Process :

When a program resides on disk then it is simply a program and when it is loaded into main memory for execution, it becomes a process and then it competes for resources such as CPU, memory, files etc. Process is a program in execution.

The OS allocates system resources and keep track of all active processes. In multi-processing system, a no. of active processes are in various stages of execution at any point. The OS keeps track of the state of each process and records all the state changes as they occur.

The process management provides the following functions :

1) creating and removing process.
2) controlling the progress of a processes.
3) tracing the interrupt during the program execution
4) Allocating hardware resources among processes.
5) Providing synchronization and communication signals among processes.

Process states :

As a process executes, it changes state. The state of a process is defined

by current activity of that process.

New :- The process is being created.
Running :- Instructions are being executed.
Waiting :- The process is waiting for some event to occur (such as an I/O completion)
Ready :- The process is waiting to be assigned to a processor.
Terminated :- The process has finished execution.

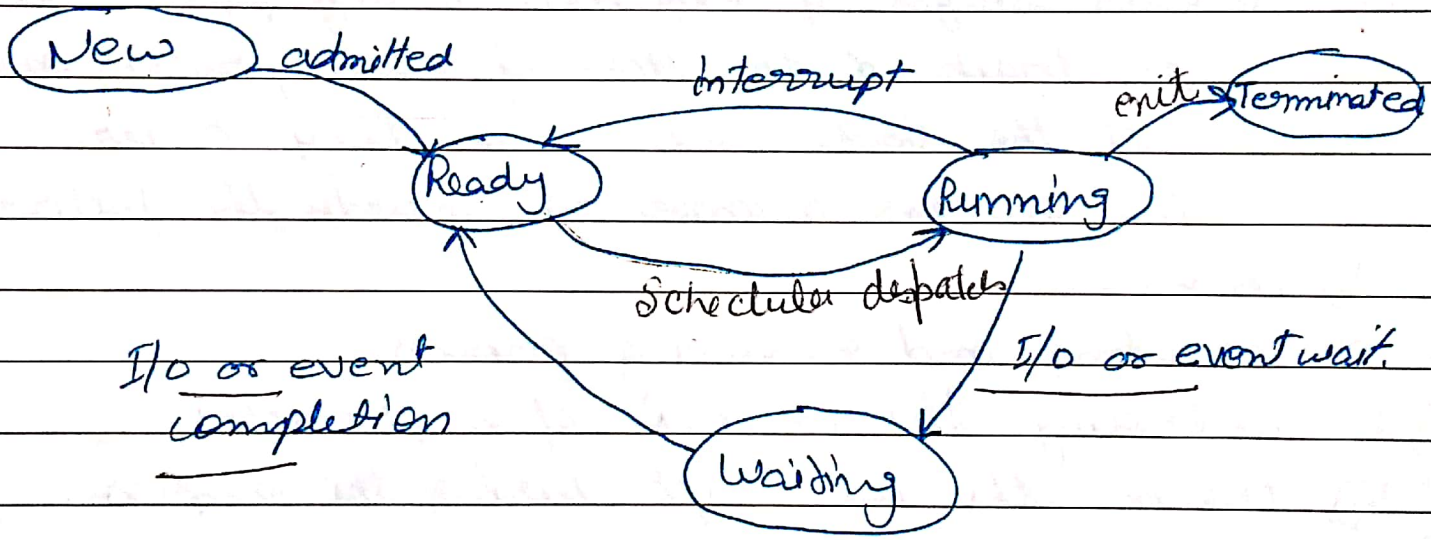


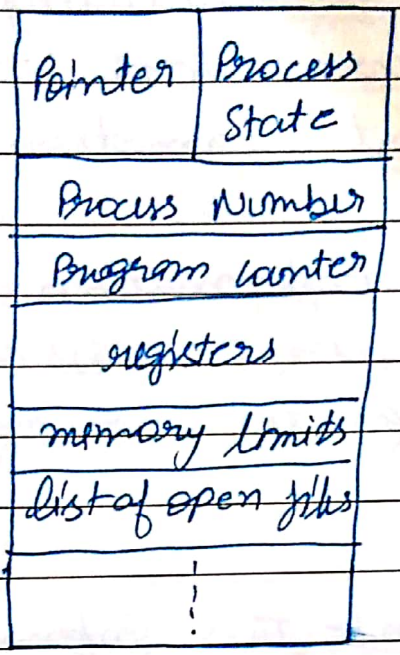
Diagram of process state.

