**Experiment No: 6**

|  |
| --- |
| **Student Name and Roll Number:** Namit Kumar |
| **Semester /Section:** V/FS -A1 |
| **Link to Code:** |
| **Date: 8**th September 2021 |
| **Faculty Signature:** |
| **Marks:** |

|  |
| --- |
| **Objective:**  Write a program to implement CPU scheduling for shortest job first (Preemptive and Non-Preemptive) |
| **Outcome:**  The students will understand the Shortest Job First scheduling mechanism |
| **Problem Statement:**  Implement the following CPU scheduling Algorithms.   * SJF (Non-Preemptive) * SJTF (shortest remaining time first -Preemptive SJF) |
| **Background Study:**   * Shortest Job first is having the advantage of a minimum average waiting time . * This algorithm associates with each process the length of the process next burst time.When CPU is available it assigned to the process that has the smallest next CPU burst time.if CPU burst time of two process is same then it follows FCFS. * It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing. * It is practically infeasible as Operating System may not know burst time and therefore may not sort them. |
| **Question Bank:**   1. [scheduling algorithm](https://t4tutorials.com/round-robin-process-scheduling-algorithm-in-operating-systems/) In multilevel feedback A. processes are not classified into groups B. a process can move to a different classified ready queue… C. classification of the ready queue is permanent D. none of the mentioned 2. Select one which algorithms tend to minimize the process flow time? A. First come First served B. Earliest Deadline First C. Shortest Job First D. Longest Job First 3. The process can be classified into many groups in A. [shortest job scheduling algorithm](https://t4tutorials.com/shortest-job-first-scheduling-sjf-process-scheduling-in-operating-systems/) B. multilevel queue scheduling algorithm C. round-robin scheduling algorithm D. priority scheduling algorithm 4. The turnaround time for short jobs during multiprogramming is usually Shortened and that for long jobs is slightly \_\_\_\_\_\_\_\_\_\_\_ A. Shortened B. Unchanged C. Lengthened D. Shortened 5. Time quantum can be said A. multilevel queue scheduling algorithm B. round-robin scheduling algorithm C. shortest job scheduling algorithm D. priority scheduling algorithm |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**SJF Non - Preemptive**

#include<stdio.h>

int main()

{

int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;

float avg\_wt,avg\_tat;

printf("Enter number of process:");

scanf("%d",&n);

printf("\nEnter Burst Time:\n");

for(i=0;i<n;i++)

{

printf("p%d:",i+1);

scanf("%d",&bt[i]);

p[i]=i+1;

}

//sorting of burst times

for(i=0;i<n;i++)

{

pos=i;

for(j=i+1;j<n;j++)

{

if(bt[j]<bt[pos])

pos=j;

}

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0;

for(i=1;i<n;i++)

{

wt[i]=0;

for(j=0;j<i;j++)

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=(float)total/n;

total=0;

printf("\nProcesst Burst Time \tWaiting Time\tTurnaround Time");

for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i];

total+=tat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

avg\_tat=(float)total/n;

printf("\n\nAverage Waiting Time=%f",avg\_wt);

printf("\nAverage Turnaround Time=%f\n",avg\_tat);

}

A screenshot of a computer

Description automatically generated with medium confidence

#include <stdio.h>

int main()

{

int arrival\_time[10], burst\_time[10], temp[10];

int i, smallest, count = 0, time, limit;

double wait\_time = 0, turnaround\_time = 0, end;

float average\_waiting\_time, average\_turnaround\_time;

printf("\nEnter the Total Number of Processes:\t");

scanf("%d", &limit);

printf("\nEnter Details of %d Processesn", limit);

for(i = 0; i < limit; i++)

{

printf("\nEnter Arrival Time:\t");

scanf("%d", &arrival\_time[i]);

printf("Enter Burst Time:\t");

scanf("%d", &burst\_time[i]);

temp[i] = burst\_time[i];

}

burst\_time[9] = 9999;

for(time = 0; count != limit; time++)

{

smallest = 9;

for(i = 0; i < limit; i++)

{

if(arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] && burst\_time[i] > 0)

{

smallest = i;

}

}

burst\_time[smallest]--;

if(burst\_time[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest];

turnaround\_time = turnaround\_time + end - arrival\_time[smallest];

}

}

average\_waiting\_time = wait\_time / limit;

average\_turnaround\_time = turnaround\_time / limit;

printf("\n\nAverage Waiting Time:\t%lf\n", average\_waiting\_time);

printf("Average Turnaround Time:\t%lf\n", average\_turnaround\_time);

return 0;

}

Text

Description automatically generated