is cheaper to store those data on one disk and to have all the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

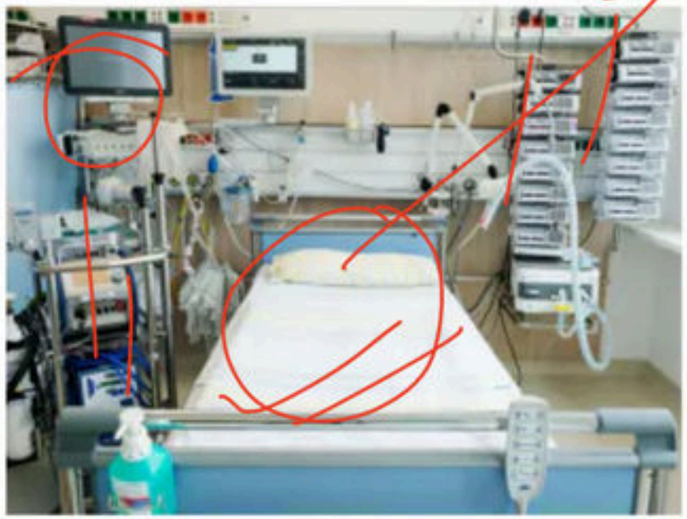
Real time Operating system

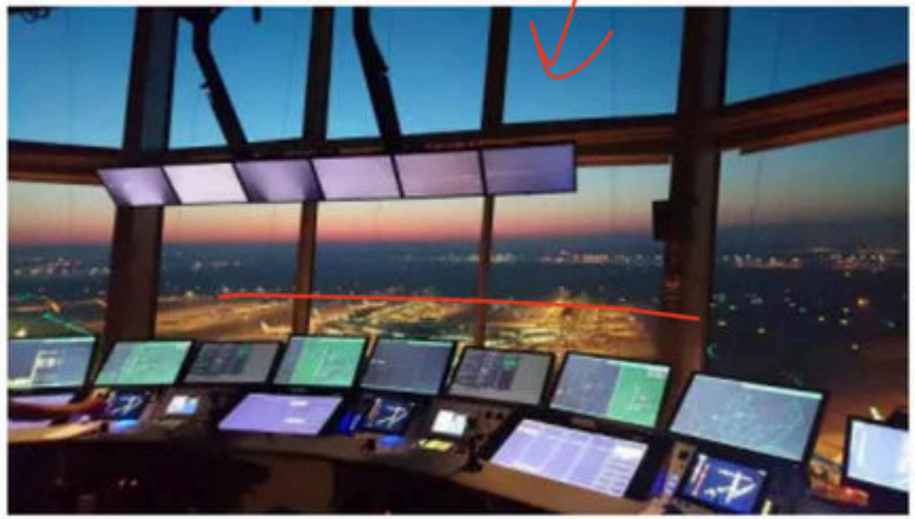
- A real time system is a time bound system which has well defined fixed time constraints. Processing must be done within the defined constraints or the system will fail. Very fast and quick respondent systems.

- 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.

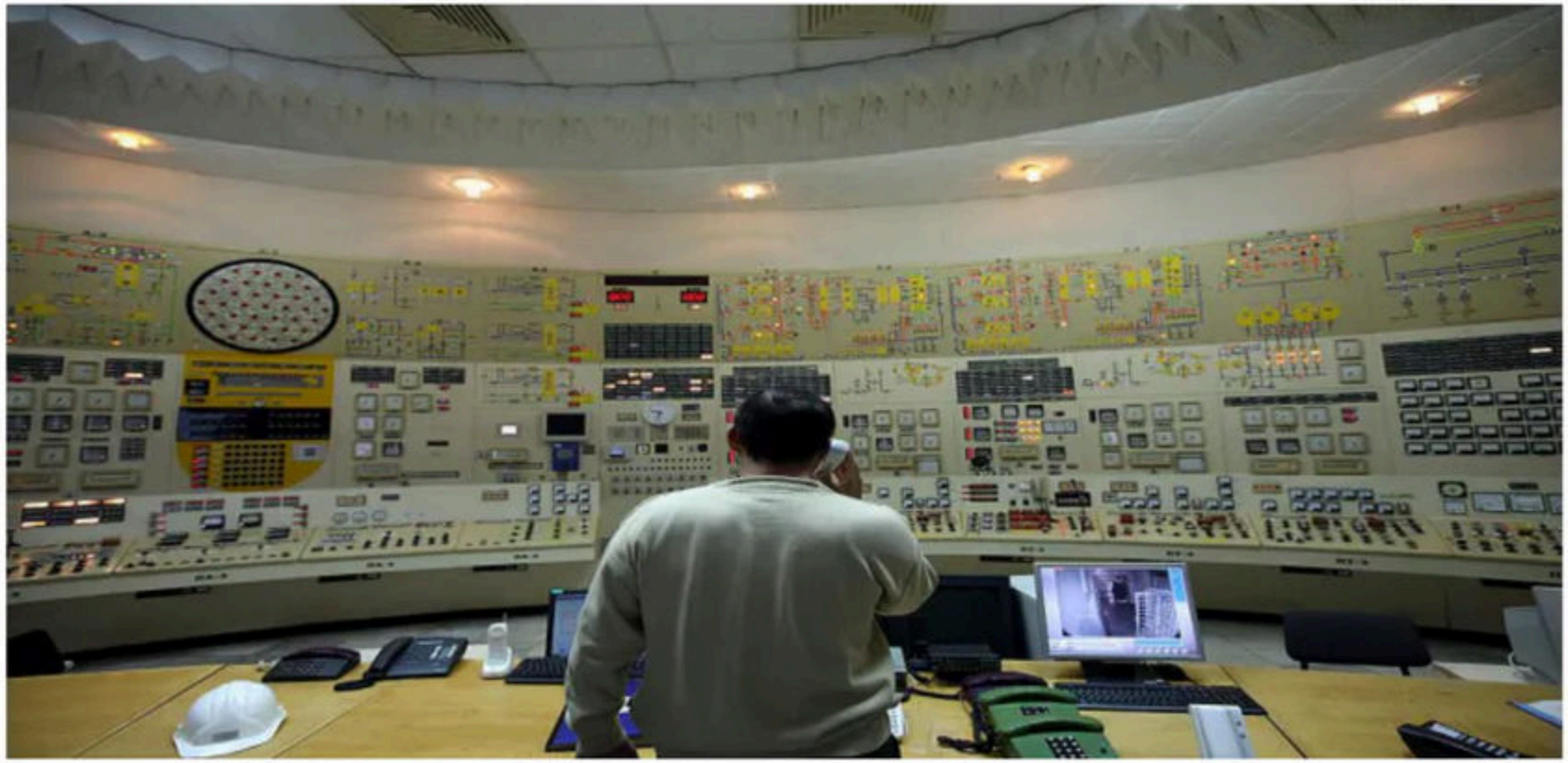
10 min



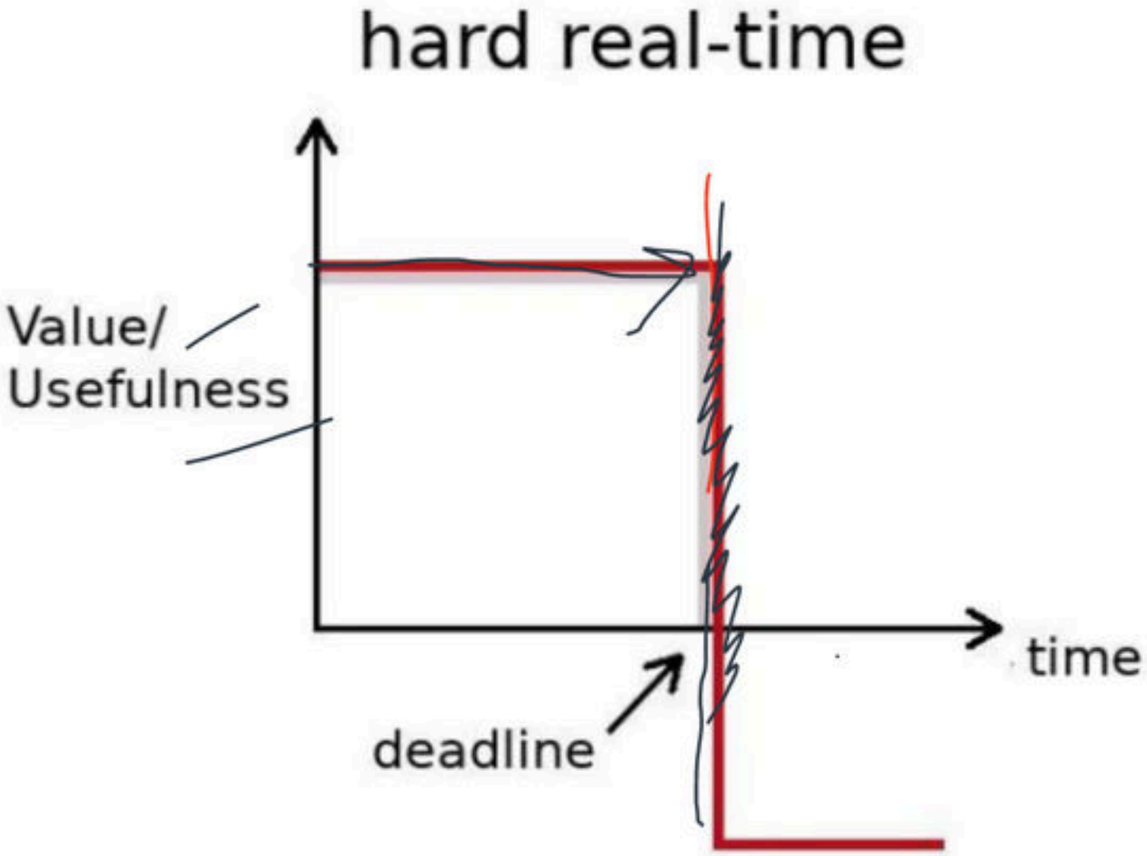


- Valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.

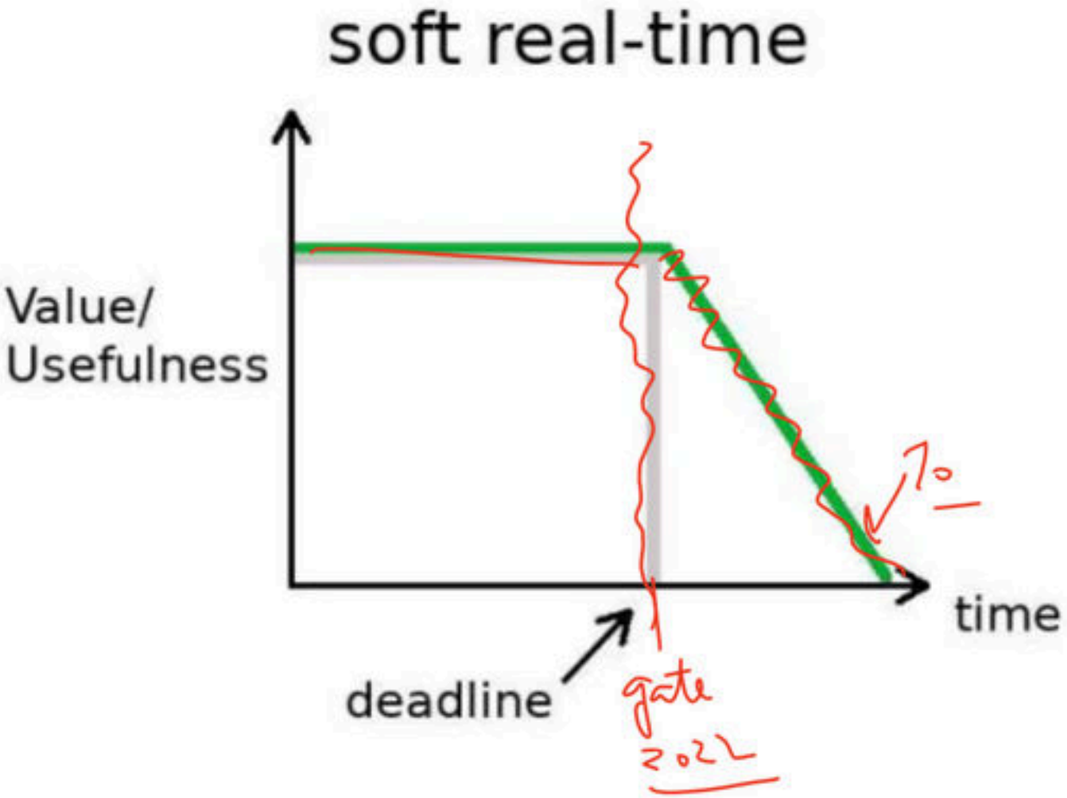
- Little swapping of programs between primary and secondary memory. Most of the time, processes remain in primary memory in order to provide quick response, therefore, memory management in real time system is less demanding compared to other systems.

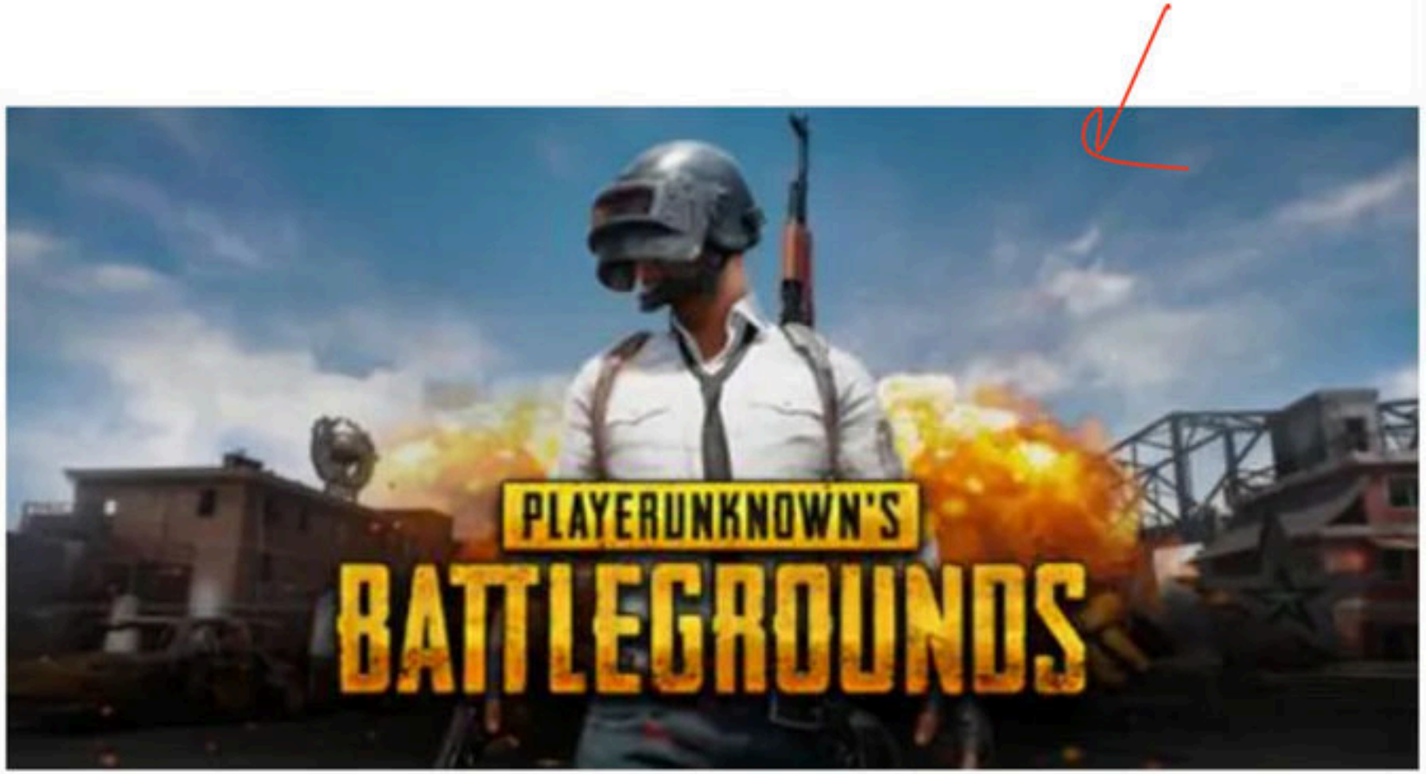


- **Hard real-time operating system** - This is also a type of OS and it is predicted by a deadline. The predicted deadlines will react at a time t = 0. Some examples of this operating system are air bag control in cars, anti-lock brake, and engine control system etc.



- 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.