Process Control Block :- (PCB)

Each process is represented in the operating system by a process control block (PCB) also called task control block. It contains many pieces of information

Dt...../...../.....

Associated with a specific process.



PCB

⇒ Process States: The state may be new, ready, running, waiting ... etc.

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

⇒ CPU registers: The registers vary in numbers and type, depending on the computer architecture. They include accumulator, index register, stack pointers and general purpose registers.

Dt...../...../....

⇒ CPU-Scheduling Information:- This information includes a process priority, pointers to scheduling queues and other scheduling parameters.

⇒ Memory-management information:- This information may include such information as the value of the base and limit registers, the page tables depending on the memory system used by the OS.

⇒ Accounting information:- This information includes the amount of CPU, time limits, account numbers, job or process numbers, and so on.

⇒ I/O Status information:- The information includes the list of I/O devices allocated to the process a list of open files etc.

CPU Scheduling

In multiprogramming A process is executed until it must wait for completion of I/O request. In simple computer CPU sit idle; all this waiting time is wasted. With multiprogramming, when one process has to wait, the OS takes the CPU away from that process and give the CPU to another process. This pattern continues.

Scheduling is the function of OS. Almost all computer resources are scheduled before use.

CPU – I/O Burst Cycle's

Process execution consists of a cycle of CPU execution and I/O wait. Processes alternate b/w these two states. Process execution begins with a CPU burst. That is followed by a I/O burst, then another CPU burst, then another I/O burst, and so on. The CPU burst will end with a system request to terminate execution, rather than another I/O burst.

Dt...../...../....

load store
add store. CPU burst } CPU burst
read from file

wait for I/O } I/O burst

store increment
index
write to file } CPU burst

wait for I/O } I/O burst

load store
add store
read from file } CPU burst

wait for I/O } I/O burst.

Alternate Sequence of CPU & I/O bursts.

CPU Scheduler:

When the CPU become idle, the OS must select one of the processes in the ready queue to be executed. The selection of process is carried out by short-term scheduler or CPU scheduler. The CPU scheduler selects from among the processes in memory that are ready to execute and allocates the CPU to one of them.

Preemptive Scheduling :-

Non-preemptive scheduling :-
Once the CPU has been allocated to a process, the process keeps the CPU until its termination or its goes to the waiting states.

Preemptive Scheduling :-
Here even if the CPU has been allocated to a certain process, it can be preempted from this process any time either due to time constraint or due to priority reasons.

Content Switching :-
Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as a content switch. The content of a process is represented in the PCB of a process; it includes the value of CPU registers, the process states and memory management information. When a content switch occurs, the Kernel saves the content of old process in its PCB and loads the saved content of the new process scheduled to run.

Scheduling Criteria's

① CPU utilization: we want to keep the CPU as busy as possible. The CPU utilization may range from 0 to 100 percent. In a real system, it should range from 40% to 90%

② Throughput :- No. of processes completed per time unit called throughput.

~~③ The to~~

③ Turnaround time: how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

Turn around time = time of completion of job − time of submission of job

④ Waiting time :- CPU-scheduling algo. does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

⇒ It is the sum of the time intervals for which the Process has to wait in the ready queue.

⑤ Response time :— is the amount of time it takes to start responding, but not the time that it takes to output that response.

We want maximize CPU utilization and throughput and to minimum turnaround time, waiting time, and response time.

