IARE '24 DISCORD TEAM

II-II

OS MODULE 2 SOLUTIONS

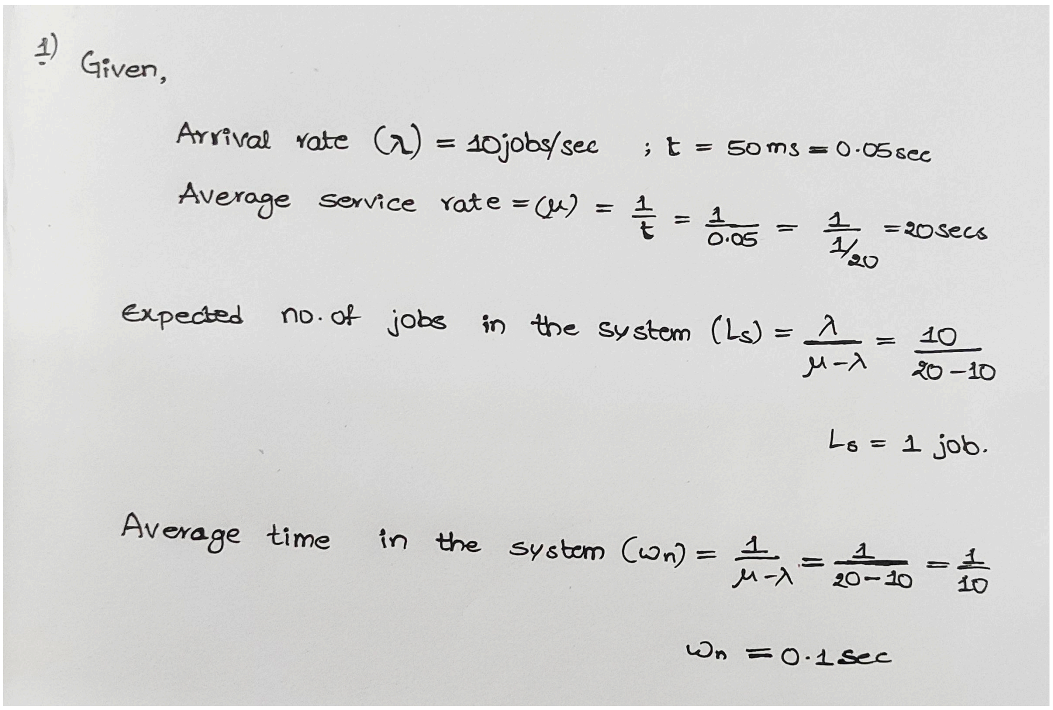
SUHRUTH • ANUSHKA • IKRAM • ABHIRAMI • UJJWAL

PROCESS AND CPU SCHEDULING,
PROCESS COORDINATION

MODULE - 2

PART - A

1. Suppose we have a single processor system and jobs arrive at a rate of 10 jobs/sec, suppose each job takes an average of 50 milliseconds to complete. Assure that both distributions are exponential. State the expected number of jobs in the system and the average time in the system.



1) Given,

Arrival rate $(\lambda) = 10$jobs/sec ; $t = 50$ ms $= 0.05$ sec

Average service rate $= (\mu) = \frac{1}{t} = \frac{1}{0.05} = \frac{1}{1/20} = 20$secs

Expected no. of jobs in the system $(L_s) = \frac{\lambda}{\mu - \lambda} = \frac{10}{20-10}$

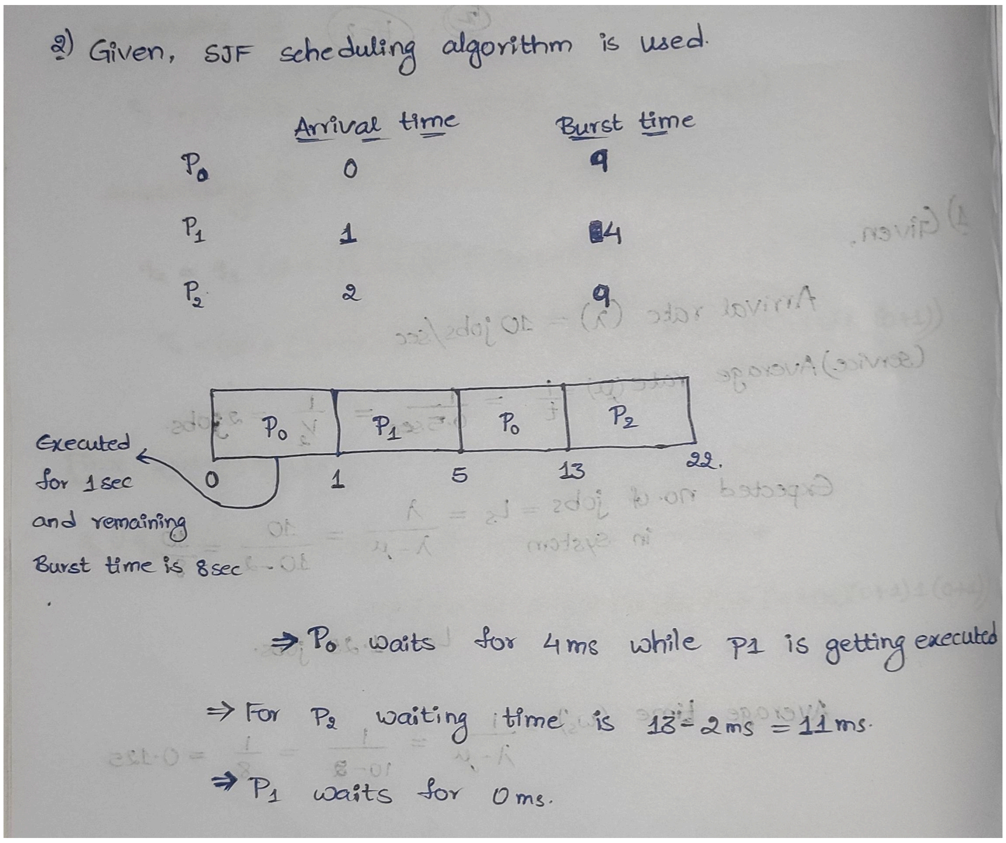
$L_s = 1$ job.

Average time in the system $(W_n) = \frac{1}{\mu - \lambda} = \frac{1}{20-10} = \frac{1}{10}$

$W_n = 0.1$sec

Answer Verified 🆗

2. Consider the following table of arrival time and burst time for three processes P0, P1 and P2. Process Arrival time and Burst Time P0-0ms, 9ms; P1-1ms, 4ms; P2-2ms, 9ms. The preemptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?



