***Assignment 2 – Scheduling***

**FCFS:**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

typedef struct

{

int pid;

int burst\_time;

int waiting\_time;

int turnaround\_time;

} Process;

void print\_table(Process p[], int n);

void print\_gantt\_chart(Process p[], int n);

int main()

{

Process p[MAX];

int i, j, n;

int sum\_waiting\_time = 0, sum\_turnaround\_time;

printf("Enter total number of process: ");

scanf("%d", &n);

printf("Enter burst time for each process:\n");

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

p[i].pid = i+1;

printf("P[%d] : ", i+1);

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

p[i].waiting\_time = p[i].turnaround\_time = 0;

}

// calculate waiting time and turnaround time

p[0].turnaround\_time = p[0].burst\_time;

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

p[i].waiting\_time = p[i-1].waiting\_time + p[i-1].burst\_time;

p[i].turnaround\_time = p[i].waiting\_time + p[i].burst\_time;

}

// calculate sum of waiting time and sum of turnaround time

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

sum\_waiting\_time += p[i].waiting\_time;

sum\_turnaround\_time += p[i].turnaround\_time;

}

// print table

puts(""); // Empty line

print\_table(p, n);

puts(""); // Empty Line

printf("Total Waiting Time : %-2d\n", sum\_waiting\_time);

printf("Average Waiting Time : %-2.2lf\n", (double)sum\_waiting\_time / (double) n);

printf("Total Turnaround Time : %-2d\n", sum\_turnaround\_time);

printf("Average Turnaround Time : %-2.2lf\n", (double)sum\_turnaround\_time / (double) n);

// print Gantt chart

puts(""); // Empty line

print\_gantt\_chart(p, n);

return 0;

}

void print\_table(Process p[], int n)

{

int i;

puts(" PID Burst Time Waiting Time Turnaround Time ");

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

printf(" %2d %2d %2d %2d \n"

, p[i].pid, p[i].burst\_time, p[i].waiting\_time, p[i].turnaround\_time );

}

}

void print\_gantt\_chart(Process p[], int n)

{

int i, j;

// printing process id in the middle

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

for(j=0; j<p[i].burst\_time - 1; j++) printf(" ");

printf("P%d", p[i].pid);

for(j=0; j<p[i].burst\_time - 1; j++) printf(" ");

printf("|");

}

printf("\n ");

// printing the time line

printf("0");

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

for(j=0; j<p[i].burst\_time; j++) printf(" ");

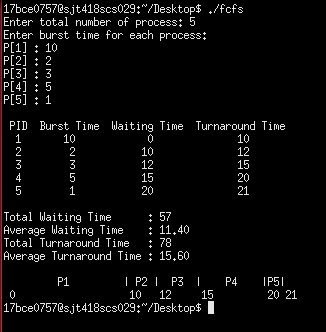
if(p[i].turnaround\_time > 9) printf("\b"); // backspace : remove 1 space

printf("%d", p[i].turnaround\_time);

}

printf("\n");

}



**SJF:**

#include<stdio.h>

struct process

{

int pid;

int waiting;

int turnaround;

int burst;

int es; //execution start

int ee; //execution end

};

struct process read(struct process p,int i)

{

    p.pid = i+1;

    printf("P%d : \nEnter the burst time : ",p.pid);

    scanf("%d",&p.burst);

    p.es = 0;

    p.ee = 0;

    return p;

}

void sort(struct process arr[],int n)

{

    int i,j,minp;

    struct process min,temp;

    for(i=0;i<n;i++)

    {

        min = arr[i];

        minp = i;

        for(j=i+1;j<n;j++)

        {

            if(arr[j].burst<min.burst)

            {

                min = arr[j];

                minp = j;

            }

        }

        arr[minp] = arr[i];

        arr[i] = min;

    }

}

int main()

{

    int n,i;

    printf("Enter number of processes to consider : ");

    scanf("%d",&n);

    struct process arr[n];

    for(i=0;i<n;i++)

     arr[i] = read(arr[i],i);

    sort(arr,n);

    int time=0;

    for(i=0;i<n;i++)

    {

        arr[i].es = time;

        time += arr[i].burst;

        arr[i].ee = time;

    }

    for(i=0;i<n;i++)