Scheduling Algorithms :—

① First come first-served (FCFS) :—

With this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS is managed with a FIFO queue. When the process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue.

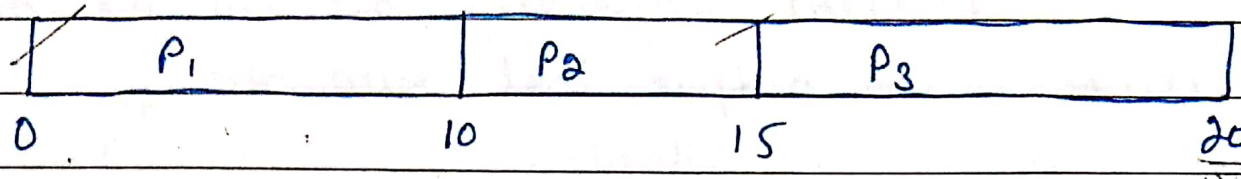
The average time under FCFS is quite long.

e.g. :—

Process	Burst time (in ms)	AT
P_1	10	0
P_2	5	1
P_3	5	2

If the process arrived in order P_1, P_2, and P_3, then the avg. waiting time for the process

avg. waiting time:



20

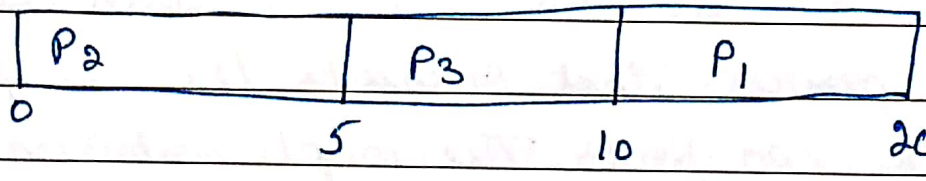
waiting time for $P_1 = 0$

" $P_2 = 10$

$P_3 = 15$

Avg. waiting time $= \frac{0+10+15}{3} = \frac{25}{3} = 8.33$ m.s.

But if process arrive in order P_2, P_3 and P_1



Avg. waiting time $= \frac{0+5+10}{3} = \frac{15}{3} = 5$ ms.

Avg waiting time will always depend upon the order in which the processes arrive. This algo never recommended where performance is a major issue.

The FCFS is non-preemptive. Once ~~Shortest Job First Scheduling~~ the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU either by terminating or by requesting I/O.

② Shortest-Job-First Algorithm:-

This algorithm associates with each process the length of the next CPU burst. When CPU is available, it is assigned to the process that has the smallest next CPU burst. If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie.

Advantage:- It gives the minimum average waiting time.

Disadvantage:-

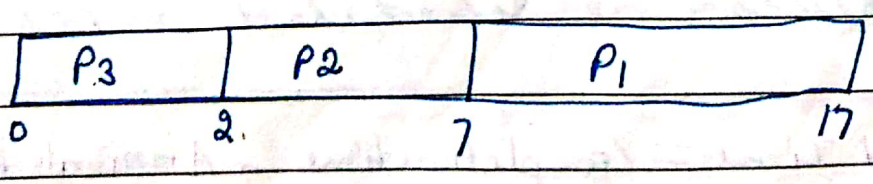
The problem is to know the length of time for which CPU is needed by a process. A prediction formula can be used to predict the amount of time for which CPU may be required by a process.

e.g.:-

P	BT
P1	6
P2	8
P3	7
P4	3

Process	CPU-burst
P_1	10
P_2	5
P_3	2

with non-preemptive SJF



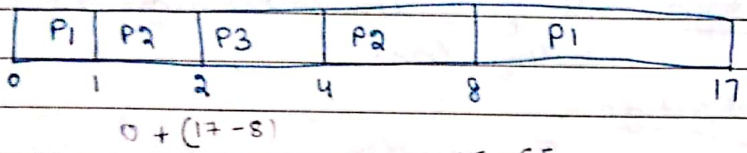
(Gantt chart)

