**EXPERIMENT NO:-5**

**AIM: -** Write a program to implement CPU Scheduling algorithms like FCFS & SJF.

**THEORY:-**

1. **FIRST-COME, FIRST-SERVE SCHEDULING (FCFS):**

In this, which process enter the ready queue first is served first. The OS maintains DS that is ready queue. It is the simplest CPU scheduling algorithm. If a process request the CPU then it is loaded into the ready queue, which process is the head of the ready queue, connect the CPU to that process.

**Algorithm for FCFS scheduling:**

**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:** Set the waiting of the first process as ‘0’ and its burst time as its turn around time

**Step 5:** for each process in the Ready Q calculate

(c) Waiting time for process(n)=waiting time of process (n-1) + Bursttime of process(n-1)

(d) Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

**Step 6:** Calculate

(e) Average waiting time = Total waiting Time / Number of process

(f) Average Turnaround time = Total Turnaround Time / Number of process

**Step 7:** Stop the process

**/\* Program to Simulate First Come First Serve CPU Scheduling Algorithm \*/**

#include<stdio.h>

#include<string.h>

int main()

{

int i,j,n,bt[10],compt[10],at[10], wt[10],tat[10];

float sumwt=0.0,sumtat=0.0,avgwt,avgtat;

printf("Enter number of processes: ");

scanf("%d",&n);

printf("Enter the burst time of %d process\n", n);

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

{

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

}

printf("Enter the arrival time of %d process\n", n);

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

{

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

}

compt[0]=bt[0]-at[0];

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

compt[i]=bt[i]+compt[i-1];

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

{

tat[i]=compt[i]-at[i];

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

sumtat+=tat[i];

sumwt+=wt[i];

}

avgwt=sumwt/n;

avgtat=sumtat/n;

printf("----------------------------------\n");

printf("PN\tBt\tCt\tTat\tWt\n");

printf("----------------------------------\n");

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

{

printf("%d\t%2d\t%2d\t%2d\t%2d\n",i,bt[i],compt[i],tat[i],wt[i]);

}

printf("----------------------------------\n");

printf(" Avgwt = %.2f\tAvgtat = %.2f\n",avgwt,avgtat);

printf("-----------------------------------\n");

return 0;

}

**OUTPUT:**

| https://lh5.googleusercontent.com/CXIFBGTwkcKruSMttBtQE158gO-RZDqMos_mArPBqGG43pyGN15WF46Qi9n326-TUlw-69ZiOVzt8wtwiz3_VGwLbX9T4aMe33kdWrpDluzPPs5Y60Tq4fmBylvjaTbPitgR2YJu |
| --- |

1. **SHORTEST JOB FIRST:**

The criteria of this algorithm are which process having the smallest CPU burst, CPU is assigned to that next process. If two process having the same CPU burst time FCFS is used to break the tie.

**Algorithm for SJF:**

**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:** For each process in the ready queue, calculate

(a) Waiting time for process(n)=waiting time of process (n-1) + Bursttime of process(n-1)

(b) Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

**Step 7:** Calculate

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

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

**Step 8:** Stop the process

**/\* Program to Simulate Shortest Job First CPU Scheduling Algorithm \*/**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

int i,j,n,bt[10],compt[10], wt[10],tat[10],temp;

float sumwt=0.0,sumtat=0.0,avgwt,avgtat;

printf("Enter number of processes: ");

scanf("%d",&n);

printf("Enter the burst time of %d process\n", n);

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

{

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

}

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

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

if(bt[i]>bt[j])

{

temp=bt[i];

bt[i]=bt[j];

bt[j]=temp;

}

compt[0]=bt[0];

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

compt[i]=bt[i]+compt[i-1];

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

{

tat[i]=compt[i];

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

sumtat+=tat[i];

sumwt+=wt[i];

}

avgwt=sumwt/n;

avgtat=sumtat/n;

printf("------------------------------\n");

printf("Bt\tCt\tTat\tWt\n");

printf("------------------------------\n");

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

{

printf("%2d\t%2d\t%2d\t%2d\n",i,bt[i],compt[i],tat[i],wt[i]);

}

printf("------------------------------\n");

printf(" Avgwt = %.2f\tAvgtat = %.2f\n",avgwt,avgtat);

printf("-------------------------------\n");

getch();

}

**OUTPUT:**

| https://lh4.googleusercontent.com/EtGy6IqFO05zSiaU29h-RzzYLxQl5Wj32JMx7ygkKY-nLej8yPymI4rXlw1-OszBbDOidb8X2HelFCiREuA09eXhCu-jZwWB8L5ZmyhTxyeyeUaMagvXiNhNy3NtChwSQjgrE1xu |
| --- |

**CONCLUSION:-** Thus we have studied FCFS & SJF scheduling algorithm and its implementation.