Meerut Institute of Engineering and Technology, Meerut

Operating System (KCS 401)

Solution: Sessional Test 1

Section – A (CO - 1) # Attempt both the questions # 30 Marks

Q.1: Attempt any **SIX** questions (Short Answer Type). Each question is of two marks. **(2 x 6 = 12 Marks)**

a) Write down different Operating System Structures.

Ans: a) Operating system consists of different type of components. These components are interconnected and melded into kernels.

These structures are:

1) Simple Structure : Simple structure operating systems are small, simple and limited systems. Example: MS-DOS System Structure.

2) Layered approach : 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. Example: Windows NT

3) Microkernels : In this architecture, all the basic OS services which are made part of user space are made to run as servers which are used by other programs in the system through inter process communication (IPC). Example: Mac Osx

b) Define the goals of the Operating System.

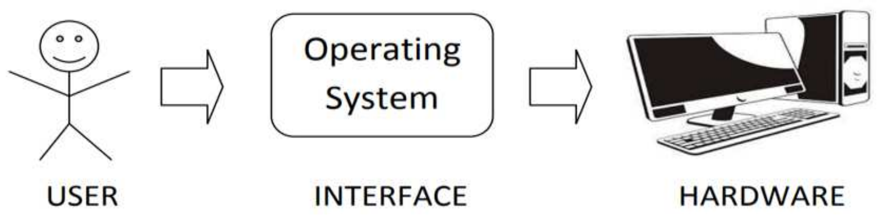
Ans: b) Objective/Functions/Goals : OS can be thought of as having three objectives:

1. Convenience : Makes a computer more convenient to use.
2. Efficiency : Resources to be used in an efficient manner.
3. Ability to evolve : Introduction of new system functions without interfering with service.

An OS provides standard services (an interface) which are implemented on the hardware, including: Processes, CPU scheduling, memory management, file systems, networking.

c) Define the Operating System and list its services.

Ans:c) OPERATING SYSTEM : "An OS is a program that controls the execution of application programs and acts as an interface between the user of the computer and the computer hardware."

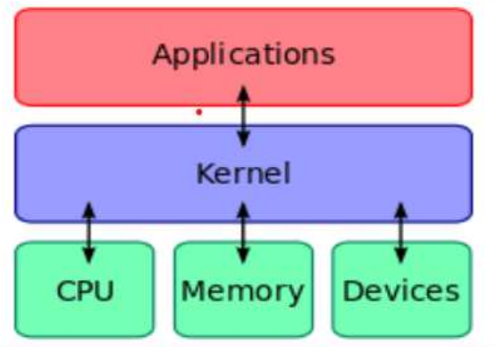


These services are

- Program execution
- I/O operations
- File-system manipulation
- Communications
- Error detection
- **Additional functions exist not for helping the user, but rather for ensuring efficient system operations.**
- Resource allocation
- Accounting
- Protection.

d) What do you mean by Kernel?

Ans: d) Kernel is the heart of OS which manages the core features of an OS while if some useful applications and utilities are added over the kernel, then the complete package becomes an OS.



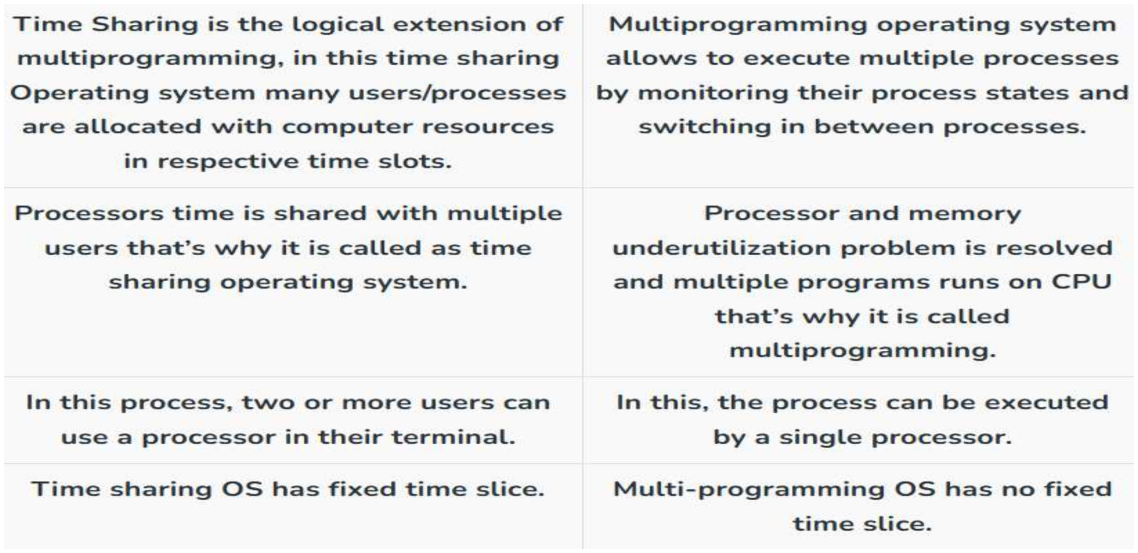
A kernel connects the application software to the hardware of a computer

- It is the part of the operating system that loads first, and it remains in main memory. Because it stays in memory, it is important for the kernel to be as small as possible.

e) Compare Multiprogramming and Time-sharing Systems.

Ans: e) In multi-programming many process reside in main memory and two process run in system one process use CPU and another use I/O Devices.