2) Given, SJF scheduling algorithm is used.

	Arrival time	Burst time
P_0	0	9
P_1	1	4
P_2	2	9

P_0	P_1	P_0	P_2

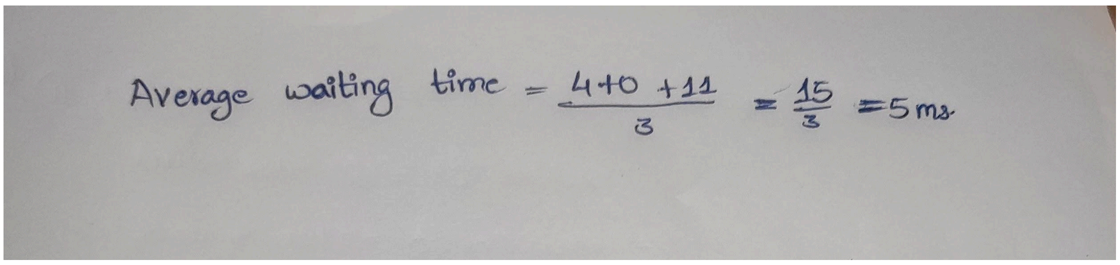
0 1 5 13 22.

Executed for 1 sec and remaining Burst time is 8 sec

$\Rightarrow$ P_0 waits for 4ms while P1 is getting executed

$\Rightarrow$ For P_2 waiting time is $13 - 2$ms $= 11$ms.

$\Rightarrow$ P_1 waits for 0 ms.



Average waiting time $= \frac{4+0+11}{3} = \frac{15}{3} = 5$ms.

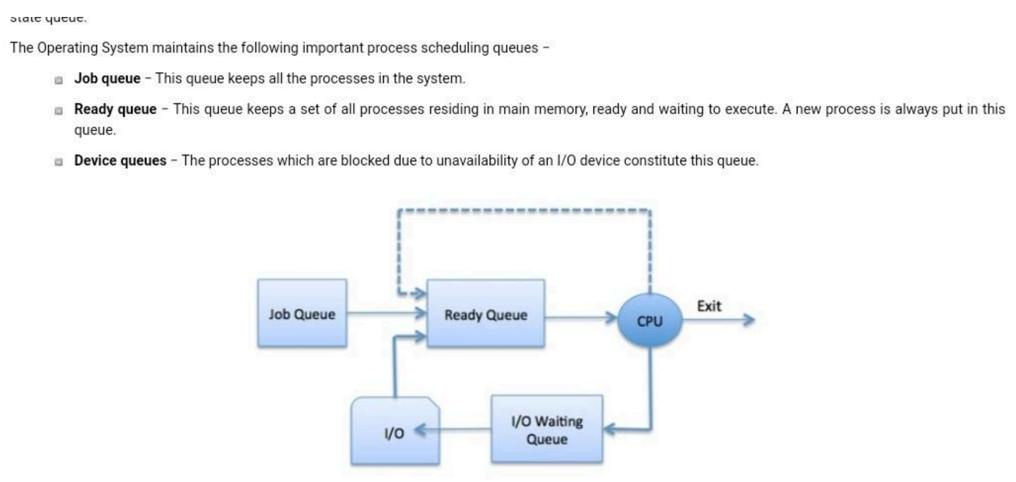
Answer Verified 🆗

3. What are the 3 different types of scheduling queues?

The 3 types of scheduling queues are:

- Job Queue
- Ready Queue
- Device Queue (or) Waiting Queue

i) Job Queue: Starting, all the processes get stored in the job queue. It is maintained in the secondary memory. The long term scheduler(Job scheduler) picks some of the jobs and puts them in the primary memory.
ii) Ready Queue: Is maintained in primary memory. The short term scheduler picks the job from the ready queue and dispatches it to the CPU for the execution.
iii) Waiting Queue: When the process needs I/O operation in order to complete its execution, the OS changes the state of the process from running to waiting. The context (PCB) associated with the process gets stored on the waiting queue which will be used by the processor when the process finishes the I/O.



state queue.

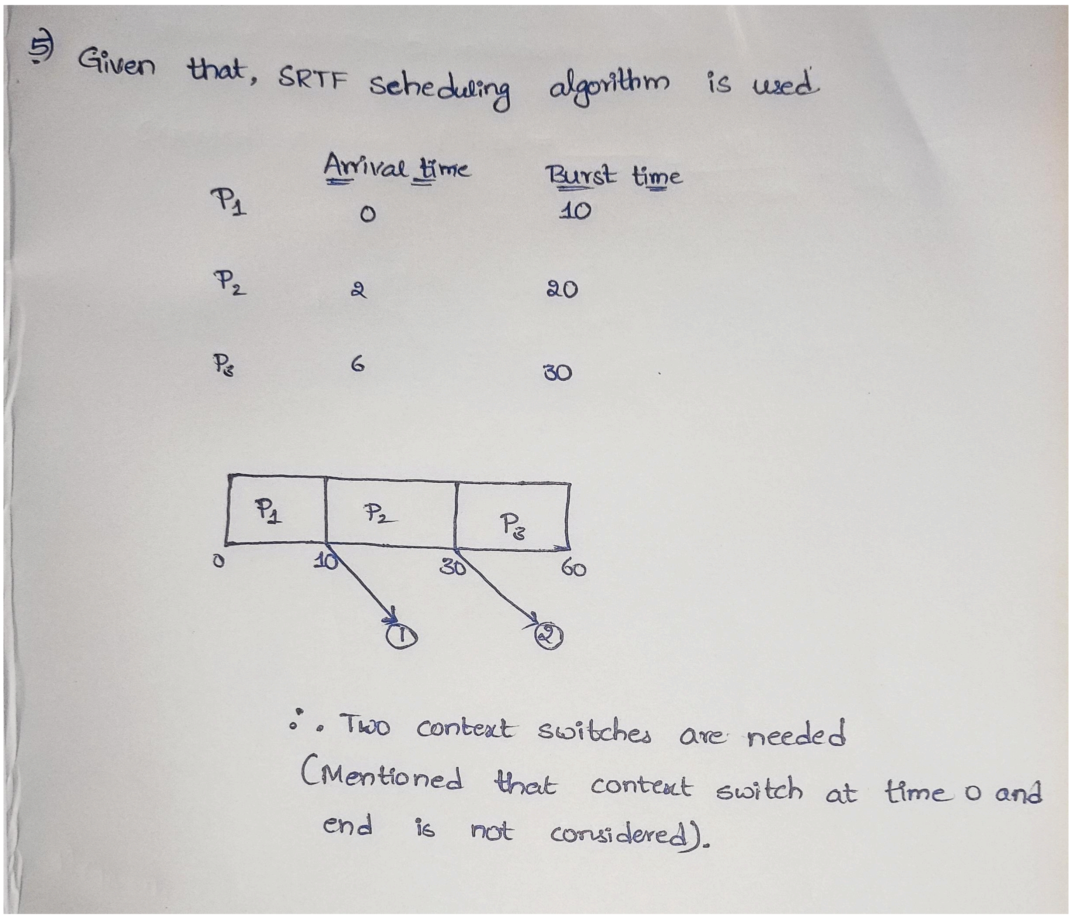
The Operating System maintains the following important process scheduling queues –

