OS Process Scheduling

Introduction

In a system, there are a number of processes that are present in different states at a particular time. Some processes may be in the waiting state, others may be in the running state and so on. Have you ever thought how CPU selects one process out of some many processes for execution ? CPU uses some kind of process scheduling algorithms to select one process for its execution amongst so many processes.

The process scheduling algorithms are used to maximise CPU utilisation by increasing throughput.

Some terms

- ▼ Burst time

  Burst time is the total time taken by the process for it's execution on the CPU. *(We ignore the I/O time and consider only the CPU time)*

- ▼ Arrival time

  Arrival time is the time when a process enters into the ready state and is ready for it's execution.

- ▼ Exit time

  Exit time is the time when a process completes its execution and exit from the system.

- ▼ Response time

  Response time is the time spent when the process is in the ready state and gets the CPU for the first time.

  Response time = Time at which the process gets the CPU for the first time - Arrival time

- ▼ Waiting time

  Waiting time is the total time spent by the process in the ready state waiting for CPU.
  Waiting time = Turnaround time - Burst time

- ▼ Turnaround time

  Turnaround time is the total amount of time spent by the process from coming in the ready state for the first time to it's completion.

  Turnaround time = Exit time - Arrival time

- ▼ Throughput

  Throughput is a way to find the efficiency of a CPU. It can be defined as the number of processes executed by the CPU in a given amount of time.

- ▼ Preemptive Scheduling

  In preemptive scheduling, the CPU will execute a process but for a limited period of time and after that, the process has to wait for its next turn i.e. the state of a process gets changed i.e. the process may go to the ready state from running state or from the waiting state to the ready state. The resources are allocated to the process for a limited amount of time and after that, they are taken back and the process goes to the ready queue if it still has some CPU burst time remaining.

- ▼ Non-preemptive Scheduling

  In non-preemptive scheduling, if some resource is allocated to a process then that resource will not be taken back until the completion of the process. Other processes that are present in the ready queue have to wait for its turn and it can't forcefully get the CPU.

- ▼ Convoy Effect

  When a number of relatively-short potential consumers of a resource get queued behind a heavyweight resource consumer. This scheduling scenario might remind you of a single line at a grocery store and what you feel like when you see the person in front of you with three carts full of provisions and a checkbook out; it’s going to be a while.

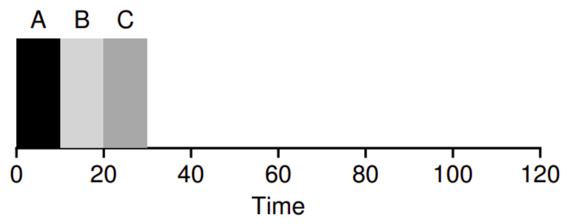
Algorithms

There are *six* popular process scheduling algorithms

▼ First-Come, First-Served (FCFS) Scheduling

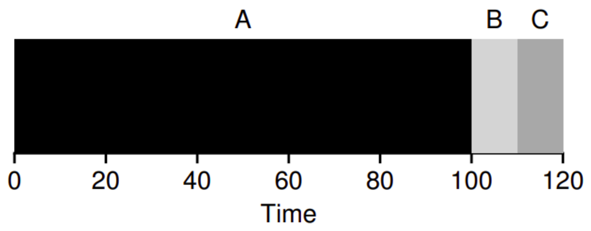
In FCFS scheduling, jobs are executed on first come, first serve basis. It can be either non-preemptive or preemptive. It is clearly simple and thus easy to implement. The implementation is based on FIFO queue.

In certain ideal cases, *FCFS* can work really well, consider 3 processes A, B, C, T_arrival = 0, for all three of them, A arrived just before B, B arrived just before C. Each process runs for say, 10 secs. *Average turnaround time* for these jobs = (10 + 20 + 30)/3 = 20.



Though jobs of different lengths can lead to trouble for FIFO scheduling. In this case the cpu performance is poor, and waiting time is high.

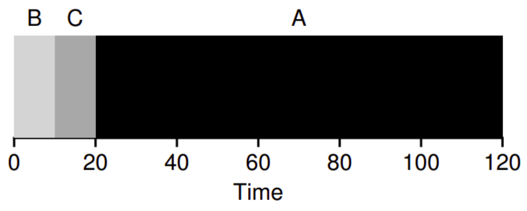
Consider the same processes A, B, C, A arrives before B, B arrives before C, but this time A runs for 100s, B and C run for 10s each. In this case Average turnaround time for these processes = (100 + 110 + 120)/3 = 110.



▼ Shortest Job First (SJF) Scheduling

As the name suggests, it runs the shortest job first, then the next shortest, and so on. t can be either non-preemptive or preemptive. SJF is one of the best approach to minimize waiting time. *Think of any line you have waited in, if the question cares about customer satisfaction, it is likely they have taken SJF into account. For example, grocery stores commonly have a* *"ten-items-or-less" line to ensure that shoppers with only a few things to purchase don't get stuck behind the family preparing for some upcoming nuclear winter.*

Consider the same example with 3 processes, A, B, C, where A arrives before B, B arrives before C, A runs for 100s, B and C runs for 10s each. In case of SJF, B runs before C and C runs before A, average turnaround time = (10 + 20 + 120)/3 = 50.



In SJF scheduling, the processor should know in advance how much time a process will take. Thus, SJF is comparatively easy to implement in batch systems where required CPU time is known in advance. On the other hand, it is impossible to implement SJF in interactive systems where required CPU time is not known.

▼ Priority Scheduling

Priority based 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.

▼ Shortest Remaining Time (SRT) Scheduling

Shortest remaining time (SRT) is the *preemptive* version of the SJF algorithm. The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion. SRT is impossible to implement in interactive systems where required CPU time is not known. It is often used in batch environments where shorter jobs are given preference.

▼ Round Robin (RR) Scheduling

Round Robin Scheduling is another preemptive process scheduling algorithm. Each process is provided a fix time to execute, it is called a **quantum**. Once a process is executed for a given time period, it is preempted and other process executes for a given time period. *Context switching* is used to save states of preempted processes.

▼ Multilevel Queues Scheduling

Multilevel Queues Scheduling is not an independent scheduling algorithm. It makes use of other existing scheduling algorithms to group and schedule jobs with common characteristics. The key points in MLQ are 1. Multiple queues are maintained for processes with common characteristics. 2. Each queue can have its own scheduling algorithms. 3. Priorities are assigned to each queue.

For *example*, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.