- Two types of multiprogramming : non-preemptive and preemptive (Time sharing/ Multitasking)

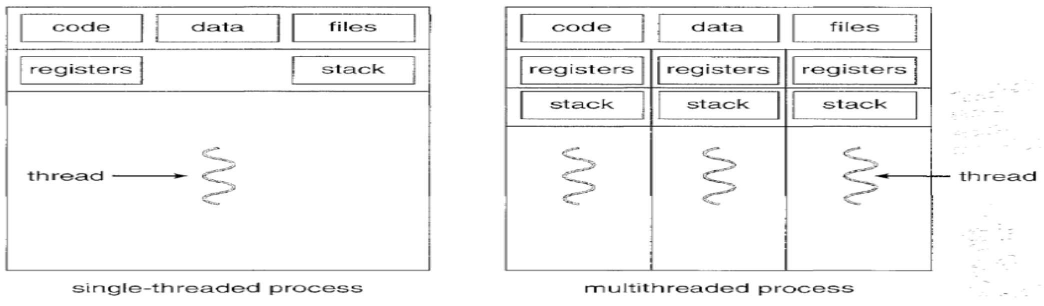


Time Sharing is the logical extension of multiprogramming, in this time sharing Operating system many users/processes are allocated with computer resources in respective time slots.	Multiprogramming operating system allows to execute multiple processes by monitoring their process states and switching in between processes.
Processors time is shared with multiple users that's why it is called as time sharing operating system.	Processor and memory underutilization problem is resolved and multiple programs runs on CPU that's why it is called multiprogramming.
In this process, two or more users can use a processor in their terminal.	In this, the process can be executed by a single processor.
Time sharing OS has fixed time slice.	Multi-programming OS has no fixed time slice.

f) Define Multithreaded OS.

Ans: f) Multithreaded Operating Systems : The concept of multi-threading needs proper understanding of these two terms – a process and a thread. A process is a program being executed. A process can be further divided into independent units known as threads.

- A thread is like a small light-weight process within a process. Or we can say a collection of threads is what is known as a process.
- A word processor may have threads for;
    - Displaying graphics
    - Responding to key stroke from user
    - Performing spelling and grammar checking



g) How is the System call handled by the System?

Ans:g) System calls provide the interface between a running program and the operating system.

- A system call instruction is an instruction that generates an interrupt. Mode bit is added to h/w of computer to indicate the current mode: Kernel (0), User (1).
- Whenever trap (interrupt) occurs h/w switches from user mode to kernel mode (change the mode bit to 0).

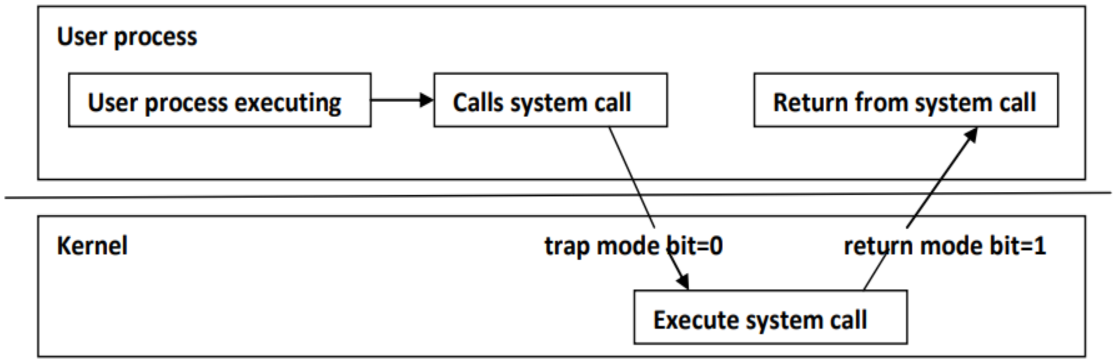


Figure: General Structure of System Call

Q.2: Attempt any **THREE** questions (Medium Answer Type). Each question is of 6 marks. **(3 x 6 = 18 Marks)**

a) Write the comparative analysis between different Operating System Structures.

Sol- **<u>Comparative Analysis of OS Structures</u>**

Points of Comparison	Monolithic	Layered	Microkernel	Re-entrant
Working	In monolithic kernel, both user services and kernel services are kept in the same address space.	The operating system is divided into several layers, each performing a specific task.	In microkernel user services and kernel, services are kept in separate address space.	Reentrant kernel is the one which allows multiple processes to be executing in the kernel mode at any given point of time and that too without causing any consistency problems among the kernel data structures
Modularity	In monolithic OS, all the components and the kernel can directly interact, making it less modular.	The layered architecture is modular as each layer performs only the tasks it is scheduled to perform.	Microkernels are modular, and the different modules can be replaced, reloaded, modified without even touching the Kernel.	Modular in nature as work is divided in two separate mode – user and kernel and different functionalities are implemented at every mode.
Modification	These are difficult to	These are easier to	In microkernel, we	Complex task as

& Updates	modify and update because all the code needs to be recompiled again after a minor update.	update since the code of the concerned layer needs to be modified only.	can add new features or patches without recompiling.	needs to have good programming skills to maintain synchronization.
Size	Monolithic kernel is larger in size.	Big in Size	Microkernel is smaller in size.	Its size is big.
Execution	Fast execution.	As every response goes through all the layers, the execution of tasks can often become tiresome and delayed.	Slow execution	Provides good performance as speed is good because its work is based on synchronization.
Security	If a service crashes, the whole system crashes in monolithic kernel. It's because user programs use the same address spaces as the kernel.	Greater Security as Debugging can be performed well with the layer that is debugged will be corrected as the layers existing below are already functioning properly.	If a service crashes, it never effect on working of microkernel. (Crash Resistant).	It works based on locking mechanism so unwanted (malicious) modification can't happen.
Example	QNX, Symbian, L4Linux, Singularity, K42, Mac OS X, Integrity, PikeOS, HURD, Minix, and Coyotos.	Windows New Technology (NT), Windows XP, Windows Vista	Linux, BSDs (FreeBSD, OpenBSD, NetBSD), Microsoft Windows (95,98,Me), Solaris, OS-9, AIX, HP-UX, DOS, OpenVMS, XTS-400 etc.	Unix

b) Differentiate between the followings:

i. Shell and Kernel

ii. Interactive and Batch OS

Sol- (i) A kernel is a type of low-level program that has its interfacing with the hardware on top of which all the applications run (disks, RAM, CPU, etc.).

