MCA Part II

Paper-XIII

Topic: Process Scheduling

Prepared by: Dr. Kiran Pandey
(School of Computer Science)

Email-id: kiranpandey.nou@gmail.com

PROCESS SCHEDULING

Scheduling is a fundamental operating system function. All computer resources are scheduled before use. Since CPU is one of the primary computer resources, its scheduling is central to operating system design. **Scheduling** refers to a set of policies and mechanisms supported by operating system that controls the order in which the work to be done is completed. A **scheduler** is an operating system program (module) that selects the next job to be admitted for execution. The main objective of scheduling is to increase CPU utilization and higher throughput. **Throughput** is the amount of work accomplished in a given **time interval.** CPU scheduling is the basis of operating system which supports multiprogramming concepts. By having a number of programs in computer

memory at the same time, the CPU may be shared among them. This mechanism improves the overall efficiency of the computer system by getting more work done in less time. In this section we will describe the scheduling objectives, the three types of schedulers and performance criteria that schedulers may use in maximizing system performance.

Scheduling Objectives

The primary objective of scheduling is to improve system performance. Various objectives of the scheduling are as follows:

- Be Fair
- 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
- Give better service to processes that have desirable behaviour patterns
- Degrade gracefully under heavy loads

The Operating System maintains the following important process scheduling queues –

- **Job queue** – This queue keeps all the processes in the system.
- **Ready queue** – This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** – The processes which are blocked due to unavailability of an I/O device constitute this queue.

The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below –

S.N.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Types of Schedulers

If we consider batch systems, there will often be more processes submitted than the number of processes that can be executed immediately. So, the incoming processes are spooled onto a disk. There are three types of schedulers. They are:

- Short term scheduler
- Long term scheduler
- Medium term scheduler

Long Term Scheduler

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

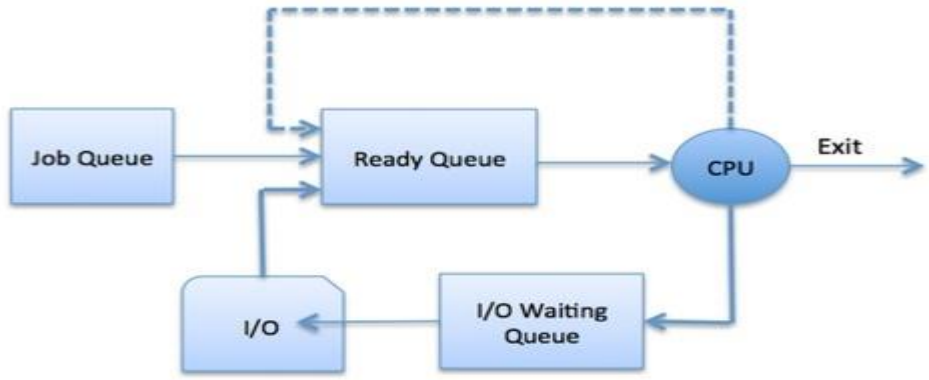


Figure 12: Job Scheduling

Comparison among 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.

Scheduling Criteria

There are many different criteria's to check when considering the "best" scheduling algorithm:

- **CPU utilization**

To make out the best use of CPU and not to waste any CPU cycle, CPU would be working most of the time (Ideally 100% of the time). Considering a real system, CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)

Processor Utilization = (Processor busy time) / (Processor busy time + Processor idle time)

- **Throughput**

  It is the total number of processes completed per unit time or rather say total amount of work done in a unit of time. This may range from 10/second to 1/hour depending on the specific processes.

  Throughput = (No. of processes completed) / (Time unit)

- **Turnaround time**

  It is the amount of time taken to execute a particular process, i.e. The interval from time of submission of the process to the time of completion of the process(Wall clock time).

  Turnaround Time = t(Process completed) – t(Process submitted)

- **Waiting time**

  The sum of the periods spent waiting in the ready queue amount of time a process has been waiting in the ready queue to acquire get control on the CPU.

  Waiting time = Turnaround Time - Processing Time

- **Response time**

  Amount of time it takes from when a request was submitted until the first response is produced. Remember, it is the time till the first response and not the completion of process execution(final response).

