SCHEDULING ALGORITHMS

When a computer is multi programmed, it frequently has multiple processes or threads competing for the CPU at the same time. This situation occurs whenever two or more of them are simultaneously in the ready state. If only one CPU is available, a choice has to be made which process to run next. The part of the operating system that makes the choice is called the scheduler, and the algorithm it uses is called the scheduling algorithm

A non-preemptive scheduling algorithm picks a process to run and then just lets it run until it blocks (either on I/O or waiting for another process) or voluntarily releases the CPU. Even if it runs for many hours, it will not be forcibly suspended.

In contrast, a preemptive scheduling algorithm picks a process and lets it run for a maximum of some fixed time. If it is still running at the end of the time interval, it is suspended and the scheduler picks another process to run (if one is available). Doing preemptive scheduling requires having a clock interrupt occur at the end of the time interval to give control of the CPU back to the scheduler. If no clock is available, non-preemptive scheduling is the only option.

Categories of Scheduling Algorithms

1. Batch.

2. Interactive.

3. Real time.

Batch systems are still in widespread use in the business world for doing payroll, inventory, accounts receivable, accounts payable, interest calculation (at banks), claims processing (at insurance companies), and other periodic tasks.

In batch systems, there are no users impatiently waiting at their terminals for a quick response to a short request. Consequently, nonpreemptive algorithms, or preemptive algorithms with long time periods for each process, are often acceptable. This approach reduces process switches and thus improves performance.

The batch algorithms are actually fairly general and often applicable to other situations as well, which makes them worth studying, even for people not involved in corporate mainframe computing.

In an environment with interactive users, preemption is essential to keep one process from hogging the CPU and denying service to the others. Even if no process intentionally ran forever, one process might shut out all the others indefinitely due to a program bug. Preemption is needed to prevent this behavior. Servers also fall into this category, since they normally serve multiple (remote) users, all of whom are in a big hurry. Computer users are always in a big hurry

In systems with real-time constraints, preemption is, oddly enough, sometimes not needed because the processes know that they may not run for long periods of time and usually do their work and block quickly. The difference with interactive systems is that real-time systems run only programs that are intended to further the application at hand. Interactive systems are general purpose and may run arbitrary programs that are not cooperative and even possibly malicious

Scheduling Algorithm Goals

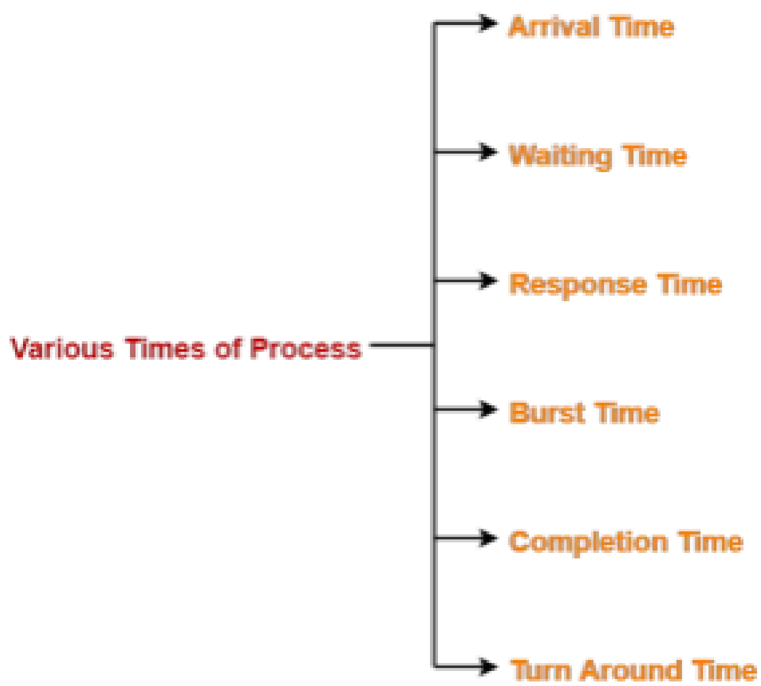
In order to design a scheduling algorithm, it is necessary to have some idea of what a good algorithm should do. Some goals depend on the environment (batch, interactive, or real time), but some are desirable in all cases.

All systems Fair ness - giving each process a fair share of the CPU Policy enforcement - seeing that stated policy is carried out Balance - keeping all parts of the system busy

Batch systems Throughput - maximize jobs per hour Turnaround time - minimize time between submission and termination CPU utilization - keep the CPU busy all the time

Interactive systems Response time - respond to requests quickly Proportionality - meet users' expectations

Real-time systems Meeting deadlines - avoid losing data Predictability - avoid quality degradation in multimedia systems



1. Arrival Time-

- Arrival time is the point of time at which a process enters the ready

queue. **2. Waiting Time-**

- Waiting time is the amount of time spent by a process waiting in the ready queue for getting the CPU.

3. Response Time-

- Response time is the amount of time after which a process gets the CPU for the first time after entering the ready queue.

4. Burst Time-

• Burst time is the amount of time required by a process for executing on CPU. • It is also called as **execution time** or **running time**.

• Burst time of a process can not be known in advance before executing the process.

• It can be known only after the process has executed.