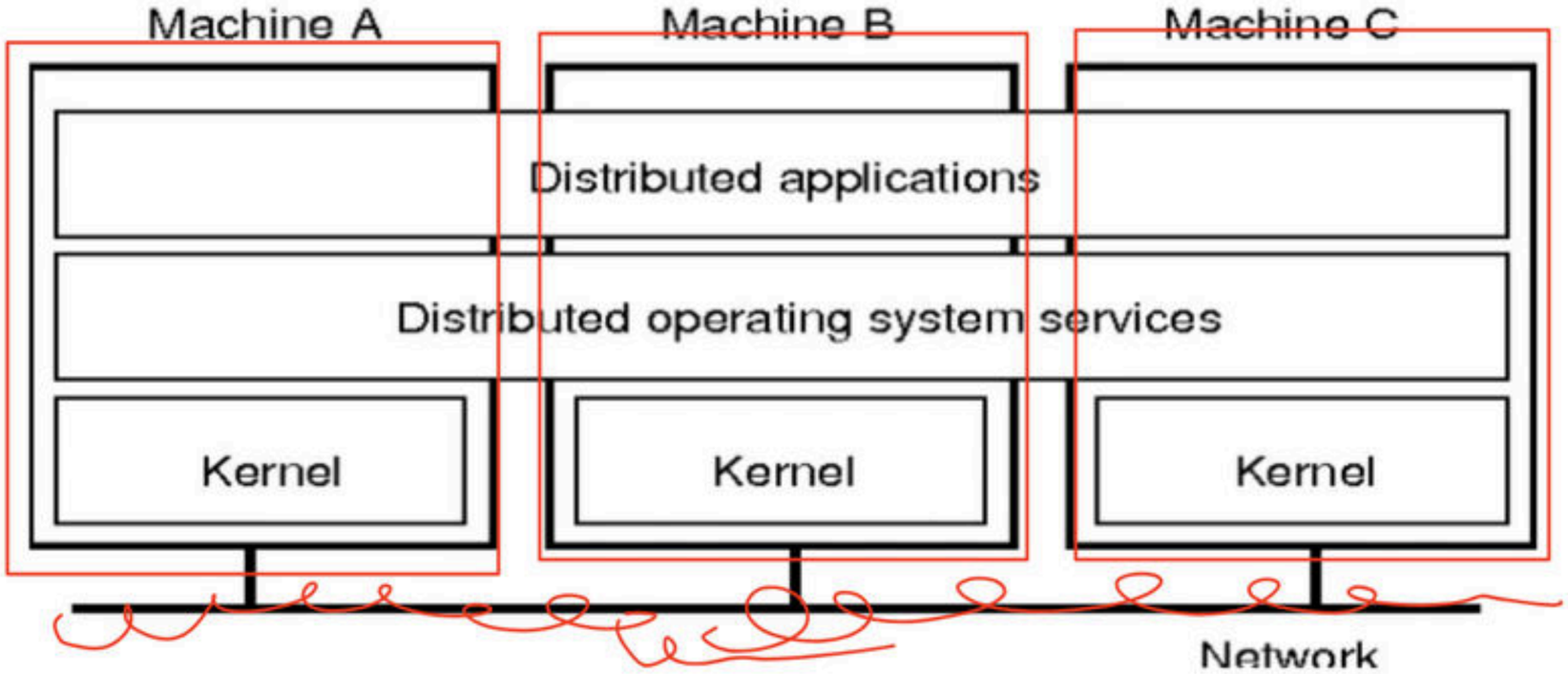
Break

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.

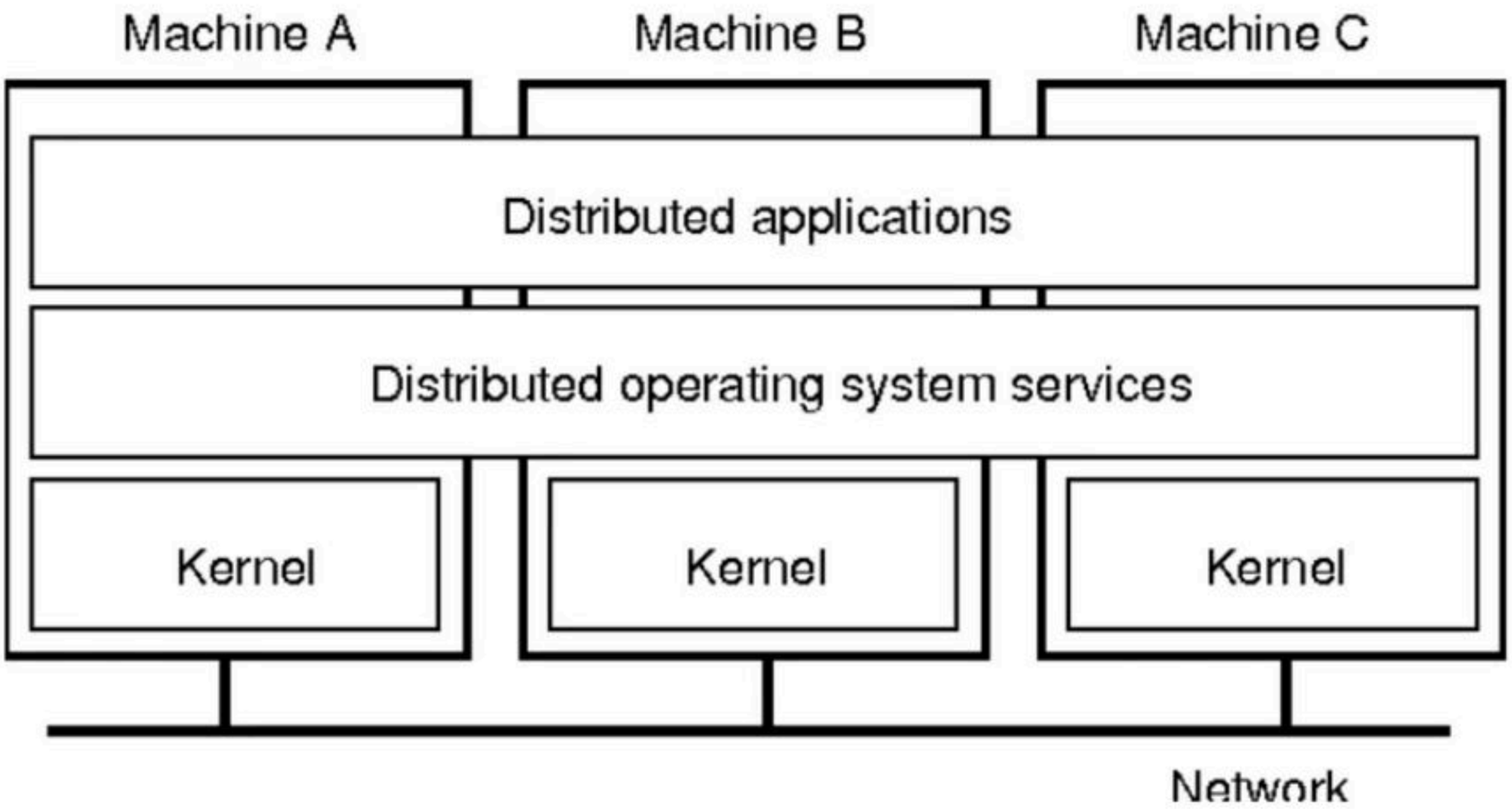


32,000,000
60

6.8

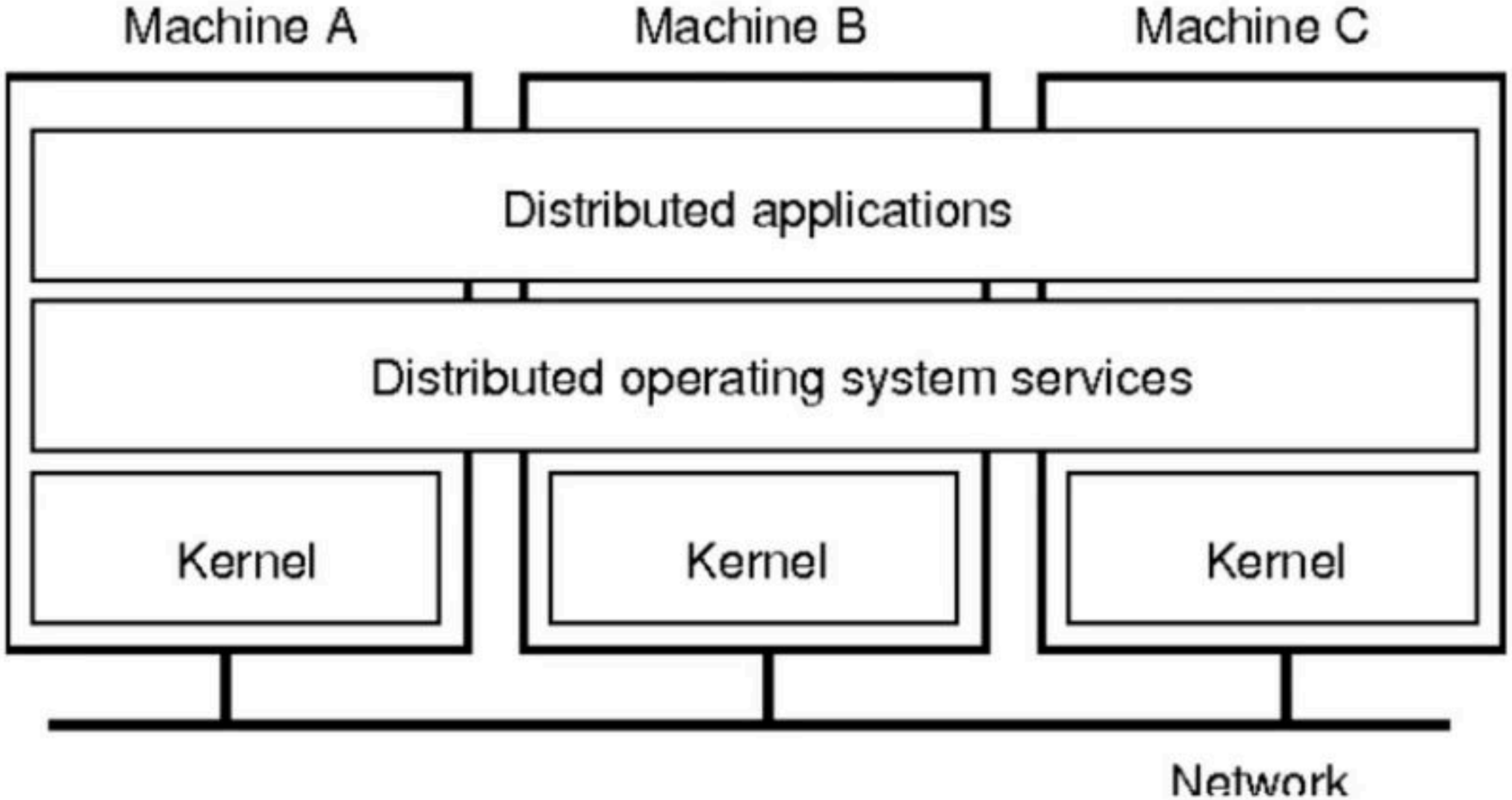
- Each subset is a composite of two distinct service provisioners. The first is a ubiquitous minimal kernel, or microkernel, that directly controls that node's hardware.

- 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.

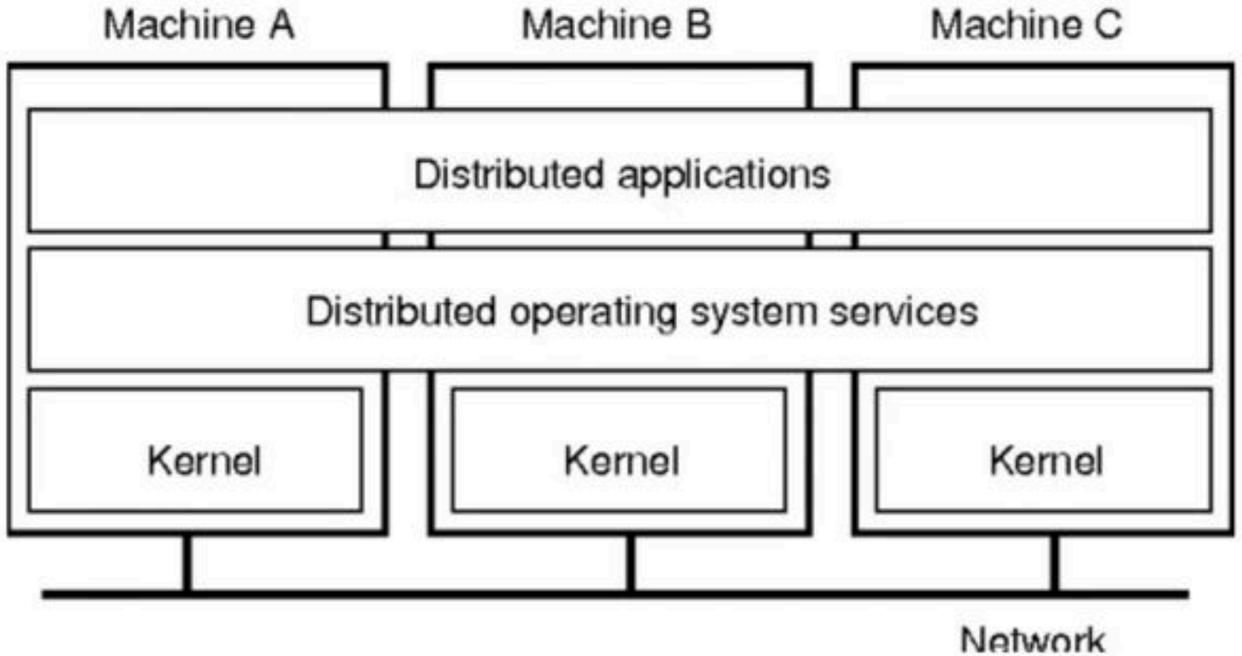
- This seamless integration of individual nodes into a global system is referred to as *transparency*, or *single system image*; describing the illusion provided to users of the global system's appearance as a single computational entity.



- 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)**

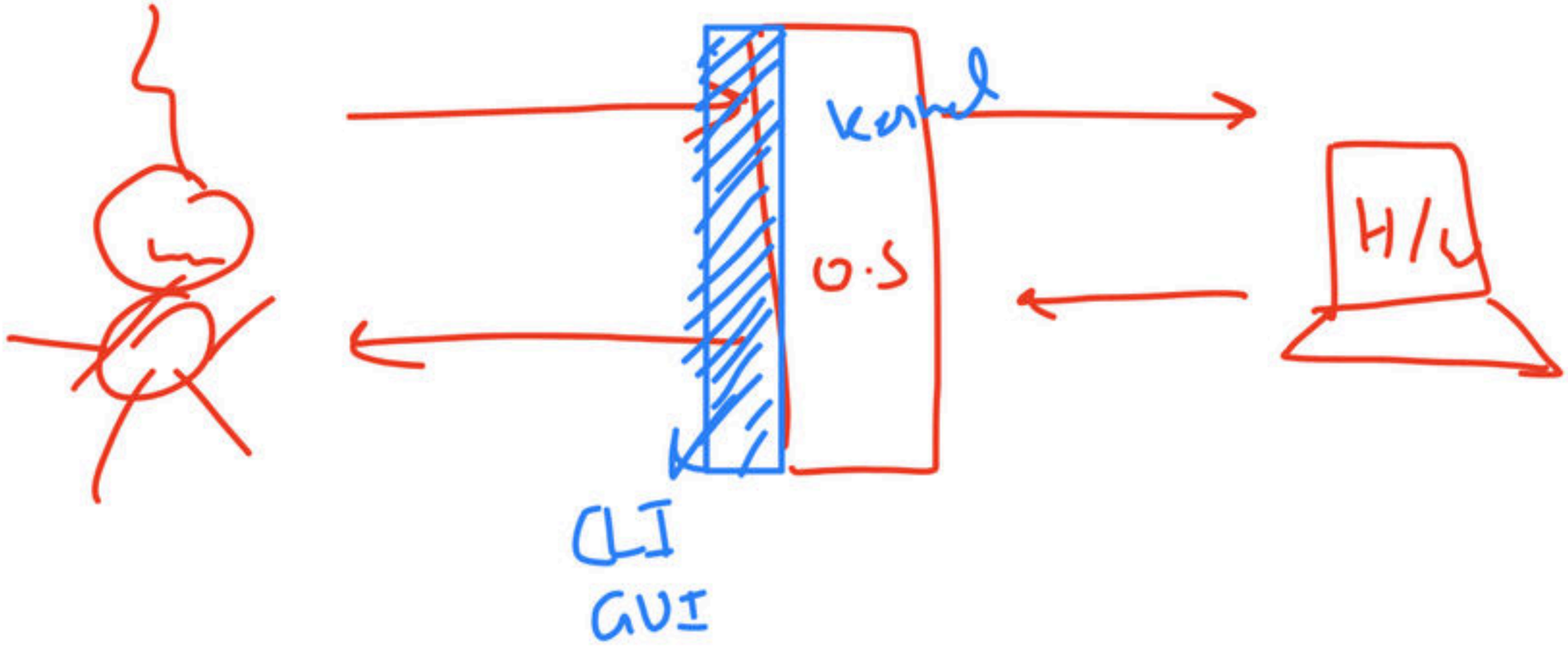
List - I	List - II
(1) Spooling	(i) Allows several jobs in memory to improve CPU utilization
(2) Multiprogramming	(ii)Access to shared resources among geographically dispersed computers in a transparent way
(3)Time sharing	(iii) Overlapping I/O and computations
(4)Distributed computing	(iv)Allows many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