- **Job queue** – This queue keeps all the processes in the system.
- **Ready queue** – This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** – The processes which are blocked due to unavailability of an I/O device constitute this queue.

4. Explain the advantage of using semaphores over Test and Set () and Swap() functions. Describe the use of wait() and signal() functions on semaphore and how these can provide the solution to the critical section problem.

5. Consider three CPU - intensive processes which require 10, 20 and 30 time units to arrive at times 0, 2 and 6 respectively. How many context switches are needed if the operating system implements a shortest

remaining time first scheduling algorithm. Do not count the context switches at time zero and at the end.



5) Given that, SRTF scheduling algorithm is used.

	Arrival time	Burst time
P_1	0	10
P_2	2	20
P_3	6	30

P_1
P_2
P_3
0
10
30
60
1
2

∴ Two context switches are needed

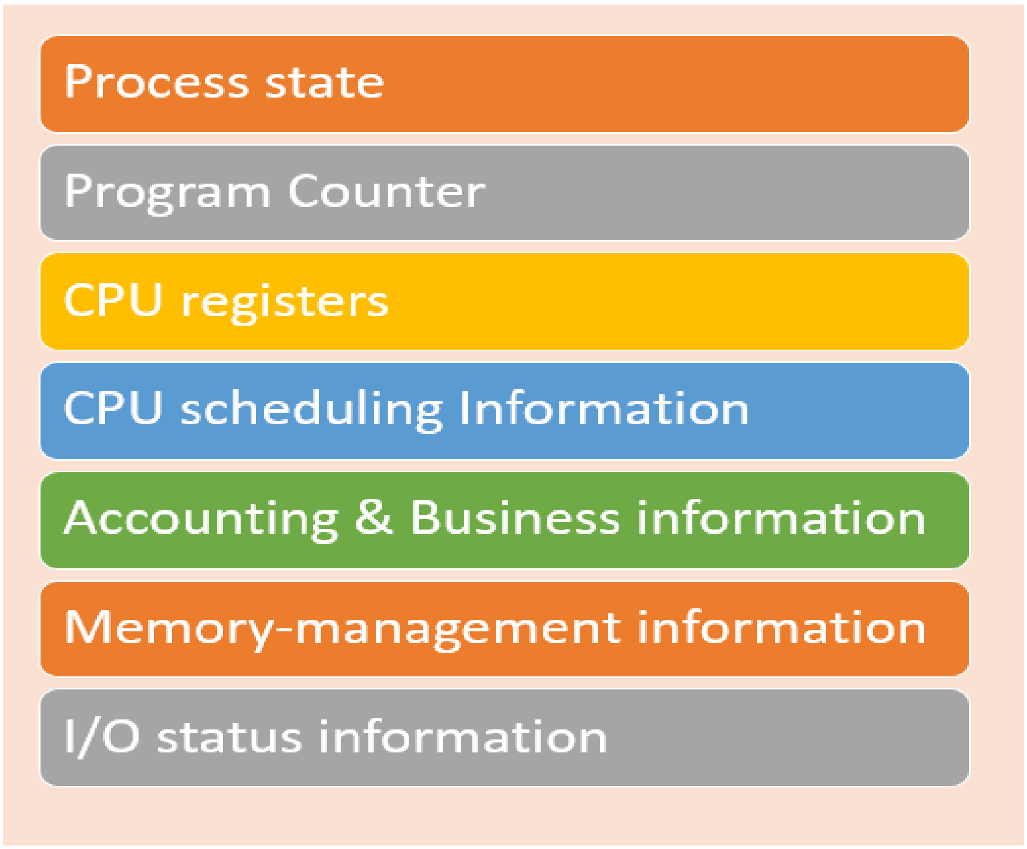
(Mentioned that context switch at time 0 and end is not considered).

6. Construct Process Control Block for any given example.

A process control block (PCB) is a data structure which contains information about the process, i.e. registers, quantum, priority, etc. The process table is an array of PCBs that logically contains a PCB for all of the current processes in the system.

Structure of the Process Control Block

The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given diagram **–**


Process state
Program Counter
CPU registers
CPU scheduling Information
Accounting & Business information
Memory-management information
I/O status information

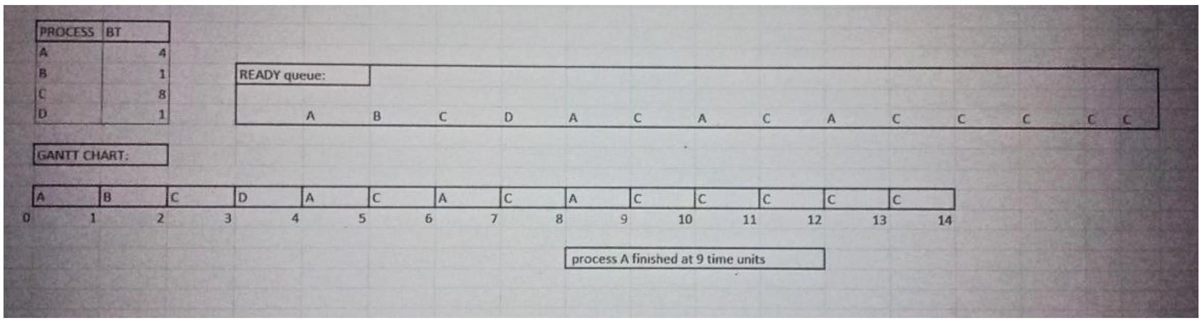
- **Process state:** A process can be new, ready, running, waiting, etc.
- **Program counter:** The program counter lets you know the address of the next instruction, which should be executed for that process.
- **CPU registers:** This component includes accumulators, index and general-purpose registers, and information of condition code.
- **CPU scheduling information:** This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.
- **Accounting and business information:** It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.
- **Memory-management information:** This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.
- **I/O status information:** This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.

