

**Operating System (CSE316)**

**Academic Task-3**

**Submitted By:**

V.Naga Vamsi

Reg.No:11717369

Scetion-K17ZH

Roll No: 63

Email :vamsichowdary2616@gmail.com

**Submitted To:**

Mr.Pradeep kumar

**Date of Submission:**

20th April 2019.

**Description**

In question 21 I have used Program for Priority Scheduling Algorithm. In priority scheduling

algorithm each process has a priority associated with it and as each process hits the queue , it

is stored in based on its priority so that process with higher priority are dealt with first . It

should be noted that equal priority processes are scheduled in” FCFS” order.

To prevent high priority process from running indefinitely the scheduler may decrease the

priority of the current running process at each clock interrupt . If this action causes its priority

to drop below that of the next highest process , a process switch occurs . Alternatively ,each

process may be assigned a maximum time quantum that is allowed to run . When this

quantum is used up ,the next highest priority process is given a chance to run.

The problem occurs when the operating system gives a particular task a very low priority, so

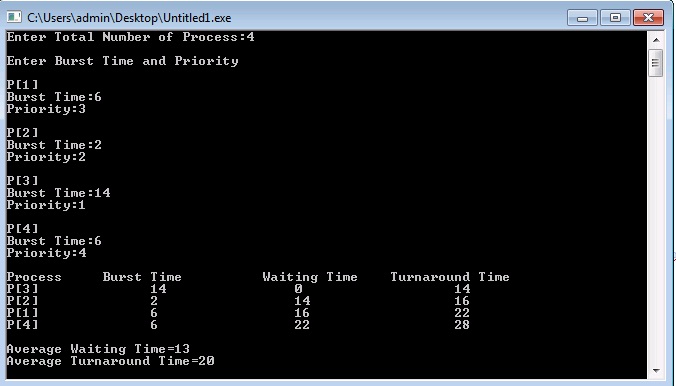
it sits in the queue for a long time ,not being dealt with by the CPU. If the process is

something that user needs , there could be a very long wait ,this process is known as

“Starvation” or “Infinite Blocking”.

**Test Cases**

**Question no.1**



**Question no.21**

#include<iostream>

using namespace std;

int main()

{

    int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg\_wt,avg\_tat;

    cout<<"Enter Total Number of Process:";

    cin>>n;

    cout<<"\nEnter Burst Time and Priority\n";

    for(i=0;i<n;i++)

    {

        cout<<"\nP["<<i+1<<"]\n";

        cout<<"Burst Time:";

        cin>>bt[i];

        cout<<"Priority:";

        cin>>pr[i];

        p[i]=i+1;           //contains process number

    }

    //sorting burst time, priority and process number in ascending order using selection sort

    for(i=0;i<n;i++)

    {

        pos=i;

        for(j=i+1;j<n;j++)

        {

            if(pr[j]<pr[pos])

                pos=j;

        }

        temp=pr[i];

        pr[i]=pr[pos];

        pr[pos]=temp;

        temp=bt[i];

        bt[i]=bt[pos];

        bt[pos]=temp;

        temp=p[i];

        p[i]=p[pos];

        p[pos]=temp;

    }

    wt[0]=0;            //waiting time for first process is zero

    //calculate waiting time

    for(i=1;i<n;i++)

    {

        wt[i]=0;

        for(j=0;j<i;j++)

            wt[i]+=bt[j];

        total+=wt[i];

    }

    avg\_wt=total/n;      //average waiting time

    total=0;

    cout<<"\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time";

    for(i=0;i<n;i++)

    {

        tat[i]=bt[i]+wt[i];     //calculate turnaround time

        total+=tat[i];

        cout<<"\nP["<<p[i]<<"]\t\t  "<<bt[i]<<"\t\t    "<<wt[i]<<"\t\t\t"<<tat[i];

    }

    avg\_tat=total/n;     //average turnaround time

    cout<<"\n\nAverage Waiting Time="<<avg\_wt;

    cout<<"\nAverage Turnaround Time="<<avg\_tat;

    return 0;

}