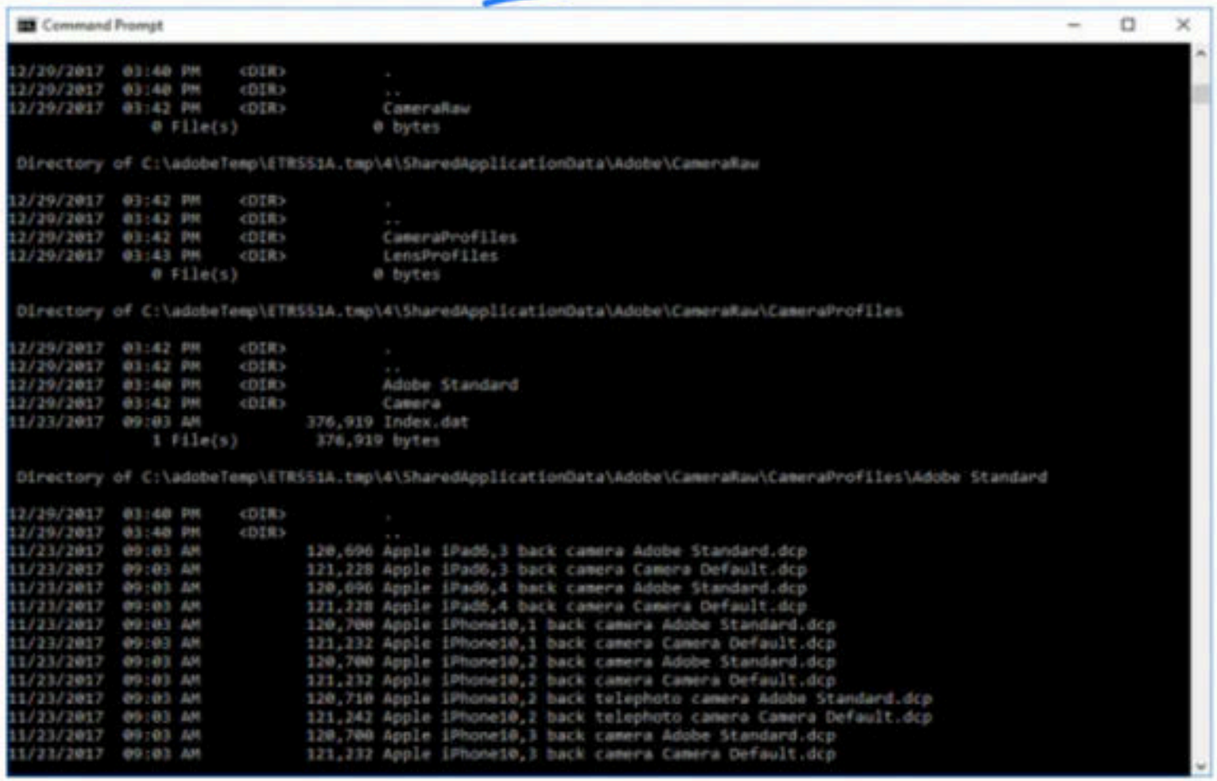
- There are several ways for users to interface with the operating system. Here, we discuss two fundamental approaches.
- One provides a command-line interface, or command interpreter, that allows users to directly enter commands to be performed by the operating system.
- The other allows users to interface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

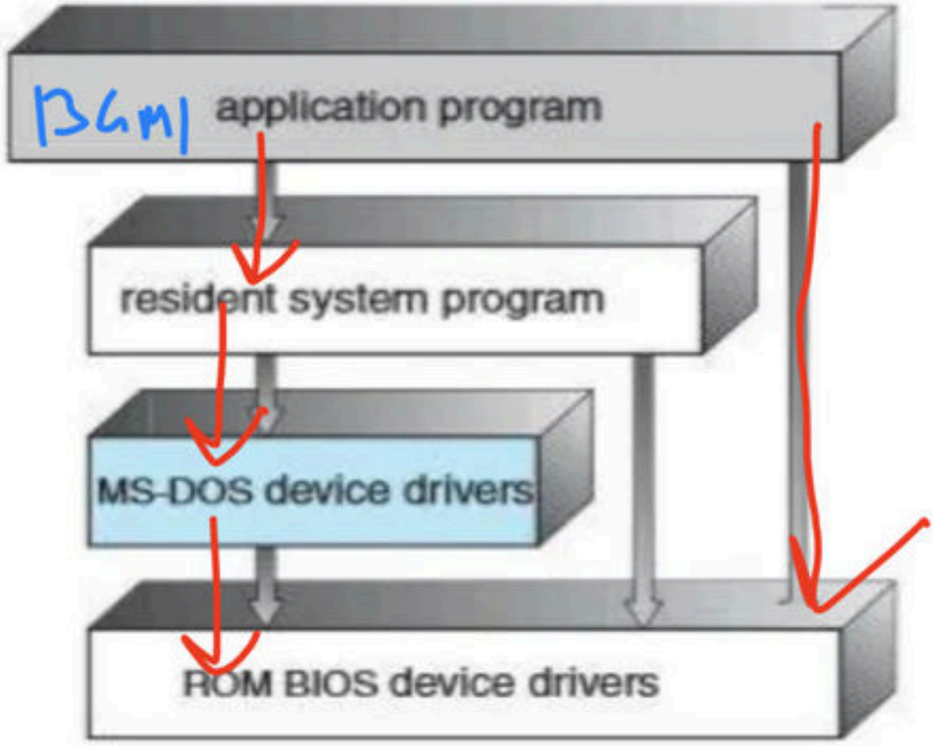
2

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.

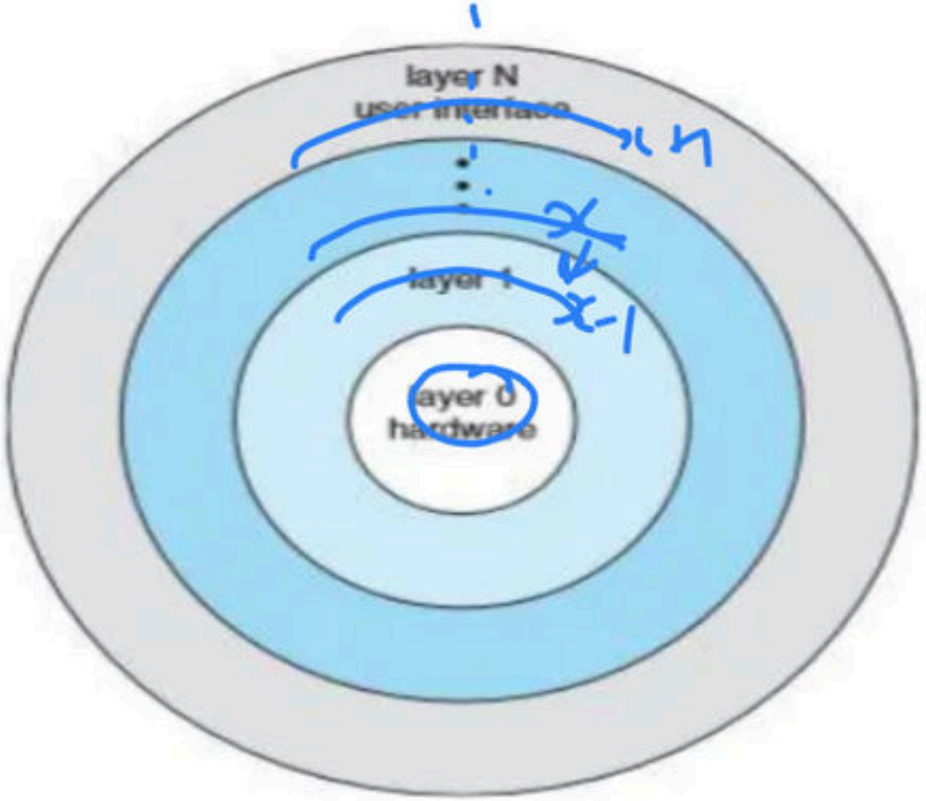
- **Layered Approach** - With proper hardware support, operating systems can be broken into pieces. The operating system can then retain much greater control over the computer and over the applications that make use of that computer.

- 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.

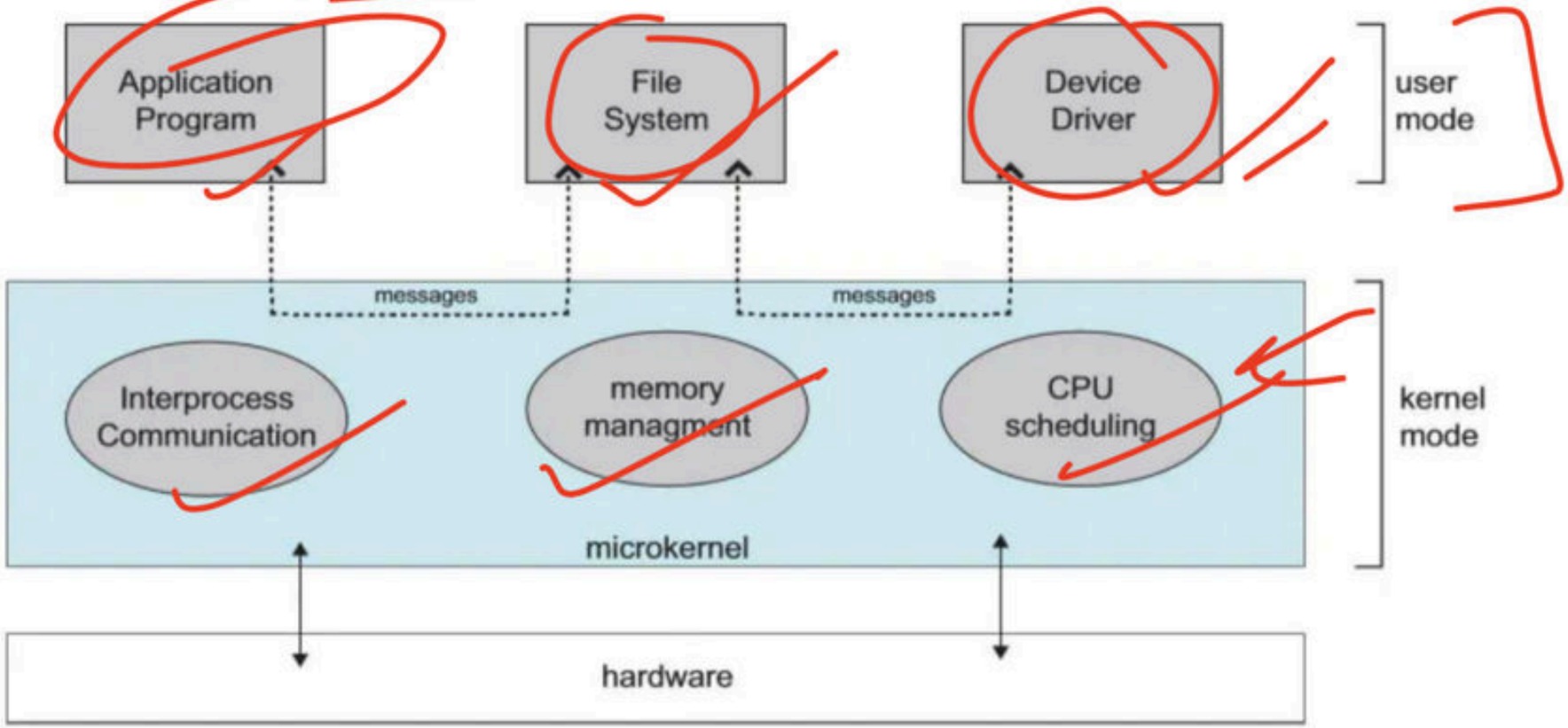


A layered operating system.

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.

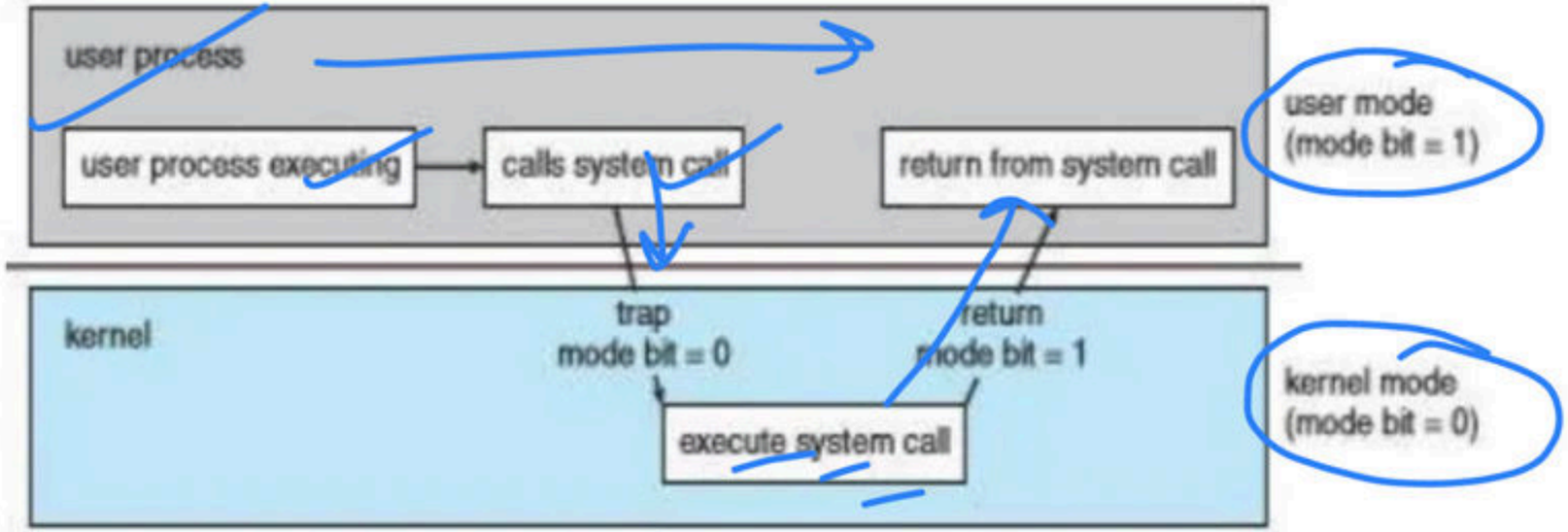
- The MINIX 3 microkernel, for example, has only approximately 12,000 lines of code.

