Unit - II (Process Management)

Process Model

Process Model is the graphical representation of process or workflows.

- PCB (Process Control Block)
  A PCB is data structure used by the operating system to manage information about a process.

→ Attributes of Process Model

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.

2. Program Counter
   It stores the address of the last instruction of the process on which the process was suspended.

3. Process State
   The Process, from its creation to the completion.

4. Priority
   Every Process has its own priority.

Unit - II (Process Management)

Process Model

Process Model is the graphical representation of process or workflows

- PCB (Process Control Block)

A PCB is data structure used by the operating system to manage information about a process

→ Attributes of Process Model

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

② Program Counter
It stores the address of the last instruction of the process on which the process was suspended

③ Process State
The Process, from its creation to the completion

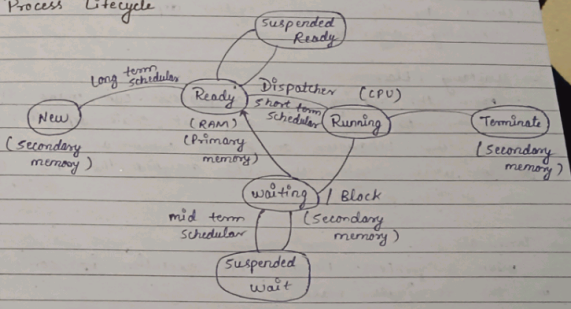
④ Priority
Every Process has its own priority.

⑤ General Purpose Registers
Every process has its own set of register which are used to hold the data.

⑥ List of Open files
During the execution, Every process uses some files which need to be present in main memory

⑦ List of Open devices
Operating system also maintain the list of open devices which are used during the execution of process

→ Process Lifecycle



→ New

A program which is going to be picked by the Operating system into the main memory

→ Ready

whenever a process is created it directly enters in the ready state in which it waits for the CPU to be assigned. Operating System pick the new process from the secondary memory and put all of them in the main memory

→ Running

One of the Process from the Ready state will be choosen by the operating system depending on the scheduling algorithm

→ Wating / Block

from the running state a process can make the transition to the block or wait state depending upon the behaviour of process when a process it in wait state the o/s assigned the CPU to the another person

→ Terminate

when a process finishes its execution it comes in the terminate state

• Suspended Ready

A process in the ready state which is move to secondary memory from the main memory due to the lack of resources is called suspended ready state

• Suspended wait

A process in the block state which is move to the secondary memory from the main memory due to lack of resources is called suspended wait

Process Scheduling

• Operating System uses various schedular for process scheduling

① Long term Scheduler

Long term scheduler is also known as Job Scheduler it chooses the process from the secondary memory and keeps them in the Ready Quee maintained in the main memory

② Short term Scheduler

Short term scheduler is also known as CPU Scheduler it select one of the process from the ready quee and dispatch to the CPU for the execution

③ Mid term Scheduler

Mid term Scheduler removes the process from the running state and make space for the other process

⑪ CPU Scheduling

CPU Scheduling is a process that allow one process to use the CPU while another process is busy in Input / Output operation. The purpose of CPU Scheduling is to make the system more efficient and faster. When CPU become ideal the Operating System must select one of the process from the ready quee.

→ CPU Scheduling Algorithm

Term :-

① Arrival Time
Time at which the process arrive in ready quee.

② Completion Time
Time at which process completes its execution.

③ Burst Time
Time required by a process for CPU execution.

④ Turn Around Time
Time difference between completion time and arrival time.

Completion time - Arrival time.

⑤ Waiting Time
Time difference between turn Around time and Burst time

Turn Around Time - Burst Time

→ CPU Scheduling Algorithm types

• There are mainly two types of scheduling methods

① Preemptive Scheduling
Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
• Round Robin

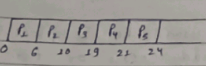
② Non - Preemptive Scheduling
Non - preemptive scheduling is used when a process terminates or when a process switches from running state to waiting state.
• FCFS • SJF

* FCFS (first come first serve)
first come first serve scheduling algorithm states that the process that request the CPU first is allocated the CPU first. It is the simplest of all operating system scheduling algorithm.