    {

        arr[i].waiting = arr[i].es;

        arr[i].turnaround = arr[i].ee;

    }

    printf("\n\nPID\tBurst\tStart\tEnd\tWaiting\tTurnaround\n");

    for(i=0;i<n;i++)

     printf("%d\t%d\t%d\t%d\t%d\t%d\n",arr[i].pid,arr[i].burst,arr[i].es,arr[i].ee,arr[i].waiting,arr[i].turnaround);

    int totalwait=0,totalturn=0;

    for(i=0;i<n;i++)

    {

        totalwait += arr[i].waiting;

        totalturn += arr[i].turnaround;

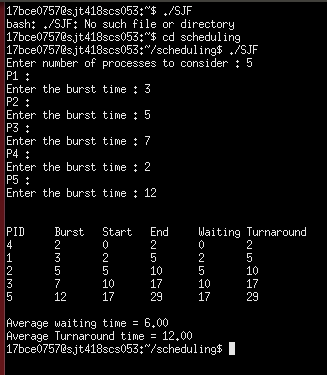
    }

    float avgwait = totalwait/n;

    float avgturn = totalturn/n;

    printf("\nAverage waiting time = %.2f\nAverage Turnaround time = %.2f\n",avgwait,avgturn);

}



**SRTF:**

#include <stdio.h>

int main()

{

int a[10],b[10],x[10],i,j,smallest,count=0,time,n;

double avg=0,tt=0,end;

printf("enter the number of Processes:\n");

scanf("%d",&n);

printf("enter arrival time\n");

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

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

printf("enter burst time\n");

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

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

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

x[i]=b[i];

b[9]=9999;

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

{

smallest=9;

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

{

if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )

smallest=i;

}

b[smallest]--;

if(b[smallest]==0)

{

count++;

end=time+1;

avg=avg+end-a[smallest]-x[smallest];

tt= tt+end-a[smallest];

}

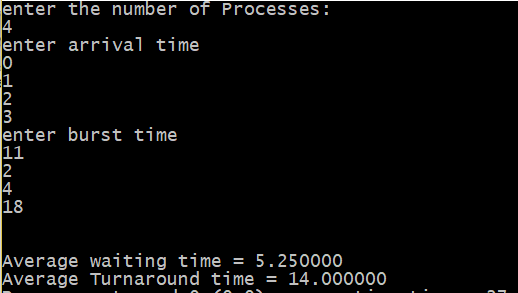
}

printf("\n\nAverage waiting time = %lf\n",avg/n);

printf("Average Turnaround time = %lf",tt/n);

return 0;

}



**Priority Scheduling:**

#include<stdio.h>

typedef struct

{

int pid;

int burst\_time;

int waiting\_time;

int turnaround\_time;

int priority;

} process;

void print\_table(process p[], int n)

{

int i;

puts(" PID Burst Time Waiting Time Turnaround Time ");

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

printf(" %2d\t%2d\t%2d\t%2d\t%2d\n"

, p[i].priority, p[i].pid, p[i].burst\_time, p[i].waiting\_time, p[i].turnaround\_time );

}

}

void print\_gantt\_chart(process p[], int n)

{

int i, j;

// printing process id in the middle

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

for(j=0; j<p[i].burst\_time - 1; j++) printf(" ");

printf("P%d", p[i].pid);

for(j=0; j<p[i].burst\_time - 1; j++) printf(" ");

printf("|");

}

printf("\n ");

// printing the time line

printf("0");

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

for(j=0; j<p[i].burst\_time; j++) printf(" ");

if(p[i].turnaround\_time > 9) printf("\b"); // backspace : remove 1 space

printf("%d", p[i].turnaround\_time);

}

printf("\n");

}

void sort(process arr[],int n)

{

    int minp,i,j;

    process min;

    for(i=0;i<n;i++)

    {

        min = arr[i];

        minp = i;

        for(j=i+1;j<n;j++)

        {

            if(arr[j].priority < min.priority)

            {

                min = arr[j];

                minp = j;

            }

        }

        arr[minp] = arr[i];

        arr[i] = min;

    }

}

void read(process \*a,int i)

{

    a->pid = i+1;

    printf("Enter burst time : ");

    scanf("%d",&a->burst\_time);

    printf("Enter priority : ");

    scanf("%d",&a->priority);