The shell is a command-line interface that allows the user to enter commands to interact with the operating system. A shell allows all of its users to establish communication with the kernel.

(ii) An interactive operating system allows the user to interact directly with the system. In this type of operating system, the user enters a command into the system, and the system executes it. Programs that allow users to enter data or commands are known as interactive systems

A batch operating system (BOS) is a type of operating system that allows multiple users to use it at the same time. It does not allow any direct communication between users, meaning that each user must complete their tasks before passing control on to the next user.

c) Whether the general user can operate a Computer System without Operating System. Discuss.

Sol- An operating system is a group of computer programs that coordinates all the activities among computer hardware device. Without Operating System the computer can't run the application and we can not do work in the computer.

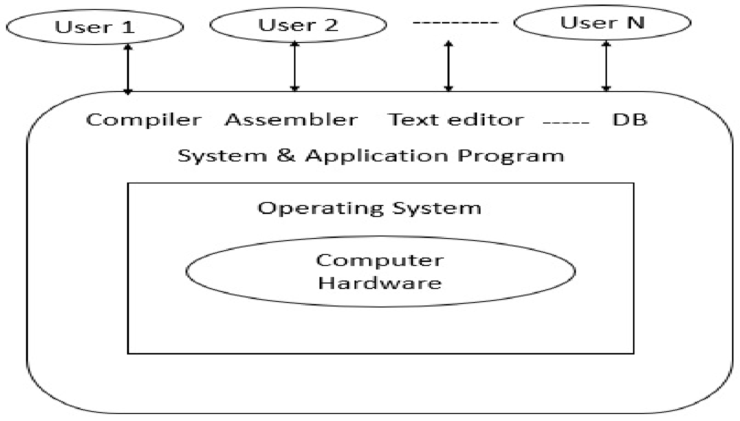
So, Without an operating system computer hardware is only an inactive electronic machine, which is inconvenient to user for execution of programs. As the computer hardware or machine understands only the machine language. It is difficult to develop each and every program in machine language in order to execute it.

d) Draw the abstract view of a computer system. Discuss various System Components.

Sol- A computer system consists of many resources like hardware and software, which are useful to complete a task. The common required resources are input/output devices, memory, file storage space, CPU etc.

The operating system acts as a manager for all the above resources and allocates them to specific programs and users, whenever necessary to perform a particular task. Therefore, the operating system is the resource manager that means it can manage the resources of a computer system internally. The resources are processor, memory, files, and I/O devices.

The abstract view of components of computer system is as follows –



Components of Computer System

A computer system can be divided into four components, which are as follows –

Hardware – The hardware is the physical part which we can touch and feel, the central processing unit (CPU), the memory, and the input/output (I/O) devices are the basic computing resources of a computer system.

Application programs – Application programs are user or programmer created programs like compilers, database systems, games, and business programs that define the ways in which these resources can be used to solve the computing problems of the users.

Users – There are different types of users like people, machines, and even other computers which are trying to solve different problems.

Operating system – An operating system is the interface between the user and the machine which controls and coordinates the use of the hardware among the various application programs for the various users.

e) Compare Symmetric and Asymmetric Multiprocessing with a suitable example.

Sol- In asymmetric multiprocessing, all the processors are not identical and they follow a master-slave relationship; whereas in symmetric multiprocessing, all the processors are identical and they share the main memory.

Asymmetric multiprocessing is the use of two or more processors handled by one master processor. All CPUs are interconnected but are not self-scheduling. It is used to schedule specific tasks to a CPU based on priority and importance of task.

Asymmetric multiprocessing refers to a type of computer architecture in which there are multiple processors, but they are not all the same. This means, one CPU might be handling the operating system codes while another CPU is performing inputting and outputting jobs.

Asymmetric multiprocessing systems are often used in embedded systems where specific tasks need to be performed concurrently, but the system does not require the same level of general-purpose computing power as a symmetric multiprocessing system.

Symmetric multiprocessing refers to a type of computer architecture in which two or more processors are connected to a shared main memory, and are able to work together to perform tasks. These processors are generally identical and are able to run any task that is assigned to them. This allows for improved performance, as tasks can be divided among the processors, allowing them to be completed more quickly.

Symmetric multiprocessing applies multiple CPUs to a task to complete in parallel and faster fashion. Therefore, in the symmetric multiprocessing system, two or more CPUs are connected to a shared main memory. Also, all these CPUs have full access to the input and output devices. In symmetric multiprocessing, the operating system considers all the processors equal.

Key	Asymmetric Multiprocessing	Symmetric Multiprocessing
CPU	All processors are not equal in precedence.	All processors are same in precedence.
OS Task	OS task is done by master processor.	OS task can be done by any processor.
Communication Overhead	No communication overhead between processors as they are controlled by master processor.	All processor communicates to each other using shared memory.
Process Scheduling	Master-Slave approach is used.	A ready queue of processes is used.
Cost	Asymmetric multiprocessing is cheaper to implement.	Symmetric multiprocessing is costlier to implement.
Design Complexity	Asymmetric multiprocessing is simpler to design.	Symmetric multiprocessing is complex to design.

