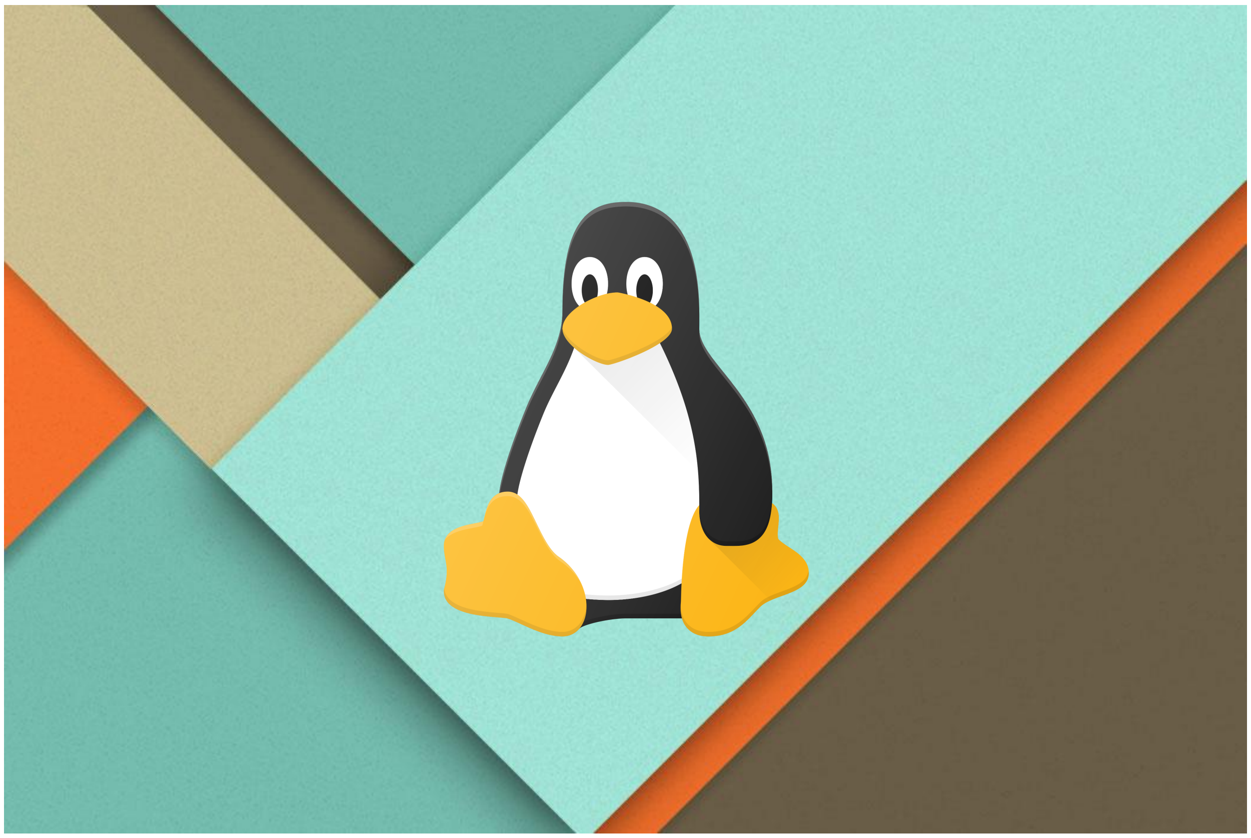


Operating Systems Lab

Experiment No. 5

15.02.2022

Professor - Dr. Shrinivas Khedkar.

Pratham Loya
201080068
IT
prloya_b20@it.vjti.ac.in

Aim

The implementation of below CPU scheduling algorithms in Shell scripting. Take user input for arrival time/ burst time / priority and produce completion time, waiting time, turn around time, average waiting time, average turnaround time and Gantt charts.

1. FCFS Scheduling
2. SJF Scheduling (Non-Preemptive and Preemptive)
3. Non- Preemptive Priority Scheduling
4. Round Robin Scheduling

Make four different scripts and finally combine all four executable scripts in a single script by taking case conditions to execute each.

