

School of Engineering and applied Science



Course: Operating System

Scheduling Algorithms

Instructor:

Dr. Sanjay Chaudhary

Mentors:

Aditya Parikh

Pooja Dhamija

Jasnoor Anand

Kaivalya Shah

Group No: 42

Kirtan Gajjar 201501029

Darshan Shah 201501094

Harsh Shah 201501096

Abstract

Nowadays we need our computer to work very fast with more programs (processes) so for that we need to improve processor and our operating system. We can manage processes with use of operating system. A typical process involves both I/O time and CPU time. In a uniprogramming system time spent waiting for I/O is wasted and CPU is free during this time. In multiprogramming systems, one process can use CPU while another is waiting for I/O. This is possible only with process scheduling.

CPU SCHEDULING is a key concept in computer multitasking and multiprocessing operating system. CPU Scheduling is real-time operating system designs. Scheduling refers to the way processes are assigned to run on the available CPUs. CPU scheduling deciding which of the processes in the ready queue is to be allocated the CPU. By switching the CPU among processes, the operating system can make the computer more productive. A multiprogramming operating system allows more than one processes to be loaded into the executable memory at a time and for the loaded processes to share the CPU using time-multiplexing(time slice).

We have created a simulator which performs various algorithms like first come first serve, round robin , shortest job first , priority wise scheduling. we have created threads which performs like a processes and each process contains some instructions to be executed and we are maintaining a log files for each algorithms.

Introduction

Cpu Scheduling is a key concept In computer multitasking, multiprocessing operating system and real time operating system designs. Scheduling refers to the way processes are assigned to run on the available CPUs, since there are typically many more processes running than there are available CPUs.

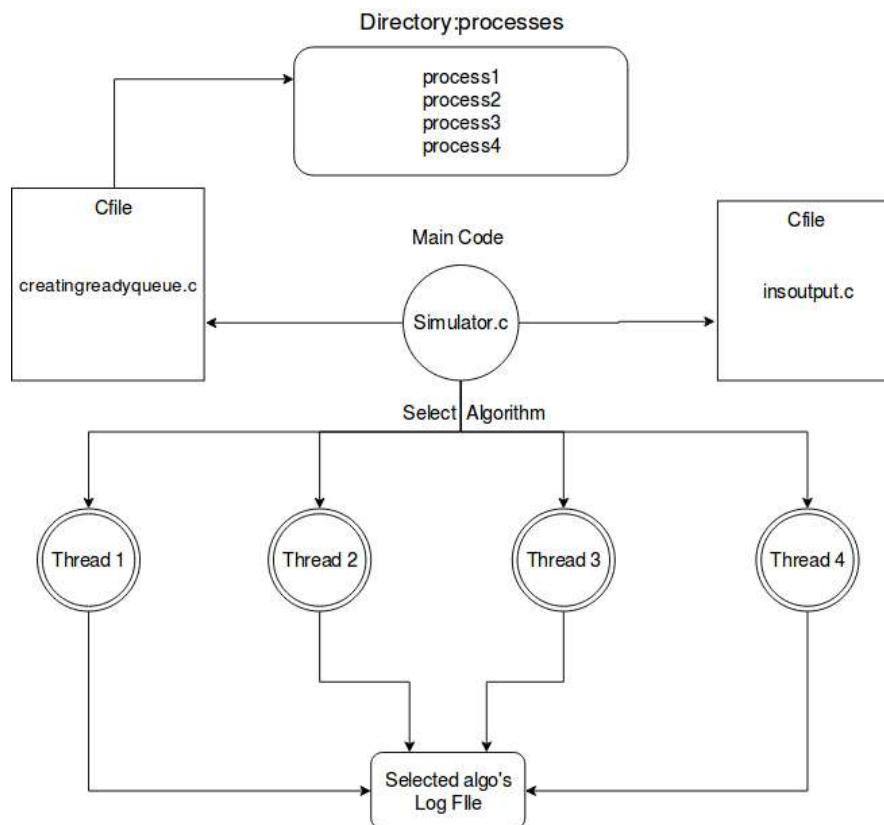
The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular Algorithm. It provides appropriate mechanism so that available resources can be used among all processes in an optimum way. It keeps each resource status and decides control over computer resources.

