20/09/2021. 06. Basics of Process.

Abhishek Sharma Notes.
Ph. 6290903490

Program: A program is a file. It resides on your harddisk. we need a copiler to convert of our program to Binary Program. and this Binary Program exists on your file. If you are using Linux then it is exists as `.OUT` file and in windows it might exist as `.exe` file.

This Binary program is loaded into ram and run by the processor.

Program →

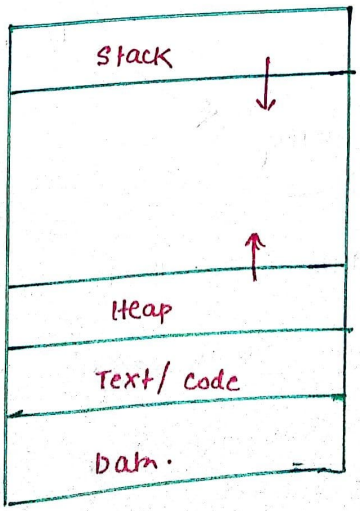
```
int main()
{
cout << "Hello";
}
```

Program.

compiler →

Binary Program.

Process: A process is a program in execution. When a program goes to Ram and when starts running then becomes a Process.



Process.

Abhishek Sharma Notes.
Ph. 6290903490.

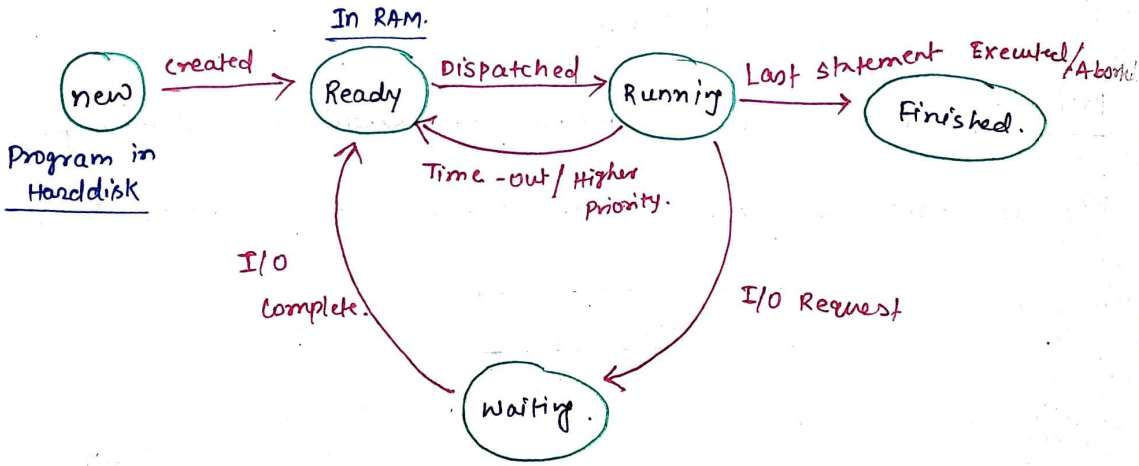
07. Process State.

* Single Tasking System (MS-DOS).

In single tasking system better utilization of computer (you like to task more command) doesn't happen. We can't do multiple process in the Ram. that's why Multi ~~tasking~~ Programming system came into picture.

* Multi ~~tasking~~ Programming System. (5-states Model)

In this system, we can do better utilization of our computer. We can do multiple processes. in RAM other than the OS.

