PROCESS SCHEDULING:

CPU is always busy in **Multiprogramming**. Because CPU switches from one job to another job. But in

simple computers CPU sit idle until the I/O request granted.

scheduling is a important OS function. All resources are scheduled before use. (cpu, memory, devices…..)

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

. **Scheduling Objectives**

Maximize throughput.

Maximize number of users receiving acceptable response times.Be predictable.

Balance resource use.

Avoid indefinite postponement.Enforce Priorities.

Give preference to processes holding key resources

SCHEDULING QUEUES:

people live in rooms. Process are present in rooms knows as queues.

There are 3types

1.job queue: when processes enter the system, they are put into a **job queue**, which consists all processes in the system. Processes in the job queue reside on mass storage and awaitthe allocation of main memory.

2.ready queue: if a process is present in main memory and is ready to be allocated tocpu forexecution, is kept in **readyqueue.**

3.device queue: if a process is present in waiting state (or) waiting for an i/o event to complete is said to bein device queue.(or)

The processes waiting for a particular I/O device is called device queue.

Schedulers : There are 3 schedulers

Long term scheduler.
Medium term scheduler
Short term scheduler.

Scheduler duties:

- Maintains the queue.
- Select the process from queues assign to CPU.

Types of schedulers

1. **1.Long term scheduler:**
   select the jobs from the job pool and loaded these jobs into main memory (ready queue).Long term scheduler is also called job scheduler.
2. **2.Short term scheduler:**
   select the process from ready queue, and allocates it to the cpu.
   If a process requires an I/O device, which is not present available then process enters device queue.
   short term scheduler maintains ready queue, device queue. Also called as cpu scheduler.

3.Medium term scheduler: if process request an I/O device in the middle of the execution, then the process removed from the main memory and loaded into the waiting queue. When the I/O operation completed, then the job moved from waiting queue to ready queue. These two operations performed by medium term scheduler

Comparison between Scheduler

S.N.	Long Term Scheduler	Short Term Scheduler	Medium Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch:

Assume, main memory contains more than one process. If cpu is executing a process, if time expires or if a high priority process enters into main memory, then the scheduler saves information about current process in the PCB and switches to execute the another process. The concept of moving CPU by scheduler from one process to other process is known as context switch.

Non-Preemptive Scheduling: **CPU** is assigned to one process, CPU do not release until the competition of that process. The CPU will assigned to some other process only after the previous process has finished.

Preemptive scheduling: here CPU can release the processes even in the middle of the execution. CPU received a signal from process p2. OS compares the priorities of p1 ,p2. If p1>p2, CPU continues the execution of p1. If p1<p2 CPU preempt p1 and assigned to p2.

Dispatcher: The main job of dispatcher is switching the cpu from one process to another process. Dispatcher connects the cpu to the process selected by the short term scheduler.

Dispatcher latency: The time it takes by the dispatcher to stop one process and start another process is known as dispatcher latency. If the dispatcher latency is increasing, then the degree of multiprogramming decreases.

SCHEDULING CRITERIA:

1.Throughput: how many jobs are completed by the cpu with in a timeperiod.

2.Turn around time : The time interval between the submission of the processand time of the completion is turn around time.

TAT = Waiting time in ready queue + executing time + waiting time in waiting queue for I/O.

3.Waiting time: The time spent by the process to wait for cpu to beallocated.

4.Response time: Time duration between the submission and firstresponse.

5.Cpu Utilization: CPU is costly device, it must be kept as busy aspossible.Eg: CPU efficiency is 90% means it is busy for 90 units, 10 units idle.

CPU SCHEDULINGALGORITHMS:

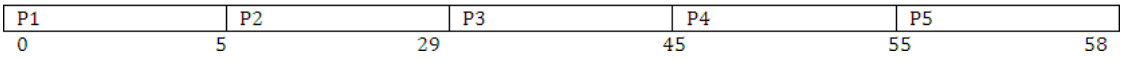
1.First come First served scheduling: (FCFS): The process that request the CPU first is holds the cpu first. If a process request the cpu then it is loaded into the ready queue, connect CPU to that process.

Consider the following set of processes that arrive at time 0, the length of the cpu burst time given in milli seconds.

burst time is the time, required the cpu to execute that job, it is in milli seconds.

Process	**Burst time(milliseconds)**
P1	5
P2	24
P3	16
P4	10
P5	3

Chart:



Average turn around time:

Turn around time= waiting time + burst time

Turn around time for p1= 0+5=5.Turn around time for p2=5+24=29
Turn around timefor p3=29+16=45
Turn around time for p4=45+10=55
Turn around time for p5=55+3=58
Average turn around time= (5+29++45+55+58/5) = 187/5 =37.5 millisecounds

Average waiting time:

waiting time= starting time- arrival time

Waiting time for p1=0
Waiting time for p2=5-0=5
Waiting time for p3=29-0=29
Waiting time for p4=45-0=45
Waiting time for p5=55-0=55
Average waiting time= 0+5+29+45+55/5 = 125/5 = 25 ms.

Average Response Time :

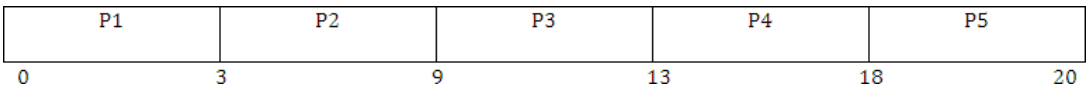
Formula : First Response - ArrivalTime Response Time for P1 =0 Response Time for P2 => 5-0 = 5 Response Time for P3 => 29-0 = 29 Response Time for P4 => 45-0 = 45 Response Time for P5 => 55-0 = 55
Average Response Time => (0+5+29+45+55)/5 =>25ms

First Come First Serve:

It is Non Primitive Scheduling Algorithm.