Avg. waiting time = $\frac{0+2+7}{3} = \frac{9}{3} = 3$ milliseconds

P2	1	4
P3	2	9
P4	3	5

If the process arrived in the order P_1, P_2, P_3 and at the time as mentioned, then avg. waiting time using preemptive SJF

		Time of Arrival
P_1	10 → 9	0
P_2	15 → 14	1
P_3	2	2



0 + (17 − 8)

AT − CT

Waiting time for P_1 = 0 + (8 − 1) = 7
" " P_2 = 1 + (4 − 2) = 3
" " P_3 = 2

Avg waiting time = $\frac{7+3+2}{3} = \frac{12}{3} = 4$ ms.

(3) Priority Scheduling:-

A priority is associated with each process and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in FCFS order.

Turnaround Time = Completion time − Arrival time
Waiting Time = Turnaround time − Burst time

1.
2.

Process	Burst Time	Priority
P_1	10	3
P_2	1	1 → 0
P_3	2	4
P_4	1	5
P_5	5 → 4	2

Avg waiting time is 8.2 ms.

Priority scheduling can be either preemptive or nonpreemptive. When a process arrives at the ready queue, its priority is compared with the priority of the currently running process. A preemptive priority-scheduling algo. will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process. A non-preemptive priority scheduling algo. will simply put the new process at the head of the ready queue.

A major problem with priority scheduling is indefinite blocking or starvation. A priority-scheduling algo. can leave some low-priority processes waiting indefinitely for the CPU.

In this problem higher priority processes can prevent a low-priority process from ever getting the CPU.

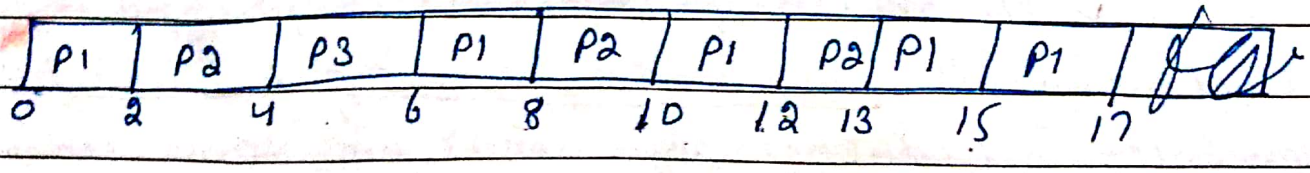
A solution to the problem of indefinite blockage of low-priority processes is aging. Aging is the technique of increasing the priority of processes that wait the system for a long time.

✓ Round-Robin Scheduling:-

RR scheduling algorithm is designed especially for time sharing systems. It is similar to FCFS scheduling, but preemption is added to switch b/w processes. A small unit of time, called a time quantum or time slice. is defined. Its implementation requires timer interrupts based on which the preemption takes place.

e.g:-

Process	CPU-Burst time
P1	10
P2	5
P3	2



Time quantum is 2 milli-seconds

RR Algo.

Dt...../...../.....

Waiting time for P1 will be = 0 + (6-2) + (10-8) + (13-12)

= 4 + 2 + 1 = 7 ms.

" " P2 = 2 + (8-4) + (12-10)

= 2 + 4 + 2 = 8 ms.

" " P3 = 4

Avg. waiting time = $\frac{7+8+4}{3} = \frac{19}{3} = 6.33$ ms.

The performance of this algo. depends upon many factors:

(1) Size of time quantum.

* If time slice size is large then algo. becomes same as FCFS algo and this performance degrades.
* If time quantum's size is very small then the no. of content switches increases which slow down the overall execution of all the processes or which is not acceptable.

So. the time quantum should not be very large and also should not be too small to achive good system performance.

(2). The no. of content switches should not be too many to slow down the overall execution of all the processes.