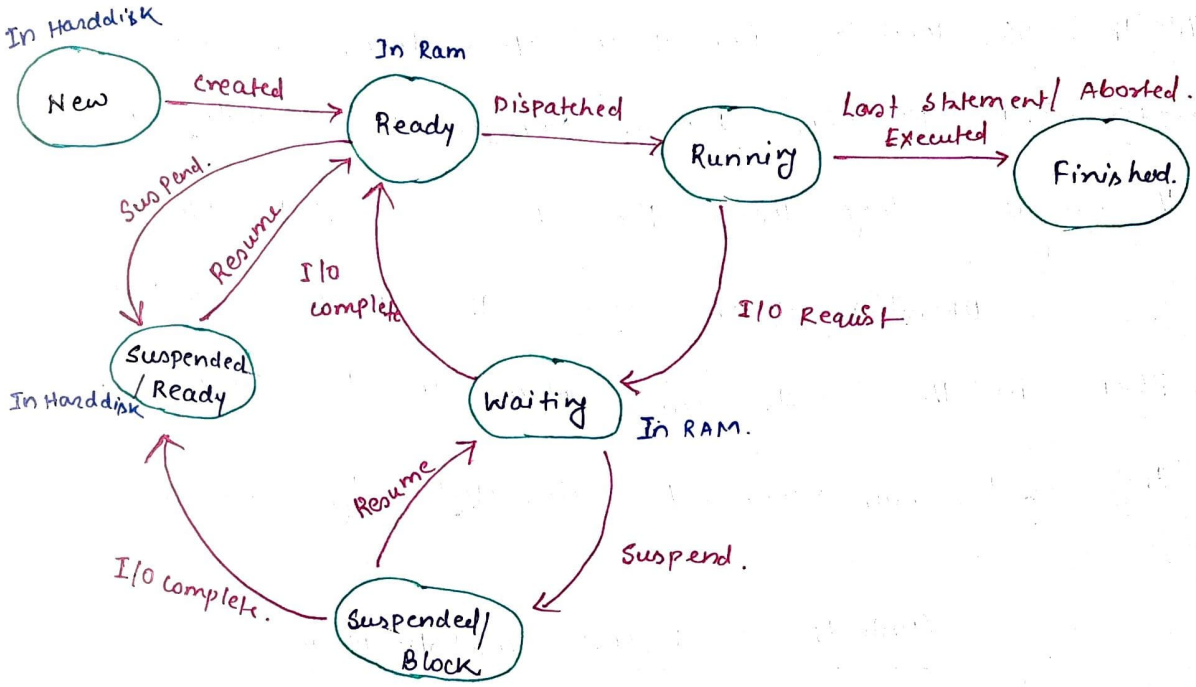


In This 5 state Model first program will be in Harddisk and it goes in the Ram and then it is running if it find any Higher priority or program will Time out then it we return back to Ready area again. After this if program is Running and I/O Request happens then it will wait for it. again it goes to Ready State and Running State and then last statement happed and Program finished. (if anything error msg. happened I/O i.e. then Aborted).

Abhishek Sharma Notes.
Ph: 6290903490.

7 - States Model.

(Multiprogramming System).



In the 7 state model we are moving some of the waiting process to the Harddisk bcz we want to run more processes, we want to increase the degree of Multiprogramming.

If A program's all process I/O is waiting for in the Ram then and if you want to run some other process you take out one of the waiting process from waiting for I/O to Harddisk. So that (suspended/Block state) you can run more processes. (Multiprogramming). If it's I/O is completed then it move to (suspended Ready state.).

And if your RAM is free then it goes to Ready state as usual. Ready process might return back to suspended/Ready (in Harddisk) bcz. it might happen some Higher Priority task which has to Run. This is about 7- States Model.

08. Process Control Block.

Abhishek Sharma Note.
ph. 6290903490.

This is the most central Datastructure of your operating system. All the OS Module i.e. it is I/O module, or memory management Module, Process Management Module, all of the module they use the Process control Block.

Operating System stores the process control Block in the most secure place in the memory. So they ~~can't~~ can never be corrupted by any user Process.

Contents of process control Block:

i) **Process ID:** It is a unique ID assigned to every Process by operating system. It is of integer type. it can be of 32-bit, 64-bit. (Depends upon OS).

ii) **process state:** It might be ready, it might be running we have talked this previously.

All the information related to that ↗ stores in the process state.

iii) **CPU Registers:** : When you have process in memory it might using some CPU Registers when it was running. One important Register is program counter. (which tells the next instruction to be executed).

iv) Account information: Account information is something like how much time CPU has consumed so far. More information related to system can also be stored in this field.

v) I/O Information: What all I/O's it is using. What is the status of the I/O's.

vi) CPU scheduling Information: All the CPU scheduling information will be kept Here.

vii) Memory Information: Where all memory blocks allocated for the process. Where were the memory blocks Where the information is stored of HardDisk. All this informations are stored in Memory Information.

There are many more contents in Process control.

09. Process Scheduler.

When we have movement from one process state to other process state. Process Schedulers do these things.

i) Long Term Scheduler: Brings process from DISK to RAM. Basically it moves process to Ready state. Once the process is came into the RAM. it is in Ready state.

ii) Short Term Scheduler: Brings process from Ready to Running state.

iii) Medium Term Scheduler: Brings process from Waiting state to suspended block state and it also resumes it and it also brings the process from Ready state to suspend/Ready state.