Section – B (CO - 2) # Attempt both the questions # 30 Marks

Q.3: Attempt any **SIX** questions (Short Answer Type). Each question is of two marks. **(2 x 6 = 12 Marks)**

a) Why do we need Scheduling?

Ans : Scheduling algorithms are used in operating systems to manage the allocation of resources to processes. The purpose of a scheduling algorithm is to provide

- Maximum CPU utilization,
- Fair allocation of CPU time to every process,
- Maximize throughput,
- Minimize the turnaround time,
- Minimize the waiting time
- Minimize the response time.

In other words, scheduling algorithms help ensure that the CPU is always busy executing a process and that every process gets a fair share of CPU time. This helps improve system performance and ensures that no process is left waiting for an excessive amount of time.

b) Describe the Convoy effect.

Ans: Convoy Effect is phenomenon associated with the First Come First Serve (FCFS) algorithm, in which the whole Operating System slows down due to few slow processes. All other process waits for the one big process to get off the CPU.FCFS algorithm is non-pre-emptive in nature, that is, once CPU time has been allocated to a process, other processes can get CPU time only after the current process has finished. This property of FCFS scheduling leads to the situation called Convoy Effect. If the CPU gets the processes of the higher burst time at the front end of the ready queue, then the processes of lower burst time may get blocked which means they may never get the CPU if the job in the execution has a very high burst time. This is called **convoy effect**. Convoy Effect, one slow process slows down the performance of the entire set of processes, and leads to wastage of CPU time and other devices.

c) List different preemptive and non-preemptive CPU Scheduling algorithms.

Ans : Non Pre-emptive Scheduling Algorithm

- First Come First Serve (FCFS)
- Shortest Job First (SJF)
- Largest Job First (LJF)
- Highest Response Ratio Next (HRRN)
- Priority Scheduling
- Multi-level Queue (MLQ)
- Multi-level Feedback Queue (MLFQ)

Pre-emptive Scheduling Algorithm

- Shortest Remaining Time First (SRTF)
- Round Robin (RR)
- Longest Remaining Time First (LRTF)
- Priority Scheduling
- Multi-level Queue (MLQ)
- Multi-level Feedback Queue (MLFQ)

d) What resources are used when a thread is created?

Ans : When a thread is created the threads does not require any new resources to execute the thread shares the resources like memory of the process to which they belong to. The benefit of code sharing is that it allows an application to have several different threads of activity all within the same address space. Whereas if a new process creation is very heavyweight because it always requires new address space to be created and even if they share the memory then the inter process communication is expensive when compared to the communication between the threads.

e) What is thread and how is it different from a process?

Ans: A **process** is an executing program that includes the program itself, data, resources such as files, and execution info such as process relation information kept by the OS. The OS allows users to create, schedule, and terminate the processes via system calls.

A **thread** is a semi-process that has its own stack and executes a given piece of code. Unlike a real process, the thread normally shares its memory with other threads. Processes usually have a different memory area for each one of them. A thread is the basic unit to which the operating system allocates processor time. A thread can execute any part of the process code, including parts currently being executed by another thread.
In summary, a process is an executing program that includes multiple threads while a thread is a lightweight process that can be managed independently by a scheduler.

f) Explain Context Switching.
Ans: Context switching is the process of storing and restoring the state (context) of a process or thread so that execution can be resumed from the same point at a later time system. In the context of **scheduling algorithms**, context switching is used to save the current state of a process so that it can resume execution later. The short-term scheduler and scheduling algorithms operate on the ready queue, where the processes are in ready state

g) What are the performance criteria in CPU Scheduling?

Ans: The criteria include the following:

- **CPU utilization:** The main purpose of any CPU algorithm is to keep the CPU as busy as possible. Theoretically, CPU usage can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the system load.
- **Throughput:** The average CPU performance is the number of processes performed and completed during each unit. This is called throughput. The output may vary depending on the length or duration of the processes.
- **Turn round Time:** For a particular process, the important conditions are how long it takes to perform that process. The time elapsed from the time of process delivery to the time of completion is known as the conversion time. Conversion time is the amount of time spent waiting for memory access, waiting in line, using CPU, and waiting for I / O.
- **Waiting Time:** The Scheduling algorithm does not affect the time required to complete the process once it has started performing. It only affects the waiting time of the process i.e., the time spent in the waiting process in the ready queue.
- **Response Time:** In a collaborative system, turnaround time is not the best option. The process may produce something early and continue to computing the new results while the previous results are released to the user. Therefore, another method is the time taken in the submission of the application process until the first response is issued. This measure is called response time.

Q.4: Attempt any **THREE** questions (Medium Answer Type). Each question is of 6 marks. **(3 x 6 = 18 Marks)**

(a) Define Process. Explain various steps involved in the change of a process state with a neat Life Cycle of the Process diagram.

Solution- **Process-** A process is a program in execution. Components of the process are:

1. Object Program
2. Data
3. Resources
4. States of the process execution

Object program i.e. code to be executed. Data is used for executing program. While executing the program, it may require some resources. Last component is used for verifying the status ofthe process execution.

- A process is more than the program code, which is sometimes known as the **text section**. It also includes the current activity, as represented by the value of the **program counter** and the contents of the processor's registers. A process generally also includes the process **stack**, which contains temporary data (such as function parameters, return addresses, and local variables), and a **data section**, which contains global variables. A process may also include a **heap**, which is memory that is dynamically allocated during process run time.