Refer- What is Process Control Block (PCB)?

or

Process Table and Process Control Block (PCB) - GeeksforGeeks

7. Explain Four jobs to be executed on a single processor system arrive at time 0 in the order A, B, C, D their burst CPU time requirements are 4, 1, 8, 1 time units respectively. The completion time of A under round robin scheduling with a time slice of one time unit is?



PROCESS	BT
A	4
B	1
C	8
D	1

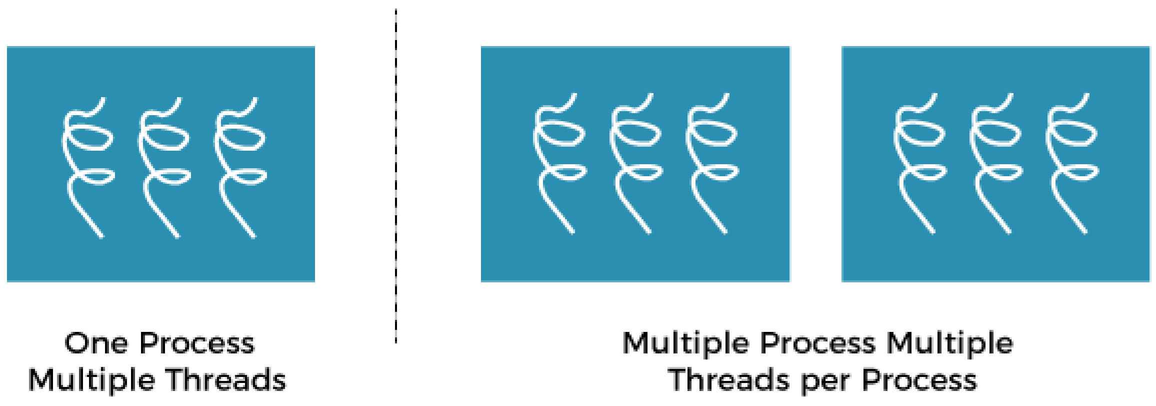
READY queue:
A B C D A C A C A C C C C C
GANTT CHART:
A B C D A C A C A C C C C C
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
process A finished at 9 time units

8. Explain the concept of multi threading. Discuss the following multi threading models.

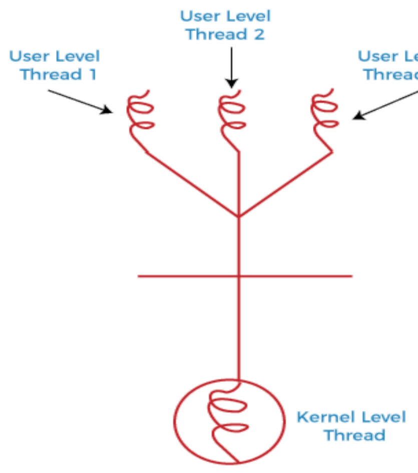
a) **Many-to-one.**
b) **One-to-one.**
c) **Many-to-many.**
d) **Two-level**

Multithreading Model:

Multithreading allows the application to divide its task into individual threads. In multi-threads, the same process or task can be done by the number of threads, or we can say that there is more than one thread to perform the task in multithreading. With the use of multithreading, multitasking can be achieved.



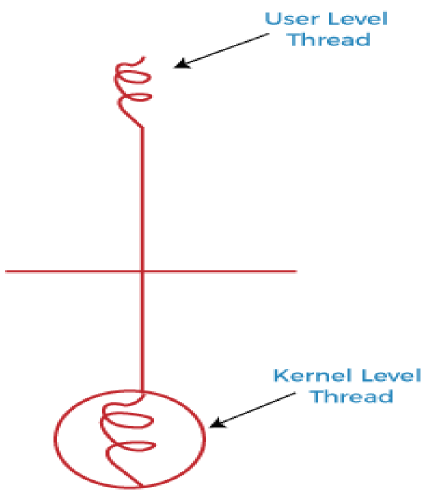
The main drawback of single threading systems is that only one task can be performed at a time, so to overcome the drawback of this single threading, there is multithreading that allows multiple tasks to be performed.



Many-to-one model

One to one multithreading model

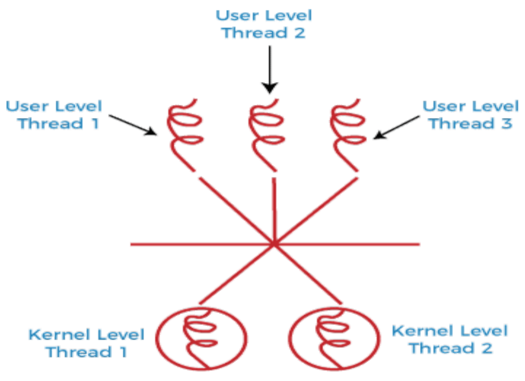
The one-to-one model maps a single user-level thread to a single kernel-level thread. This type of relationship facilitates the running of multiple threads in parallel. However, this benefit comes with its drawback. The generation of every new user thread must include creating a corresponding kernel thread causing an overhead, which can hinder the performance of the parent process



One-to-one model

Many to Many Model multithreading model

In this type of model, there are several user-level threads and several kernel-level threads. The number of kernel threads created depends upon a particular application. The developer can create as many threads at both levels but may not be the same. The many to many model is a compromise between the other two models. In this model, if any thread makes a blocking system call, the kernel can schedule another thread for execution

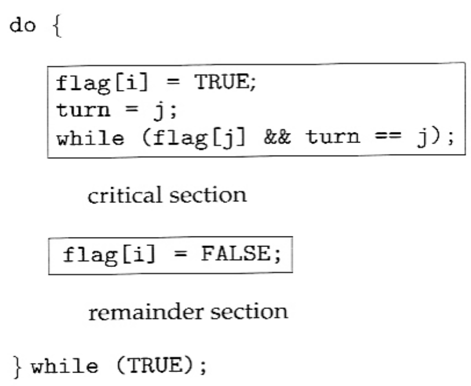


Many-to-Many model

9. Write about Peterson's solution.

Peterson's algorithm (or Peterson's solution) is a concurrent programming algorithm for mutual exclusion that allows two or more processes to share a single-use resource without conflict, using only shared memory for

communication. It was formulated by Gary L. Peterson in 1981. While Peterson's original formulation worked with only two processes, the algorithm can be generalised for more than two.
REFER– Peterson's Problem



```
do {
    flag[i] = TRUE;
    turn = j;
    while (flag[j] && turn == j);

        critical section

    flag[i] = FALSE;

        remainder section

} while (TRUE);
```

10. Distinguish between monitor and semaphore. Explain in detail a monitor with notify and broadcast functions using an example.

