# INTRODUCTION TO OPERATING SYSTEMS

An Operating System is a program that manages the Computer hardware. It controls and coordinates the use of the hardware among the various application programs for the various users.

A Process is a program in execution. As a process executes, it changes state

- New: The process is being created
- Running: Instructions are being executed
- Waiting: The process is waiting for some event to occur
- Ready: The process is waiting to be assigned to a process
- Terminated : The process has finished execution

Apart from the program code, it includes the current activity represented by

- Program Counter,
- Contents of Processor registers,
- Process Stack which contains temporary data like function parameters, return addresses and local variables
- Data section which contains global variables
- Heap for dynamic memory allocation

A Multi-programmed system can have many processes running simultaneously with the CPU multiplexed among them. By switching the CPU between the processes, the OS can make the computer more productive. There is Process Scheduler which selects the process among many processes that are ready, for program execution on the CPU. Switching the CPU to another process requires performing a state save of the current process and a state restore of new process, this is Context Switch.

**Scheduling Algorithms**

CPU Scheduler can select processes from ready queue based on various scheduling algorithms. Different scheduling algorithms have different properties, and the choice of a particular algorithm may favour one class of processes over another. The scheduling criteria include

- CPU utilization:

- Throughput: The number of processes that are completed per unit time.
- Waiting time: The sum of periods spent waiting in ready queue.
- Turnaround time: The interval between the time of submission of process to the time of completion.
- Response time: The time from submission of a request until the first response is produced.

The different scheduling algorithms are

1. **FCFS: First Come First Serve Scheduling**

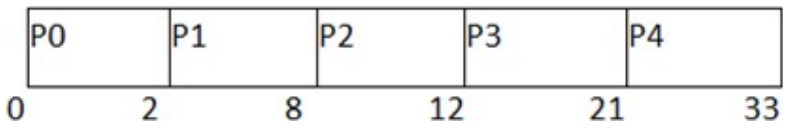
- It is the simplest algorithm to implement.

- The process with the minimal arrival time will get the CPU first.
- The lesser the arrival time, the sooner will the process gets the CPU.
- It is the non-pre-emptive type of scheduling.
- The Turnaround time and the waiting time are calculated by using the following formula.

***Turn Around Time = Completion Time - Arrival Time***
***Waiting Time = Turnaround time - Burst Time***

| Process ID | Arrival Time | Burst Time | Completion Time | Turn Around Time | Waiting Time |
|---|---|---|---|---|---|
| 0 | 0 | 2 | 2 | 2 | 0 |
| 1 | 1 | 6 | 8 | 7 | 1 |
| 2 | 2 | 4 | 12 | 8 | 4 |
| 3 | 3 | 9 | 21 | 18 | 9 |
| 4 | 4 | 12 | 33 | 29 | 17 |

Avg Waiting Time=31/5




**2. SJF: Shortest Job First Scheduling**

- The job with the shortest burst time will get the CPU first.
- The lesser the burst time, the sooner will the process get the CPU.
- It is the non-pre-emptive type of scheduling.
- However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.
- In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

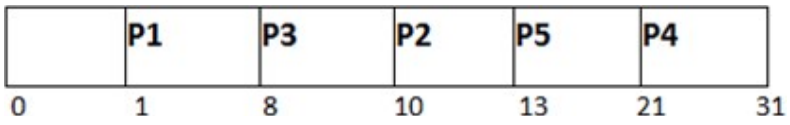
| Process ID | Arrival Time | Burst Time | Completion Time | Turn Around Time | Waiting Time |
|---|---|---|---|---|---|
| 1 | 1 | 7 | 8 | 7 | 0 |
| 2 | 3 | 3 | 13 | 10 | 7 |
| 3 | 6 | 2 | 10 | 4 | 2 |

| | | | | | |
|---|---|---|---|---|---|
| 4 | 7 | 10 | 31 | 24 | 14 |
| 5 | 9 | 8 | 21 | 12 | 4 |

Since, No Process arrives at time 0 hence; there will be an empty slot in the **Gantt chart** from time 0 to 1 (the time at which the first process arrives)

.

- According to the algorithm, the OS schedules the process which is having the lowest burst time among the available processes in the ready queue.
- Till now, we have only one process in the ready queue hence the scheduler will schedule this to the processor no matter what is its burst time.
- This will be executed till 8 units of time.
- Till then we have three more processes arrived in the ready queue hence the scheduler will choose the process with the lowest burst time.
- Among the processes given in the table, P3 will be executed next since it is having the lowest burst time among all the available processes.




Avg Waiting Time = 27/5

**3. SRTF: Shortest Remaining Time First Scheduling**

- It is the pre-emptive form of SJF. In this algorithm, the OS schedules the Job according to the remaining time of the execution

**4. Priority Scheduling**

- In this algorithm, the priority will be assigned to each of the processes.
- The higher the priority, the sooner will the process get the CPU.
- If the priority of the two processes is same then they will be scheduled according to their

arrival time.

### 5. Round Robin Scheduling

- In the Round Robin scheduling algorithm, the OS defines a time quantum (slice).
- All the processes will get executed in the cyclic way.
- Each of the process will get the CPU for a small amount of time (called time quantum) and then get back to the ready queue to wait for its next turn. It is a pre-emptive type of scheduling.

### 6. Multilevel Queue Scheduling

- A multi-level queue scheduling algorithm partitions the ready queue into several separate queues.
- The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.
- Each queue has its own scheduling algorithm.

### 7. Multilevel Feedback Queue Scheduling

- Multilevel feedback queue scheduling, however, allows a process to move between queues.
- The idea is to separate processes with different CPU-burst characteristics.
- If a process uses too much CPU time, it will be moved to a lower-priority queue.
- Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue.
- This form of aging prevents starvation.

## EX.NO.3 SYSTEM CALLS OF UNIX OPERATING SYSTEM

### (a) Stat:

**AIM :**

To Execute a Unix Command in a 'C' program using stat() system call.

**ALGORITHM**:

1. Start the program