That we seen 2 extra steps in 7-state Model.

10. Background for scheduling Algorithms.

Abhishek Sharma MCA
Ph: 6290903490.

Queues in operating system: There are different Queue in OS. The most imported Queue for scheduling perspective is Ready Queue. Since there are multiple Process in the Ready state. we put them all in a Queue and it is called **Ready Queue**

Job Queue: Queue of jobs which One to be created as Process.

I/O Queue: This Queue is for I/O devices.

Here in scheduling Algorithms we only Read about- **Ready Queue** we only focus about on Ready Queue.

Short Term Scheduler: It's job is to pick one of the processes from Ready Queue. Then Dispatcher comes into the picture.

Dispatcher: It makes to the program jump to the line where to begin with. Exampl. A program might gone in the waiting state then when it resumes it begins from the next line. This work is done by the Dispatcher.

* These Algorithms are only for short term schedules Or for the jobs in the Ready Queue.

When we are talking about scheduling Algorithms, we are

Q) When is short time scheduler called?

Abhishek Sharma Notes.

Ans a) When some other process moves from running to waiting.

b) When some other process moves from running to ready.

c) When a new/existing process move to ready State.
↳ (Higher priority process).

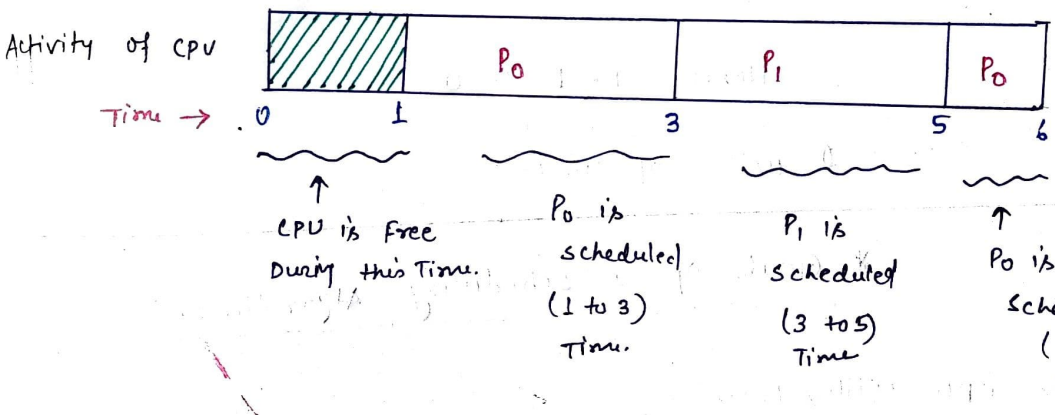
d) When a process terminates.

Different Times in scheduling Algorithms.

Time Frame in CPU is scheduling are generally Represend in DENT chart

In ~~Dent~~ Gantt chart we represent the activity of CPU.

Example: of ~~DENT~~ Gantt CHART.



When we Analysis Process 'P0'

i) Arrival Time : (When P0) is arrived : 1

ii) Completion Time: (When Process P0 is completed) : 6

] Point of Time frame.

iii) Brust Time: (Time of CPU That the Process takes) : 3

Here for P0 CPU takes 1→3 = 2, 5→6 = 1 } = 3 (2+1)

] Difference b/w Two frames.

iv) Turn Arround Time: (Differen b/w completion time and Arrival Time).

= (completion Time − Arrival Time)

= (6 − 1) = 5 unit time. for P0

v) **Waiting Time:** The Time that the process spends in ready Queue.

Here P_0 is waiting when P_1 start processes starts. Hence $(3 \to 5\ P_0)$ is waiting.

= 2 unit of Time.

Formula = Turn Around − Burst Time ⇒ 5−3 = 2

= 2 units of Time for P_0.

vi) **Response Time:** Difference b/w Arrival Time and The first Time when the process gets in CPU.

Difference b/w Two frames.

Here the process P_0 arrived at time 1 and gets into the CPU at time 1.

Hence 1 − 1 = 0

= 0 unit of Time.

---

* Goals of a Scheduling Algorithms.

i) Max CPU utilization

ii) Max Throughput (No. of jobs finished per unit of Time)

iii) Min Turnaround Time. (Discussed earlier).

iv) Min. waiting Time. (" " ")

v) Min. Response Time. (" " ")

vi) Fair CPU Allocation. (No starvation)

(There is a job which is not picked by any sort term scheduler for long long Time. bcz there are many jobs)

11. FCFS Scheduling.

Abhishek Sharma Notes.

