Experiment No:06

Experiment Name: Shortest job First non preemptive scheduling algorithm

Aim and Objectives:

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

* Shortest Job first has the advantage of having minimum average waiting time among all scheduling algorithms.
* It is a Greedy Algorithm.
* It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of aging.
* It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

Code:

#include<bits/stdc++.h>

using namespace std;

int main()

{

int p, i, j, sum=0, min, index;

float awt=0, atat=0;

cout<<"\nEnter The Total Number of Process: ";

cin>>p;

int proc[p];

int \*cbt = new int[p];

int \*wt = new int[p];

int \*gc = new int[p];

int \*tat = new int[p];

int \*tmp = new int[p];

cout<<"\nEnter bursttime of Process:\n";

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

{ cin>>cbt[i];

tmp[i]=cbt[i];

}

sort(cbt, cbt+p);

cout<<"\n========================================================\n";

cout<<"\t\tGantt. Chart";

cout<<"\n========================================================\n";

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

{

min=100;

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

{

if(min>tmp[i]&&tmp[i]!=-1)

{

min=tmp[i];

index=i;

}

}

gc[j]=sum;

wt[j]=sum;

sum+=tmp[index];

tat[j]=sum;

tmp[index]=-1;

if(j==p)

break;

cout<<'P'<<index+1<<" | ";

proc[j]=index+1;

}

cout<<"\n--------------------------------------------------------\n";

sum=0;

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

{

if(gc[j]<10)

cout<<0;

cout<<gc[j]<<" ";

sum+=gc[j];

}

cout<<endl;

atat=(sum\*1.0)/p;

cout<<"\n--------------------------------------------------------";

cout<<"\nProcess\t\tbursttime\tWaiting Time\tTurn Around Time";

cout<<"\n--------------------------------------------------------\n";

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

{

cout<<"P["<<proc[i]<<"]\t\t"<<cbt[i]<<"\t"<<wt[i]<<"\t\t"<<tat[i]<<endl;

awt=awt+wt[i];

}

awt=(awt\*1.0)/p;

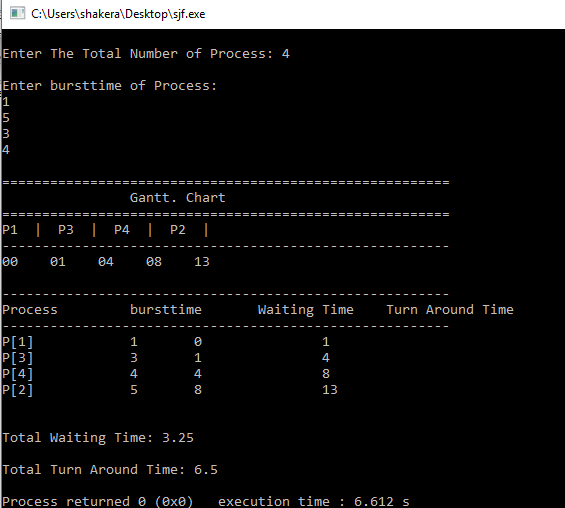
cout<<"\n\nTotal Waiting Time: "<<awt;

cout<<"\n\nTotal Turn Around Time: "<<atat<<endl;

return 0;

}

Output:



Discussion:

In this lab we learn about cpu scheduling sjf. This is the second scheduling that we are learn. SJF is provably optimal; it yields a minimum average waiting time for any set of processes.