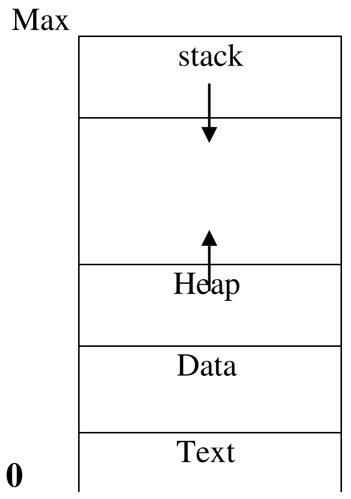


Figure: Process in memory

A program is a passive entity, such as a file containing a list of instructions stored on disk (often called an executable file), whereas *a process is an active entity*, with a program counter specifying the next instruction to execute and a set of associated resources. A program becomes a process when an executable file is loaded into memory.

NOTE: Two processes may be associated with the same program, but considered as two separate execution sequences. For instance, several users may invoke many copies of the Web browser program. Each of these is a separate process; and although the text sections are equivalent, the data, heap, and stacksections vary.

Various steps involved in the change of a process state-

Five State Process Model: Each 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/0 completion or reception of a signal).
- ***Ready:*** The process is waiting to be assigned to a processor.
- ***Terminated:*** The process has finished execution.

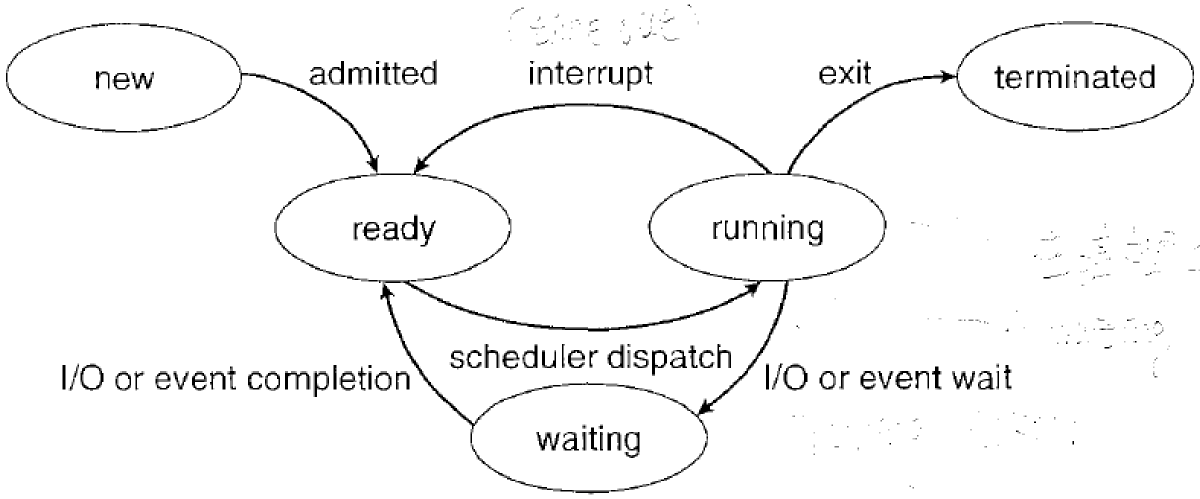


Figure: Five State Process Model Transition Diagram

Five State Model Transitions:

- *Null – New*: A new process is created due to any of four reasons; New Batch Job, Interactive Login, to provide service and spawning.
- *New – Ready*: Operating system moves a process from the new state to the Ready state when it is prepared to take on an additional process.
- *Ready – Running*: Any process can be moved from ready to running state whenever it is scheduled. This is the job of the scheduler or dispatcher.
- *Running – Exit*: The currently running process is terminated by the OS if the process indicates that it has completed, or if it aborts.
- *Running – Ready*: The most commonly known situation is that currently running process has taken its share of time for execution (Time Out). Also in some events a process may have to be admitted from running to ready if a high priority process has occurred.
- *Running – Blocked*: A process is moved to the blocked state if it requested something (data) for which it may have to wait.
- *Blocked – Ready*: A process in the blocked state is moved to the ready state when the event for which it has been waiting occurs.

- *Ready – Exit*: This is the case for example a parent process has generated a single or multiple children processes and they are in the ready state. Now during the execution of the process it may terminate any child process, therefore it will directly go to exit state.
- *Blocked – Exit*: Similarly as above, during the execution of a parent process any child process waiting for an event occur may directly go to exit if the parent itself terminates.

b) **Differentiate between the followings:**

i. User-level thread and Kernel-level thread

ii. Asynchronous and Deferred Cancellation

Sol: (i) Difference between User-level thread and Kernel-level thread-

Sr. No.	**User level threads**	**Kernel level threads**
1.	User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
2.	User threads are supported above thekernel and are managed without kernel support.	kernel threads are supported and managed directly by the operating system.
3.	User level threads can run on any operating system.	Kernel level threads are specific to the operating system.
4.	Multithreaded applications cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.

(ii). Difference between Asynchronous and Deferred Cancellation-

Cancellation of a target thread may occur in two different scenarios:
Asynchronous cancellation *- One thread immediately terminates the target thread.* In asynchronous cancellation, the difficulty with cancellation occurs in situations where resources have been allocated to a canceled thread or a thread is canceled while in the midst of updating data it is sharing with other threads.

Deferred cancellation *- The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an orderly fashion.* With deferred cancellation, cancellation occurs only after the target thread haschecked a *flag* to determine whether or not it should be canceled.

(c) Define Process Control Block (PCB). Draw and explain the Process state transition diagram.