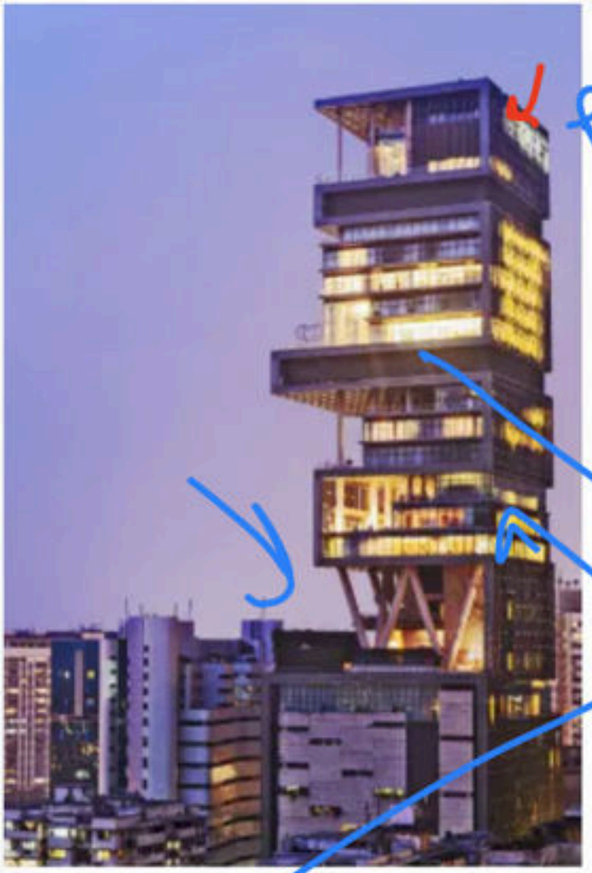
- Developer Andrew S. Tanenbaum

Break

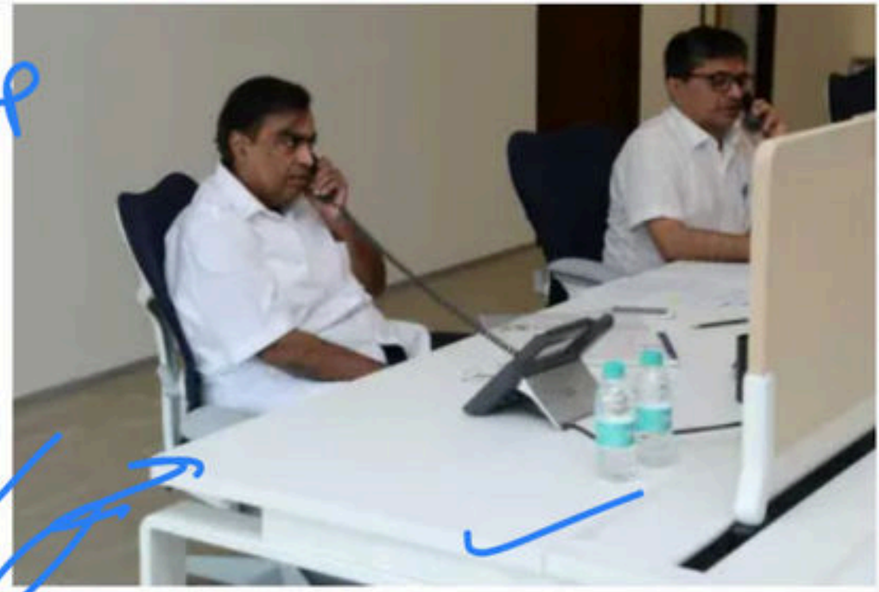
Mode bit

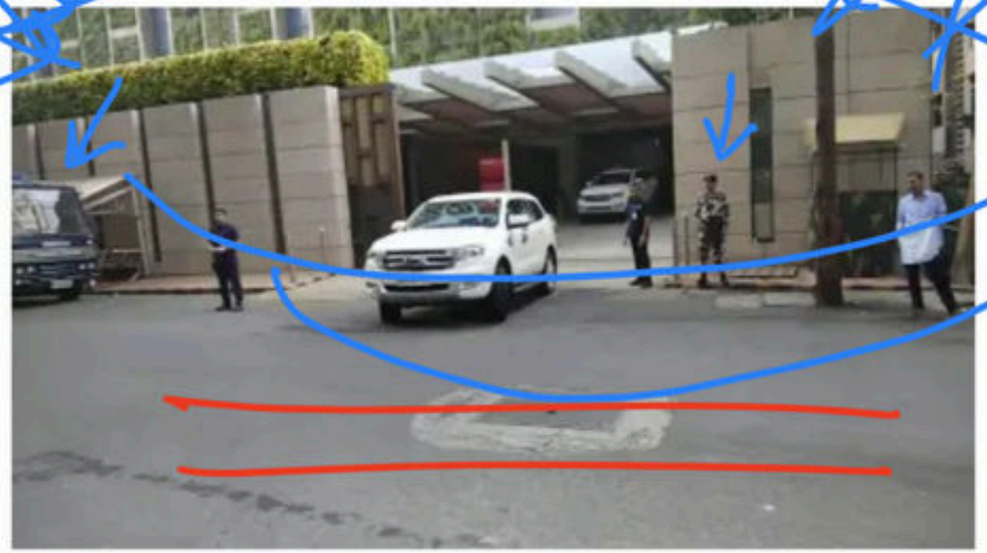
- In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user- defined code.
- At the very least, we need two separate *modes* of operation: user mode and kernel mode (also called supervisor mode, system mode, 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





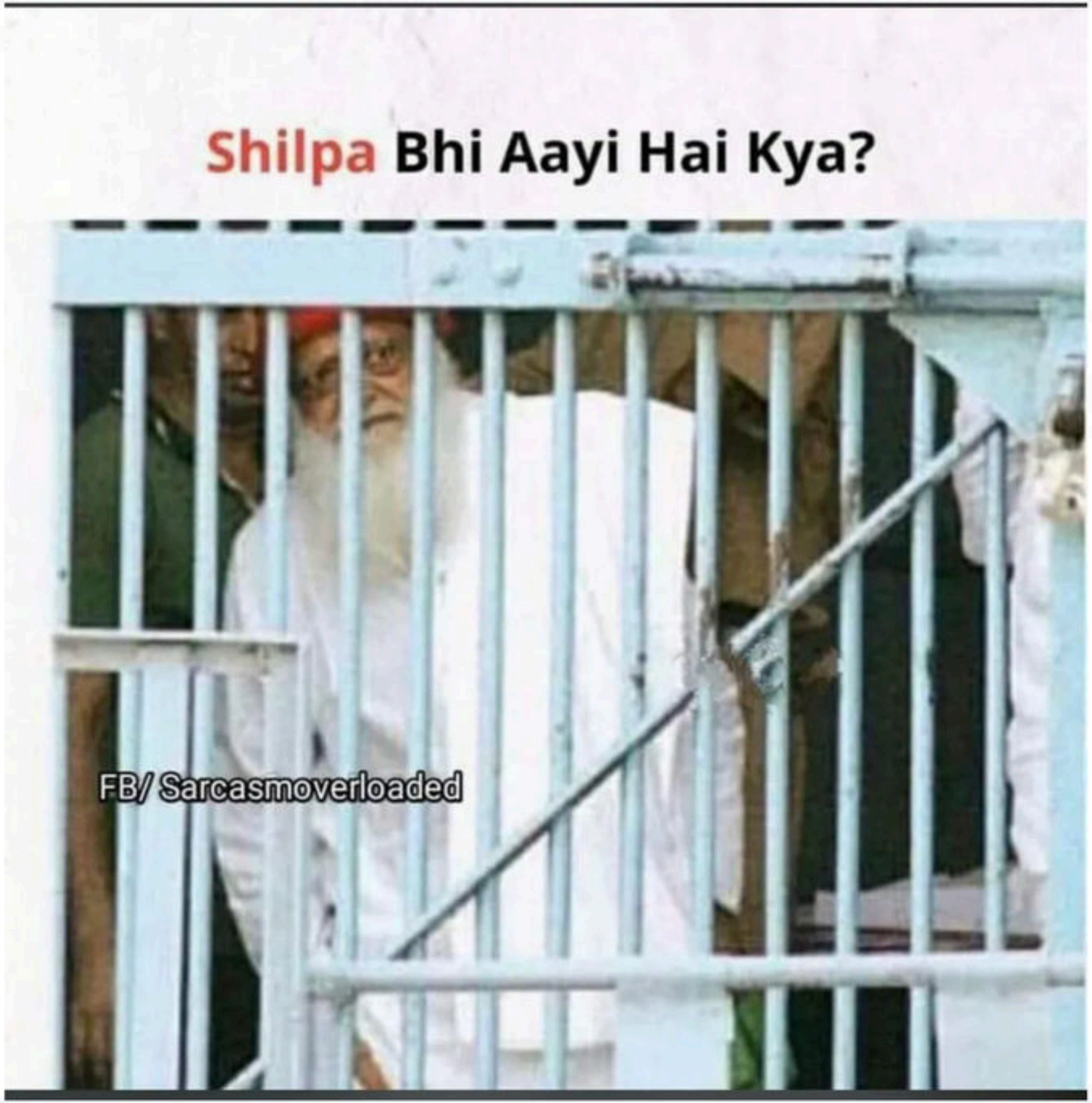
Kernel Mode

- 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).

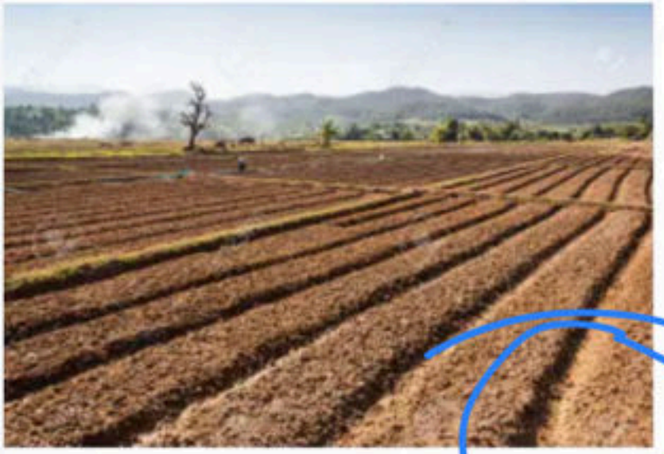
- The hardware allows privileged instructions to be executed only in kernel mode. If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

- 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


Shilpa Bhi Aayi Hai Kya?
FB/ Sarcasmoverloaded

Farmer



Instruction set programmer

Industry(Gov / Private)

OS designer(API)

Chef

Programmer

Instruction set programmer

OS designer(API)

Programmer

<tr>
<td>
</td>
</tr>

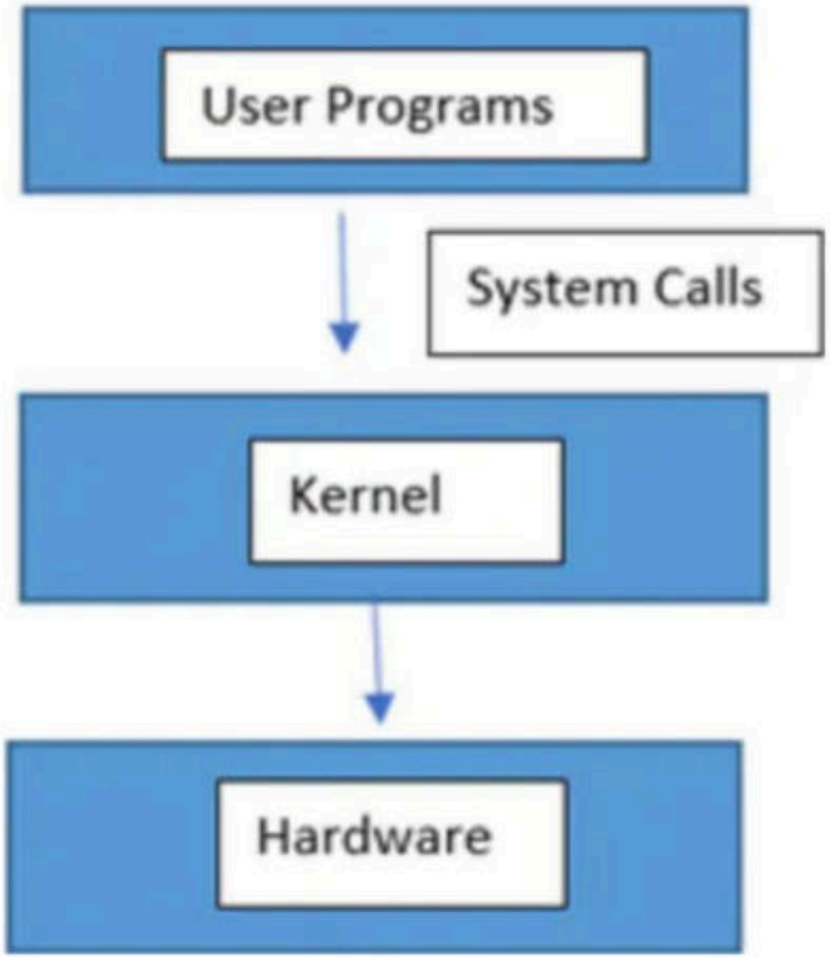
Break

System call

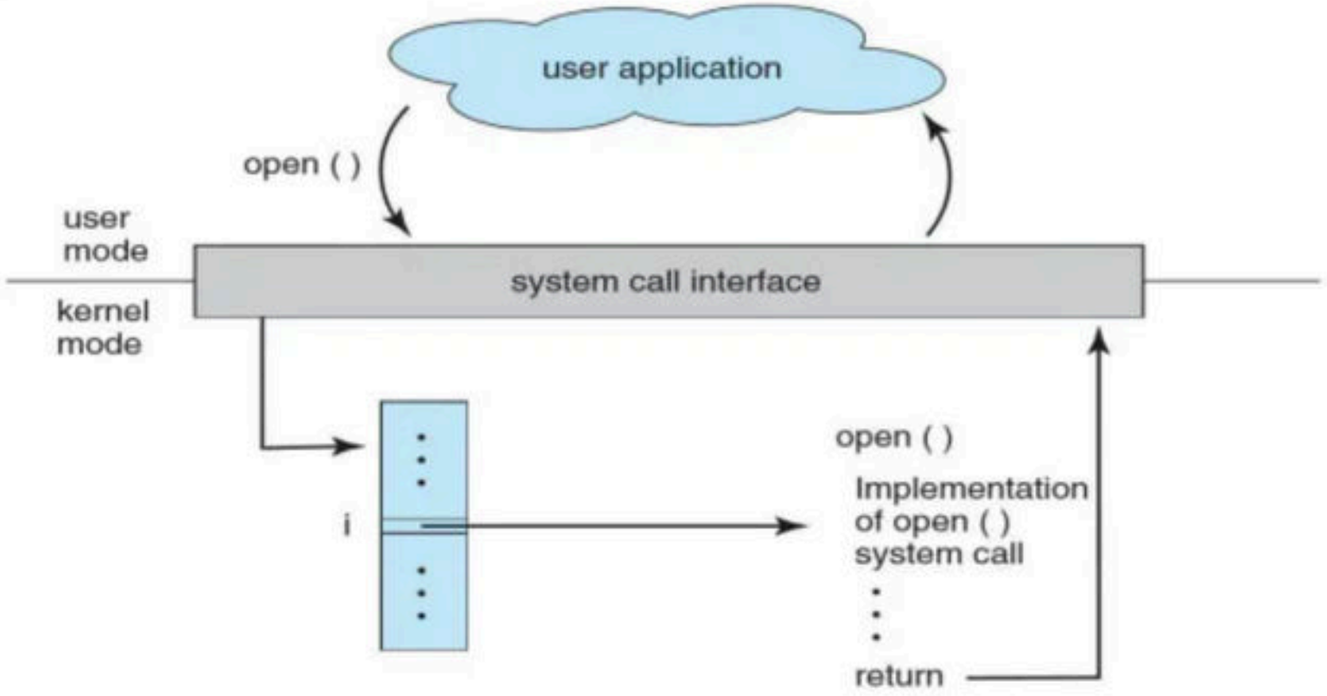
- System calls provide the means for a user program to ask the operating system to perform tasks reserved for the operating system on the user program's behalf.

- 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.

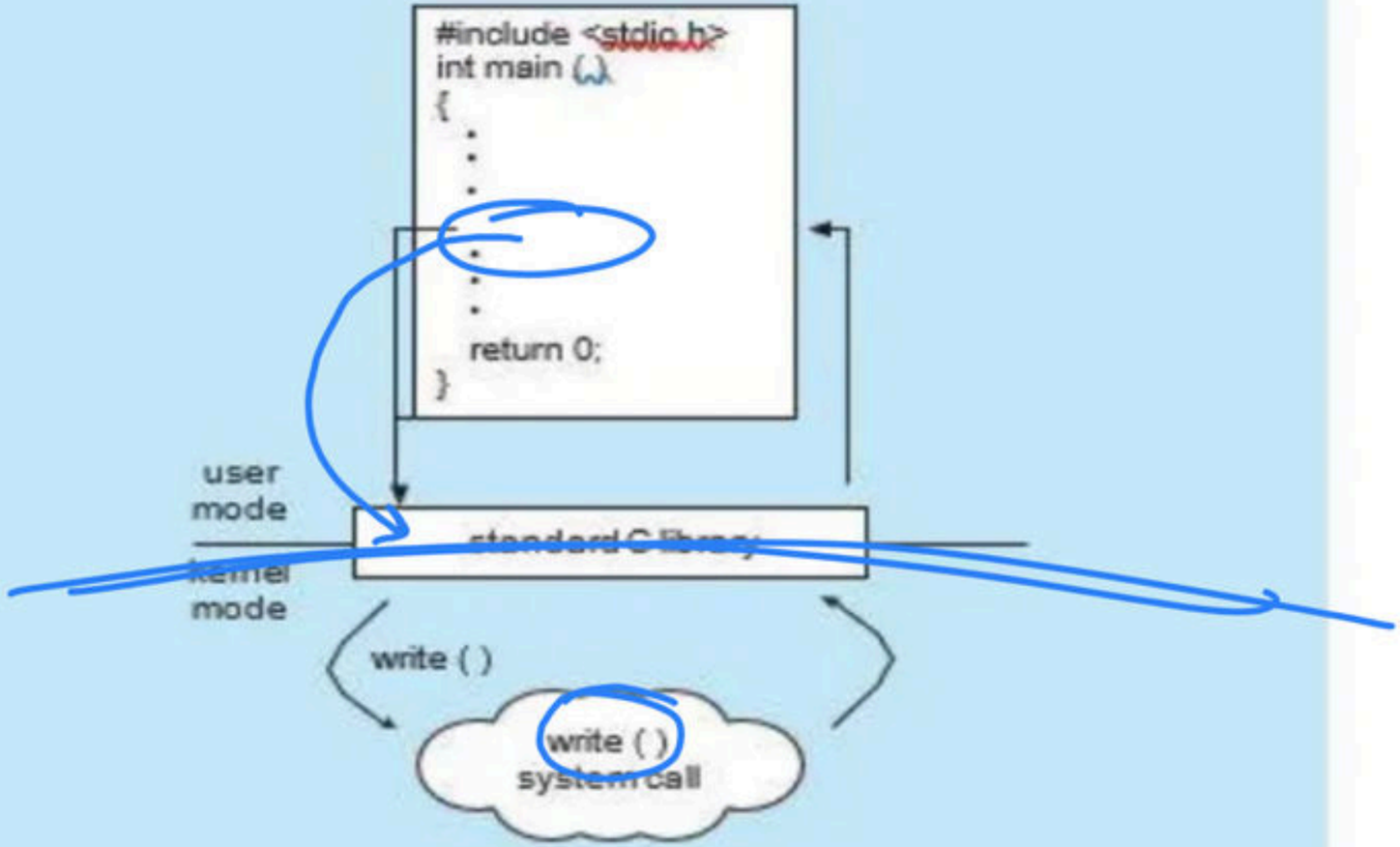


- Three of the most common APIs available to application programmers are the
  - Windows API for Windows systems
  - POSIX API for POSIX-based systems (which include virtually all versions of UNIX, Linux, and Mac OS X)
  - The Java API for programs that run on the Java virtual machine.
- The caller need 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