Some of the main differences between semaphore and monitor are as follows:

Features	Semaphore	Monitor
Definition	A semaphore is an integer variable that allows many processes in a parallel system to manage access to a common resource like a multitasking OS.	It is a synchronisation process that enables threads to have mutual exclusion and the wait() for a given condition to become true.

Syntax	`// Wait Operation`
	`wait(Semaphore S) {`
	`while (S<=0);`
	`S--;`
	`}`
	`// Signal Operation`
	`signal(Semaphore S) {`
	`S++;`
	`}`
		`monitor {`
		`//shared variable`
		`declarations`
		`data variables;`
		`Procedure P1() { ... }`
		`Procedure P2() { ... }`
		`.`
		`.`
		`.`
		`Procedure Pn() { ... }`
		`}`

Basic	Integer variable	Abstract data type
Access	When a process uses shared resources, it calls the wait() method on S, and when it releases them, it uses the signal() method on S.	When a process uses shared resources in the monitor, it has to access them via procedures.
Action	The semaphore's value shows the number of shared resources available in the system.	The Monitor type includes shared variables as well as a set of procedures that operate on them.
Condition Variable	No condition variables.	It has condition variables.

notify():

The notify() method chooses one thread that is waiting on the monitor held by the current thread and wakes it up. Typically, the waiting thread will grab the monitor and proceed.

Example:

```
public synchronized int get() {
    while (available == false) {
        try {
            wait();
        }
        catch (InterruptedException e) {
        }
    }
    available = false;
    notify();
    return contents;
}
```

broadcast()

The broadcast operation wakes up every thread waiting for a particular resource. This generally makes sense only with sharable resources. Perhaps a writer just completed so all of the readers can be awakened.

Example:

```
void broadcast (condition *c)
  {
    thread_id tid;
    mutex_acquire (c->listLock);
    while (&c->next){
      tid = dequeue(&c->next, &c->prev);
      thr_continue (tid);
    }
   mutex_release (c->listLock); /* done with the queue */
  }
```

PART - B

1. Describe process scheduling. Explain the various levels of scheduling.

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Various levels of process scheduling are:

1) **Long term scheduler**

   It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

2) **Short term scheduler**

   It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

   Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

3) **Medium Term Scheduler**

   Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

   A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion.

2. Explain the process state transition diagram with examples.

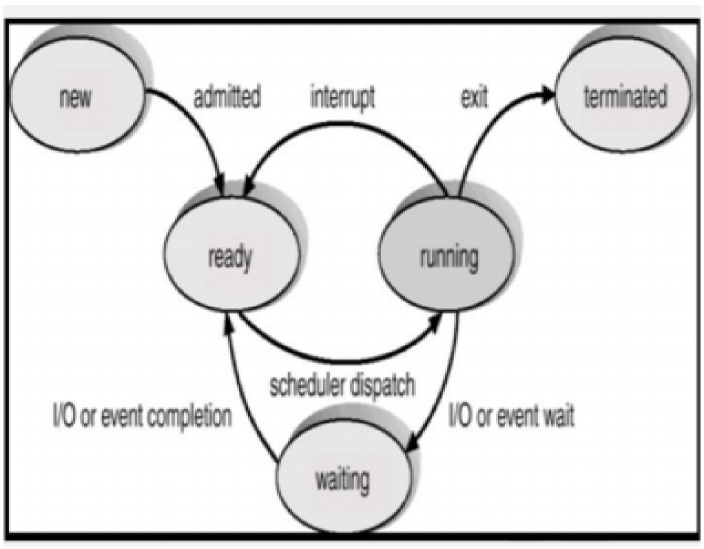


Figure 2.1: Process state diagram

1. Whenever a new process is created, it is admitted into ready state.

2. If no other process is present at running state, it is dispatched to running based on scheduler dispatcher.

3. If any higher priority process is ready, the uncompleted process will be sent to the waiting state from the running state.

4. Whenever I/O or event is completed the process will be sent back to ready state based on the interrupt signal given by the running state.

5. Whenever the execution of a process is completed in running state, it will exit to terminate state, which is the completion of process.

3. Explain handling pruning in detail with examples.

The handle is the substring that matches the body of a production whose reduction represents one step along with the reverse of a Rightmost derivation.
Left sentential and right sentential form:

- A left-sentential form is a sentential form that occurs in the leftmost derivation of some sentence.

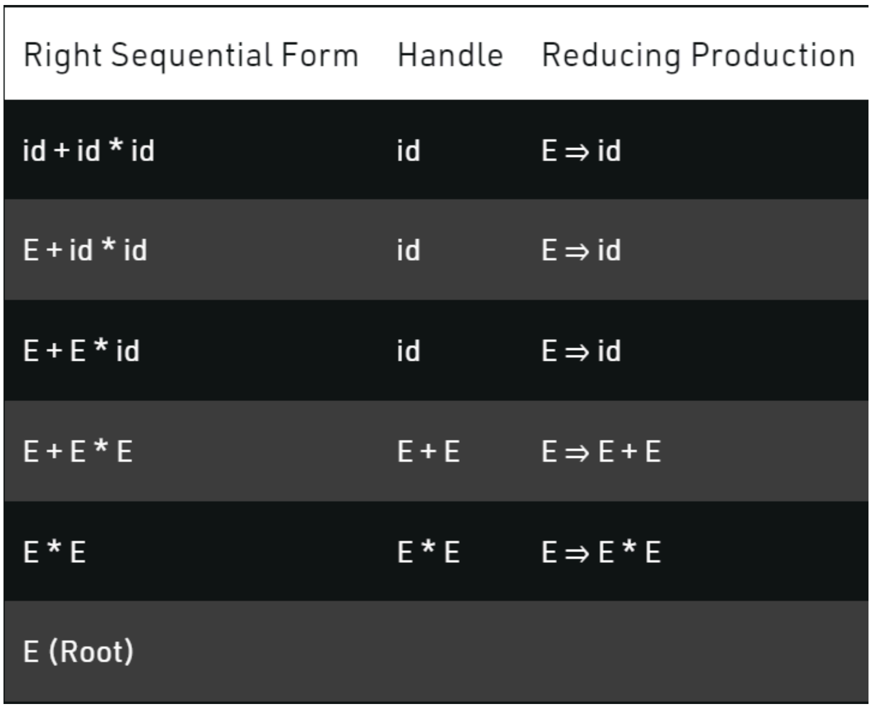
- A right-sentential form is a sentential form that occurs in the rightmost derivation of some sentence.