Solution- **Process Control Block-** Each process is represented in the operating system by a process control block (PCB) – alsocalled a *task control block*.

Pointer
Process state
Process number
Program counter
registers
Memory limit
List of open files
• • • •

Figure: Process Control Block (PCB)

It contains following information associated with a specific process:

- **Process state**: The state may be new, ready, running, waiting, terminated, and so on.
- **Pointer**: Each PCB includes a pointer field that points to the next PCB in the ready queue.
- **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 interruptoccurs, to allow the process to be continued correctly afterward.

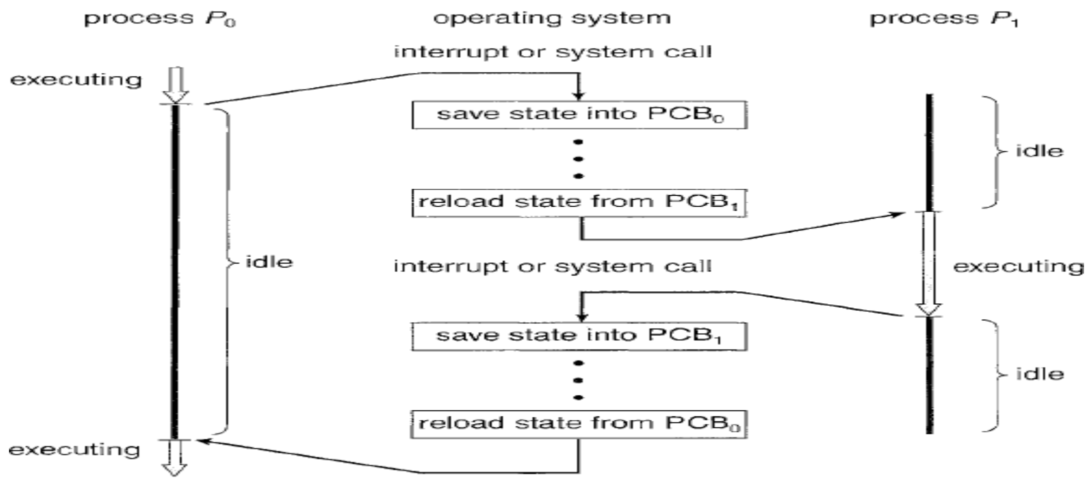


Fig.: Diagram showing CPU switch from process to process

- **CPU-scheduling information**: This information includes a process priority, and any other scheduling parameters.
- **Memory-management information**: This include the information of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
- **Accounting information**: This information includes the amount of CPU used, time limits, job or process numbers, and so on.
- **I/O status information**: This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

In brief, the PCB simply serves as the repository for any information that may vary from process to process.

Process Transition Diagram-

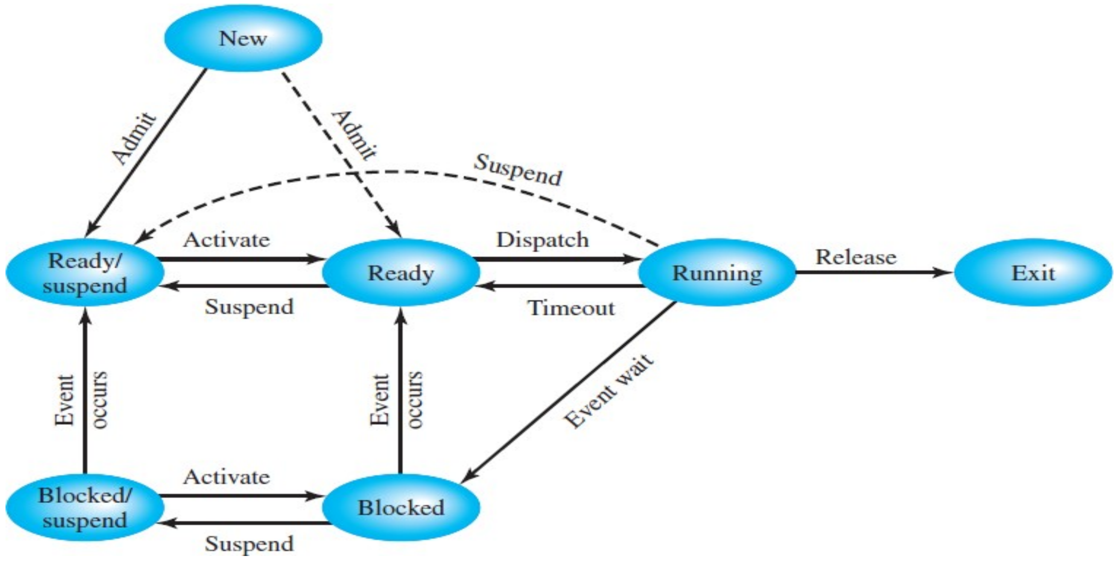


Fig.: Process State Transition Diagram

Each 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 in main memory and awaiting an event.
- ***Ready:*** The process is in main memory and available for execution.
- ***Terminated/Exit:*** The process has finished execution.
- ***Blocked/Suspend***: The process is in secondary memory and awaiting an event.
- ***Ready/Suspend***: The process is in secondary memory but is available for execution as soon as it is loaded into main memory.