Ques

Process ID	Arrival Time	Burst Time
P_1	0	6
P_2	0	4
P_3	0	9
P_4	0	2
P_5	0	3

Gantt chart :-



Turn Around Time :-

C.T − A.T

$P_1 = 6 - 0 = 6$

$P_2 = 10 - 0 = 10$

$P_3 = 19 - 0 = 19$

$P_4 = 21 - 0 = 21$

$P_5 = 24 - 0 = 24$

Avg. TAT $= \frac{6 + 10 + 19 + 21 + 24}{5}$

$= \frac{80}{5}$

$= 16$

Waiting Time :-

T.A.T − B.T

$P_1 = 6 - 6 = 0$

$P_2 = 10 - 4 = 6$

$P_3 = 19 - 9 = 10$

$P_4 = 21 - 2 = 19$

$P_5 = 24 - 3 = 21$

Avg. WT $= \frac{0 + 6 + 10 + 19 + 21}{5}$

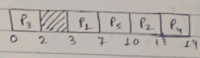
$= \frac{56}{5}$

$= 11.2$

Ques

Process ID	Arrival Time	Burst Time
P_1	3	4
P_2	5	3
P_3	0	2
P_4	5	1
P_5	4	3

Gantt chart :-



Turn Around Time :-

C.T − A.T

$P_1 = 7 - 3 = 4$

$P_2 = 13 - 5 = 8$

$P_3 = 2 - 0 = 2$

$P_4 = 14 - 5 = 9$

$P_5 = 10 - 4 = 6$

Avg. TAT $= \frac{4 + 8 + 2 + 9 + 6}{5}$

$= \frac{29}{5}$

$= 5.8$

Waiting Time

TAT − BT

$P_1 = 4 - 4 = 0$

$P_2 = 8 - 3 = 5$

$P_3 = 2 - 2 = 0$

$P_4 = 9 - 1 = 8$

$P_5 = 6 - 3 = 3$

Avg. WT $= \frac{0 + 5 + 0 + 8 + 3}{5}$

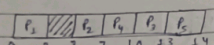
$= \frac{16}{5}$

$= 3.2$

Ques

Process ID	Arrival Time	Burst Time
P_2	3	4
P_3	5	3
P_1	0	2
P_5	5	1
P_4	4	3

Gantt chart :-



Turn Around Time :-

C.T − A.T

P_1 = 2 − 0 = 2
P_2 = 7 − 3 = 4
P_3 = 13 − 5 = 8
P_4 = 10 − 4 = 6
P_5 = 14 − 5 = 9

Avg. TAT = $\frac{2+4+8+6+9}{5}$

= $\frac{29}{5}$

= 5.8

Waiting Time :-

TAT − B.T

P_1 = 2 − 2 = 0
P_2 = 4 − 4 = 0
P_3 = 8 − 3 = 5
P_4 = 6 − 3 = 3
P_5 = 9 − 1 = 8

Avg. WT = $\frac{0+0+5+3+8}{5}$

= $\frac{16}{5}$

= 3.2

→ Convoy Effect

When a higher burst time effect arrives in ready quee before the process with smaller burst time then the smaller process has to wait for long time to get in CPU

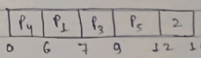
* SJF (Shortest Job First)

- Shortest job first is a scheduling process that selects the waiting process with the smallest execution time to execute next.

Ques

Process ID	Arrival Time	Burst Time
P_1	3	1
P_2	1	4
P_3	4	2
P_4	0	6
P_5	2	3

Gratt Chart :-



Turn Around Time

$P_1 = 7 - 3 = 4$
$P_2 = 16 - 1 = 15$
$P_3 = 9 - 4 = 5$
$P_4 = 6 - 0 = 6$
$P_5 = 12 - 2 = 10$

Waiting Time

$P_1 = 4 - 1 = 3$
$P_2 = 15 - 4 = 11$
$P_3 = 5 - 2 = 3$
$P_4 = 6 - 6 = 0$
$P_5 = 10 - 3 = 7$

Avg TAT $= \frac{4+15+5+6+10}{5}$

$= \frac{40}{5}$

$= 8$

Avg. WT $= \frac{3+11+3+0+7}{5}$

$= \frac{24}{5}$

$= 4.8$

Ques

Process ID	Arrival Time	Burst Time
P_1	3	1
P_2	9	1
P_3	2	2
P_4	0	6
P_5	2	3

Gantt Chart :-

P_4	P_1	P_3	P_2	P_5

0 6 7 9 10 13

Turn Around Time

$P_1 = 7 - 3 = 4$
$P_2 = 10 - 9 = 1$
$P_3 = 9 - 2 = 7$
$P_4 = 6 - 0 = 6$
$P_5 = 13 - 2 = 11$

Waiting Time

$P_1 = 4 - 1 = 3$
$P_2 = 1 - 1 = 0$
$P_3 = 7 - 2 = 5$
$P_4 = 6 - 6 = 0$
$P_5 = 11 - 3 = 8$

Avg. TAT $= \frac{4+1+7+6+11}{5}$

$= \frac{29}{5}$

$= 5.8$

Avg. WT $= \frac{3+0+5+0+8}{5}$

$= \frac{16}{5}$

$= 3.2$

* Round - Robin

Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. Round Robin CPU algorithm generally focused on Time sharing technique.