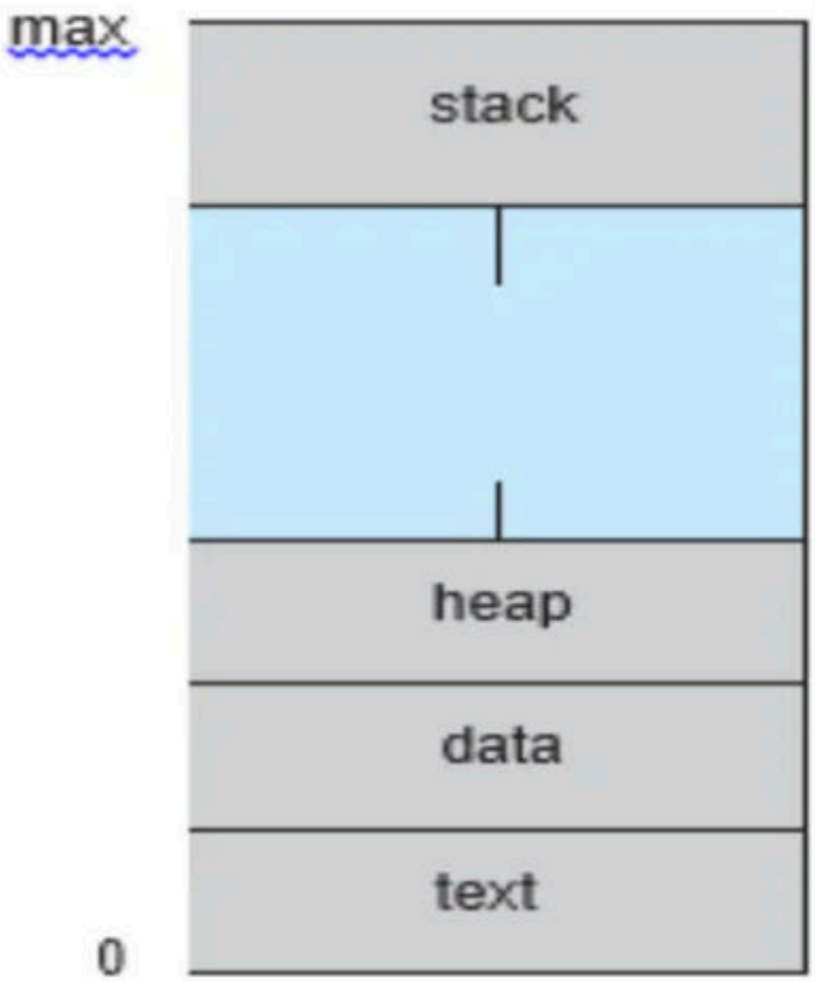
Process

- In general, a process is a program in execution.
- A Program is not a Process by default. A program is a passive entity, i.e. a file containing a list of instructions stored on disk (secondary memory) (often called an executable file).
- A program becomes a Process when an executable file is loaded into main memory and when it's PCB is created.
- A process on the other hand is an Active Entity, which require resources like main memory, CPU time, registers, system buses etc.

- Even if two processes may be associated with same program, they will be considered as two separate execution sequences and are totally different process.

- For instance, if a user has invoked many copies of web browser program, each copy will be treated as separate process. even though the text section is same but the data, heap and stack sections can vary.

- A Process consists of following sections:
  - **Text section:** also known as Program Code.
  - **Stack:** which contains the temporary data (Function Parameters, return addresses and local variables).
  - **Data Section:** containing global variables.
  - **Heap:** which is memory dynamically allocated during process runtime.



Q Process is a: **(ISRO 2009)**

a) A program in high level language kept on disk

b) Contents of main memory

c) A program in execution

d) A job in secondary memory

Break

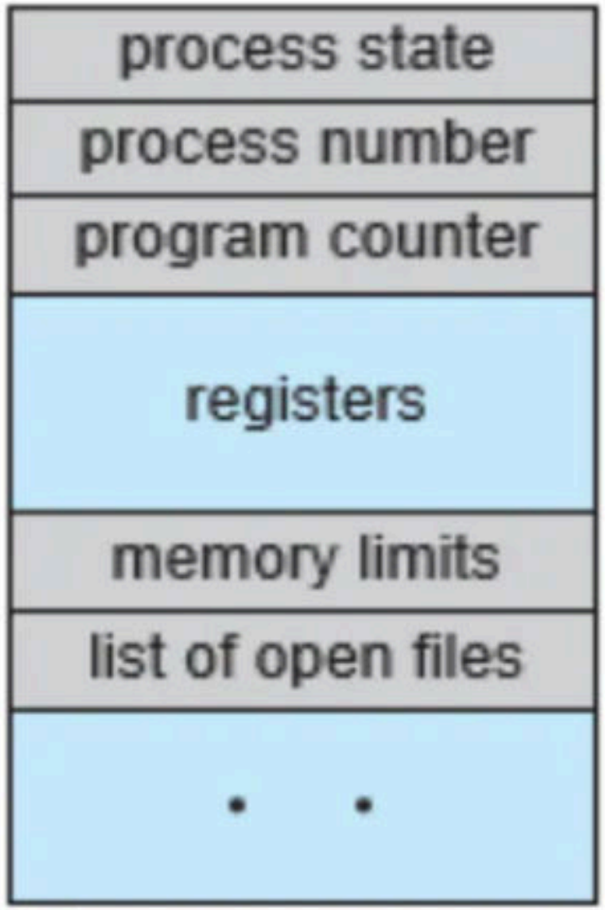
Process Control Block (PCB)

- Each process is represented in the operating system by a process control block (PCB) — also called a task control block.

- PCB simply serves as the repository for any information that may vary from process to process. It contains many pieces of information associated with a specific process, including these:
  - **Process state:** The state may be new, ready, running, waiting, halted, and so on.

  - **Program counter:** The counter indicates the address of the next instruction to be executed for this process.

  - **CPU registers:** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.

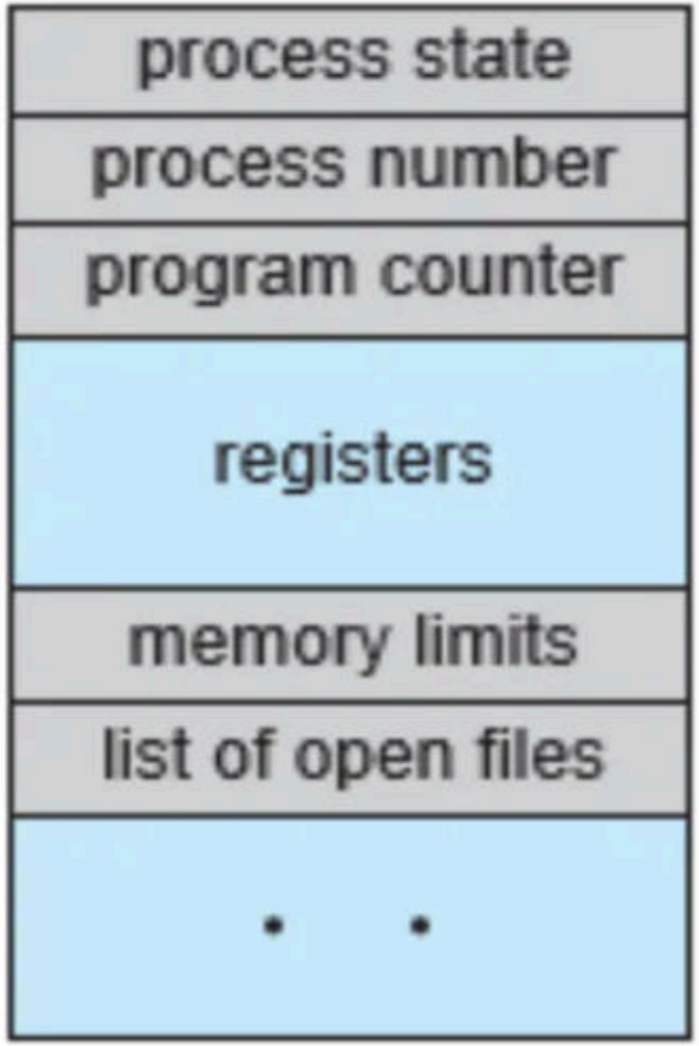


- **<u>CPU-scheduling information</u>:** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

- **<u>Memory-management information</u>:** This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system.

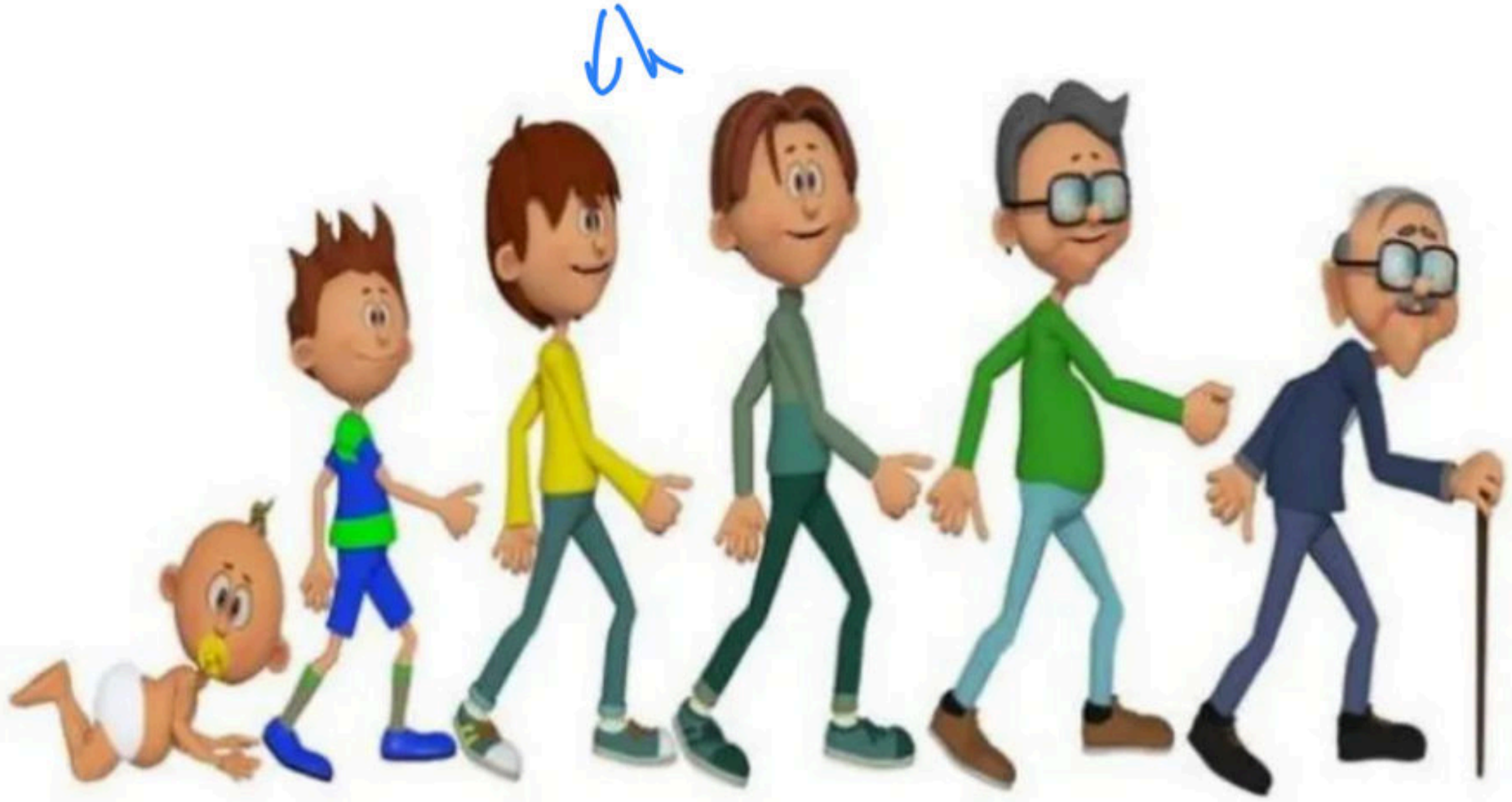
- **<u>Accounting information</u>:** This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

- **<u>I/O status information</u>:** This information includes the list of I/O devices allocated to the process, a list of open files, and so on.