It handles I/O programming and controls allocation of resources among various users and tasks and during execution it manages those resources also.

The main aim of scheduling algorithms is to focus on following parameters :

- Utilization/Efficiency: To utilize CPU at every time for effective processes.
- Throughput: To maximize the number of jobs processed for some interval of time.
- Turnaround time: To minimize process completion time. It is the amount needed for execution of a single process.
- Waiting time: It is the amount of time a process waits in the ready queue. To minimize process allocation time.
- Cpu burst Time: It is the amount of time process spend in cpu till it completes.
- Response Time: Time from submission till the first response is produced, minimize response time for interactive users

Architecture/model/diagrams



Technical Specifications:

There are three Types of scheduler:

1. Short term Scheduler
2. Middle term Scheduler
3. Long term Scheduler

(i) Short term Scheduler:

It is also called as CPU scheduler. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

(ii) Middle term Scheduler

Medium-term scheduling is a part of swapping. Basically it Schedules processes from suspended (block/ready) state to ready state.

(iii) 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.

Moreover there are two types of scheduling:

1. Non-Preemptive Scheduling
2. Preemptive Scheduling

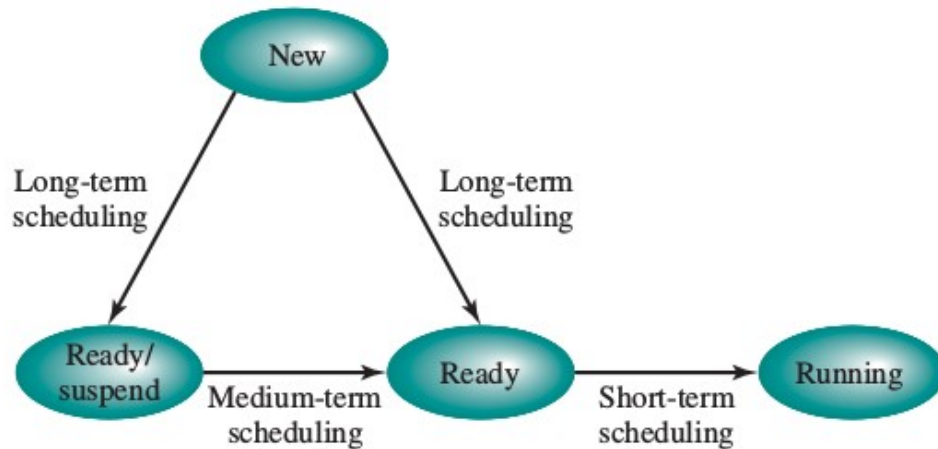
(i) Non-preemptive Scheduling :

In non-preemptive mode, once if a process enters into running state, it continues to execute until it terminates or blocks itself to wait for Input/Output or by requesting some operating system service.

(ii) Preemptive Scheduling :

In preemptive mode, currently running process may be interrupted and moved to the ready State by the operating system.

Among the algorithms that we have implemented FCFS, Priority based scheduling are Non-Preemptive Scheduling algorithms. But Round Robin Scheduling algorithms is Preemptive Scheduling algorithm. SJF scheduling algorithms can be Preemptive or Non-Preemptive algorithm.



In our project we have created a scheduler for four algorithms FCFS, SJF, RR and Priority based scheduling. Initially we have four files(which is like a process) which contains some instructions(like Add, Sub, Mul, Div,Modulo) and we have created threads for each process which means the threads are running as individual processes and the instructions of the processes will work according to the scheduling algorithm.

There are three .c file and one process directory which we made.

.c Files:

- I. simulator.c
- II. creatingreadyqueue.c
- III. insoutput.c .

directory : processes(process1.txt,process2.txt,process3.txt,process4.txt)

Creatingreadyqueue.c :

This file will create ready queue of the any algorithm which user will select. In every algorithm we need ready queue which will create this file. For each process this file create one process control block which will contains following variables (). This file will fetch the processes information from the “processes” directory and will create ready queue for that processes.

