**Peterson solution for 2 THREAD**

#include<pthread.h>  
#include<stdio.h>  
void \*func1(void \*);  
void \*func2(void \*);  
int flag[2];  
int turn=0;  
int global=100;  
int main()  
{  
    pthread\_t tid1,tid2;  
    pthread\_create(&tid1,NULL,func1,NULL);  
    pthread\_create(&tid2,NULL,func2,NULL);  
    pthread\_join(tid1,NULL);  
    pthread\_join(tid2,NULL);  
}  
  
void \*func1(void \*param)  
{  
    int i=0;  
    while(i<2)  
    {  
        flag[0]=1;  
        turn=1;  
        while(flag[1]==1 && turn==1);  
        global+=100;  
        printf("FT: g: %d",global);  
        flag[0]=0;  
        i++;  
    }  
}  
void \*func2(void \*param)  
{  
    int i=0;  
    while(i<2)  
    {  
        flag[1]=1;  
        turn=0;  
        while(flag[0]==1 && turn==0);  
        global-=75;  
        printf("SP: g: %d",global);  
        flag[1]=0;  
        i++;  
    }  
}

**PRIORITY SCHEDULING**

#include<stdio.h>

int main()

{

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

    printf("Enter Total Number of Process:");

    scanf("%d",&n);

    printf("\nEnter Burst Time and Priority\n");

    for(i=0;i<n;i++)

    {

        printf("\nP[%d]\n",i+1);

        printf("Burst Time:");

        scanf("%d",&bt[i]);

        printf("Priority:");

        scanf("%d",&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;

    printf("\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];

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

    }

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

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

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

return 0;

}

**SHORTEST JOB FIRST ALGORITHM**

#include<stdio.h>

void 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;           //contains process number

    }

    //sorting burst time in ascending order using selection sort

    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;            //waiting time for first process will be 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=(float)total/n;      //average waiting time

    total=0;

    printf("\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];

        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;     //average turnaround time

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

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

}

**ROUND ROBIN ALGORITHM**

#include<stdio.h>

int main()

{

  int count,j,n,time,remain,flag=0,time\_quantum;

  int wait\_time=0,turnaround\_time=0,at[10],bt[10],rt[10];

  printf("Enter Total Process:\t ");

  scanf("%d",&n);

  remain=n;

  for(count=0;count<n;count++)

  {

    printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);

    scanf("%d",&at[count]);

    scanf("%d",&bt[count]);

    rt[count]=bt[count];

  }

  printf("Enter Time Quantum:\t");

  scanf("%d",&time\_quantum);

  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");

  for(time=0,count=0;remain!=0;)

  {

    if(rt[count]<=time\_quantum && rt[count]>0)

    {

      time+=rt[count];

      rt[count]=0;

      flag=1;

    }

    else if(rt[count]>0)

    {

      rt[count]-=time\_quantum;

      time+=time\_quantum;

    }

    if(rt[count]==0 && flag==1)

    {

      remain--;

      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);

      wait\_time+=time-at[count]-bt[count];

      turnaround\_time+=time-at[count];

      flag=0;

    }

    if(count==n-1)

      count=0;

    else if(at[count+1]<=time)

      count++;

    else

      count=0;

  }

  printf("\nAverage Waiting Time= %f\n",wait\_time\*1.0/n);

  printf("Avg Turnaround Time = %f",turnaround\_time\*1.0/n);

  return 0;

}