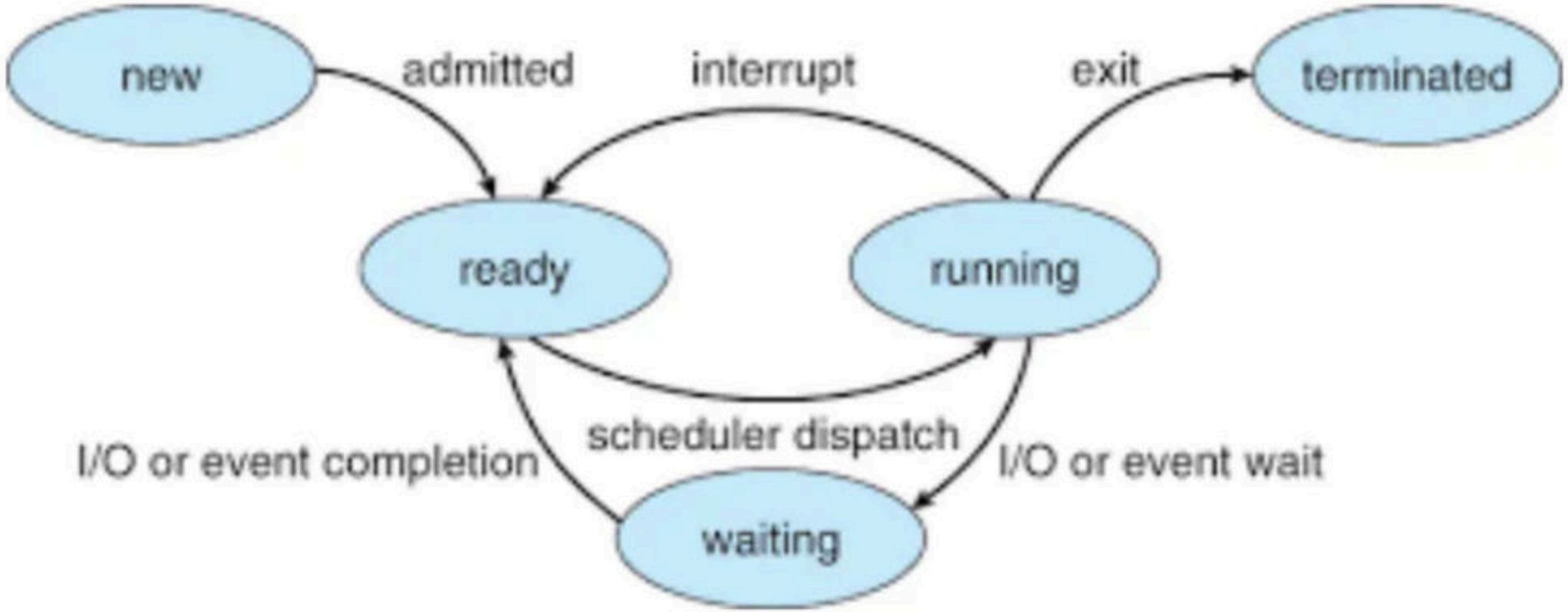
Break

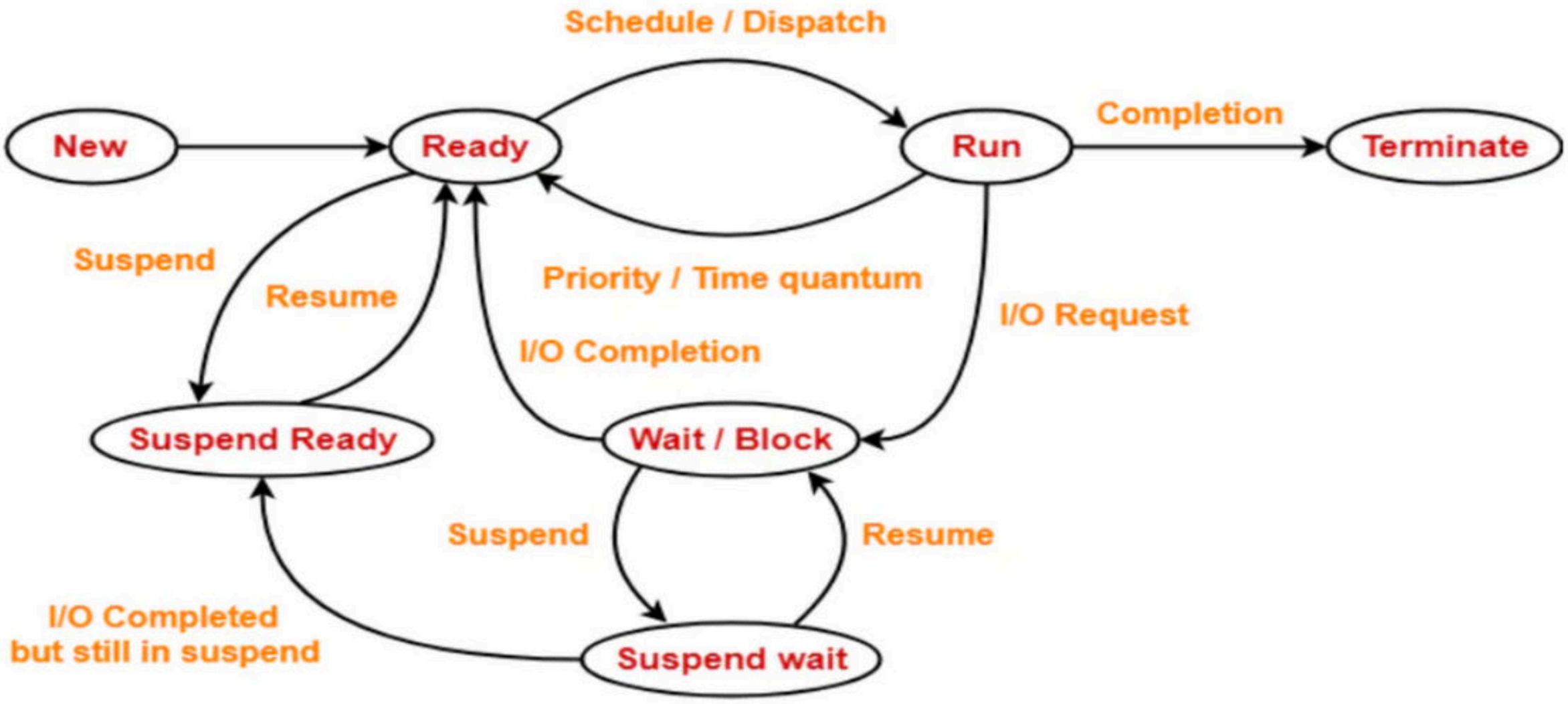
The Human Life Cycle



Process States

- A Process changes states as it executes. The state of a process is defined in parts by the current activity of that process. A process may be in one of the following states:
  - **New:** The process is being created.
  - **Running:** Instructions are being executed.
  - **Waiting (Blocked):** The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
  - **Ready:** The process is waiting to be assigned to a processor.
  - **Terminated:** The process has finished execution.


new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting



Process State Diagram

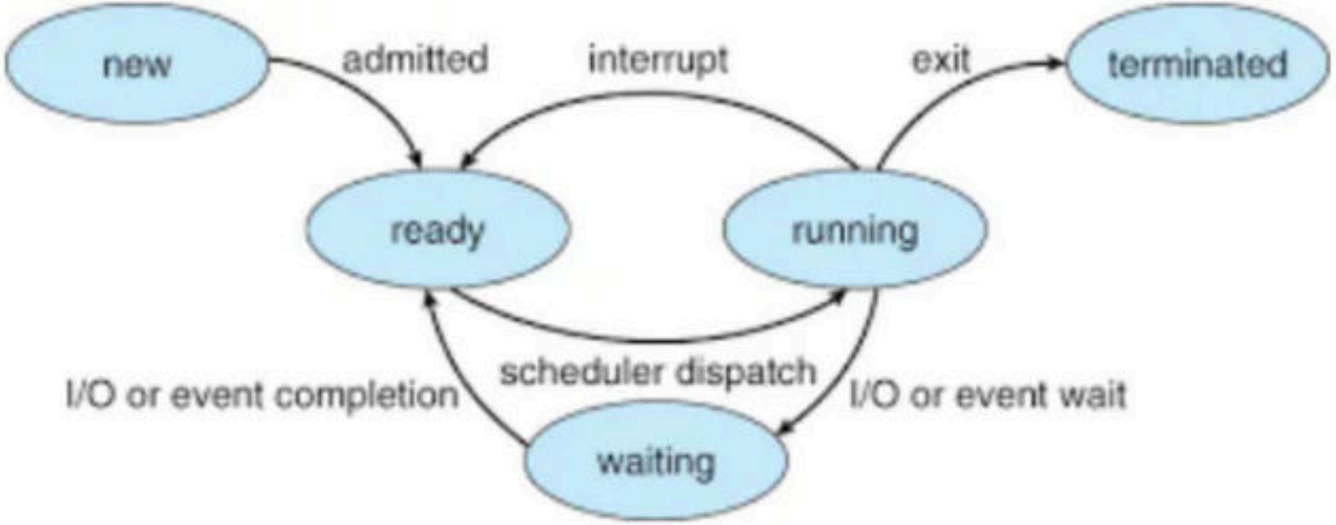
Break

Q How many states can a process be in? **(NET-DEC-2007)**

a) 2 **b)** 3 **c)** 4 **d)** 5

Q What is the minimum and maximum number of processes that can be in the ready, run, and blocked states, if total number of process is n?

	Min	Max
Ready		
Run		
Block		



Q On a system with P CPUs and n processes, what is the minimum and maximum number of processes that can be in the ready, run, and blocked states?

	Min	Max
Ready		
Run		
Block		

Q The state of a process after it encounters an I/O instruction is **(ISRO 2013)**

a) ready **b)** blocked **c)** idle **d)** running

Q There are three processes in the ready queue. When the currently running process requests for I/O how many process switches take place? **(ISRO 2011)**

a) 1 **b)** 2 **c)** 3 **d)** 4

Q What is the ready state of a process?

a) when process is scheduled to run after some execution

b) when process is using the CPU

c) when process is unable to run until some task has been completed

d) none of the mentioned

Q A task in a blocked state

a) is executable

b) is running

c) must still be placed in the run queues

d) is waiting for some temporarily unavailable resources

Break

Q Which combination of the following features will suffice to characterize an OS as a multi-programmed OS? **(GATE-2002) (2 Marks)**

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

(b) If a program waits for certain events such as I/O, another program is immediately scheduled for execution.

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

(A) a **(B)** a and b **(C)** a and c **(D)** a, b and c

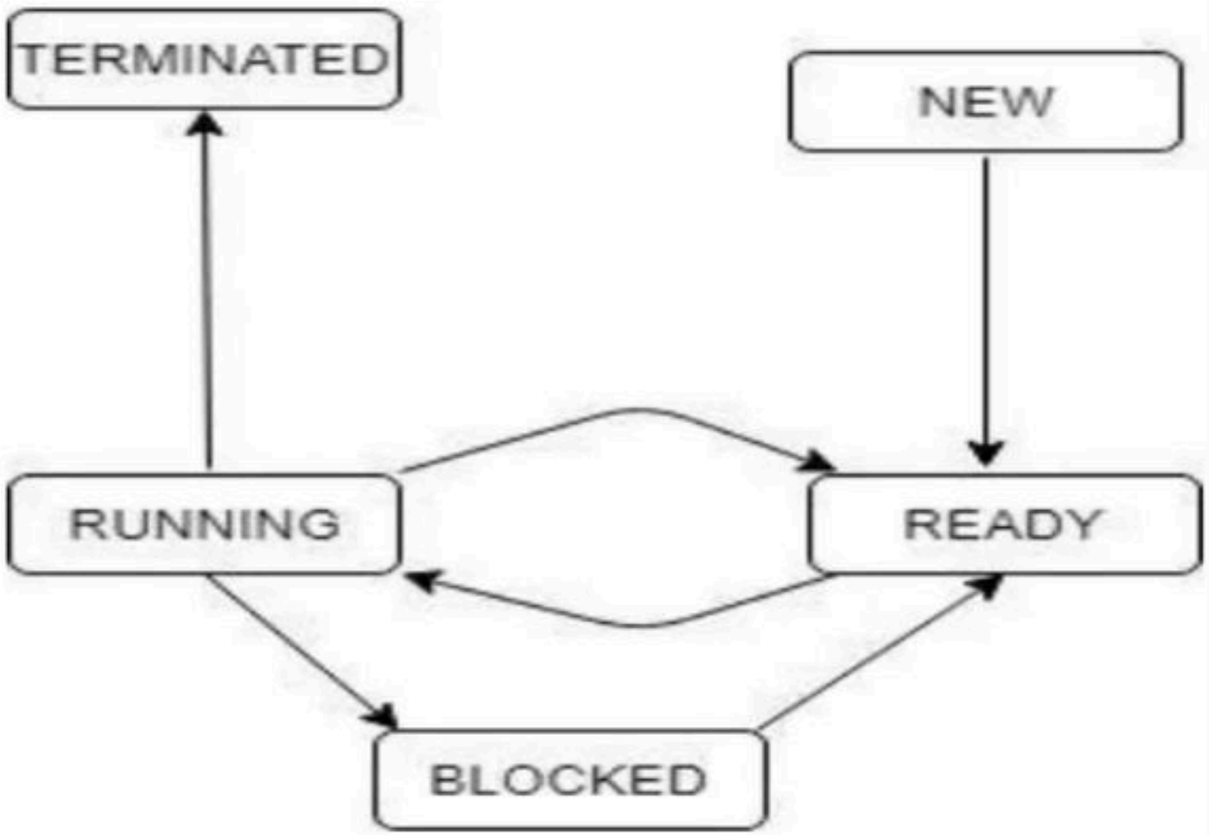
Q The process state transition diagram in below figure is representative of Untitled Diagram **(GATE-1996) (1 Marks)**

a) a batch operating system

b) an operating system with a preemptive schedular

c) an operating system with a non-preemptive schedular

d) a uni-programmed operating system



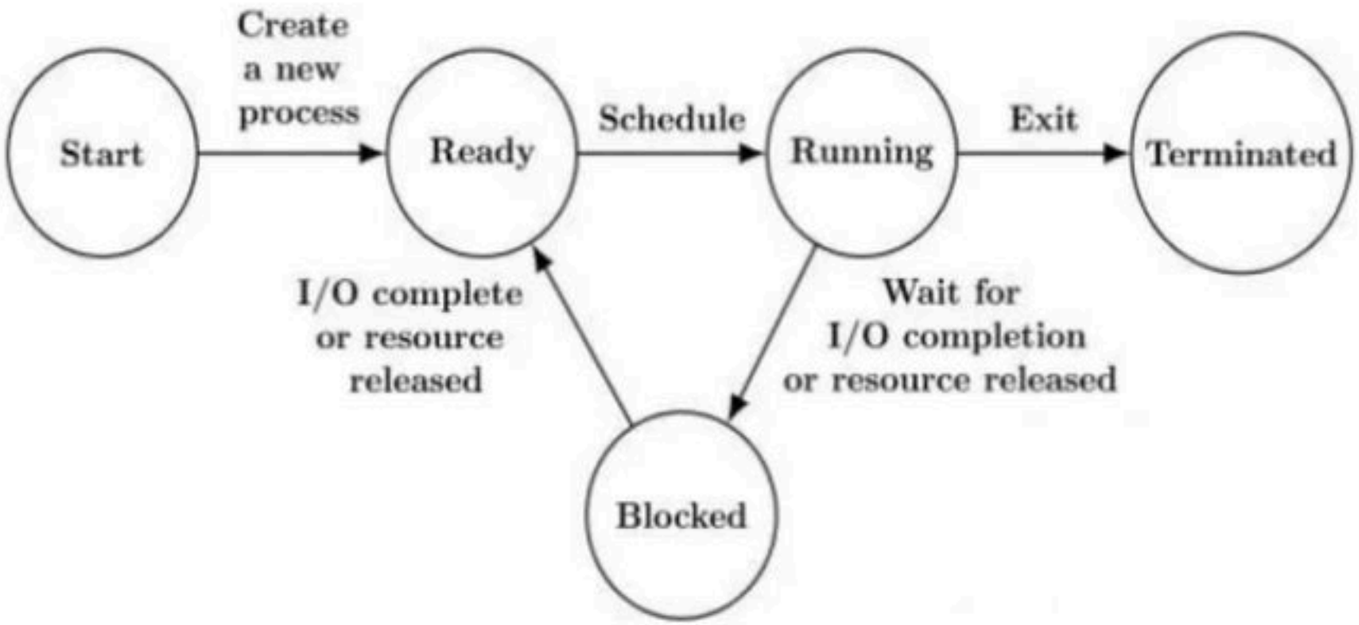
Q The process state transition diagram of an operating system is as given below. Which of the following must be FALSE about the above operating system? **(GATE-2006) (1 Marks)**

a) It is a multiprogram operating system

b) It uses preemptive scheduling

c) It uses non-preemptive scheduling

d) It is a multi-user operating system



Q In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state:

Now consider the following statements:

I) If a process makes a transition D, it would result in another process making transition A immediately.

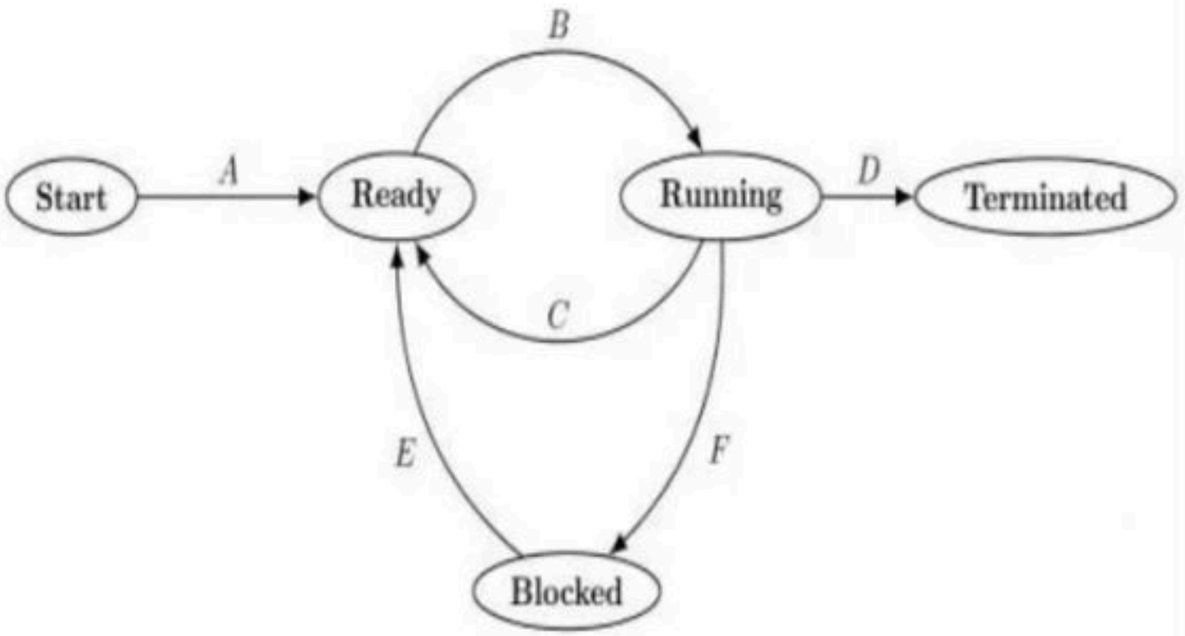
II) A process P2 in blocked state can make transition E while another process P1 is in running state.

III) The OS uses preemptive scheduling.

IV) The OS uses non-preemptive scheduling.

Which of the above statements are TRUE? **(GATE-2009) (2 Marks)**

a) I and II **b)** I and III **c)** II and III **d)** II and IV



Q A computer handles several interrupt sources of which the following are relevant for this question.