Insoutput.c :

This file will give output of process instruction which is stored in any file in the “processes” directory. This file takes instruction as an input parameter and decode that instruction and return the output of that instruction

Simulator.c :

This file is main file of the project which contain all the algorithms which we implemented so user have to select the algorithm from the menu and execute that algorithm. This file contain c program for First Come First Serve (FCFS), Round Robin (RR), shortest job first (SJF) and Priority scheduling algorithm using threads. Threads will execute as user selected algorithm.

Processes Directory :

There are four files in this directory which contains different types of instructions like Add, Sub, Mul, Div and Modulo.

Algorithms

We have one main file which contains all the algorithms and according to user selection selected algorithm will be executed.

Assumptions :

- Each instruction will take one second for execution.
- Time slice for round robin algorithm is 1 second.

First Come First Serve (FCFS):

FCFS is a non preemptive scheduling algorithm. It uses First in- First out- FIFO strategy to assign the priority to processes in the order, that is same as the request made by process for the processor. The process or job that requests the CPU first is allocated the CPU first and other if in the queue has to wait until the CPU is free. Algorithm follows the given step:

Step1: Make an array from ready queue.

Step2: Sort array according to arrival time of the process.

Step3: Take out process (thread) one by one and give it to CPU(insoutput.c).

Step4: Lock CPU till, process is completed.

Step5: Do step3 and step4 until ready queue is null.

Shortest Job First(SJF):

This algorithm is not for implementation purpose because we don't have knowledge of the CPU burst time of any process. In SJF technique the shortest amongst the entire ready queue job is executed first. The benefit if this is that waiting time is minimal for the shorter jobs. The SJF is especially appropriate for the batch jobs for which the run time(cpu_burst) are known in advance.

Algorithm follows the given step:

- Step1: Make an array from ready queue.
- Step2: Sort array according to burst time of the process.
- Step3: Take out process(thread) one by one and give it to CPU(input/output.c)
- Step4: Lock CPU till, process is completed.
- Step5: Do step3 and step4 ready until queue is null.

Priority:

In Priority scheduling algorithm each process is assigned priority by either an outer agency or as per their system requirements and as soon as each process hits the queue it is sorted in based on its priority so that process with higher priority are dealt with first. In case two processes arrive with same priority in different order than they are executed in FCFS order. The main advantage of Priority scheduling is that the important jobs can be finished first. In our case we have taken lower integer value as higher priority i.e 1 has higher priority than 2

Algorithm follows the given step:

- Step1: Make an array from ready queue.
- Step2: Sort array according to the priority of the process.
- Step3: Take out process (thread) one by one and give it to CPU(input/output.c).
- Step4: Lock CPU till, process is completed.
- Step5: Do step3 and step4 until ready queue is null.

Round Robin(RR):

In this approach a fixed time slot is defined before the execution of processes starts, which is a normally small unit of time. In each time slice (quantum) the CPU executes the current process only up to the end of time slice. If that process is having less burst time than the time slice then it is completed and is discarded from the queue and the next process in queue is handled by CPU.

We have created a times array which initially have no. of instructions of each process. Whenever the thread will execute the times array will be decremented by time slice(one second).

Algorithm follows following step:

Step1: Make an array(times) for number of instructions in each process.

Step2: Create thread(process)

Step3: Execute thread for one second(time slice) and then that thread will pause.

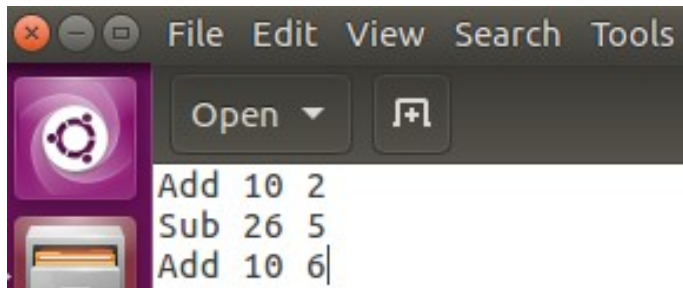
Step4: Decrement the times array by 1 for that particular process.