Ques

Process ID	Arrival Time	Burst Time
P_1	0	5
P_2	1	3
P_3	2	1
P_4	3	2
P_5	4	3

Time Quantum (TQ) = 2ns

Gantt Chart :-

P_1	P_2	P_3	P_4	P_5	P_2	P_1	P_5
0	2	4	5	7	9	11	12	13	14

Turn Around Time :-

$P_1 = 13 - 0 = 13$
$P_2 = 12 - 1 = 11$
$P_3 = 5 - 2 = 3$
$P_4 = 9 - 3 = 6$
$P_5 = 14 - 4 = 10$

Avg TAT $= \frac{13+11+3+6+10}{5}$

$= \frac{43}{5}$

$= 8.6$

Waiting Time :-

$P_1 = 13 - 5 = 8$
$P_2 = 11 - 3 = 8$
$P_3 = 3 - 1 = 2$
$P_4 = 6 - 2 = 4$
$P_5 = 10 - 3 = 7$

Avg. WT $= \frac{8+8+2+4+7}{5}$

$= \frac{29}{5}$

$= 5.8$

Ques

Process ID	Arrival Time	Burst Time
P_1	0	5
P_2	1	7
P_3	2	6
P_4	3	3
P_5	4	2

Time Quantum (TQ)
3ns

Gantt Chart :-

P_2	P_1	P_3	P_4	P_5	P_2	P_3	P_2
0	3	6	9	12	14	16	19	22	23

Turn Around Time :-

$P_1 = 14 - 0 = 14$
$P_2 = 23 - 1 = 22$
$P_3 = 22 - 2 = 20$
$P_4 = 12 - 3 = 9$
$P_5 = 16 - 4 = 12$

Avg. TAT $= \frac{14+22+20+9+12}{5}$

$= \frac{77}{5}$

$= 15.4$

Waiting Time :-

$P_1 = 14 - 5 = 9$
$P_2 = 22 - 7 = 15$
$P_3 = 20 - 6 = 14$
$P_4 = 9 - 3 = 6$
$P_5 = 12 - 2 = 10$

Avg. WT $= \frac{9+15+14+6+10}{5}$

$= \frac{54}{5}$

$= 10.8$

Process Synchronization

Process synchronization is the coordination of execution of multiple process in a multiprocessing system to ensure that they access shared memories or resource in a controlled manner without interfearing with each other to prevent race condition.

- **Race Condition**

  It is a situation when device or system attempt to perform two or more operation at the same time.

On the basis of synchronization there are two categories of process :-

1. **Independent**

   The execution of one process does not effect the other process.

2. **Dependent / Cooperative**

   A process that can affect the execution of other process.

Critical Section Problem

- Critical section is a code segment that can be access one process at a time.
- The critical section contain shared variable that need to be synchronize.
- The solution to the Critical Section problem must require 3 condition.

① Mutual Exclusion

If a process is executing in critical section than no other process is allowed to execute in critical section.

② Progress

If no process is executing in critical section and other process are waiting than they can access critical section and selection cannot be postpond indefinitly.

③ Bounded Wait

A bound wait must exist on the number of times that the other process are allowed to enter their critical section.

Synchronization Hardware

- The process synchronization problem occurs when more than one process tries to access the same resource or variable.

If more than one process tries to update a variable at the same time than the data inconsistency problem can occur. This process synchronization is called Synchronization Hardware.

→ Hardware Solution is as follows :-

- Test and Set

Test and Set algorithm uses a boolean variable 'lock' which is initially initialized to false. This lock variable determine the entry of process inside the critical section of code.

- Swap

Swap function uses two boolean variable lock and key. Both lock and key variable are initially initialized to false.

The swap algorithm uses a temporary variable to set lock to true when a process enters the critical section of the program.

- Unlock and lock

Unlock and lock algorithm uses the test and set method to control the value of lock. All the process are maintained in a ready queue before entering into critical section.

Semaphores

Semaphore are integer variable that are used to solve the critical section problem by using two atomic operation, wait and signal that are used for process synchronization

→ 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.