- Interrupt from CPU temperature sensor (raises interrupt if CPU temperature is too high)
- Interrupt from Mouse (raises interrupt if the mouse is moved or a button is pressed)
- Interrupt from Keyboard (raises interrupt when a key is pressed or released)
- Interrupt from Hard Disk (raises interrupt when a disk read is completed)

Which one of these will be handled at the HIGHEST priority? **(GATE - 2011) (1 Marks)**

(A) Interrupt from Hard Disk

(B) Interrupt from Mouse

(C) Interrupt from Keyboard

(D) Interrupt from CPU temperature sensor

Q Match the following: **(NET-SEP-2013)**

List – I	List - II
Process state	Reason for transition
a) Ready → Running	i. Request made by the process is satisfied or an event for which it was waiting occurs.
b) Blocked → Ready	ii. Process wishes to wait for some action by another process.
c) Running → Blocked	iii. The process is dispatched.
d) Running → Ready	iv. The process is pre-empted.

	a	b	c	d
a)	iii	i	ii	iv
b)	iv	i	iii	ii
c)	iv	iii	i	ii
d)	Iii	iii	ii	i

Q Which is the correct definition of a valid process transition in an operating system? **(NET-JUNE-2005)**

a) Wake up : Ready → Running

b) Dispatch : Ready→ Running

c) Block: Ready → Running

d) Timer run out: Ready → Blocked

Break

- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.

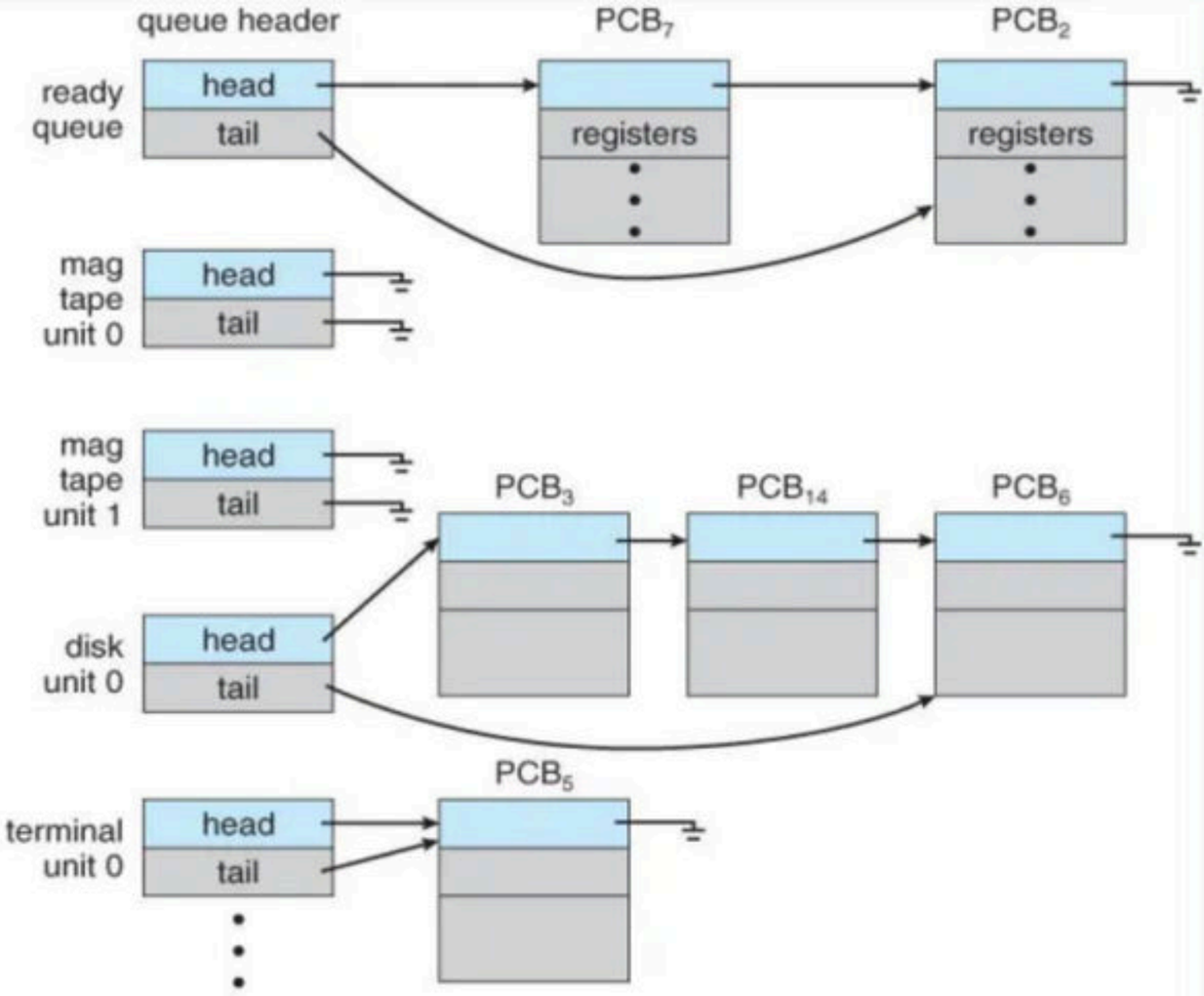
- The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running.

- To meet these objectives, the process scheduler selects an available process (possibly from a set of several available processes) for execution on the CPU. Note: For a single-processor system, there will never be more than one running process.

- Scheduling Queues - As processes enter the system, they are put into a job queue, which consists of all processes in the system.

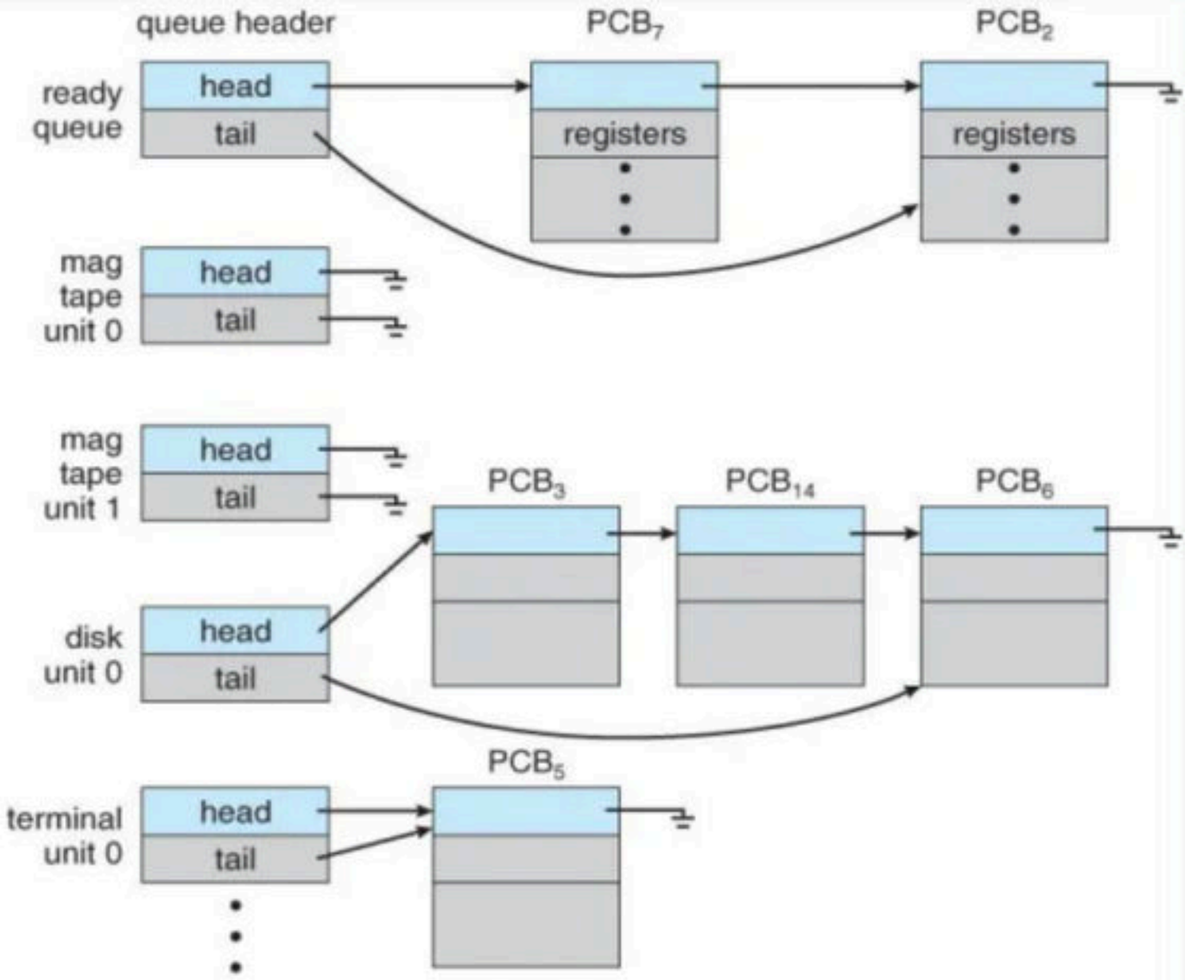
- The processes that are residing in main memory and are ready to execute are kept on a list called the ready queue. This queue is generally stored as a linked list.

- A ready-queue header contains pointers to the first and final PCBs in the list. Each PCB includes a pointer field that points to the next PCB in the ready queue.

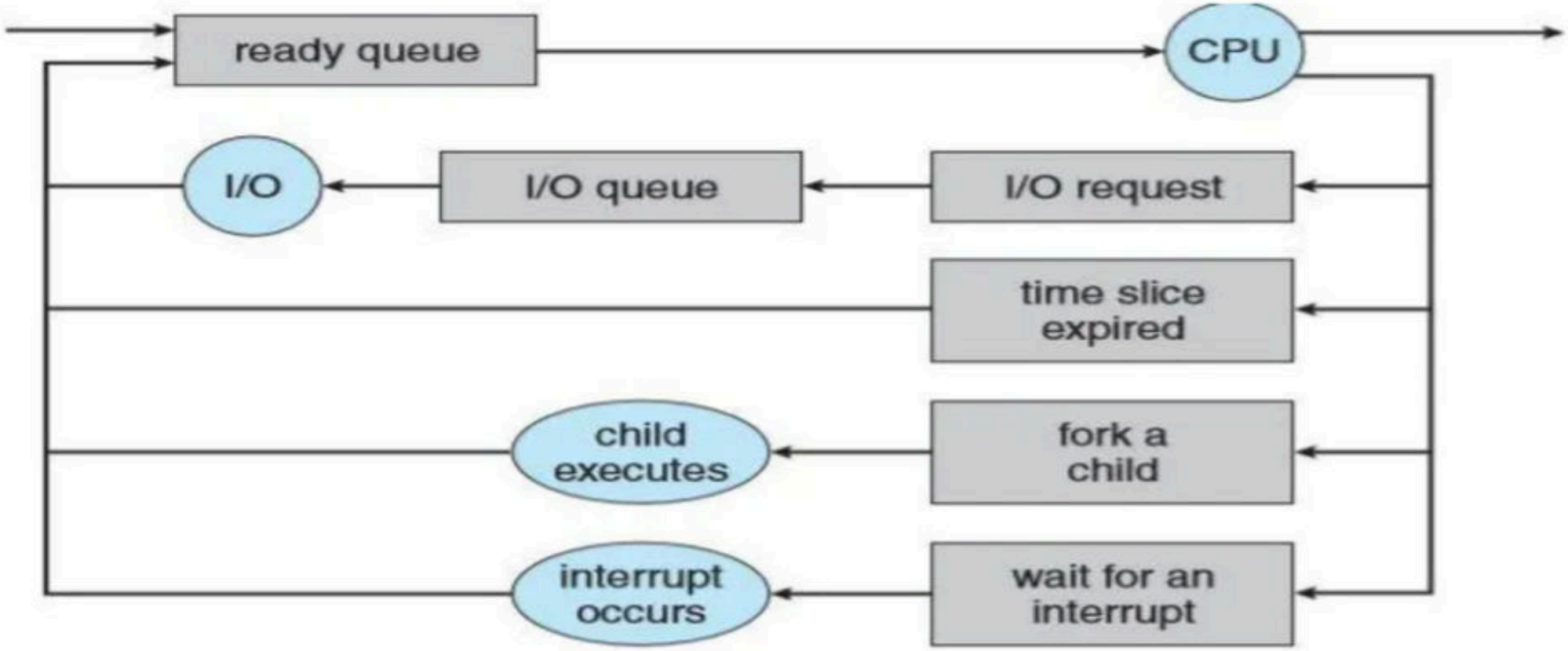


- The system also includes other queues. When a process is allocated the CPU, it executes for a while and eventually quits, is interrupted, or waits for the occurrence of a particular event, such as the completion of an I/O request.

- Suppose the process makes an I/O request to a shared device, such as a disk. Since there are many processes in the system, the disk may be busy with the I/O request of some other process. The process therefore may have to wait for the disk. The list of processes waiting for a particular I/O device is called a device queue. Each device has its own device queue.

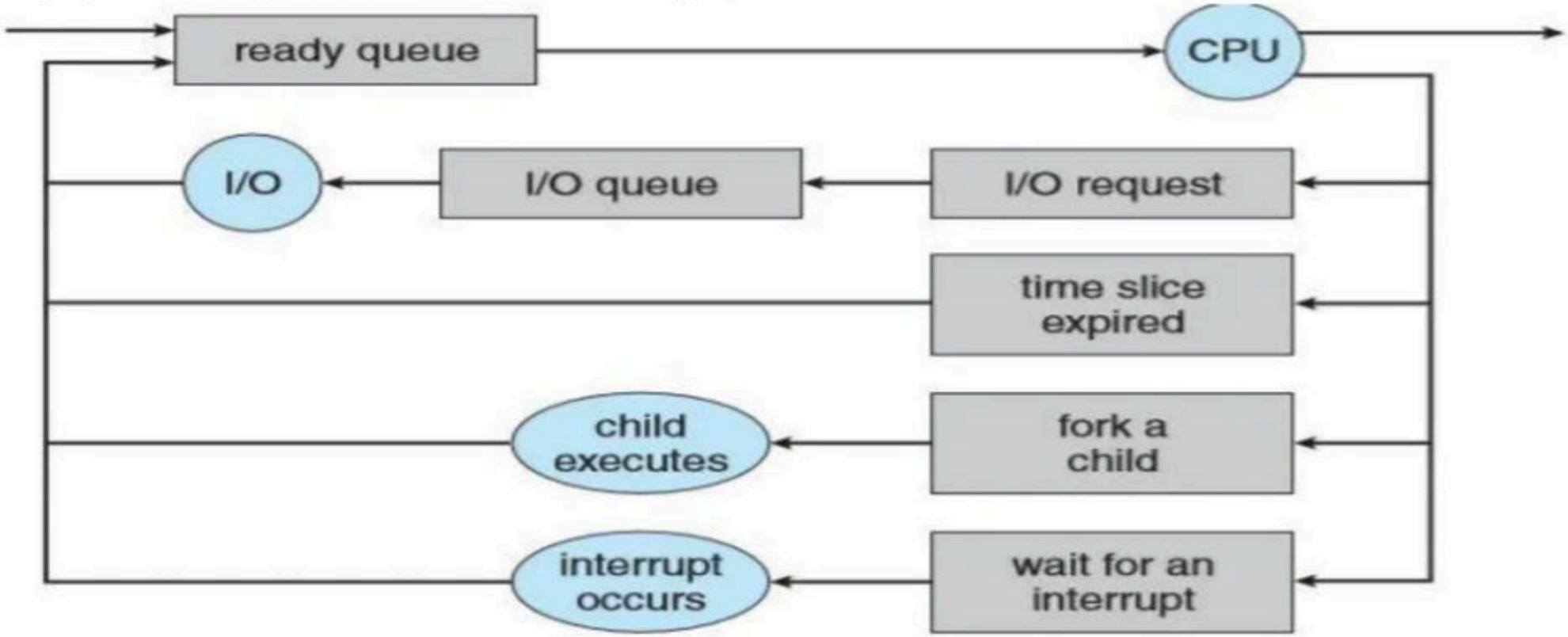


- Each rectangular box represents a queue. Two types of queues are present: the ready queue and a set of device queues. The circles represent the resources that serve the queues. A new process is initially put in the ready queue. It waits there until it is selected for execution, or dispatched.



Queueing-diagram representation of process scheduling.