<u>Suspend Process Transitions:</u>

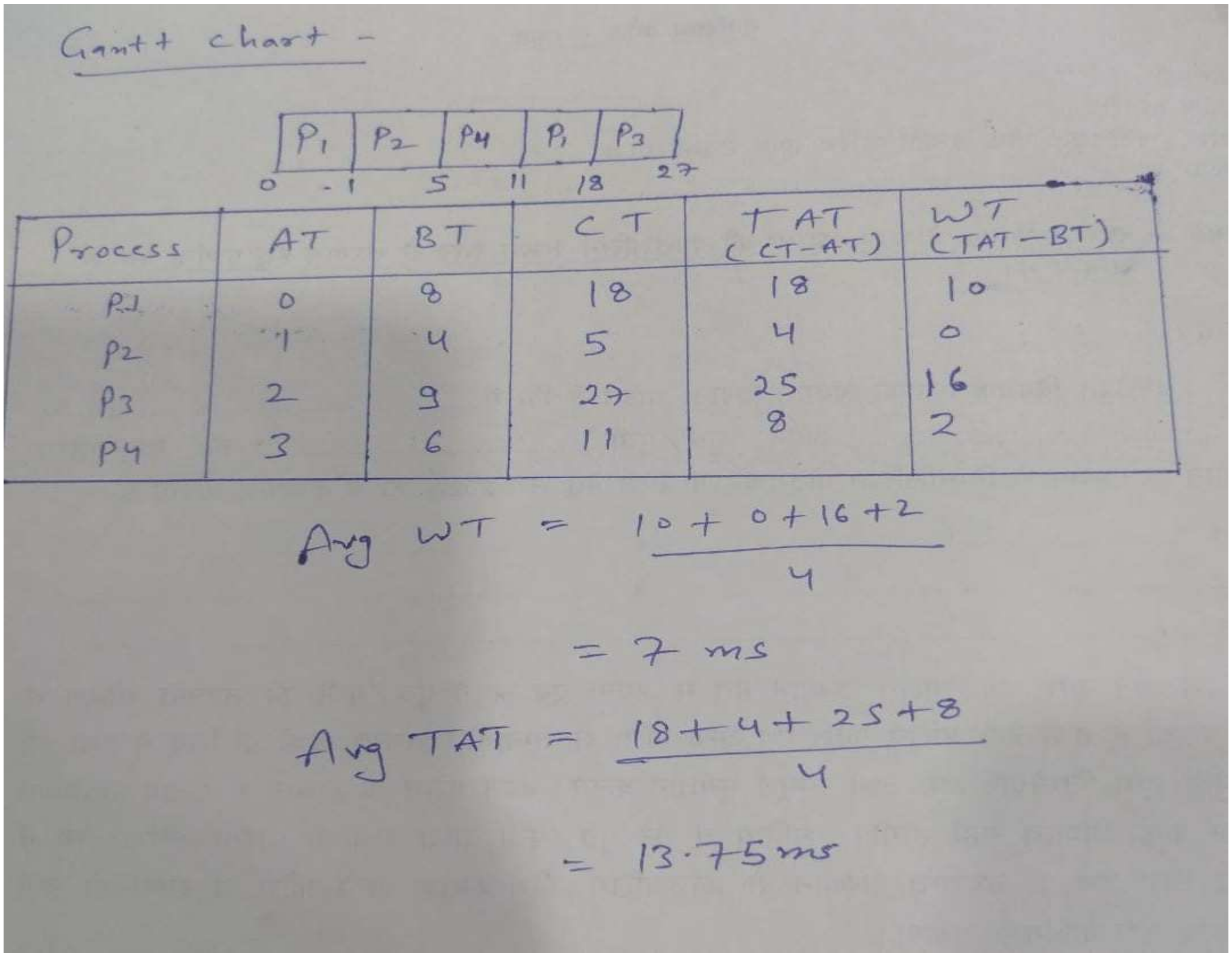
- *Blocked - Blocked/Suspend*: If there are no ready processes, then at least one blocked process is swapped out to make room for another process that is not blocked.
- *Blocked/Suspend - Ready/Suspend*: A process in the Blocked/Suspend state is moved to the Ready/Suspend state when the event for which it has been waiting occurs.
- *Ready/Suspend - Ready*: When there are no ready processes in main memory, the OS will need to bring one in to continue execution. In addition, it might be the case that a process in the Ready/Suspend state has higher priority than any of the processes in the Ready state.
- *Ready - Ready/Suspend*: it may be necessary to suspend a ready process if that is the only way to free up a sufficiently large block of main memory. Also, the OS may choose to suspend a lower-priority ready process rather than a higher priority blocked process if it believes that the blocked process will be ready soon.
- *New - Ready/Suspend and New - Ready*: When a new process is created, it can either be added to the Ready queue or the Ready/Suspend queue.
- *Blocked/Suspend - Blocked*: Consider a situation that, a process terminates, freeing up some main memory. There is a process in the (Blocked/Suspend) queue with a higher priority than any of the processes in the (Ready/Suspend) queue and the OS has reason to believe that the blocking event for that process will occur soon. Under these circumstances, it would seem reasonable to bring a blocked process into main memory in preference to a ready process.
- *Running - Ready/Suspend*: Normally, a running process is moved to the Ready state when its time allocation expires. If, however, the OS is preempting the process because a higher-priority process on the Blocked/Suspend queue has just become unblocked, the OS could move the running process directly to the (Ready/Suspend) queue and free some main memory.
- *Any State - Exit*: Typically, a process terminates while it is running, either because it has completed or because of some fatal fault condition. However, in some operating systems, a process may be terminated by the process that created it or when the parent process is itself terminated.

(d) Consider the following processes:

Processes	Arrival Time (milli seconds)	Burst Time (milli seconds)
P1	0	8
P2	1	4
P3	2	9
P4	3	6

Draw a Gantt chart and calculate the average waiting time and turnaround time for these processes with **Preemptive SJF Scheduling**.