Step5: Thread will resume after it will get a chance to execute.

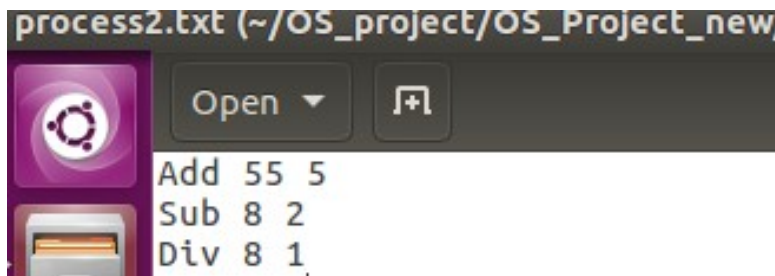
Step6: Do step3, step4,step5 till value of times array is greater than zero.

Test Results:

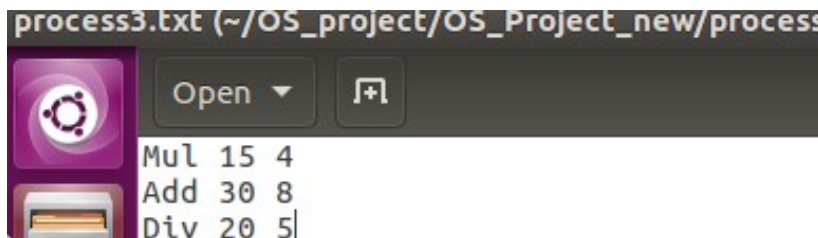
Process 1 instructions



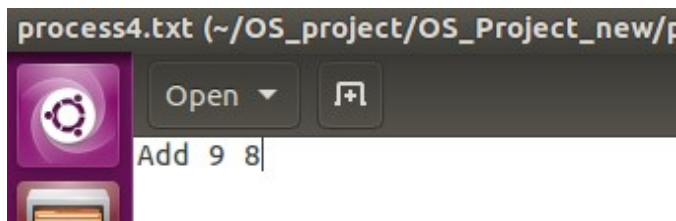
Process 2 instructions



Process 3 instructions

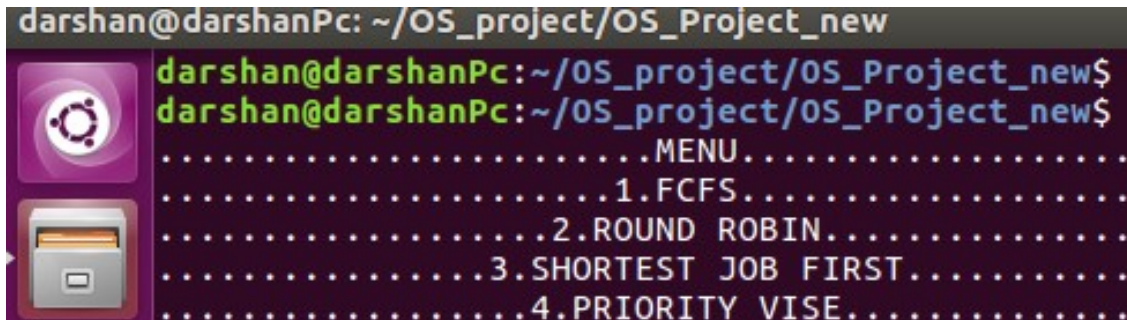


Process 4 Instruction



Main Simulator Menu

```

darshan@darshanPc: ~/OS_project/OS_Project_new

darshan@darshanPc:~/OS_project/OS_Project_new$
darshan@darshanPc:~/OS_project/OS_Project_new$
.....MENU.....
.....1.FCFS.....
.....2.ROUND ROBIN.....
.....3.SHORTEST JOB FIRST.....
.....4.PRIORITY VISE.....