Theory

The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. The idea is relatively simple.

A process is executed until it must wait, typically for the completion of some I/O request. In a simple computer system, the CPU then just sits idle. All this waiting time is wasted; no useful work is accomplished.

With multiprogramming, we try to use this time productively. Several processes are kept in memory at one time. When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process.

Every time one process has to wait, another process can take over use of the CPU. This is known as CPU scheduling.

CPU-scheduling decisions may take place under the following four circumstances:

1. When a process switches from the running state to the waiting state (for example, as the result of an I/O request or an invocation of wait() for the termination of a child process).
2. When a process switches from the running state to the ready state (for example, when an interrupt occurs)
3. When a process switches from the waiting state to the ready state (for example, at completion of I/O).
4. When a process terminates.

It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time.

In most cases, we optimize the average measure. However, under some circumstances, we prefer to optimize the minimum or maximum values rather than the average.

To fulfill varying needs, we have different scheduling algorithms as mentioned above.

Types of Scheduling

FCFS Scheduling

FCFS strategy assigns priority to processes in the order in which they request the processor. The process that requests the CPU first is allocated the CPU first. When a process comes in, add its PCB to the tail of the ready queue. When the running process terminates, dequeue the process (PCB) at the head of the ready queue and run it.

FCFS.sh

SJF Scheduling (Non-Preemptive and Preemptive)

The SJF algorithm takes processes that use the shortest CPU time first.

SJF_NP.sh

SJF_P.sh

Non- Preemptive Priority Scheduling

A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Priority can be defined either internally or externally. Internally defined priorities use some measurable quantities to compute the priority of a process.

Priority_NP.sh

Round Robin Scheduling

Round-robin scheduling is really the easiest way of scheduling. All processes form a circular array and the scheduler gives control to each process at a time. It is of course very easy to implement and causes almost no overhead, when compared to all other algorithms. But response time is very low for the processes that need it.