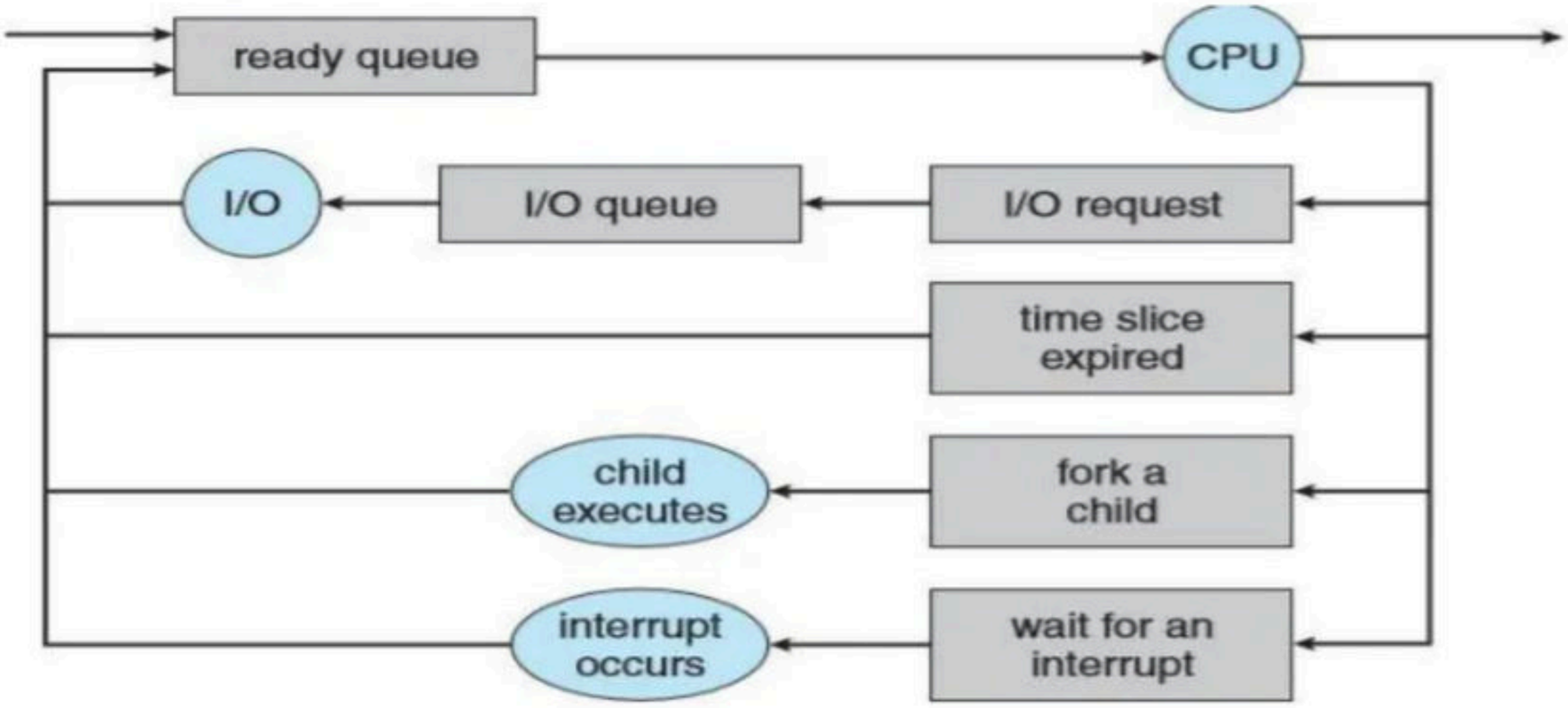
- Once the process is allocated the CPU and is executing, one of several events could occur:
  - The process could issue an I/O request and then be placed in an I/O queue.
  - The process could create a new child process and wait for the child's termination.
  - Time slice expired, the process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.



Queueing-diagram representation of process scheduling.

- In the first two cases the process eventually switches from the waiting state to the ready state and is then put back in the ready queue.

- A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.



Queueing-diagram representation of process scheduling.

Break

Schedulers

- **Schedulers:** A process migrates among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler.

- **Types of Schedulers**
  - **Long Term Schedulers (LTS)/Spooler:** In multiprogramming os, more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device (typically a disk), where they are kept for later execution. The long-term scheduler, or job scheduler, selects processes from this pool and loads them into memory for execution.

  - **Short Term Scheduler (STS):** The short-term scheduler, or CPU scheduler, selects from among the processes that are ready to execute and allocates the CPU to one of them.

- **Difference between LTS and STS** - The primary distinction between these two schedulers lies in frequency of execution.

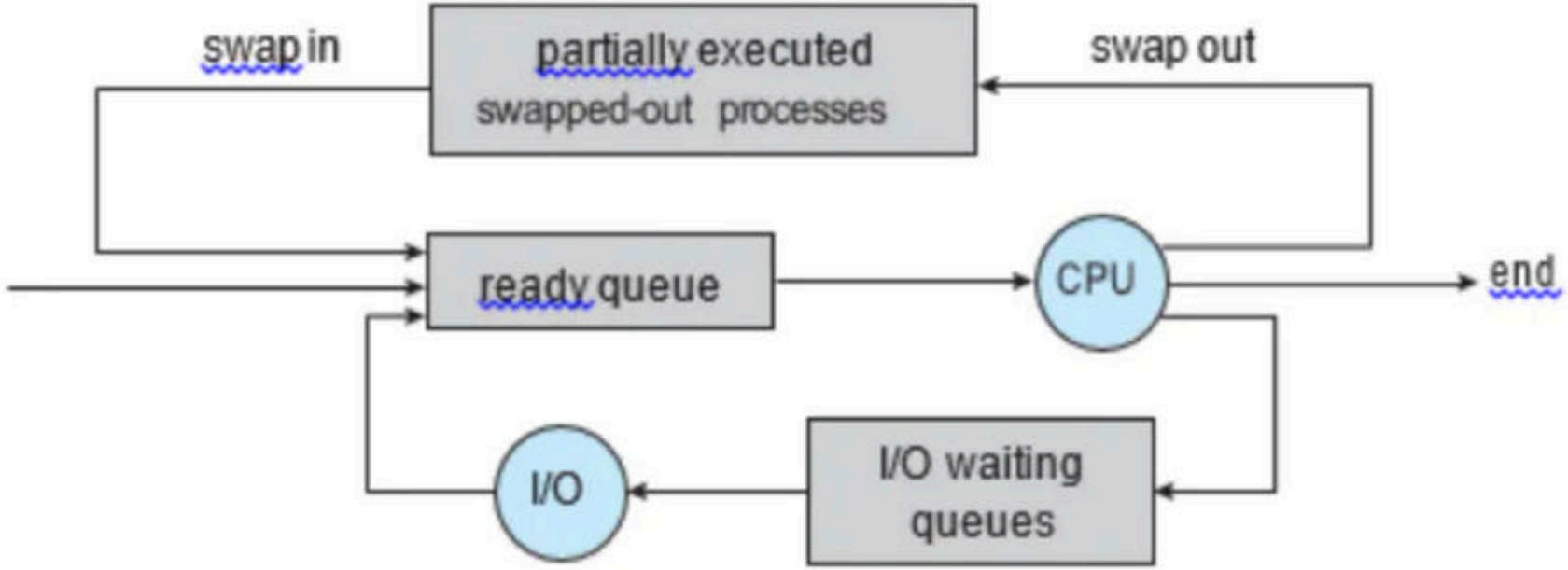
- The short-term scheduler must select a new process for the CPU frequently. A process may execute for only a few milliseconds before waiting for an I/O request. Often, the short-term scheduler executes at least once every 100 milliseconds.

- Because of the short time between executions, the short-term scheduler must be fast. The long-term scheduler on the other hand executes much less frequently; minutes may separate the creation of one new process and the next.

Degree of Multiprogramming - The number of processes in memory is known as Degree of Multiprogramming.

- The long-term scheduler controls the degree of multiprogramming as it is responsible for bringing in the processes to main memory.

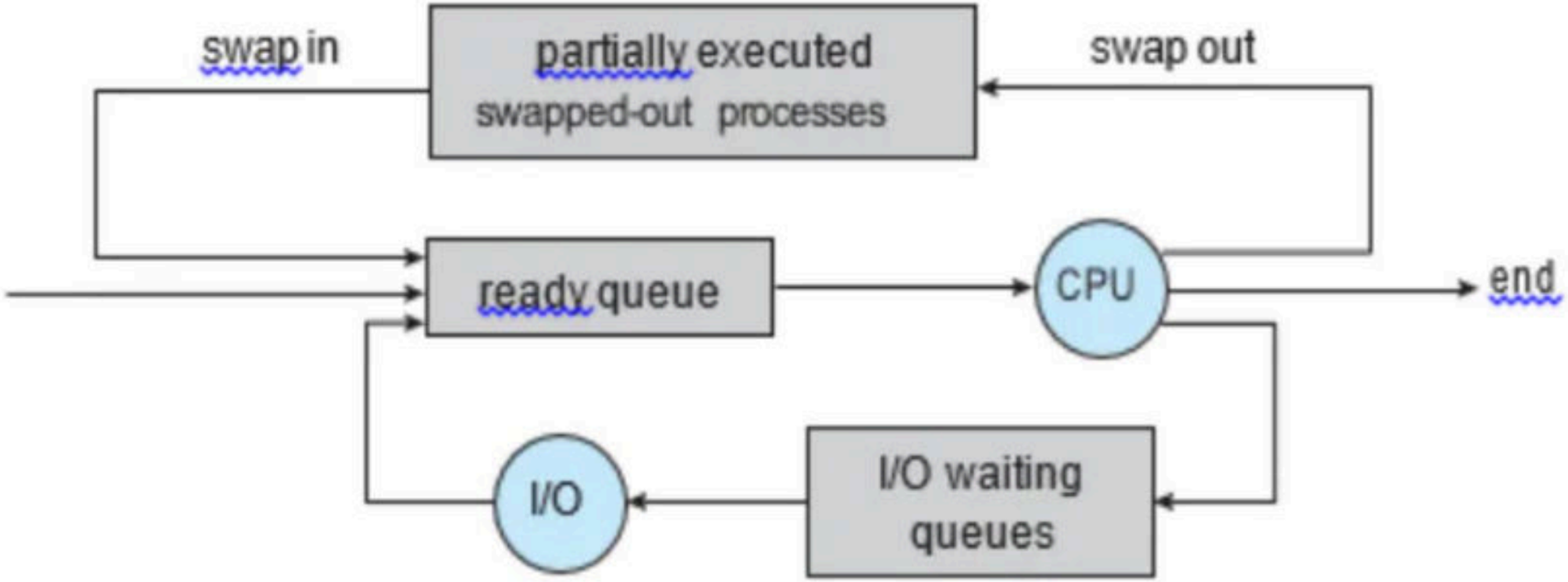
- If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system. So, this means the long-term scheduler may need to be invoked only when a process leaves the system. Because of the longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution.



Medium-term scheduler: The key idea behind a medium-term scheduler is that sometimes it can be advantageous to remove a process from memory (and from active contention for the CPU) and thus reduce the degree of multiprogramming.

Later, the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called swapping.

The process is swapped out, and is later swapped in, by the medium-term scheduler. Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up.



- **Dispatcher** - The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler.

- This function involves the following: Switching context, switching to user mode, jumping to the proper location in the user program to restart that program.

- The dispatcher should be as fast as possible, since it is invoked during every process switch. The time it takes for the dispatcher to stop one process and start another running is known as the dispatch latency.

Q A software to create a Job Queue is called..... **(NET-DEC-2006)**

a) Linkage editor

b) Interpreter

c) Driver

d) Spooler

Q A special software that is used to create a job queue is called **(NET-JUNE-2011)**

(A) Drive

(B) Spooler

(C) Interpreter

(D) Linkage editor

Q Which module gives control of the CPU to the process selected by the short - term scheduler? **(NET-NOV-2017)**

a) Dispatcher

b) Interrupt

c) Scheduler

d) Threading

Break

CPU Bound and I/O Bound Processes

- A process execution consists of a cycle of CPU execution or wait and i/o execution or wait.
- Normally a process alternates between two states.
- Process execution begin with the CPU burst that may be followed by a i/o burst, then another CPU and i/o burst and so on.
- Eventually in the last will end up on CPU burst. So, process keep switching between the CPU and i/o during execution.

- **<u>I/O Bound Processes:</u>** An I/O-bound process is one that spends more of its time doing I/O than it spends doing computations.

- **<u>CPU Bound Processes</u>:** A CPU-bound process, generates I/O requests infrequently, using more of its time doing computations.

- It is important that the long-term scheduler select a good process mix of I/O-bound and CPU-bound processes.

- If all processes are I/O bound, the ready queue will almost always be empty, and the short-term scheduler will have little to do.

- Similarly, if all processes are CPU bound, the I/O waiting queue will almost always be empty, devices will go unused, and again the system will be unbalanced.

- So, to have the best system performance LTS needs to select a good combination of I/O and CPU Bound processes.

Break

Context Switch

- When an interrupt occurs, the system needs to save the current context of the process running on the CPU so that it can restore that context when its processing is done.

- Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run. Context-switch time is pure overhead, because the system does no useful work while switching.

- Switching speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions (such as a single instruction to load or store all registers). A typical speed is a few milliseconds.

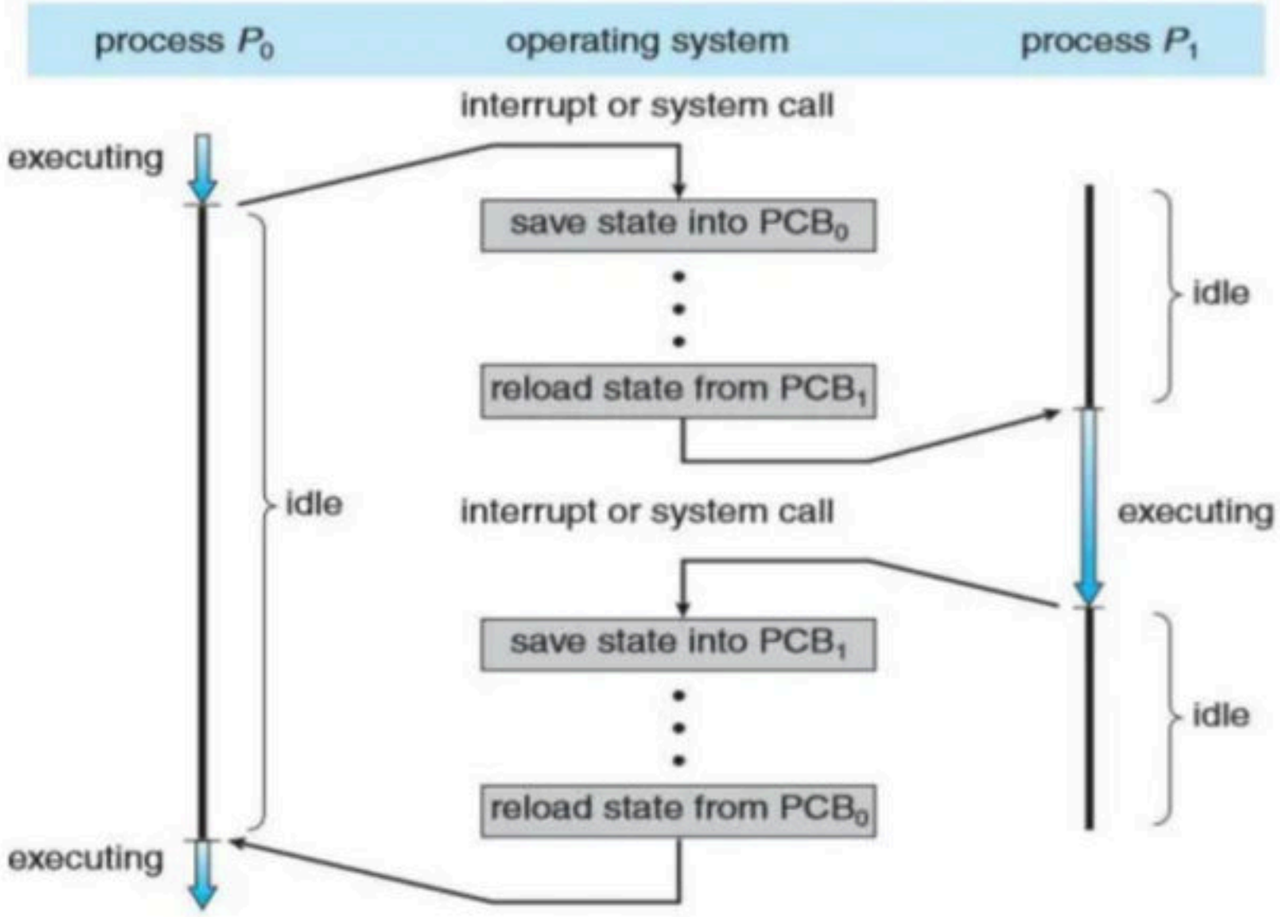


Diagram showing CPU switch from process to process.

Q Which of the following does not interrupt a running process? **(GATE-2001) (2 Marks)**

(A) A device

(B) Timer

(C) Scheduler process

(D) Power failure

Q Which is the correct definition of a valid process transition in an operating system?

a) Wake up: ready → running

b) Dispatch: ready → running

c) Block: ready → running

d) Timer runout: ready → running

Q A process stack does not contain

a) function parameters

b) local variables

c) return addresses

d) PID of child process

Q Loading operating system from secondary memory to primary memory is called..... **(NET-DEC-2006)**

(A) Compiling

(B) Booting

(C) Refreshing

(D) Reassembling