5. Completion Time-

• Completion time is the point of time at which a process completes its execution on the CPU and takes exit from the system.

• It is also called as **exit time**.

6. Turn Around Time-

• Turn Around time is the total amount of time spent by a process in the system. • When present in the system, a process is either waiting in the ready queue for getting the CPU or it is executing on the CPU.

In FCFS Scheduling,

• The process which arrives first in the ready queue is firstly assigned the CPU. • In case of a tie, process with smaller process id is executed first. • It is always non-preemptive in nature

Advantages-

• It is simple and easy to understand.

• It can be easily implemented using queue data structure.

• It does not lead to starvation.

Disadvantages-

• It does not consider the priority or burst time of the processes.

• It suffers from the **convoy effect**.

Convoy Effect

In convoy effect,

• Consider processes with higher burst time arrived before the processes with smaller burst time.

• Then, smaller processes have to wait for a long time for longer processes to release the CPU.

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:

- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Suppose that the processes arrive in the order:

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:

- Waiting time for P_1 = 6; P_2 = 0, P_3 = 3
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- Convoy effect - short process behind long process
  - Consider one CPU-bound and many I/O-bound processes

Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	2
P2	3	1
P3	5	6

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Gantt Chart-

Gantt Chart

Here, black box represents the idle time of CPU.

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	2	2 – 0 = 2	2 – 2 = 0
P2	4	4 – 3 = 1	1 – 1 = 0
P3	11	11- 5 = 6	6 – 6 = 0

SJF Scheduling

In SJF Scheduling,

- Out of all the available processes, CPU is assigned to the process having smallest burst time.
- In case of a tie, it is broken by **FCFS Scheduling**.

SJF Scheduling can be used in both preemptive and non-preemptive mode.

Advantages-

- SRTF is optimal and guarantees the minimum average waiting time.
- It provides a standard for other algorithms since no other algorithm performs better than it.

Disadvantages-

- It can not be implemented practically since the burst time of the processes cannot be known in advance.
- It leads to starvation for processes with larger burst time.

• Priorities cannot be set for the processes.
• Processes with larger burst time have poor response time.

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Also read- Various Times of Process

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 1 = 3
P2	16	16 – 1 = 15	15 – 4 = 11
P3	9	9 – 4 = 5	5 – 2 = 3
P4	6	6 – 0 = 6	6 – 6 = 0
P5	12	12 – 2 = 10	10 – 3 = 7

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart

- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

<u>SHORTEST REMAINING TIME FIRST (SRTF)</u>

> Consider the set of 3 processes whose arrival time and burst time are given below-
>
> | Process Id | Arrival time | Burst time |
> |---|---|---|
> | P1 | 0 | 9 |
> | P2 | 1 | 4 |
> | P3 | 2 | 9 |
>
> If the CPU scheduling policy is SRTF, calculate the average waiting time and average turn around time.

Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 9 = 4
P2	5	5 – 1 = 4	4 – 4 = 0
P3	22	22- 2 = 20	20 – 9 = 11

Now,

- Average Turn Around time = (13 + 4 + 20) / 3 = 37 / 3 = 12.33 unit
- Average waiting time = (4 + 0 + 11) / 3 = 15 / 3 = 5 unit

Round Robin Scheduling

In Round Robin Scheduling,

• CPU is assigned to the process on the basis of FCFS for a fixed amount of time. • This fixed amount of time is called as **time quantum** or **time slice**. • After the time quantum expires, the running process is preempted and sent to the ready queue.

• Then, the processor is assigned to the next arrived process.

• It is always preemptive in nature.

Advantages-

• It gives the best performance in terms of average response time.

• It is best suited for time sharing system, client server architecture and interactive system.

Disadvantages-

• It leads to starvation for processes with larger burst time as they have to repeat the cycle many times.

• Its performance heavily depends on time quantum.

• Priorities can not be set for the processes.

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turn around time.

Gantt Chart-

Ready Queue-

P5, P1, P2, P5, P4, P1, P3, P2, P1

Process Id	Exit time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Now,

- Average Turn Around time = (13 + 11 + 3 + 6 + 10) / 5 = 43 / 5 = 8.6 unit
- Average waiting time = (8 + 8 + 2 + 4 + 7) / 5 = 29 / 5 = 5.8 unit

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.

• Processes with same priority are executed on first come first served basis. • Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Consider the below table to processes with their respective CPU burst times and the priorities.

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,

The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

Types of Priority Scheduling Algorithm

Priority scheduling can be of two types:

1. **Preemptive Priority Scheduling**: If the new process arrived at the ready queue has a higher priority than the currently running process, the CPU is preempted, which means the processing of the current process is stoped and the incoming new process with higher priority gets the CPU for its execution.

2. **Non-Preemptive Priority Scheduling**: In case of non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue, which means after the execution of the current process it will be processed

Problem with Priority Scheduling Algorithm

In priority scheduling algorithm, the chances of **indefinite blocking** or **starvation**.

A process is considered blocked when it is ready to run but has to wait for the CPU as some other process is running currently.

But in case of priority scheduling if new higher priority processes keeps coming in the ready queue then the processes waiting in the ready queue with lower priority may have to wait for long durations before getting the CPU for execution.