PROCESS	BURST TIME	ARRIVAL TIME
P1	3	0
P2	6	2
P3	4	4
P4	5	6
P5	2	8

Process arrived in the order P1, P2, P3, P4, P5.P1 arrived at 0 ms.
P2 arrived at 2 ms. P3 arrived at 4 ms. P4 arrived at 6 ms. P5 arrived at 8 ms.



Average Turn Around Time

Formula : Turn around Time =**:** waiting time + burst timeTurn Around Time for P1 => 0+3= 3 Turn Around Time for P2 => 1+6 = 7 Turn Around Time for P3 => 5+4 = 9 Turn Around Time for P4 => 7+ 5= 12 Turn Around Time for P5 => 2+ 10=12

Average Turn Around Time => (3+7+9+12+12)/5 =>43/5 = 8.50 ms.

Average Response Time :

Formula : Response Time = First Response - Arrival TimeResponse Time of P1 = 0 Response Time of P2 => 3-2 = 1 Response Time of P3 => 9-4 = 5 Response Time of P4 => 13-6 = 7 Response Time of P5 => 18-8 =10

Average Response Time => (0+1+5+7+10)/5 => 23/5 = 4.6 ms

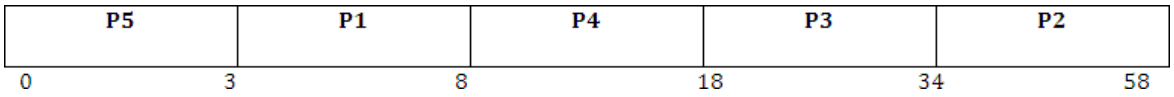
Advantages: Easy to Implement, Simple.

Disadvantage: Average waiting time is very high.

2).Shortest Job First Scheduling (SJF):

Which process having the smallest CPU burst time, CPU is assigned to that process . Iftwo process having the same CPU burst time, FCFS is used.

PROCESS	CPU BURST TIME
P1	5
P2	24
P3	16
P4	10
P5	3



P5 having the least CPU burst time (3ms). CPU assigned to that (P5). After completion of P5 short term scheduler search for nest (P1).....

Average Waiting Time :

Formula = Staring Time - Arrival Timewaiting Time for P1 => 3-0 = 3

waiting Time for P2 => 34-0 = 34waiting Time for P3 => 18-0 = 18 waiting Time for P4 =>8-0=8 waiting time for P5=0
Average waiting time => (3+34+18+8+0)/5 => 63/5 =12.6 ms

Average Turn Around Time :

Formula = waiting Time + burst Time

Turn Around Time for P1 => 3+5 =8Turn Around for P2 => 34+24 =58 Turn Around for P3 => 18+16 = 34

Turn Around Time for P4 => 8+10 =18

Turn Around Time for P5 => 0+3 = 3

Average Turn around time => (8+58+34+18+3)/5 => 121/5 = 24.2 ms

Average Response Time :

Formula : First Response - Arrival Time

First Response time for P1 =>3-0 = 3
First Response time for P2 => 34-0 = 34 First Response time for P3 => 18-0 = 18 First Response time for P4 => 8-0 = 8
First Response time for P5 = 0
Average Response Time => (3+34+18+8+0)/5 => 63/5 = 12.6 msSJF is Non primitive scheduling algorithm
Advantages : **Least average waiting time Least average turn around time Leastaverage response time**
Average waiting time (FCFS) = 25 ms
Average waiting time (SJF) = 12.6 ms 50% time saved in SJF.
Disadvantages:

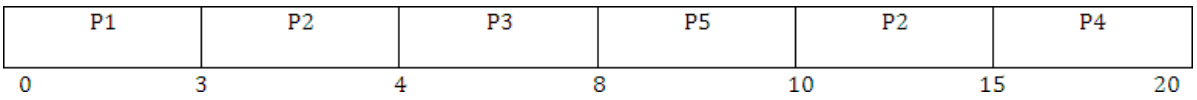
- Knowing the length of the next CPU burst time is difficult.
- Aging (Big Jobs are waiting for long time for CPU)

3).Shortest Remaining Time First (SRTF):

This is primitive scheduling algorithm.

Short term scheduler always chooses the process that has term shortest remaining time. When a new process joins the ready queue , short term scheduler compare the remaining time of executing process and new process. If the new process has the least CPU burst time, The scheduler selects that job and connect to CPU. Otherwise continue the old process.

PROCESS	BURST TIME	ARRIVAL TIME
P1	3	0
P2	6	2
P3	4	4
P4	5	6
P5	2	8



P1 arrives at time 0, P1 executing First , P2 arrives at time 2. Compare P1 remaining time and P2 (3-2 = 1) and 6. So, continue P1 after P1, executing P2, at time 4, P3 arrives, compare P2 remaining time (6-1=5
) and 4 (4<5) .So, executing P3 at time 6, P4 arrives. Compare P3 remaining time and P4 (4- 2=2) and 5 (2<5). So, continue P3 , after P3, ready queue consisting P5 is the least out of three. So execute P5, next P2, P4.
FORMULA : Finish time - ArrivalTime Finish Time for P1 => 3-0 = 3 Finish Time for P2 => 15-2 = 13 Finish Time for P3 => 8-4 =4
Finish Time for P4 => 20-6 = 14Finish Time for P5 => 10-8 = 2

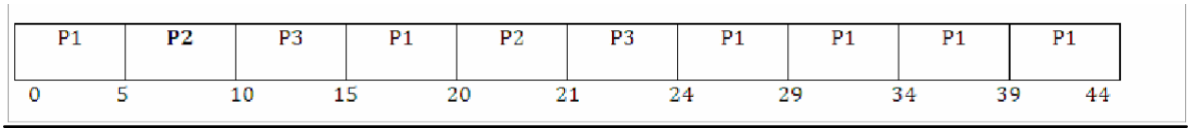
Average Turn around time => 36/5 = 7.2 ms.