```

FCFS Simulation

```

ENTER YOUR CHOICE
1
processes/process1.txt enters in readyqueue
Process id: 1, Process name:processes/process
Arrival time of this Process:1.000000
Priority of this Process: 3
CPU burst time of this Process:3.000000

processes/process2.txt enters in readyqueue
Process id: 2, Process name:processes/process
Arrival time of this Process:2.000000
Priority of this Process: 3
CPU burst time of this Process:4.000000

processes/process3.txt enters in readyqueue
Process id: 3, Process name:processes/process
Arrival time of this Process:3.000000
Priority of this Process: 9
CPU burst time of this Process:3.000000

processes/process4.txt enters in readyqueue
Process id: 4, Process name:processes/process
Arrival time of this Process:4.000000
Priority of this Process: 6
CPU burst time of this Process:1.000000

output of process1 of processes/process1.txt
output of process2 of processes/process1.txt
output of process3 of processes/process1.txt
processes/process1.txt is completed
output of process1 of processes/process2.txt
output of process2 of processes/process2.txt
output of process3 of processes/process2.txt
output of process4 of processes/process2.txt

```

Round Robin Simulation

```
ENTER YOUR CHOICE
2
processes/process1.txt enters in readyqueue
Process id: 1, Process name:processes/process1
Arrival time of this Process:1.000000
Priority of this Process: 6
CPU burst time of this Process:3.000000

processes/process2.txt enters in readyqueue
Process id: 2, Process name:processes/process2
Arrival time of this Process:2.000000
Priority of this Process: 7
CPU burst time of this Process:4.000000

processes/process3.txt enters in readyqueue
Process id: 3, Process name:processes/process3
Arrival time of this Process:3.000000
Priority of this Process: 0
CPU burst time of this Process:3.000000

processes/process4.txt enters in readyqueue
Process id: 4, Process name:processes/process4
Arrival time of this Process:4.000000
Priority of this Process: 7
CPU burst time of this Process:1.000000

output of process1 of processes/process1.txt i
output of process1 of processes/process2.txt i
output of process1 of processes/process3.txt i
output of process1 of processes/process4.txt i
processes/process4.txt is completed
output of process2 of processes/process1.txt i
output of process2 of processes/process2.txt i
output of process2 of processes/process3.txt i
```


Shortest Job First Simulation

```
ENTER YOUR CHOICE
3
processes/process1.txt enters in readyqueue
Process id: 1, Process name:processes/process1.txt
Arrival time of this Process:1.000000
Priority of this Process: 8
CPU burst time of this Process:3.000000

processes/process2.txt enters in readyqueue
Process id: 2, Process name:processes/process2.txt
Arrival time of this Process:2.000000
Priority of this Process: 4
CPU burst time of this Process:4.000000

processes/process3.txt enters in readyqueue
Process id: 3, Process name:processes/process3.txt
Arrival time of this Process:3.000000
Priority of this Process: 6
CPU burst time of this Process:3.000000

processes/process4.txt enters in readyqueue
Process id: 4, Process name:processes/process4.txt
Arrival time of this Process:4.000000
Priority of this Process: 7
CPU burst time of this Process:1.000000

output of process1 of processes/process4.txt
processes/process4.txt is completed
output of process1 of processes/process3.txt
output of process2 of processes/process3.txt
output of process3 of processes/process3.txt
processes/process3.txt is completed
output of process1 of processes/process1.txt
output of process2 of processes/process1.txt
```

Priority Based Scheduling Simulation

```
ENTER YOUR CHOICE
4
processes/process1.txt enters in readyqueue
Process id: 1, Process name:processes/process1.txt
Arrival time of this Process:1.000000
Priority of this Process: 6
CPU burst time of this Process:3.000000

processes/process2.txt enters in readyqueue
Process id: 2, Process name:processes/process2.txt
Arrival time of this Process:2.000000
Priority of this Process: 1
CPU burst time of this Process:4.000000

processes/process3.txt enters in readyqueue
Process id: 3, Process name:processes/process3.txt
Arrival time of this Process:3.000000
Priority of this Process: 9
CPU burst time of this Process:3.000000

processes/process4.txt enters in readyqueue
Process id: 4, Process name:processes/process4.txt
Arrival time of this Process:4.000000
Priority of this Process: 6
CPU burst time of this Process:1.000000

output of process1 of processes/process2.txt
output of process2 of processes/process2.txt
output of process3 of processes/process2.txt
output of process4 of processes/process2.txt
processes/process2.txt is completed
output of process1 of processes/process1.txt
output of process2 of processes/process1.txt
output of process3 of processes/process1.txt
```