Round_Robin.sh

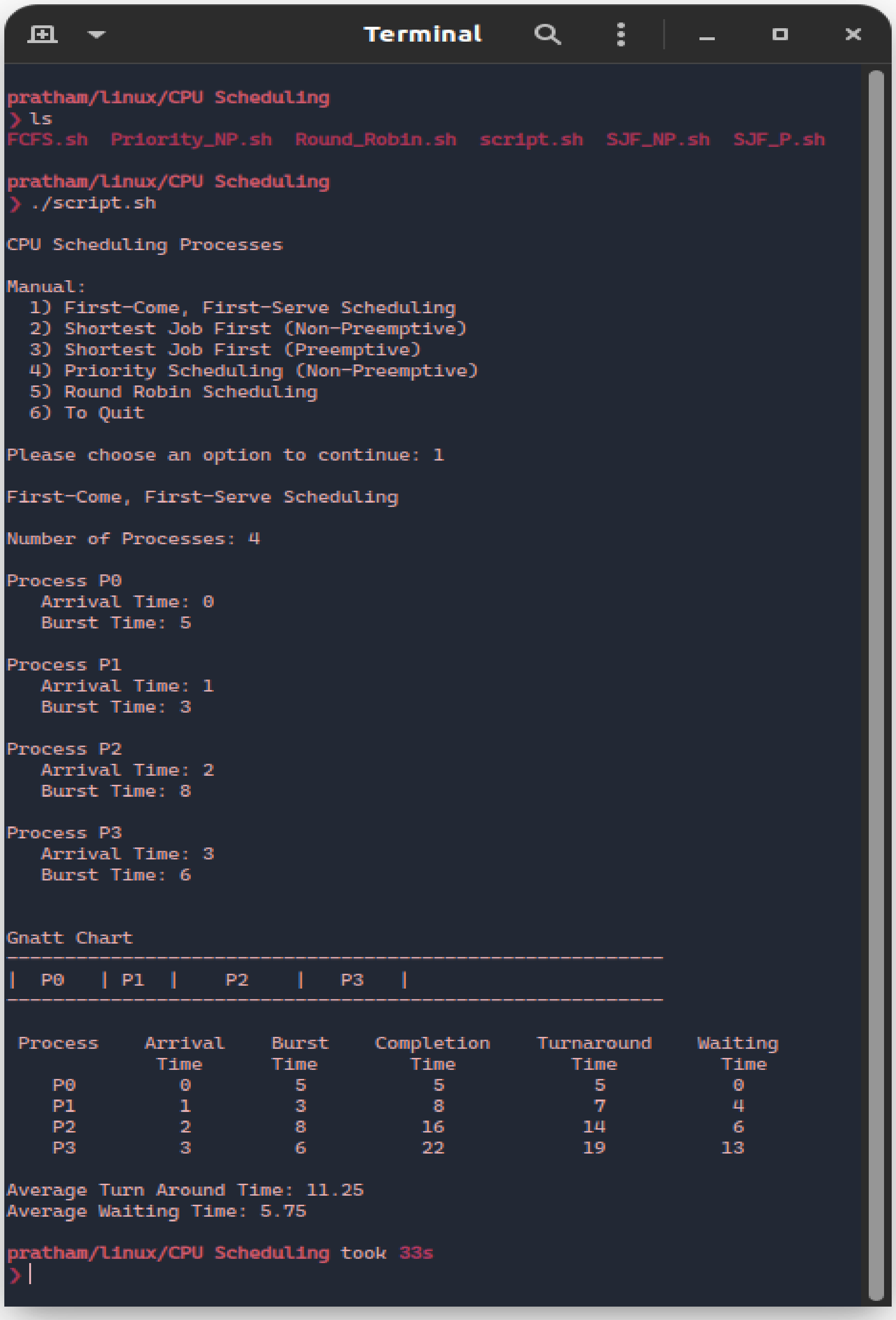
Program

All Individual scripts are linked above and even submitted in assignment.

Final script.sh that binds all the above script is as follows:

```
#! /usr/bin/bash
echo -e "\nCPU Scheduling Processes\n"
echo "Manual: "
   echo "  1) First-Come, First-Serve Scheduling"
   echo "  2) Shortest Job First (Non-Preemptive)"
   echo "  3) Shortest Job First (Preemptive)"
   echo "  4) Priority Scheduling (Non-Preemptive)"
   echo "  5) Round Robin Scheduling"
   echo -e "  6) To Quit\n"
read -p "Please choose an option to continue: " option
case "$option" in
   1 )
       ./FCFS.sh;;
   2 )
       ./SJF_NP.sh;;
   3 )
       ./SJF_P.sh;;
   4 )
       ./Priority_NP.sh;;
   5 )
       ./Round_Robin.sh;;
   6 )
       exit 1;;
   * )
       echo "Invalid Input";;
esac
```

Output



```
pratham/linux/CPU Scheduling
❯ ls
FCFS.sh  Priority_NP.sh  Round_Robin.sh  script.sh  SJF_NP.sh  SJF_P.sh

pratham/linux/CPU Scheduling
❯ ./script.sh

CPU Scheduling Processes

Manual:
  1) First-Come, First-Serve Scheduling
  2) Shortest Job First (Non-Preemptive)
  3) Shortest Job First (Preemptive)
  4) Priority Scheduling (Non-Preemptive)
  5) Round Robin Scheduling
  6) To Quit

Please choose an option to continue: 1

First-Come, First-Serve Scheduling

Number of Processes: 4

Process P0
   Arrival Time: 0
   Burst Time: 5

Process P1
   Arrival Time: 1
   Burst Time: 3

Process P2
   Arrival Time: 2
   Burst Time: 8

Process P3
   Arrival Time: 3
   Burst Time: 6


Gnatt Chart
----------------------------------------------------------
|  P0   | P1  |    P2    |    P3    |
----------------------------------------------------------

 Process    Arrival    Burst    Completion    Turnaround    Waiting
              Time      Time       Time          Time         Time
    P0         0         5          5             5            0
    P1         1         3          8             7            4
    P2         2         8          16            14           6
    P3         3         6          22            19           13

Average Turn Around Time: 11.25
Average Waiting Time: 5.75

pratham/linux/CPU Scheduling took 33s
❯ |
```