4)ROUND ROBIN SCHEDULING ALGORITHM :

It is designed especially for time sharing systems. Here CPU switches between the processes. When the time quantum expired, the CPU switched to another job. A small unit of time, called a time quantum or time slice. A time quantum is generally from 10 to 100 ms. The time quantum is generally depending on OS. Here ready queue is a circular queue. CPU scheduler picks the first process from ready queue, sets timer to interrupt after one time quantum and dispatches the process.

PROCESS	BURST TIME
P1	30

P2	6
P3	8



AVERAGE WAITING TIME :

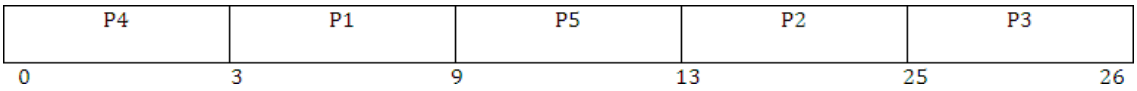
Waiting time for P1 => 0+(15-5)+(24-20) => 0+10+4 = 14
Waiting time for P2 => 5+(20-10) => 5+10 = 15
Waiting time for P3 => 10+(21-15) => 10+6 = 16Average waiting time => (14+15+16)/3 = 15 ms.

AVERAGE TURN AROUND TIME : FORMULA :
Turn around time = waiting time + burst TimeTurn around time for P1 => 14+30 =44
Turn around time for P2 => 15+6 = 21Turn around time for P3 => 16+8 = 24
Average turn around time => (44+21+24)/3 = 29.66 ms

5) PRIORITY SCHEDULING :

PROCESS	BURST TIME	PRIORITY
P1	6	2
P2	12	4
P3	1	5
P4	3	1
P5	4	3

P4 has the highest priority. Allocate the CPU to process P4 first next P1, P5, P2, P3.



AVERAGE WAITING TIME :

Waiting time for P1 => 3-0 =3 Waiting time for P2 => 13-0 = 13 Waiting time for P3 => 25-0 = 25 Waiting time for P4 => 0
Waiting time for P5 => 9-0 =9

Average waiting time => (3+13+25+0+9)/5 = 10 ms

AVERAGE TURN AROUND TIME:

Turn around time for P1 =>3+6 = 9 Turn around time for P2 => 13+12= 25Turn around time for P3 => 25+1 = 26 Turn around time for P4 => 0+3= 3Turn around time for P5 => 9+4 = 13

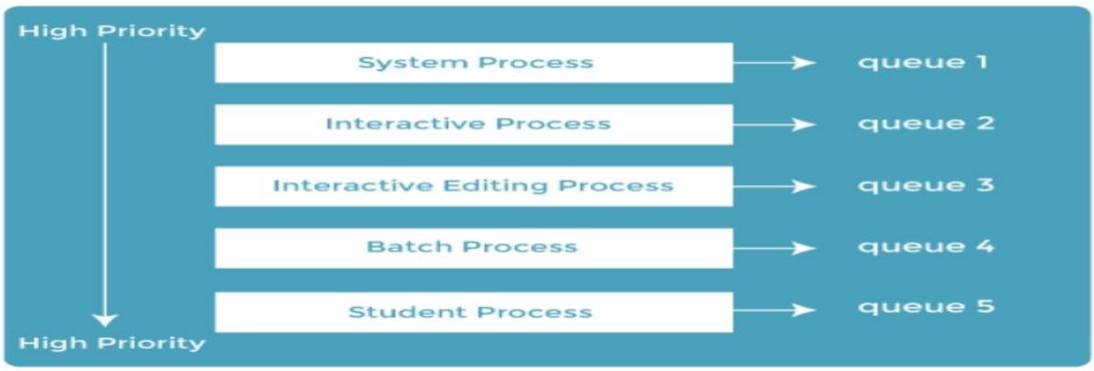
Average Turn around time => (9+25+26+3+13)/5 = 15.2 ms

Disadvantage: Starvation

Starvation means only high priority process are executing, but low priorityprocess are waiting for the CPU for the longest period of the time.

6). Multilevel Queue Scheduling:

ready queue is partitioned into number of ready queues . Each ready queue is capable to load same type of jobs. Each ready queue has its own scheduling algorithm. ready queue is partitioned into 4 ready queues. one ready queue for system processes, one ready queue for background processes, one ready queue for foreground processes, another ready queue for student



1. **System process**
2. **Interactive processes**
3. **Interactive editing processes**
4. **Batch processes**
5. **Student processes**

processes. No student process run unless system, foreground, background process were all empty. Each queue gets a certain portion of the cpu time , which it can then schedule among its various processes.

Inter Process communication:

Process synchronization refers to the idea that multiple processes are to join up or handshake at a certain point, in order to reach an agreement or commit to a certain sequence of action. Coordination of simultaneous processes to complete a task is known as process synchronization.

The critical section problem

Consider a system, assume that it consisting of n processes. Each process having a segment of code. This segment of code is said to be critical section.

E.G: Railway Reservation System.

Two persons from different stations want to reserve their tickets, the train number, destination is common, the two persons try to get the reservation at the same time. Unfortunately, the available berths are only one; both are trying for that berth.

It is also called the critical section problem. Solution is when one process is executing in its critical section, no other process is to be allowed to execute in its critical section.

The critical section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of code implementing this request is the **entry section**. The critical section may be followed by an **exit section**. The remaining code is the **remainder section.**

```
do {
    entry section
        critical section
    exit section
        remainder section
} while (1);
```

Figure General structure of a typical process P_i.

A solution to the critical section problem must satisfy the following 3requirements:

1. **Mutual exclusion:**

   Only one process can execute their critical section at any time.
2. **Progress:**

   When no process is executing a critical section for a data, one of the processeswishing to enter a critical section for data will be granted entry.
3. **Bounded wait:**

   No process should wait for a resource for infinite amount of time.