Log File of FCFS

logs_fcfs (~/.OS_project/OS_Project_new) - gedit

| | | | | |
|------------------------|-----------|-------------------------|--------------------------|----------------|
| processes/process1.txt | process's | Arrival Time : 1.000000 | Waiting Time : 0.000000 | cpu_burst_time |
| processes/process2.txt | process's | Arrival Time : 2.000000 | Waiting Time : 3.000000 | cpu_burst_time |
| processes/process3.txt | process's | Arrival Time : 3.000000 | Waiting Time : 7.000000 | cpu_burst_time |
| processes/process4.txt | process's | Arrival Time : 4.000000 | Waiting Time : 10.000000 | cpu_burst_time |

Log File of Round Robin

File Edit View Search Tools Documents Help

| | | | | |
|------------------------|-----------|-------------------------|-----------------------|----------------|
| processes/process4.txt | process's | Arrival Time : 4.000000 | Waiting Time:0.000000 | cpu_burst_time |
| processes/process1.txt | process's | Arrival Time : 1.000000 | Waiting Time:3.000000 | cpu_burst_time |
| processes/process3.txt | process's | Arrival Time : 3.000000 | Waiting Time:1.000000 | cpu_burst_time |
| processes/process2.txt | process's | Arrival Time : 2.000000 | Waiting Time:2.000000 | cpu_burst_time |

Log File of SJF

logs_sjf (~/.OS_project/OS_Project_new) - gedit

| | | | | |
|------------------------|-----------|-----------------------|-----------------------|-------------------------|
| processes/process4.txt | process's | Arrival Time:1.000000 | Waiting Time:0.000000 | cpu_burst_time:1.000000 |
| processes/process3.txt | process's | Arrival Time:2.000000 | Waiting Time:1.000000 | cpu_burst_time:3.000000 |
| processes/process1.txt | process's | Arrival Time:3.000000 | Waiting Time:4.000000 | cpu_burst_time:3.000000 |
| processes/process2.txt | process's | Arrival Time:4.000000 | Waiting Time:7.000000 | cpu_burst_time:4.000000 |

Log File of Priority based Scheduling

logs_priority (~/.OS_project/OS_Project_new) - gedit

| | | | | | |
|------------------------|-----------|------------|-------------------------|-------------------------|----------------|
| processes/process3.txt | process's | Priority:1 | Arrival Time : 3.000000 | Waiting Time : 0.000000 | cpu_burst_time |
| processes/process4.txt | process's | Priority:1 | Arrival Time : 4.000000 | Waiting Time : 3.000000 | cpu_burst_time |
| processes/process1.txt | process's | Priority:5 | Arrival Time : 1.000000 | Waiting Time : 4.000000 | cpu_burst_time |
| processes/process2.txt | process's | Priority:7 | Arrival Time : 2.000000 | Waiting Time : 7.000000 | cpu_burst_time |

Analysis of algorithms:

We have created one file “analysis.c” which will randomly generate burst time, priority for each process and this program will calculate avg. turn around time, waiting time for different algorithms(FCFS, SJF, RR, Priority based) . We have taken no of process=5 and time quantum = 3.

```
Enter total number of process
Enter Time Quantum: 3
```

FCFS Output

```
-----FCFS-----
```

| Process | Burst Time | Waiting Time | Turnar |
|---------|------------|--------------|--------|
| P[1] | 27 | 0 | 27 |
| P[2] | 93 | 27 | 120 |
| P[3] | 91 | 120 | 211 |
| P[4] | 62 | 211 | 273 |
| P[5] | 85 | 273 | 358 |

Average Waiting Time:126

SJF

```
-----SJF-----
```

| Process | Burst Time | Waiting Time | Turnar |
|---------|------------|--------------|--------|
| p[1] | 27 | 0 | 27 |
| p[4] | 62 | 27 | 89 |
| p[5] | 85 | 89 | 174 |
| p[3] | 91 | 174 | 265 |
| p[2] | 93 | 265 | 358 |