Syntax :-
```
Wait (S)
{
while (S<=0);
S--;
}
```

→ Signal

The signal operation increment the value of its argument S.

Syntax :-
```
Signal (S)
{
S++;
}
```

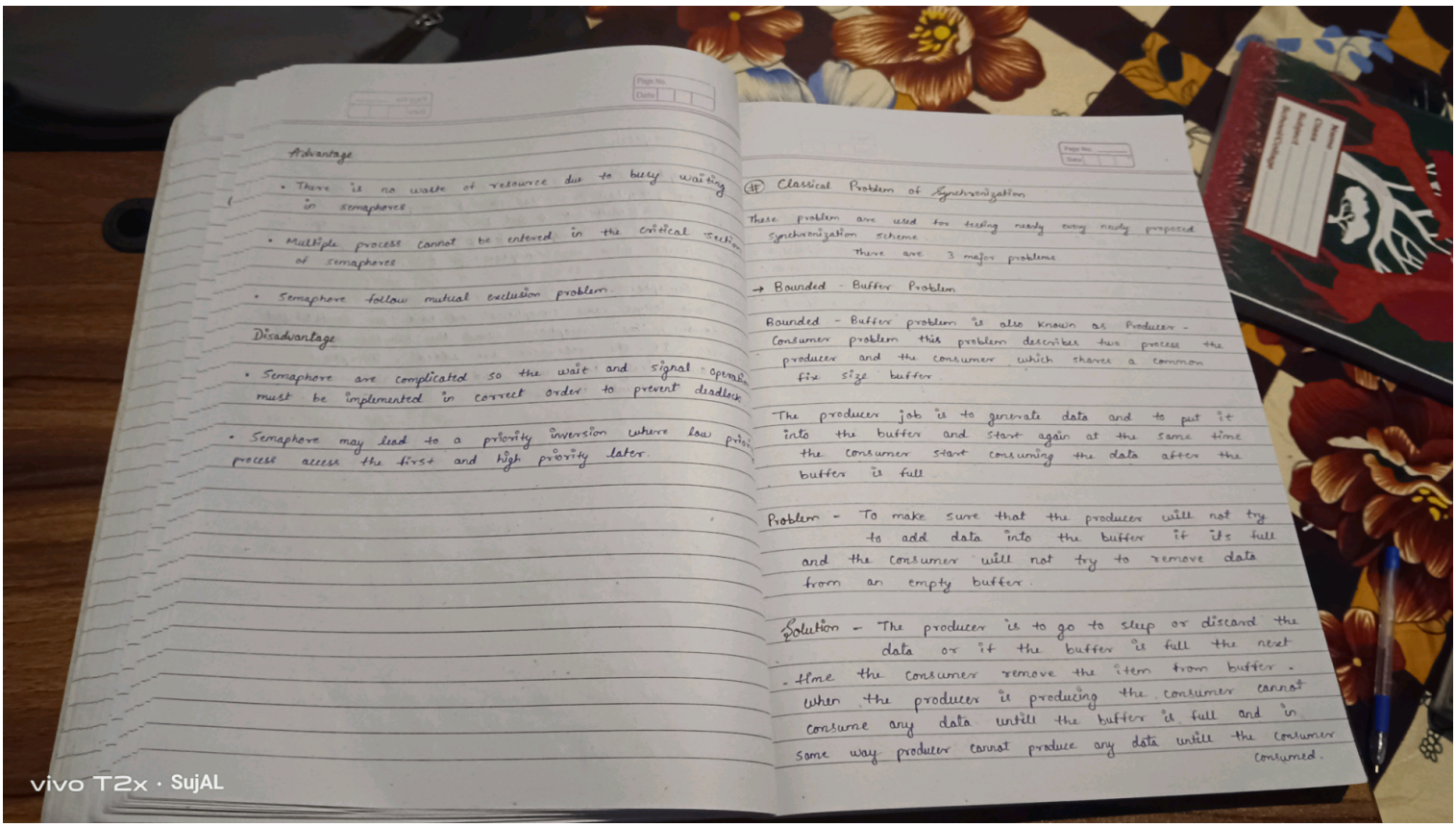
Types of Semaphore

→ Binary Semaphores

The binary semaphore are like counting semaphore but their value is restricted to 0 and 1. The wait wait operation only work when the semaphore is 1 and the signal operation works when the semaphore is 0.

→ Counting Semaphores

These are integer value semaphores and have an unrestricted value domain. The semaphores count the number of available resources. If the resources are added, semaphore count automatically incremented and if the resource are removed, the count is decremented.