Critical section:
The portion in any program that accesses a shared resource is called as critical section (or) critical region.

Petersons Solution:
Peterson solution is one of the solutions to critical section problem involving two processes. This solution states that when one process is executing its critical sectionthen the other process executes the rest of the code and vice versa.
Peterson solution requires two shared data items:
1). turn: indicates whose turn it is to enterinto the critical section. If turn == i then process i is allowed into their critical section.

2). flag: indicates when a process wants to enter into critical section. when process i wants to enter their critical section, it sets flag[i] to true.

```
do {
flag[i] = TRUE;
turn = j;
while (flag[j] && turn == j);
critical section
flag[i] = FALSE;
remainder section
} while (TRUE);
```

Synchronization hardware

In a uniprocessor multiprogrammed system, mutual exclusion can be obtained by disabling the interrupts before the process enters its critical section and enabling them after it has exited the critical section**.**

Disable interrupts Critical section Enable interrupts

Once a process is in critical section it cannot be interrupted. This solution cannot be used in multiprocessor environment. since processes run independently on different processors.

In multiprocessor systems, **Testandset** instruction is provided,it completes execution without interruption. Each process when entering their critical section must set **lock**,to prevent other processes from entering their critical sections simultaneously and must release the lock when exiting their critical sections.

do {

acquire lock critical section release lock remaindersection

} while (TRUE);

A process wants to enter critical section and value of lock is false then **testandset** returns false and the value of lock becomes true. thus for other processes wanting to enter their critical sections **testandset** returns true and the processes do busy waiting until the process exits critical section and sets the value of lock to false.

- **Definition:**

boolean TestAndSet(boolean&lock){boolean temp=lock;
Lock=true; return temp;
}

Algorithm for TestAndSet

do{
while testandset(&lock)
//do nothing
//critical sectionlock=false
remainder section
}while(TRUE);

Swap instruction can also be used for mutual exclusionDefinition

```
Void swap(boolean &a, boolean &b)
{
boolean temp=a;a=b;
b=temp;
}
```

Algorithm

```
do
{
key=true;
while(key=true)
swap(lock,key);
critical section
lock=false;
remainder section
}while(1);
```

lock is global variable initialized to false.each process has a local variable key. A process wants to enter critical section,since the value of lock is false and key is true.

lock=falsekey=true
after swap instruction,
lock=truekey=false

now key=false becomes true,process exits repeat-until,and enter into critical section.When process is in critical section (lock=true),so other processes wanting to enter critical section will have
lock=truekey=true
Hence they will do busy waiting in repeat-until loop until the process exits criticalsection and sets the value of lock to false.

Semaphores

A semaphore is an integer variable.semaphore accesses only through two operations.

1) **wait:** wait operation decrements the count by1.
If the result value is negative,the process executing the wait operation is blocked.

2) **Signal operation:**
Signal operation increments by 1,if the value is not positive then one of theprocess blocked

in wait operation unblocked.

```
wait (S) {
while S <= 0 ; //no-op
S--;
}

signal (S)
{

S++;
}
```

In binary semaphore count can be 0 or 1. The value of semaphore isinitialized to 1.

```
do {
wait (mutex);
// Critical Sectionsignal (mutex);
// remainder section

} while (TRUE);
```

First process that executes wait operation will be immediately granted sem.count to 0.If some other process wants critical section and executes wait() then it is blocked,since value becomes -1. If the process exits critical section it executes signal().sem.count is incremented by 1.blocked process is removed from queue and added to ready queue.

Problems:

1) **Deadlock**
Deadlock occurs when multiple processes are blocked.each waiting for a resourcethat can only be freed by one of the other blocked processes.
2) **Starvation**
one or more processes gets blocked forever and never get a chance to take theirturn in the critical section.
3) **Priority inversion**
If low priority process is running ,medium priority processes are waiting for low priority process,high priority processes are waiting for medium priority processes.this is called Priority inversion.

The two most common kinds of semaphores are **counting semaphores** and **binary**

semaphores. Counting semaphores represent multiple resources, while binary semaphores, as the name implies, represents two possible states(generally 0 or 1; locked or unlocked).

Classic problems of synchronization

1) Bounded-buffer problem

Two processes share a common, fixed –size buffer.

Producer puts information into the buffer, consumer takes it out.

The problem arise when the producer wants to put a new item in the buffer, but it is already full. The solution is for the producer has to wait until the consumer has consumed atleast one buffer. similarly if the consumer wants to remove an item from the buffer and sees that the buffer is empty, it goes to sleep until the producer puts something in the buffer and wakes it up.

synchronisation problems:

i) we must guard against attempting to write data to the buffer when the buffer is full; ie the producer must wait for an 'empty space'.

ii) we must prevent the consumer from attempting to read data when the buffer is empty; ie, the consumer must wait for 'data available'.

To provide for each of these conditions, we require to employ three semaphores which are defined in the following table:

Semaphore	*Purpose*	*Initial Value*
free	mutual exclusion for buffer access	1
space	space available in buffer	N
data	data available in buffer	0

The structure of the producer process

```
do {
// produce an item in nextp wait (empty);
wait (mutex);
// add the item to thebuffer signal (mutex); signal (full);
} while (TRUE);
```

The structure of the consumer process

```
do {

wait (full);
wait (mutex);
```

// remove an item from buffer tonextc signal (mutex);
signal (empty);
// consume the item in nextc
} while (TRUE);

2) The readers-writers problem

A database is to be shared among several concurrent processes. some processes maywant only to read the database, some may want to update the database. If two readersaccess the shared data simultaneously no problem. if a write, some other process accesses the database simultaneously problem arised. Writes have exclusive access to the shared database while writing to the database. This problem is known asreaders- writes problem.