  Response time = t(first response) – t(submission of request)

In general CPU utilization and Throughput are maximized and other factors are reduced for proper optimization.

SCHEDULING ALGORITHMS

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated to the CPU. There are several scheduling algorithms which will be examined in this section. A major division among scheduling algorithms is that whether they support **pre-emptive** or **non-preemptive scheduling** discipline.

Preemptive Scheduling means once a process started its execution, the currently running process can be paused for a short period of time to handle some other process of higher priority, it means we can preempt the control of CPU from one process to another if required.

A computer system implementing this supports multi-tasking as it gives the user impression that the user can work on multiple processes.

It is practical because if some process of higher priority is encountered then the current process can be paused to handle that process.

Examples:- SRTF, Priority, Round Robin, etc.

Non-Preemptive Scheduling means once a process starts its execution or the CPU is processing a specific process it cannot be halted or in other words we cannot preempt (take control) the CPU to some other process.

A computer system implementing this cannot support the execution of process in a multi task fashion. It executes all the processes in a sequential manner. It is not practical as all processes are not of same priority and are not always known to the system in advance.

Examples:- FCFS, SJF, Priority, etc.

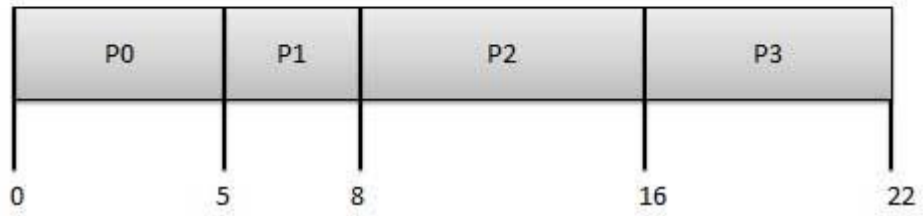
A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter –

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0

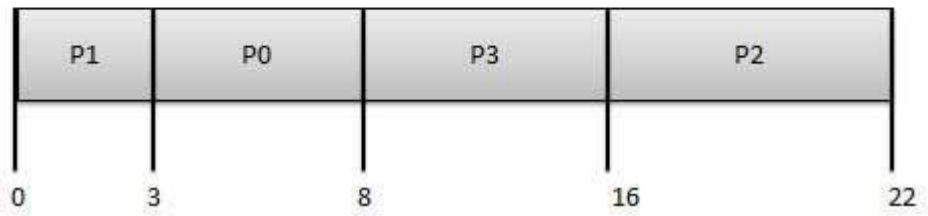
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8



Wait time of each process is as follows –

Process	**Wait Time : Service Time - Arrival Time**

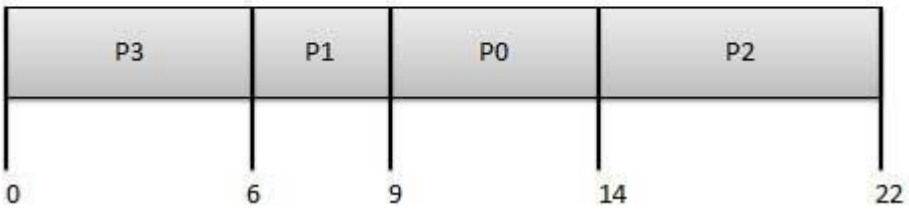
P0	3 - 0 = 3
P1	0 - 0 = 0
P2	16 - 2 = 14
P3	8 - 3 = 5

Average Wait Time: (3+0+14+5) / 4 = 5.50

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.

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Wait time of each process is as follows –

Process	**Wait Time : Service Time - Arrival Time**

P0	9 - 0 = 9
P1	6 - 1 = 5
P2	14 - 2 = 12
P3	0 - 0 = 0

Average Wait Time: (9+5+12+0) / 4 = 6.5

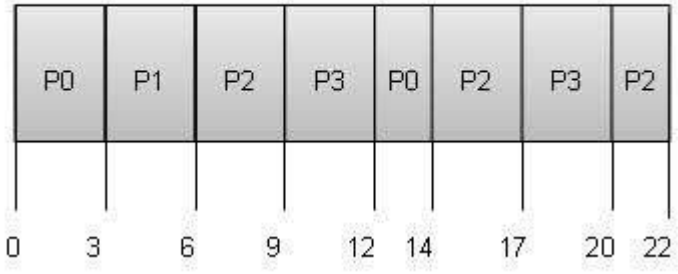
Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN 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.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the 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.