Advantage

- There is no waste of resource due to busy waiting in semaphores.
- Multiple process cannot be entered in the critical section of semaphores.
- Semaphore follow mutual exclusion problem.

Disadvantage

- Semaphore are complicated so the wait and signal operation must be implemented in correct order to prevent deadlock.
- Semaphore may lead to a priority inversion where low priority process access the first and high priority later.

Classical Problem of Synchronization

These problem are used for testing nearly every newly proposed synchronization scheme.

There are 3 major problems

→ Bounded - Buffer Problem

Bounded - Buffer problem is also known as Producer - Consumer problem this problem describes two process the producer and the consumer which shares a common fix size buffer.

The producer job is to generate data and to put it into the buffer and start again at the same time the consumer start consuming the data after the buffer is full.

Problem - To make sure that the producer will not try to add data into the buffer if it's full and the consumer will not try to remove data from an empty buffer.

Solution - The producer is to go to sleep or discard the data or if the buffer is full the next time the consumer remove the item from buffer.

When the producer is producing the consumer cannot consume any data until the buffer is full and in some way producer cannot produce any data until the consumer consumed.

→ Dining - Philosphers Problem

The dining philosopher problem state that K no. of philospher seated around a circular table with one chopstick between one pair of chopstick. A philospher may eat if he can pick up the two chopstick near to him. These problem involves the allocation of limited resources to a group of process in a dead lock free and starvation free manner.

→ Reader - Writer Problem

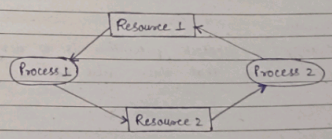
Suppose that a data base to be shared among several concurrent process some of this process may want only to read database wher other may want to update the database there problem known as Reader Writer problem

Solution :-

If a process is writing no other process can read it. If a process is Reading no other process can write it.

Deadlock

A deadlock is a situation where a set of process are blocked because each process is holding a resource and waiting for another resource acquired by some other process.



* Necessary Conditions for the Occurance of Deadlock

- Mutual Exclusion
  Two or more resources are non-shareable.
  (only one process can use at a time)

- Hold and Wait
  A process is holding atleast one resource and waiting for another resource

- No preemption
  A resource cannot be taken from a process unless the process release the resource

- Circular Wait
  A set of processes waiting for each other in circular form to release the resources

→ Handling the Deadlock

- Deadlock Prevention
  The strategy of deadlock prevention is to design the system in such a way that their will be no possibility of deadlock

① Prevention by mutual exclusion

② Prevention by hold and wait condition
  the condition can be prevented by requiring a process request all its required resource at one time and blocking the process untill all of its request can be granted at the same time simultaneously.

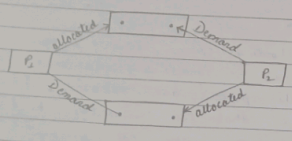
③ Prevention By No preemption
  If a process is holding some resource, requests another resource that cannot be immediately allocated to it all resource held by this process are released, if a process request a resource that can currently held by another process the Operating System can preempt the second process.

④ Prevention By Circular Wait
  To prevent the circular wait each process should request for the resource in increasing order.

- Deadlock Avoidance

• Deadlock detection

→ Resource allocation Graph (RAG) :-

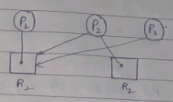


The resource allocation is factorial representation of the state. It gives the complete information about all the process which are holding some resource and waiting for another resource. It also contain the information about all the instance of all the resource whether they are available or been used by process. It is represented by circle and resource are represented by rectangle and the instance is represented by dot.

→ Eages in RAG :- [⟶]

Eages in RAG are also of two type one represent assignment of resource and other represent wait process for a resource

Ex:- Let consider three process P_1, P_2 and P_3 and two type of resource R_1 and R_2, the resource having one instance each. Draw the RAG, According to graph R_1 is used by P_1 and P_2 is holding R_2 and waiting for P_2, P_2 is waiting for R_1 and R_1



• Deadlock Recovery

In order to recover the system from the deadlock we can follow these resource approaches :-

→ For Resource

• Preempt the Resource
• Role Back to Saftey state

The movement we get into the deadlock we will role back all the allocation to get in the previous safe state

→ For Process

• Kill a process

Killing a process can solve our problem but the bigger concern to decide which process to kill. Generally O/S kill a process which has done least amount of work untill now.

• Kill all process {Rare condition}

• Deadlock Ignorance

In this approach the O/S assume the deadlock never occurs. It simply ignore deadlock this approach is best for a single hand user. System where user uses the system only for normal work.