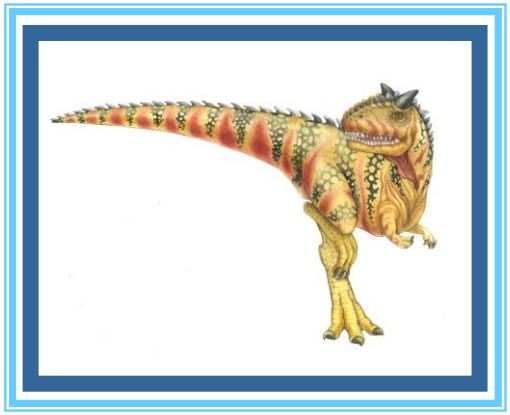
Operating System LAB-02

Prepared By

Md. Mustafizur Rahman

Lecturer, CSE,DCC.



EXPERIMENT NO-2

CPU SCHEDULINGALGORITHMS

SHORTEST JOB FIRST(SJF)

- AIM: To write a program to stimulate the CPU scheduling algorithm Shortest job first (Non- Preemption)

EXPERIMENT NO-2

DESCRIPTION

To calculate the average waiting time in the shortest job first algorithm the sorting of the process based on their burst time in ascending order then calculate the waiting time of each process as the sum of the bursting times of all the process previous or before to that process.

EXPERIMENT NO-2: ALGORITHM

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as _0‘ and its turnaround time as its burst time.

Step 6: Sort the processes names based on their Burt time

EXPERIMENT NO-2: ALGORITHM

Step 7: For each process in the ready queue,

calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)

Step 8: Calculate

c) Average waiting time = Total waiting Time / Number of process

d)Average Turnaround time = Total Turnaround Time / Number of process

Step 9: Stop the process

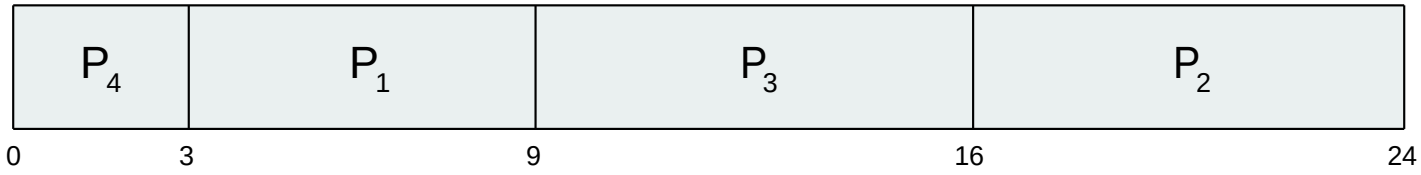
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

Example of SJF

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

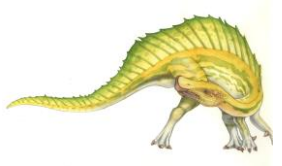
- SJF scheduling chart



- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

EXPERIMENT NO-2: C PROGRAM

```
#include<stdio.h>
int main()
{
int p[20], bt[20], wt[20], tat[20], i, k, n, temp; float wtavg,
tatavg;
printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
p[i]=i;
printf("Enter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);
}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(bt[i]>bt[k])
```



EXPERIMENT NO-2: C PROGRAM

```
{
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] = wt[i-1] +bt[i-1];
tat[i] = wt[i] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}
```

EXPERIMENT NO-2: C PROGRAM

```
printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);
return 0;
}
```

End of Lab 02