First readers-writers problem

No reader be kept waiting unless a writer has already obtained permission touse the shared resource.

Second readers-writes problem:

Once writer is ready, that writer performs its write as soon as possible.

A process wishing to modify the shared data must request the lock in write mode. multiple processes are permitted to concurrently acquire a reader-writer lock in read mode. A reader writer lock in read mode. but only one process may acquirethe lock for writing as exclusive access is required for writers.

Semaphore mutex initialized to 1

- Semaphore wrt initialized to 1
- Integer read count initialized to 0

The structure of a writer process

```
do {
wait (wrt) ;
//                    writing   isperformed signal (wrt) ;
} while (TRUE);
```

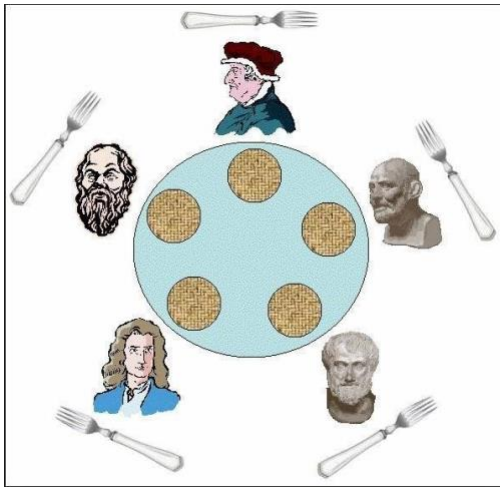
The structure of a reader process

```
do {
wait (mutex) ;readcount ++ ;
if (readcount == 1)wait (wrt) ;
signal (mutex)
// reading is performed wait (mutex) ;readcount
- - ;
if (readcount == 0)signal (wrt) ; signal (mutex) ;
```

} while (TRUE);

3) Dining Philosophers problem

Five philosophers are seated on 5 chairs across a table. Each philosopher has a plate full of noodles. Each philosopher needs a pair of forks to eat it. There are only5 forks available all together. There is only one fork between any two plates of noodles.
In order to eat, a philosopher lifts two forks, one to his left and the other to his right. if he is successful in obtaining two forks, he starts eating after some time, he stops eating and keeps both the forks down.



What if all the 5 philosophers decide to eat at the same time ?
All the 5 philosophers would attempt to pick up two forks at the same time. So,none of them succeed.

One simple solution is to represent each fork with a semaphore.a philosopher tries to grab a fork by executing wait() operation on that semaphore.he releases his forks by executing the signal() operation.This solution guarantees that no two neighbours are eating simultaneously.
Suppose all 5 philosophers become hungry simultaneously and each grabs his left fork,he will be delayed forever.

The structure of Philosopher *i*:

```
do{
wait ( chopstick[i] );
wait ( chopStick[ (i + 1) % 5] );
// eat
signal ( chopstick[i] );
signal (chopstick[ (i + 1) % 5] );
```

// think
} while (TRUE);

Several remedies:

1) Allow at most 4 philosophers to be sitting simultaneously at the table.
2) Allow a philosopher to pick up his fork only if both forks are available.
3) An odd philosopher picks up first his left fork and then right fork. an even philosopher picks up his right fork and then his left fork.

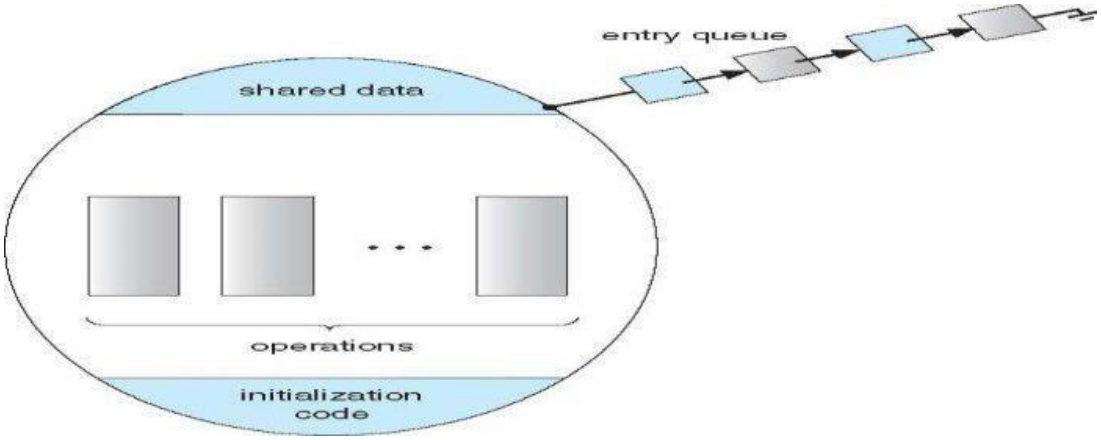
MONITORS

The disadvantage of semaphore is that it is unstructured construct. Wait and signal operations can be scattered in a program and hence debugging becomes difficult.
A monitor is an object that contains both the data and procedures needed to perform allocation of a shared resource. To accomplish resource allocation using monitors, a process must call a **monitor entry routine**. Many processes may want to enter the monitor at the same time. but only one process at a time is allowed to enter. Data inside a monitor may be either global to all routines within the monitor (or) local to a specific routine. Monitor data is accessible only within the monitor. There is no way for processes outside the monitor to access monitor data. This is a form of information hiding.