Quantum = 3



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- 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 Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

Performance Evaluation of the Scheduling Algorithms

Performance of an algorithm for a given set of processes can be analysed if the appropriate information about the process is provided. But how do we select a CPU-scheduling algorithm for a particular system? There are many scheduling algorithms so the selection of an algorithm for a particular system can be difficult. To select an algorithm there are some specific criteria such as:

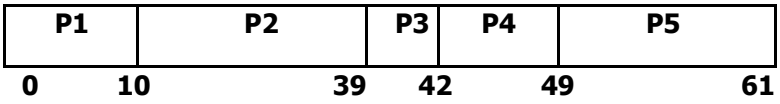
- Maximize CPU utilization with the maximum response time.
- Maximize throughput.

For example, assume that we have the following five processes arrived at time 0, in the order given with the length of CPU time given in milliseconds.

Process	Processing time
P1	10
P2	29
P3	03
P4	07
P5	12

First consider the FCFS scheduling algorithm for the set of processes.

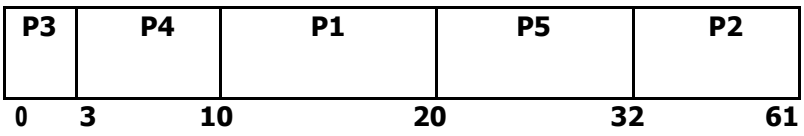
For FCFS scheduling the Gantt chart will be:



Process	Processing time	Waiting time
P1	10	0
P2	29	10
P3	03	39
P4	07	42
P5	12	49

Average Waiting Time: (0+10+39+42+49) / 5 = 28 milliseconds.

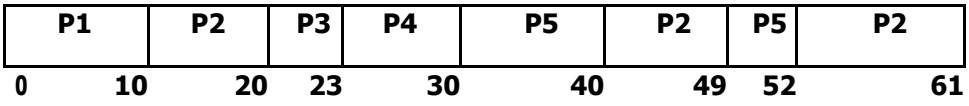
Now consider the SJF scheduling, the Gantt chart will be:



Process	Processing time	Waiting time
P1	10	10
P2	29	32
P3	03	00
P4	07	03
P5	12	20

Average Waiting Time: (10+32+0+3+20)/5 = 13 milliseconds.

Now consider the Round Robin scheduling algorithm with a quantum of 10 milliseconds. The Gantt chart will be:



Process	Processing time	Waiting time
P1	10	0
P2	29	32
P3	03	20
P4	07	23
P5	12	40

Average waiting time = (0+32+20+23+40) / 5 = 23 milliseconds

Now if we compare average waiting time above algorithms, we see that SJF policy results in less than one half of the average waiting time to that of FCFS scheduling; the RR algorithm gives us an intermediate value.

Therefore the performance of algorithm can be measured when all the necessary information is provided.

Multiple-Processor Scheduling

Till now we focused on the problems of scheduling the CPU in a system with a single processor. If multiple CPUs are available, load sharing becomes possible; however, the scheduling problem becomes correspondingly more complex, as we saw with single-processor CPU scheduling, there is no one best solution. Here, we will discuss several concerns in multiprocessor scheduling. We will concentrate on systems in which the processors are identical-homogeneous-in terms of their functionality; we can then use any available processor to run any process in the queue. (However, even with homogeneous multiprocessors, there are sometimes limitations on scheduling. Consider a system with an l/0 device attached to a private bus of one processor. Processes that wish to use that device must be scheduled to run on that processor.)

QUESTIONS

1. What are the basic criteria of scheduling? Discuss.

2. Differentiate between preemptive scheduling and non-preemptive scheduling.

3. Explain the difference in how much the following scheduling algorithms discriminate in favour of short process:

   - FCFS
   - RR

4.Discuss how the following pairs of scheduling criteria conflict in certain settings.

   - CPU utilization and response time
   - Average turnaround time and maximum waiting time.