Handle contains two things:

- Production
- Position

Removing the children of the left-hand side non-terminal from the parse tree is called Handle Pruning. A rightmost derivation in reverse can be obtained by handling pruning.

Example:



Right Sequential Form	Handle	Reducing Production
id + id * id	id	E ⇒ id
E + id * id	id	E ⇒ id
E + E * id	id	E ⇒ id
E + E * E	E + E	E ⇒ E + E
E * E	E * E	E ⇒ E * E
E (Root)		

4. What do you mean by PCB? Where is it used? What are its contents? Explain.

Process Control Block is a data structure that contains information of the process related to it. The process control block is also known as a task control block, entry of the process table, etc.

It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

PCB must be kept in an area of memory protected from normal process access. In some operating systems the PCB is placed at the bottom of the process stack.

- CPU registers Accumulators, index registers, stack pointer, general purpose registers

- CPU scheduling information Process priority, scheduling parameters

- Memory-management information Value of base and limit registers

- Accounting information Amount of CPU used, time limits, process numbers

- I/O status information List of I/O devices allocated to the process, list of open files and so on.

5. Explain direct and indirect communication of messages passing systems.

Direct communication:

With direct communication each process that requires communication must explicitly name the recipient or sender of the communication. The send and receive primitives are

- Send (P,message) - Send a message to process P.
- Receive (Q,message) - Receive a message from process Q.

Indirect communication:

With indirect communication, the messages are sent to and received from mailboxes or ports. A mailbox can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed. Every mailbox has a unique identification. Two processes can communicate only if they share a mailbox. The primitives are

- Send (A, message) - Send a message to mailbox A.
- Receive (A, message) - Receive a message from mailbox A.

6. What is a process? Draw and explain the process state diagram.

A process is a program in execution. The execution of a process progresses in a sequential fashion. A program is a passive entity while a process is an active entity. A process includes much more than just the program code. A process includes the text section, stack, data section, program counter, register contents and so on.

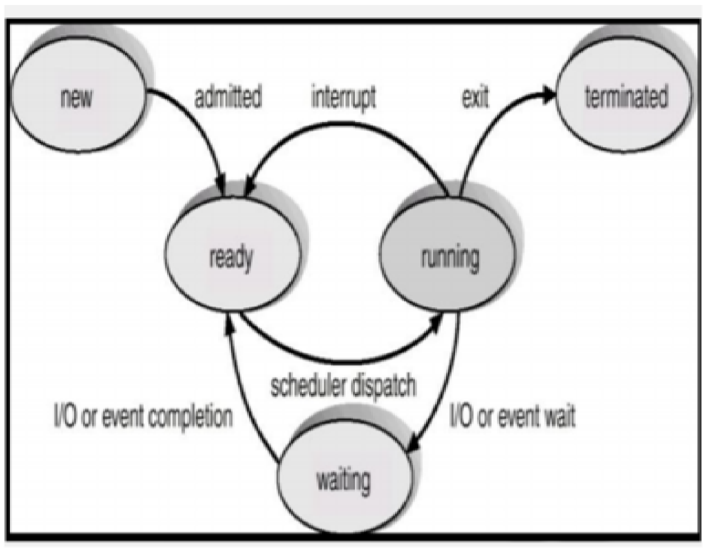


Figure 2.1: Process state diagram

Process state:

As a process executes, it changes state. The state of a process refers to what the process currently does. A process can be in one of the following states during its lifetime:

New: The process is being created.

Running: Instructions are being executed.

Waiting: The process is waiting for some event to occur or waiting for the I/O operation.

Ready: The process is waiting to be assigned to a processor.

Terminated: The process has finished execution.

7. Distinguish between monitor and semaphore.

Monitor:
Monitors are used for process synchronisation. With the help of programming languages, we can use a monitor to achieve mutual exclusion among the processes. In other words, monitors are defined as the construct of programming language, which helps in controlling shared data access. The Monitor is a module or package which encapsulates shared data structure, procedures, and the synchronisation between the concurrent procedure invocations.
Semaphores:
Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronisation. The definitions of wait and signal are as follows –
• Wait :The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.
• Signal: The signal operation increments the value of its argument S.

8. Discuss the attributes of the process. Describe the typical elements of the process control block.

1. Process ID

When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.

2. Program counter

A program counter stores the address of the last instruction of the process on which the process was suspended. The CPU uses this address when the execution of this process is resumed.

3. Process State

The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting. We will discuss them later in detail.

4. Priority

Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.

5. General Purpose Registers

Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

6. List of open files

During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

7. List of open devices

OS also maintains the list of all open devices which are used during the execution of the process.

Components of Process Control Block(PCB):

- Process Privileges–allowed/disallowed access to system resources
- Process State–new, ready, running, waiting, dead
- Process Number (PID)–unique identification number for each process (also known as Process ID)
- Program Counter (PC)–A pointer to the address of the next instruction to be executed for this process.

9. What is the purpose of the system calls and system programs?

Purpose of System Calls:

The interface between a process and an operating system is provided by system calls. In general, system calls are available as assembly language instructions. They are also included in the manuals used by the assembly level programmers. System calls are usually made when a process in user mode requires access to a resource. Then it requests the kernel to provide the resource via a system call.

Purpose of System Programs:

The system program serves as a part of the operating system. It traditionally lies between the user interface and the system calls. The user view of the system is actually defined by system programs and not system calls because that is what they interact with and system programs are closer to the user interface. System programs as well as application programs form a bridge between the user interface and the system calls.

10. List out the various process states and briefly explain the same with a state diagram.

1. New

A program which is going to be picked up by the OS into the main memory is called a new process.

2. Ready

Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary memory and put all of them in the main memory.

3. Running

One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm. Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one.