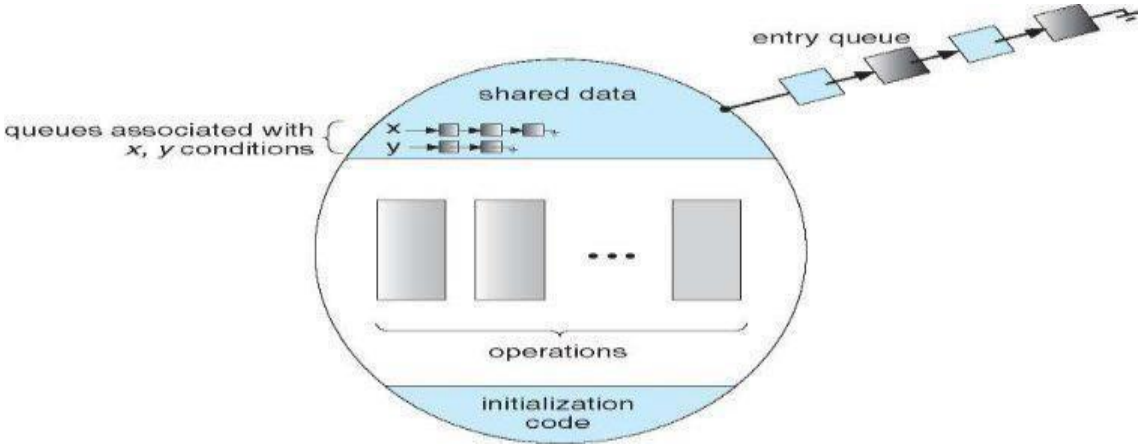
If a process calls a monitor entry routine while no other processes are executing inside the monitor, the process acquires a lock on the monitor and enters it. while a process is in the monitor, other processes may not enter the monitor to acquire the resource. If a process calls a monitor entry routine while the other monitor is locked the monitor makes the calling process wait outside the monitor until the lock on the monitor is released. The process that has the resource will call a monitor entry routine to release the resource. This routine could free the resource and wait for another requesting process to arrive monitor entry routine calls signal to allow one of the waiting processes to enter the monitor and acquire the resource. Monitor gives high priority to waiting processes than to newly arriving ones.

Structure:
monitor monitor-name
{
// shared variable declarationsprocedure P1 (…) { …. } procedurePn (…) {……} Initialization code (…) { … }
}
}
Processes can call procedures p1,p2,p3……They cannot access the local variables of the monitor

Schematic view of a Monitor



Monitor with Condition Variables



Monitor provides condition variables along with two operations on them i.e. wait and signal.

wait(condition variable) signal(condition variable)

Every condition variable has an associated queue.A process calling wait on a particular condition variable is placed into the queue associated with that condition variable.A process calling signal on a particular condition variable causes a process waiting on that condition variable to be removed from the queue associated with it.

Solution to Producer consumer problem using monitors:

```
monitor producerconsumercondition full,empty;
int count;
procedure insert(item)
{
if(count==MAX) wait(full) ; insert_item(item);count=count+1; if(count==1) signal(empty);
}
procedure remove()
{
if(count==0) wait(empty); remove_item(item);count=count-1; if(count==MAX-1)
signal(full);
}
procedure producer()
{
producerconsumer.insert(item);
}
procedure consumer()
{
producerconsumer.remove();
}
```

Solution to dining philosophers problem using monitors

```
        void test(int i) {
           if ((state[(i + 4) % 5] != eating) &&
              (state[i] == hungry) &&
              (state[(i + 1) % 5] != eating)) {
                 state[i] = eating;
                 self[i].signal();
           }
        }

        void init() {
           for (int i = 0; i < 5; i++)
              state[i] = thinking;
        }
     }
```

Figure A monitor solution to the dining-philosopher problem.

```
      test((i + 1) % 5);
   }
```

A philosopher may pickup his forks only if both of them are available.A philosopher can eat only if his two neighbours are not eating.some other philosopher can delay himself when he is hungry.

Diningphilosophers.Take_forks( ) : acquires forks ,which may block the process.

Eat noodles ( )

Diningphilosophers.put_forks( ): releases the forks.

Resuming processes within a monitor

If several processes are suspended on condion x and x.signal( ) is executed by some process.then

how do we determine which of the suspended processes should be resumed next ? solution is FCFS(process that has been waiting the longest is resumed first).In many circumstances, such simple technique is not adequate. alternate solution is to assign priorities and wake up the process with the highest priority.

Resource allocation using monitorboolean inuse=false; conditionavailable; //conditionvariable

```
monitorentry void get resource()
{
if(inuse)        //is resource inuse
{
wait(available);        //wait until available issignaled
}
inuse=true;    //indicate resource is now inuse
}
monitor entry void return resource()
{
inuse=false;    //indicate resourceis not in use signal(available); //signal a waiting process to proceed
}
```

SHORT ANSWER QUESTIONS:

1. What is Multi programming?
2. What is Multi-tasking?
3. What is Scheduling?
4. What is Job Queue?
5. What is Ready Queue?
6. What is Device Queue?
7. What is Long Term Scheduler?
8. What is Medium Term Scheduler?
9. What is Short Term Scheduler?
10. What is Preemptive Scheduling?
11. What is Non-Preemptive Scheduling?
12. What is Turn Around Time?
13. What is Waiting Time?
14. What is Response Time?
15. What is Semaphore?
16. What is Monitor?
17. What is Mutual Exclusion?
18. What is Concurrency?

LONG ANSWER QUESTIONS:

1. What is Scheduling? Explain all types of Schedulers?
2. Explain Scheduling Criteria.
3. Explain First Come First Served Scheduling Algorithm with an example.
4. Explain Shortest Remaining Time First Scheduling Algorithm with an example.
5. Explain Round Robin Scheduling Algorithm with an example.
6. Explain Priority Scheduling Algorithm with an example.
7. Write a short note on Multi Queue Scheduling.
8. What are the three requirements that must satisfy the critical section problem.
9. Explain the Peterson's solution for critical section problem.
10. Write a short note on Semaphores.
11. Write a short note on Monitors.
12. Explain about Inter Process Communication.
13. What is synchronization? Explain the classic problems of Synchronization.
14. Write a short note on readers-writers problem.
15. Write a short note on Bounded-Buffer problem.