Average Waiting Time=111

Priority

```
-----Priority-----
```

| Process | Priority | Burst Time | Waiting Time | Tur |
|---------|----------|------------|--------------|-----|
| P[5] | 11 | 85 | 0 | 85 |
| P[2] | 39 | 93 | 85 | 178 |
| P[3] | 44 | 91 | 178 | 269 |
| P[1] | 45 | 27 | 269 | 296 |
| P[4] | 65 | 62 | 296 | 358 |

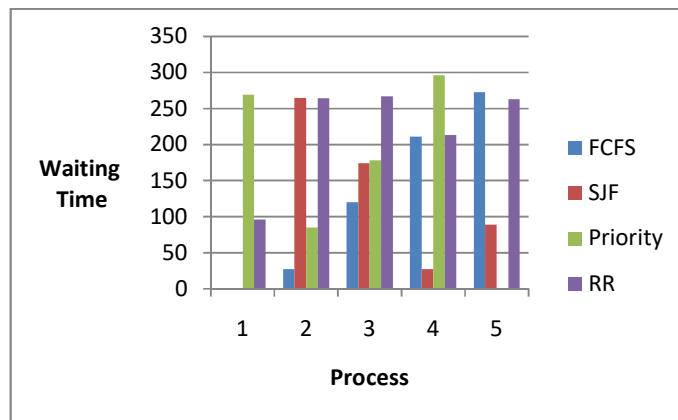
Average Waiting Time=165

RR

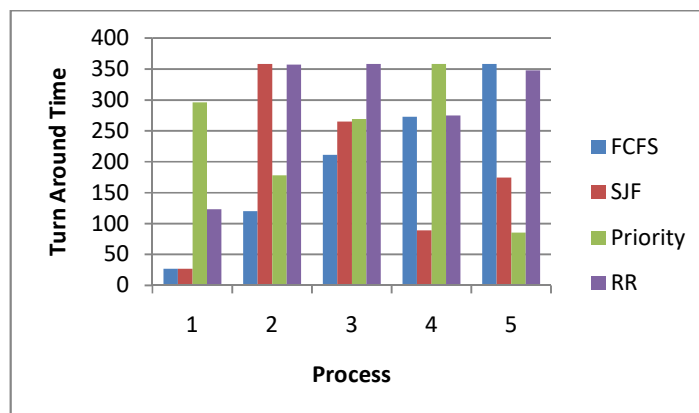
| -----RR----- | | | |
|---------------------------|------------|--------------|------------|
| Process | Burst Time | Waiting Time | Turnaround |
| P[1] | 27 | 96 | 123 |
| P[4] | 62 | 213 | 275 |
| P[5] | 85 | 263 | 348 |
| P[2] | 93 | 264 | 357 |
| P[3] | 91 | 267 | 358 |
| Average Waiting Time= 220 | | | |

Comparison between Various algorithm

(i) Graph Analysis For Waiting time



(ii) Graph Analysis For Turnaround time



➤ Some Observations:

- First Come First Serve is easy to implement and mostly favourable for batch processing systems where waiting time is large but it will not fairly dealing with the processes.
- Short Job First is not applicable because we don't have knowledge of burst time but Short Job First works with optimum scheduling criteria and gives minimum average waiting time.
- The priority scheduling algorithm is based on the priority criteria of execution from highest priority to lowest.
- If we want fair dealing with all process then we need to use Round Robin. Round robin scheduling is preemptive and based on policy of fair dealing of CPU to each process evenly. It works with interactive time sharing system. There is no starvation in Round Robin. Round robin scheduling is generally used now days because it deals with process that is fair and timing of execution of any processes depends on time slice so in dynamic round robin scheduling algorithm we can change time slice according to process load and balance scheduling of process.

References:

- Operating Systems: Internals and Design Principles, 7th Edition by William Stallings
- https://www.tutorialspoint.com/operating_system/os_process_scheduling.htm
- <http://www.geeksforgeeks.org/multithreading-c-2/>
- <https://www.techopedia.com/definition/27857/thread-operating-systems>