Department of
Computer Science and Engineering

Title: Non-preemptive CPU Scheduling Algorithms to find Turnaround Time and Waiting Time.

Operating System Lab

CSE 310

Green University of Bangladesh

1 Objective(s)

- To gather knowledge of different non-preemptive CPU scheduling algorithms.
- To implement First Come First Served (FCFS) and Shortest Job First (SJF) CPU scheduling algorithms.
- To implement Round Robin (RR) and Priority Scheduling algorithms.

2 Problem analysis

Assume that all the processes arrive at the same time.

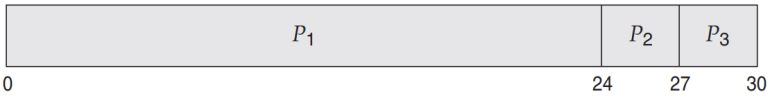
2.1 FCFS CPU Scheduling Algorithm

For FCFS scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. The scheduling is performed on the basis of arrival time of the processes irrespective of their other parameters. Each process will be executed according to its arrival time. Calculate the waiting time and turnaround time of each of the processes accordingly.

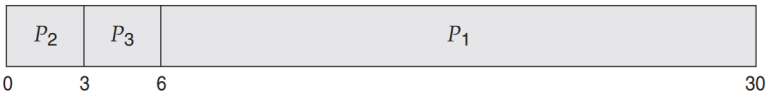
Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	24
P_2	3
P_3	3

If the processes arrive in the order P_1, P_2, P_3, and are served in FCFS order, we get the result shown in the following Gantt chart, which is a bar chart that illustrates a particular schedule, including the start and finish times of each of the participating processes:



The waiting time is 0 milliseconds for process P_1, 24 milliseconds for process P_2, and 27 milliseconds for process P_3. Thus, the average waiting time is $(0 + 24 + 27)/3 = 17$ milliseconds. If the processes arrive in the order P_2, P_3, P_1, however, the results will be as shown in the following Gantt chart:



The average waiting time is now $(6 + 0 + 3)/3 = 3$ milliseconds. This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and may vary substantially if the processes' CPU burst times vary greatly.

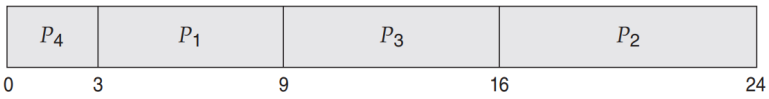
2.2 SJF CPU Scheduling Algorithm

For SJF scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. Arrange all the jobs in order with respect to their burst times. There may be two jobs in queue with the same execution time, and then FCFS approach is to be performed. Each process will be executed according to the length of its burst time. Then calculate the waiting time and turnaround time of each of the processes accordingly.

As an example of SJF scheduling, consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Using SJF scheduling, we would schedule these processes according to the following Gantt chart:



The waiting time is 3 milliseconds for process P_1, 16 milliseconds for process P_2, 9 milliseconds for process P_3, and 0 milliseconds for process P_4. Thus, the average waiting time is $(3 + 16 + 9 + 0)/4 = 7$ milliseconds. By comparison, if we were using the FCFS scheduling scheme, the average waiting time would be 10.25 milliseconds.

3 Implementation in C

```c
/* FCFS CPU Scheduling Algorithm code */
#include<stdio.h>
int main()
{
    int p[20],bt[20],wt[20],tat[20],i,n;
    float wtavg, tatavg;

    printf("Enter the number of processes: ");
    scanf("%d",&n);
    printf("\n");

    for(i=0; i<n; i++)
    {
        printf("Enter the Burst Time: ",i);
        scanf("%d",&bt[i]);
    }

    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];

    for(i=1; i<n; i++)
    {
        wt[i] = tat[i-1];
        tat[i] = bt[i] + wt[i];
        wtavg = wtavg + wt[i];
        tatavg = tatavg + tat[i];
    }

    printf("\nPROCESS\t\tBURST TIME\tWAITING TIME\tTURNAROUND TIME");

    for(i=0; i<n; i++)
    {
        printf("\nP%d \t\t %d \t\t %d \t\t %d ",i,bt[i],wt[i],tat[i]);
    }

    printf("\n\n");
    printf("Average Waiting Time --> %f \n", wtavg/n);