CPU Scheduling MCQ – SET 1

1. In priority scheduling algorithm _____

a) CPU is allocated to the process with highest priority

b) CPU is allocated to the process with lowest priority

c) Equal priority processes cannot be scheduled

d) None of the mentioned

Answer: CPU is allocated to the process with highest priority

2. Process are classified into different groups in _____

a) shortest job scheduling algorithm

b) round robin scheduling algorithm

c) priority scheduling algorithm

d) multilevel queue scheduling algorithm

Answer: multilevel queue scheduling algorithm

3. The interval from the time of submission of a process to the time of completion is termed as _____

a) waiting time

b) turnaround time

c) response time

d) throughput

Answer: turnaround time

4. In multilevel feedback scheduling algorithm _____

a) a process can move to a different classified ready queue

b) classification of ready queue is permanent

c) processes are not classified into groups

d) none of the mentioned

Answer: a process can move to a different classified ready queue

5. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called _____

a) job queue

b) ready queue

c) execution queue

d) process queue

Answer: ready queue

6. In priority scheduling algorithm, when a process arrives at the ready queue, its priority is compared with the priority of _____

a) all process

b) currently running process

c) parent process

d) init process

Answer: currently running process

7. Which module gives control of the CPU to the process selected by the short-term scheduler?

a) dispatcher

b) interrupt

c) scheduler

d) none of the mentioned

Answer: dispatcher

8. Which one of the following cannot be scheduled by the kernel?

a) kernel level thread

b) user level thread

c) process

d) none of the mentioned

Answer: user level thread

9. Which algorithm is defined in Time quantum?

a) shortest job scheduling algorithm

b) round robin scheduling algorithm

c) priority scheduling algorithm

d) multilevel queue scheduling algorithm

Answer: round robin scheduling algorithm

10. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?

a) first-come, first-served scheduling

b) shortest job scheduling

c) priority scheduling

d) none of the mentioned

Answer: first-come, first-served scheduling

CPU Scheduling Benefits MCQ- SET 2

1. The switching of the CPU from one process or thread to another is called _____

a) process switch

b) task switch

c) context switch

d) all of the mentioned

Answer: all of the mentioned

2. An I/O bound program will typically have _____

a) a few very short CPU bursts

b) many very short I/O bursts

c) many very short CPU bursts

d) a few very short I/O bursts

Answer: many very short CPU bursts

3. Scheduling is done so as to _____

a) increase CPU utilization

b) decrease CPU utilization

c) keep the CPU more idle

d) none of the mentioned

Answer: increase CPU utilization

4. Scheduling is done so as to _____

a) increase the turnaround time

b) decrease the turnaround time

c) keep the turnaround time same

d) there is no relation between scheduling and turnaround time

Answer: decrease the turnaround time

5. CPU scheduling is the basis of _____

a) multiprocessor systems

b) multiprogramming operating systems

c) larger memory sized systems

d) none of the mentioned

Answer: multiprogramming operating systems

6. Scheduling is done so as to _____

a) increase the throughput

b) decrease the throughput

c) increase the duration of a specific amount of work

d) none of the mentioned

Answer: increase the throughput

7. Scheduling is done so as to _____

a) increase the waiting time

b) keep the waiting time the same

c) decrease the waiting time

d) none of the mentioned

Answer: decrease the waiting time

8. With multiprogramming _____ is used productively.

a) time

b) space

c) money

d) all of the mentioned

Answer: time

9. A process is selected from the _____ queue by the _____ scheduler, to be executed.

a) blocked, short term

b) wait, long term

c) ready, short term

d) ready, long term

Answer: ready, short term

10. What is Response time?

a) the total time taken from the submission time till the completion time

b) the total time taken from the submission time till the first response is produced

c) the total time taken from submission time till the response is output

d) none of the mentioned

Answer: the total time taken from the submission time till the first response is produced

11. What is Dispatch latency?

a) the speed of dispatching a process from running to the ready state

b) the time of dispatching a process from running to ready state and keeping the CPU idle

c) the time to stop one process and start running another one

d) none of the mentioned

Answer: the time to stop one process and start running another one

12. What is Waiting time?

a) the total time in the blocked and waiting queues

b) the total time spent in the ready queue

c) the total time spent in the running queue

d) the total time from the completion till the submission of a process

Answer: the total time spent in the ready queue

13. What are the two steps of a process execution?

a) I/O & OS Burst

b) CPU & I/O Burst

c) Memory & I/O Burst

d) OS & Memory Burst

Answer: CPU & I/O Burst

14. What is Turnaround time?

a) the total waiting time for a process to finish execution

b) the total time spent in the ready queue

c) the total time spent in the running queue

d) the total time from the completion till the submission of a process

Answer: the total time from the completion till the submission of a process

15. In the following cases non – preemptive scheduling occurs?

a) When a process switches from the running state to the ready state

b) When a process goes from the running state to the waiting state

c) When a process switches from the waiting state to the ready state

d) All of the mentioned

Answer: When a process goes from the running state to the waiting state

CPU Scheduling Algorithms MCQ- SET 3

1. The strategy of making processes that are logically runnable to be temporarily suspended is called _____

a) Non preemptive scheduling

b) Preemptive scheduling

c) Shortest job first

d) First come First served

Answer: Preemptive scheduling

2. A solution to the problem of indefinite blockage of low – priority processes is _____

a) Starvation

b) Wait queue

c) Ready queue

d) Aging

Answer: Aging

3. Preemptive Shortest Job First scheduling is sometimes called _____

a) Fast SJF scheduling

b) EDF scheduling – Earliest Deadline First

c) HRRN scheduling – Highest Response Ratio Next

d) SRTN scheduling – Shortest Remaining Time Next

Answer: SRTN scheduling – Shortest Remaining Time Next

4. Round robin scheduling falls under the category of _____

a) Non-preemptive scheduling

b) Preemptive scheduling

c) All of the mentioned

d) None of the mentioned

Answer: Preemptive scheduling

5. An SJF algorithm is simply a priority algorithm where the priority is _____

a) the predicted next CPU burst

b) the inverse of the predicted next CPU burst

c) the current CPU burst

d) anything the user wants

Answer: the predicted next CPU burst

6. The real difficulty with SJF in short term scheduling is _____

a) it is too good an algorithm

b) knowing the length of the next CPU request

c) it is too complex to understand

d) none of the mentioned

Answer: knowing the length of the next CPU request

7. With round robin scheduling algorithm in a time shared system _____

a) using very large time slices converts it into First come First served scheduling algorithm