4. Block or wait

From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behaviour of the process.

5. Completion or termination

When a process finishes its execution, it comes in the termination state. All the context of the process (Process Control Block) will also be deleted and the process will be terminated by the Operating system.

6. Suspend ready

A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspended ready state.

7. Suspend wait

Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory.

State Diagram:

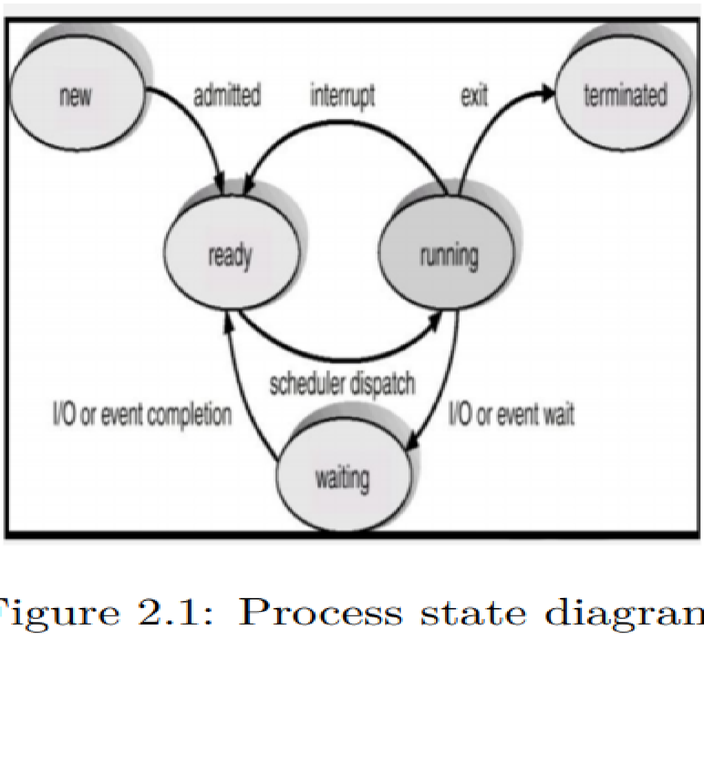


Figure 2.1: Process state diagram

11. What is the purpose of a command interpreter? Why is it usually separate from the kernel?

A command interpreter allows the user to interact with a program using commands in the form of text lines. It reads commands from the user or from a file of commands and executes them, usually by turning them into one or more system calls. The main purpose of it is to understand the command inputs and then turn it into system calls. Users can directly enter the command in the form of text lines. It contains code to execute a command or uses a command to read an external file having a code. It is usually not part of the kernel since the command interpreter is subject to changes. The Unix Shell and Windows command prompt are some examples of command interpreters.

12. What are the differences between user level and kernel supported threads?

User-level threads	Kernel-level threads
User-level threads are managed without kernel support by the run-time system.	Kernel-level threads are supported and managed by the operating system.
The scheduler cannot schedule the process properly as the kernel is unaware of user-level threads.	The scheduler handles the process better as the kernel is fully aware of kernel-level threads.
User-level threads are faster to create.	Kernel-level threads are slower to create.
User-level threads are more efficient.	Kernel-level threads are not so efficient.
User-level threads are easy to manage.	Kernel-level threads are not as easy to manage as user-level threads.
The context switching time is less.	The context switching time is more.
User-level threads are generic and can run on any Operating System.	Kernel-level threads are specific to the Operating System.
When one user-level thread performs a blocking operation, the entire process gets blocked.	The kernel can still schedule another thread for execution if one thread is blocked.
User-level threads cannot take full advantage of multiprocessing.	Kernel-level threads take full advantage of multiprocessing.
User-level threads are designed as dependent threads.	Kernel-level threads are designed as independent threads.
Hardware support is needed for context switches.	No hardware support is needed.

13. What is a Scheduler? What is a dispatcher?

Scheduler:
Schedulers in Operating Systems are the process which decides which task and process should be accessed and run at what time by the system resources. It is required to maintain the multi-tasking capabilities of a computer and to keep its performance at the highest level by scheduling the process according to their preferences and needs. The Schedulers in Operating System

are the algorithms which help in the system optimisation for maximum performance.

There are three types of schedulers, they are:

1. Long term scheduler
2. Short term scheduler
3. Middle term scheduler

Dispatcher:
A dispatcher is a special program which comes into play after the scheduler. When the scheduler completes its job of selecting a process, it is the dispatcher which takes that process to the desired state/queue. The dispatcher is the module that gives a process control over the CPU after it has been 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

14. Discuss the following CPU scheduling algorithms

a) Round Robin

b) Shortest job

Round-Robin Scheduling:
Round robin is the oldest, simplest scheduling algorithm. The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turn. It is mostly used for scheduling algorithms in

multitasking. This algorithm method helps for starvation free execution of processes.

Shortest Job First:
SJF is a full form of (Shortest job first) is a scheduling algorithm in which the process with the shortest execution time should be selected for execution next. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution.

15. Discuss the following CPU scheduling algorithms

a) **First come First serve**
b) **Priority**

First Come First Serve:

First Come First Serve is the full form of FCFS. It is the easiest and most simple CPU scheduling algorithm. In this type of algorithm, the process which requests the CPU gets the CPU allocation first. This scheduling method can be managed with a FIFO queue.

As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue. So, when the CPU becomes free, it should be assigned to the process at the beginning of the queue.

Priority Based Scheduling:

Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler selects the tasks to work as per the priority.
Priority scheduling also helps OS to involve priority assignments. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority can be decided based on memory requirements, time requirements, etc.

16. Discuss the following.

- **a) CPU-i/O burst cycle**
- **b) CPU schedule**
- **c) Preemptive and Nonpreemptive scheduling**
- **d) Dispatcher**

a) CPU i/o Burst Cycle

Process execution consists of a cycle of CPU execution and I/O wait. The state of process under execution is called CPU burst and the state of process under I/O request & its handling is called I/O burst.Processes alternate between these two states. Process execution begins with a CPU burst. That is followed by an I/O burst and repeats so on.

b) CPU Schedule

CPU Scheduling is a process of determining which process will own a CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least selects one of the processes available in the ready queue for execution. The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.

c) Preemptive and Non Preemptive Scheduling

Preemptive Scheduling is a CPU scheduling technique that works by dividing time slots of CPU to a given process. The time slot given might be able to complete the whole process or might not be able to. When the burst time of the process is greater than CPU cycle, it is placed back into the ready queue and will execute in the next chance. This scheduling is used when the process switches to ready state.