Sol-



Gantt chart -

P1	P2	P4	P1	P3

0 1 5 11 18 27

Process	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)
P1	0	8	18	18	10
P2	1	4	5	4	0
P3	2	9	27	25	16
P4	3	6	11	8	2

$$\text{Avg WT} = \frac{10 + 0 + 16 + 2}{4}$$

$$= 7 \text{ ms}$$

$$\text{Avg TAT} = \frac{18 + 4 + 25 + 8}{4}$$

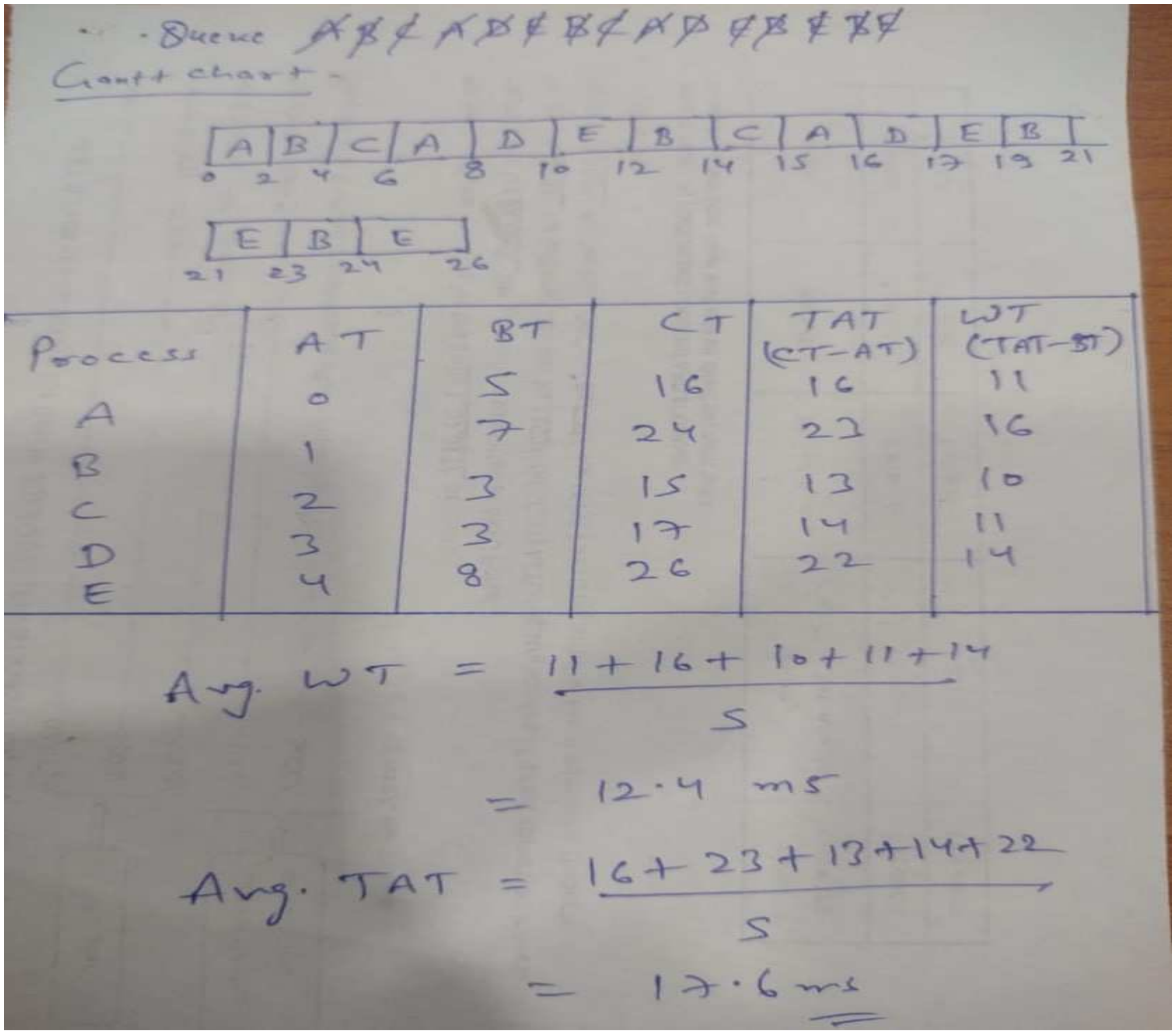
$$= 13.75 \text{ ms}$$

(e) Consider the following processes:

Processes	Arrival Time (milli seconds)	Burst Time (milli seconds)
A	0	5
B	1	7
C	2	3
D	3	3
E	4	8

Draw a Gantt chart and calculate the average waiting time and turnaround time for these processes with Round-Robin scheduling (Time Quantum/Time Slice = 2 milli seconds).

Sol-



∴ Queue ~~A~~ ~~B~~ ~~C~~ ~~A~~ ~~D~~ ~~E~~ ~~B~~ ~~C~~ ~~A~~ ~~D~~ ~~E~~ ~~B~~ ~~E~~ ~~B~~ ~~E~~

Gantt chart -

A	B	C	A	D	E	B	C	A	D	E	B

0 2 4 6 8 10 12 14 15 16 17 19 21

E	B	E

21 23 24 26

Process	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)
A	0	5	16	16	11
B	1	7	24	23	16
C	2	3	15	13	10
D	3	3	17	14	11
E	4	8	26	22	14

$$\text{Avg. WT} = \frac{11+16+10+11+14}{5}$$

$$= 12.4 \text{ ms}$$

$$\text{Avg. TAT} = \frac{16+23+13+14+22}{5}$$

$$= 17.6 \text{ ms}$$