b) using very small time slices converts it into First come First served scheduling algorithm

c) using extremely small time slices increases performance

d) using very small time slices converts it into Shortest Job First algorithm

Answer: using very large time slices converts it into First come First served scheduling algorithm

8. The FCFS algorithm is particularly troublesome for _____

a) time sharing systems

b) multiprogramming systems

c) multiprocessor systems

d) operating systems

Answer: multiprogramming systems

9. The portion of the process scheduler in an operating system that dispatches processes is concerned with _____

a) assigning ready processes to CPU

b) assigning ready processes to waiting queue

c) assigning running processes to blocked queue

d) all of the mentioned

Answer: assigning ready processes to CPU

10. Complex scheduling algorithms _____

a) are very appropriate for very large computers

b) use minimal resources

c) use many resources

d) all of the mentioned

Answer: are very appropriate for very large computers

11. Orders are processed in the sequence they arrive if _____ rule sequences the jobs.

a) earliest due date

b) slack time remaining

c) first come, first served

d) critical ratio

Answer: first come, first served

12. Under multiprogramming, turnaround time for short jobs is usually _____ and that for long jobs is slightly _____

a) Lengthened; Shortened

b) Shortened; Lengthened

c) Shortened; Shortened

d) Shortened; Unchanged

Answer: Shortened; Lengthened

13. What is FIFO algorithm?

a) first executes the job that came in last in the queue

b) first executes the job that came in first in the queue

c) first executes the job that needs minimal processor

d) first executes the job that has maximum processor needs

Answer: first executes the job that came in first in the queue

14. What is Scheduling?

a) allowing a job to use the processor

b) making proper use of processor

c) all of the mentioned

d) none of the mentioned

Answer: allowing a job to use the processor

15. There are 10 different processes running on a workstation. Idle processes are waiting for an input event in the input queue. Busy processes are scheduled with the Round-Robin time sharing method. Which out of the following quantum times is the best value for small response times, if the processes have a short runtime, e.g. less than 10ms?

a) tQ = 15ms

b) tQ = 40ms

c) tQ = 45ms

d) tQ = 50ms

Answer: tQ = 15ms

16. Which of the following algorithms tends to minimize the process flow time?

a) First come First served

b) Shortest Job First

c) Earliest Deadline First

d) Longest Job First

Answer: Shortest Job First

17. Which of the following statements are true?

I. Shortest remaining time first scheduling may cause starvation

II. Preemptive scheduling may cause starvation

III. Round robin is better than FCFS in terms of response time

a) I only

b) I and III only

c) II and III only

d) I, II and III

Answer: I, II and III

18. Which is the most optimal scheduling algorithm?

a) FCFS – First come First served

b) SJF – Shortest Job First

c) RR – Round Robin

d) None of the mentioned

Answer: SJF – Shortest Job First

19. Consider the following set of processes, the length of the CPU burst time given in milliseconds.

Process	Burst time
P1	6
P2	8
P3	7
P4	3

Assuming the above process being scheduled with the SJF scheduling algorithm.

a) The waiting time for process P1 is 3ms

b) The waiting time for process P1 is 0ms

c) The waiting time for process P1 is 16ms

d) The waiting time for process P1 is 9ms

Answer: The waiting time for process P1 is 3ms

20. Choose one of the disadvantages of the priority scheduling algorithm?

a) it schedules in a very complex manner

b) its scheduling takes up a lot of time

c) it can lead to some low priority process waiting indefinitely for the CPU

d) none of the mentioned

Answer: it can lead to some low priority process waiting indefinitely for the CPU

21. What is ‘Aging’?

a) keeping track of cache contents

b) keeping track of what pages are currently residing in memory

c) keeping track of how many times a given page is referenced

d) increasing the priority of jobs to ensure termination in a finite time

Answer: increasing the priority of jobs to ensure termination in a finite time

22. Which of the following statements are true?

i) Shortest remaining time first scheduling may cause starvation

ii) Preemptive scheduling may cause starvation

iii) Round robin is better than FCFS in terms of response time

a) i only

b) i and iii only

c) ii and iii only

d) i, ii and iii

Answer: i, ii and iii

23. Which of the following scheduling algorithms gives minimum average waiting time?

a) FCFS

b) SJF

c) Round – robin

d) Priority

Answer: SJF

24. Using Priority Scheduling algorithm, find the average waiting time for the following set of processes given with their priorities in the order: Process : Burst Time : Priority respectively .

P1 : 10 : 3

P2 : 1 : 1

P3 : 2 : 4

P4 : 1 : 5

P5 : 5 : 2

a) 8 milliseconds

b) 8.2 milliseconds

c) 7.75 milliseconds

d) 3 milliseconds

Answer: 8.2 milliseconds

25. Which of the following is a criterion to evaluate a scheduling algorithm?

a) CPU Utilization: Keep CPU utilization as high as possible

b) Throughput: number of processes completed per unit time

c) Waiting Time: Amount of time spent ready to run but not running

d) All of the above

Answer: All of the above