
VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

# SWE3001 – Operating Systems Laboratory Manual

# Lab - 06

| | |
|---|---|
| **Student Name** | Sam Prince Franklin |
| **Reg Number** | 20MIS1115 |
| **Subject Code** | SWE3001 |
| **Slot** | L37,38 |
| **Faculty** | Dr.M.Braveen |

**SWE3001 – Operating Systems**

**Lab – 06– CPU Scheduling**

---

Shortest Job First (SJF) with Priority (non-preemptive – with Zero Arrival Time)

```
#include <stdio.h>
struct Process
{
    int pid;
    int priority;
    int burstTime;
    int arrivalTime;
    int turnAroundTime;
    int waitingTime;
    int processFinished;
};
int main()
{
    struct Process p[] = {
        {1, 5, 8},
        {2, 2, 4},
        {3, 4, 6},
        {4, 1, 7},
        {5, 3, 5}};
    int no_of_processes = sizeof(p) / sizeof(p[0]);
    int ganttChart[5];
    int ganttChartProcessesCount = 0;
    int totalCompletionTime = 0;
    for (int i = 0; i < no_of_processes; i++)
    {
        if (i % 2 == 0)
        {
            // do sjf
            int minBurstTime = 10000000;
            int minBTProcessId = -1;
            for (int j = 0; j < no_of_processes; j++)
            {
                if (p[j].processFinished == 0)
```

```
            {
                if (p[j].burstTime < minBurstTime)
                {
                    minBurstTime = p[j].burstTime;
                    minBTProcessId = p[j].pid;
                }
            }
        }
        for (int j = 0; j < no_of_processes; j++)
        {
            if (p[j].pid == minBTProcessId)
            {
                p[j].processFinished = 1;
                ganttChart[ganttChartProcessesCount++] =
                    p[j].pid;
                totalCompletionTime += p[j].burstTime;
                p[j].completionTime = totalCompletionTime;
                p[j].turnAroundTime = p[j].completionTime -
                                p[j].arrivalTime;
                p[j].waitingTime = p[j].turnAroundTime -
                            p[j].burstTime;
            }
        }
    }
    else
    {
        // do priority
        int minPriority = 10000000;
        int minPriorProcessID = -1;
        for (int j = 0; j < no_of_processes; j++)
        {
            if (p[j].processFinished == 0)
            {
                if (p[j].priority < minPriority)
                {
                    minPriority = p[j].priority;
                    minPriorProcessID = p[j].pid;
                }
            }
        }
```

```
            for (int j = 0; j < no_of_processes; j++)
            {
                if (p[j].pid == minPriorProcessID)
                {
                    p[j].processFinished = 1;
                    ganttChart[ganttChartProcessesCount++] =
                        p[j].pid;
                    totalCompletionTime += p[j].burstTime;
                    p[j].completionTime = totalCompletionTime;
                    p[j].turnAroundTime = p[j].completionTime -
                                          p[j].arrivalTime;
                    p[j].waitingTime = p[j].turnAroundTime -
                                       p[j].burstTime;
                }
            }
        }
    }
    // GanttChart Process
    printf("GanttChart Process Sequence: ");
    for (int j = 0; j < sizeof(ganttChart) / sizeof(int); j++)
    {
        printf("%d ", ganttChart[j]);
    }
    printf("\n");
    // Process Table
    printf(" %-10s \t %-8s \t %-10s \t %-12s \t %-15s \t %-16s \t %-12s\n ", " Process ID ", " Priority ", " Burst Time
", " Arrival Time ", "Completion Time", "Turn Around Time", "Waiting Time");
    for (int j = 0; j < no_of_processes; j++)
    {
        printf("%-10d \t %-8d \t %-10d \t %-12d \t %-15d \t %16d \t %-12d\n ", p[j].pid, p[j].priority, p[j].burstTime,
p[j].arrivalTime, p[j].completionTime, p[j].turnAroundTime,p[j].waitingTime);
    }
    // calculate avg turnAroundTime and avg waitingTime
    int totalTurnAroundTime = 0;
    int totalWaitingTime = 0;
    for (int i = 0; i < no_of_processes; i++)
    {
        totalTurnAroundTime += p[i].turnAroundTime;
        totalWaitingTime += p[i].waitingTime;
    }
```

```c
    float avgTurnAroundTime = (float)totalTurnAroundTime / no_of_processes;
    float avgWaitingTime = (float)totalWaitingTime / no_of_processes;
    printf("Average Turn Around Time = %.2f\n",avgTurnAroundTime);
    printf("Average Waiting Time = %.2f", avgWaitingTime);
    return 0;
}
```

```
samprincefranklin@Sams-MacBook-Air Lab % ./sjf_priority
GanttChart Process Sequence: 2 4 5 3 1
  Process ID      Priority        Burst Time      Arrival Time            Completion Time         Turn Around Time        Waiting Time
  1               5               8               0               30                                      30              22
  2               2               4               0               4                                       4               0
  3               4               6               0               22                                      22              16
  4               1               7               0               11                                      11              4
  5               3               5               0               16                                      16              11
  Average Turn Around Time = 16.60
  Average Waiting Time = 10.60
samprincefranklin@Sams-MacBook-Air Lab %
```