Non-preemptive Scheduling is a CPU scheduling technique where the process takes the resource (CPU time) and holds it till the process gets terminated or is pushed to the waiting state. No process is interrupted until it is completed, and after that processor switches to another process.

d) Dispatcher

A dispatcher is a special program which comes into play after the scheduler. When the scheduler completes its job of selecting a process, it is the dispatcher which takes that process to the desired state/queue. The dispatcher is the module that gives a process control over the CPU after it has been selected by the short-term scheduler.

17. Compare between Job-scheduling and CPU-scheduling.

Job Scheduling	CPU Scheduling
The job scheduling is the mechanism to select which process has to be brought into the ready queue.	The CPU scheduling is the mechanism to select which process has to be executed next and allocates the CPU to that process.
The job scheduling is also known as long-term scheduling.	The CPU scheduling is also known as short-term scheduling
The job scheduling is done by the long-term scheduler or the job scheduler.	The CPU scheduling is done by the short-term scheduler or the CPU scheduler
The process transfers from new state to ready state in job scheduling.	The process transfers from ready state to running state in CPU scheduling.
More control over multiprogramming in Job Scheduling.	Less control over multiprogramming in CPU Scheduling.
It regulates the programs which are selected to system for processing.	It ensures which program is suitable or important for processing.

Speed is less than the short-term scheduler.	Speed is very fast as compared to a long-term scheduler.

18. Discuss the attributes of the process. Describe the typical elements of the process control block.

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them. This is also called context of the process. Attributes which are stored in the PCB are described below:

1. Process ID

   When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.

2. Program counter

   A program counter stores the address of the last instruction of the process on which the process was suspended. The CPU uses this address when the execution of this process is resumed.

3. Process State

   The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting. We will discuss them later in detail.

4. Priority

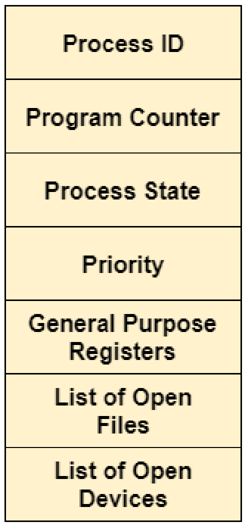
   Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.

5. General Purpose Registers

Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

6. List of open files
   During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

7. List of open devices
   OS also maintains the list of all open devices which are used during the execution of the process.



Process Attributes

19. Define semaphore. Explain the method of applications of semaphore for process synchronisation.

Semaphore:
Semaphore is simply an integer variable that is shared between threads. This variable is used to solve the critical section problem and to achieve process synchronisation in the multiprocessing environment.

Semaphores are of two types:

1. Binary Semaphore –

   This is also known as mutex lock. It can have only two values – 0 and 1. Its value is initialised to 1. It is used to implement the solution of critical section problems with multiple processes.

2. Counting Semaphore –

   Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

20. Discuss about the following?

- **a) Process**
- **b) Components of process**
- **c) Program versus process**
- **d) Process states**

a) Process

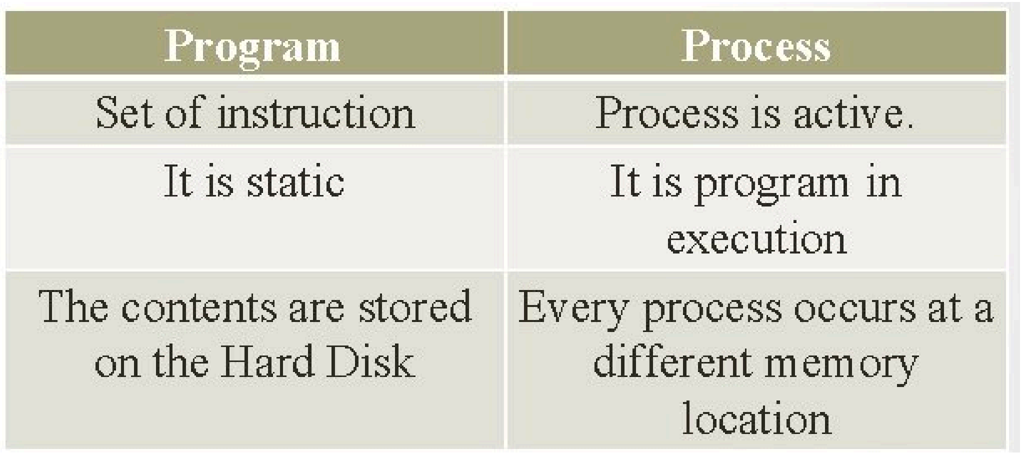
A process is defined as an entity which represents the basic unit of work to be implemented in the system. It is basically a program in execution. The execution of a process must progress in a sequential fashion.

b) Components of Process

S.N.	Component & Description

1	Stack The process Stack contains the temporary data such as method/function parameters, return address and local variables.
2	Heap This is dynamically allocated memory to a process during its run time.
3	Text This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	Data This section contains the global and static variables.

c) Program vs Process



Program	Process
Set of instruction	Process is active.
It is static	It is program in execution
The contents are stored on the Hard Disk	Every process occurs at a different memory location

d) Process States

The process executes when it changes the state. The state of a process is defined by the current activity of the process. Each process may be in any one of the following states –

- New – The process is being created.
- Running – In this state the instructions are being executed.
- Waiting – The process is in waiting state until an event occurs like I/O operation completion or receiving a signal.
- Ready – The process is waiting to be assigned to a processor.
- Terminated – the process has finished execution.

PART - C

1. Define process.

2. Define thread.

3. Describe context switching.

4. What is the information maintained in a PCB?

5. Define the process state and mention the various states of a process.

6. Define CPU scheduling.

7. State critical section problem

8. Distinguish between thread and process.

9. Distinguish between user threads and kernel threads.

10. Define turnaround time.

11. List the various scheduling criteria for CPU scheduling.

12. Describe entry and exit sections of a criteria section.

13. Define semaphores.

14. Explain different ways in which a thread can be cancelled.

15. Write about Scheduling queues.

16. Explain the use of job queues, ready queues and device queues.

17. Explain bounded waiting in critical regions.

18. State the factors on which the performance of the Round Robin CPU scheduling algorithm depends.

19. Distinguish between semaphore and binary semaphore.

20. Distinguish between preemptive and nonpreemptive scheduling techniques.