PROCESS SCHEDULING

- **CPU scheduling is used in multiprogrammed operating systems.**
- **By switching CPU among processes, efficiency of the system can be improved.**
- **Some scheduling algorithms are FCFS, SJF, Priority, Round-Robin, etc.**
- **Gantt chart provides a way of visualizing CPU scheduling and enables to understand better.**

First Come First Serve (FCFS)

- **Process that comes first is processed first**
- **FCFS scheduling is non-preemptive**
- **Not efficient as it results in long average waiting time.**
- **Can result in starvation, if processes at beginning of the queue have long bursts.**

Shortest Job First (SJF)

- **Process that requires smallest burst time is processed first.**
- **SJF can be preemptive or non–preemptive**
- **When two processes require same amount of CPU utilization, FCFS is used to break the tie.**
- **Generally efficient as it results in minimal average waiting time.**
- **Can result in starvation, since long critical processes may not be processed.**

Priority

- **Process that has higher priority is processed first.**
- **Prioirty can be preemptive or non–preemptive**
- **When two processes have same priority, FCFS is used to break the tie.**
- **Can result in starvation, since low priority processes may not be processed.**

Round Robin

- **All processes are processed one by one as they have arrived, but in rounds.**
- **Each process cannot take more than the time slice per round.**
- **Round robin is a fair preemptive scheduling algorithm.**
- **A process that is yet to complete in a round is preempted after the time slice and put at the end of the queue.**
- **When a process is completely processed, it is removed from the queue.**

Exp# 4a **FCFS Scheduling**

Aim

To schedule snapshot of processes queued according to FCFS (First Come First Serve) scheduling.

Algorithm

1. Define an array of structure *process* with members *pid*, *btime*, *wtime* & *ttime*.
2. Get length of the ready queue, i.e., number of process (say *n*)
3. Obtain *btime* for each process.
4. The *wtime* for first process is 0.
5. Compute *wtime* and *ttime* for each process as:
   a. $wtime_{i+1} = wtime_i + btime_i$
   b. $ttime_i = wtime_i + btime_i$
6. Compute average waiting time *awat* and average turnaround time *atur*
7. Display the *btime*, *ttime* and *wtime* for each process.
8. Display GANTT chart for the above scheduling
9. Display *awat* time and *atur*
10. Stop

Result

Thus waiting time & turnaround time for processes based on FCFS scheduling was computed and the average waiting time was determined.