Now we will see Different Scheduling Algorithms:

FCFS: FCFS is the simplest scheduling Algorithm. The idea is that the process ↓ comes first is scheduled first.

FCFS is non-~~preemtive~~ premptive Process.

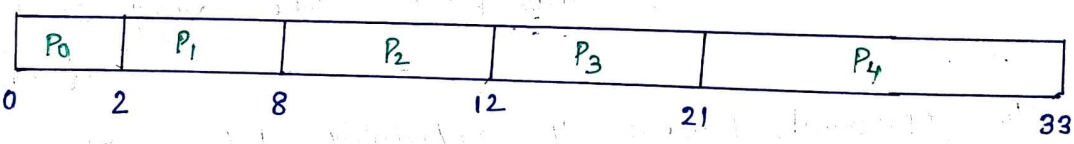
Example: We have some processes like this

Process	Arrival Time.	Burst Time.
P_0	0	2
P_1	1	6
P_2	2	4
P_3	3	9
P_4	4	12

Time for CPU that the process takes.

When you assign process to a processer. You can't turn around back.

Let's Draw Gantt chart for the processes



Now From this information we will complete the Analysis of Algorithm.

Process	Arrival Time	Burst Time (Process Takes)	Completion Time (Process completed)	Turn Around Time (diff b/w C.T − A.T)	Waiting Time (diff b/w T.A.T − B.T)	Avg. Waiting Time	Avg. Turn Around Time
P_0	0	2	2	2	0	~~How much~~ $\frac{(0+1+6+9+17)}{5}$ $= \frac{33}{5}$	$\frac{(2+7+10+8+29)}{5}$ $= \frac{74}{5}$
P_1	1	6	8	7	1		
P_2	2	4	12	10	6		
P_3	3	9	21	18	9		
P_4	4	12	33	29	17		

Avg. Waiting Time: How much Time every Process ~~is~~ spending in the ready Queue. waiting for the processor to get.

Avg. Turn Around Time: How much Time every Process is spending on your system.

* Some Important points about FCFS Algorithm.

i) Simple and easy to impliment.

ii) Non - preemptive. (If you assign a process to the processor you can't take it back).

and This causes certain problem.

i.e. convoy Effect.

iii) **convoy Effect**: (It might happen that when you using FCFS, it might happen that their is cpu Bond Process)

↓

This process takes lot of Time of cpu.

and along with cpu Bond process their might be multiple I/o Bond processes.

↳ This process take very small amount of Time.

As cpu Bond Process is scheduling before I/o Bond. So it will execute procesed first and I/o Bond Process have to wait for the cpu for execution.

This is convoy effect. (Like in convoy, for some V.I.P persons, small persons can have to wait).

This problem occurs in FCFS. Scheduling

iv) Avg. waiting Time: Avg. waiting Time for FCFS might not be good some Time. (If you have cpu Bond Process befor I/o Bond process) bcz you have to wait for more Time.

12. Shortest Job First Algorithms.

Abhishek Sharma Notes.

In FCFS Algo. we have some problem like convoy Effect. (i.e. I/O Bond processes have to wait for CPU Bond processes to finish).

The idea of shortest Job First Algorithms is to consider the jobs in increasing order of their CPU Burst Time.

* We have all the jobs and their Burst Time and we need to consider them in their increasing order of their CPU Burst Time. So that other jobs do not have to wait for too long.

* Their are Two versions of shortest job first

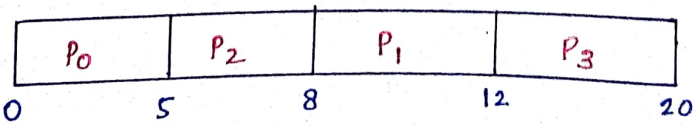
i) premptive

ii) Non - premptive.

Non - Premptive Shortest Job First Algorithms :

Process	Arrival Time	Burst Time.
P_0	0	5
P_1	1	4
P_2	2	3
P_3	3	8

Gantt chart.



This is non-premeptive (we can't go back)

and then P_3 is filled. and so on.

After P_0 we will find which Process has Least Burst Time. that is P_2 (3) So it is filled. , after P_2 we will see which " " " " " " P_1 (4).