```

```
    printf("\n");
    printf("Average Turnaround Time --> %f \n", tatavg/n);
    printf("\n");

    return 0;
}
```

```
/* SJF CPU Scheduling Algorithm code */
#include<stdio.h>
#include<conio.h>
main()
{
    int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
    float wtavg, tatavg;

    printf("\n  Enter the number of processes: ");
    scanf("%d", &n);
    printf("\n");

    for(i=0; i<n; i++)
    {
        p[i]=i;
        printf("  Enter Burst Time for Process P%d: ", i);
        scanf("%d", &bt[i]);
    }

    for(i=0; i<n; i++)
    {
      for(k=i+1; k<n; k++)
        {
           if(bt[i]>bt[k])
            {
                temp=bt[i];
                bt[i]=bt[k];
                bt[k]=temp;

                temp=p[i];
                p[i]=p[k];
                p[k]=temp;

            }
        }
    }

    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];

    for(i=1; i<n; i++)
    {
        wt[i] = tat[i-1];
        tat[i] = wt[i] +bt[i];
        wtavg = wtavg + wt[i];
        tatavg = tatavg + tat[i];
    }

    printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
```

```
50
51      for(i=0; i<n; i++)
52      {
53          printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
54      }
55
56      printf("\n\n");
57      printf("Average Waiting Time --> %f \n", wtavg/n);
58
59      printf("\n");
60      printf("Average Turnaround Time --> %f \n", tatavg/n);
61      printf("\n");
62
63      getch();
64  }
```

4 Input/Output

Output of the FCFS CPU Scheduling Algorithm program is given below.

```
Enter the number of processes: 3

Enter the Burst Time: 24
Enter the Burst Time: 3
Enter the Burst Time: 3

PROCESS         BURST TIME      WAITING TIME    TURNAROUND TIME
P0               24              0               24
P1               3               24              27
P2               3               27              30

Average Waiting Time --> 17.000000

Average Turnaround Time --> 27.000000
```

Output of the SJF CPU Scheduling Algorithm program is given below.

```
Enter the number of processes: 4

Enter Burst Time for Process P0: 6
Enter Burst Time for Process P1: 8
Enter Burst Time for Process P2: 7
Enter Burst Time for Process P3: 3

       PROCESS        BURST TIME       WAITING TIME    TURNAROUND TIME

       P3              3               0               3
       P0              6               3               9
       P2              7               9               16
       P1              8               16              24

Average Waiting Time --> 7.000000

Average Turnaround Time --> 13.000000
```

5 Discussion & Conclusion

Based on the focused objective(s) to understand about FCFS and SJF CPU scheduling algorithms, the additional lab exercises made me more confident towards the fulfilment of the objectives(s).

6 Lab Task (Please implement yourself and show the output to the instructor)

1. Implement Priority CPU Scheduling Algorithm.
2. Implement Round Robin CPU Scheduling Algorithm.

6.1 Problem analysis

For round robin scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the size of the time slice. Time slices are assigned to each process in equal portions and in circular order, handling all processes execution. This allows every process to get an equal chance. Calculate the waiting time and turnaround time of each of the processes accordingly.

7 Lab Exercise (Submit as a report)

- Implement the following problem using Scheduling Algorithms (Priority > SJF > FCFS).

Table 1: **Sample Input**

Process	**Burst Time**	**Priority**
P1	10	4
P2	13	1
P3	7	3
P4	15	2
P5	6	3
P6	10	4

Table 2: **Sample Output**

Process	**Burst Time**	**Priority**	**Waiting Time**	**Turnaround Time**
P2	13	1	0	13
P4	15	2	13	28
P5	6	3	28	34
P3	7	3	34	41
P1	10	4	41	51
P6	10	4	51	61

8 Policy

Copying from the Internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.