}

int main()

{

    int n,i;

    printf("Enter number of processes to consider : ");

    scanf("%d",&n);

    process arr[n];

    for(i=0;i<n;i++)

     read(&arr[i],i);

    sort(arr,n);

    int time = 0;

    for(i=0;i<n;i++)

    {

        arr[i].waiting\_time = time;

        time += arr[i].burst\_time;

        arr[i].turnaround\_time = time;

    }

    int sum\_waiting\_time=0,sum\_turnaround\_time=0;

    for(i=0; i<n; i++)

    {

sum\_waiting\_time += arr[i].waiting\_time;

sum\_turnaround\_time += arr[i].turnaround\_time;

    }

    // print table

    puts(""); // Empty line

    print\_table(arr, n);

    puts(""); // Empty Line

    printf("Total Waiting Time : %-2d\n", sum\_waiting\_time);

    printf("Average Waiting Time : %-2.2lf\n", (double)sum\_waiting\_time / (double) n);

    printf("Total Turnaround Time : %-2d\n", sum\_turnaround\_time);

    printf("Average Turnaround Time : %-2.2lf\n", (double)sum\_turnaround\_time / (double) n);

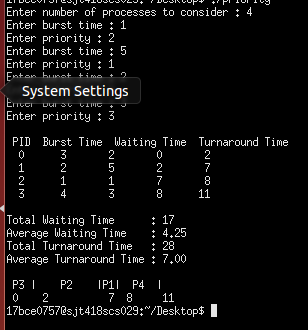
    // print Gantt chart

    puts(""); // Empty line

    print\_gantt\_chart(arr, n);

    return 0;

}

****

**Preemptive priority:**

#include<stdio.h>

struct process

{

char process\_name;

int arrival\_time, burst\_time, ct, waiting\_time, turnaround\_time, priority;

int status;

}process\_queue[10];

int limit;

void Arrival\_Time\_Sorting()

{

struct process temp;

int i, j;

for(i = 0; i < limit - 1; i++)

{

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

{

if(process\_queue[i].arrival\_time > process\_queue[j].arrival\_time)

{

temp = process\_queue[i];

process\_queue[i] = process\_queue[j];

process\_queue[j] = temp;

}

}

}

}

void main()

{

int i, time = 0, burst\_time = 0, largest;

char c;

float wait\_time = 0, turnaround\_time = 0, average\_waiting\_time, average\_turnaround\_time;

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

scanf("%d", &limit);

for(i = 0, c = 'A'; i < limit; i++, c++)

{

process\_queue[i].process\_name = c;

printf("\nEnter Details For Process[%C]:\n", process\_queue[i].process\_name);

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

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

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

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

printf("Enter Priority:\t");

scanf("%d", &process\_queue[i].priority);

process\_queue[i].status = 0;

burst\_time = burst\_time + process\_queue[i].burst\_time;

}

Arrival\_Time\_Sorting();

process\_queue[9].priority = -9999;

printf("\nProcess Name\tArrival Time\tBurst Time\tPriority\tWaiting Time");

for(time = process\_queue[0].arrival\_time; time < burst\_time;)

{

largest = 9;

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

{

if(process\_queue[i].arrival\_time <= time && process\_queue[i].status != 1 && process\_queue[i].priority > process\_queue[largest].priority)

{

largest = i;

}

}

time = time + process\_queue[largest].burst\_time;

process\_queue[largest].ct = time;

process\_queue[largest].waiting\_time = process\_queue[largest].ct - process\_queue[largest].arrival\_time - process\_queue[largest].burst\_time;

process\_queue[largest].turnaround\_time = process\_queue[largest].ct - process\_queue[largest].arrival\_time;

process\_queue[largest].status = 1;

wait\_time = wait\_time + process\_queue[largest].waiting\_time;

turnaround\_time = turnaround\_time + process\_queue[largest].turnaround\_time;

printf("\n%c\t\t%d\t\t%d\t\t%d\t\t%d", process\_queue[largest].process\_name, process\_queue[largest].arrival\_time, process\_queue[largest].burst\_time, process\_queue[largest].priority, process\_queue[largest].waiting\_time);

}

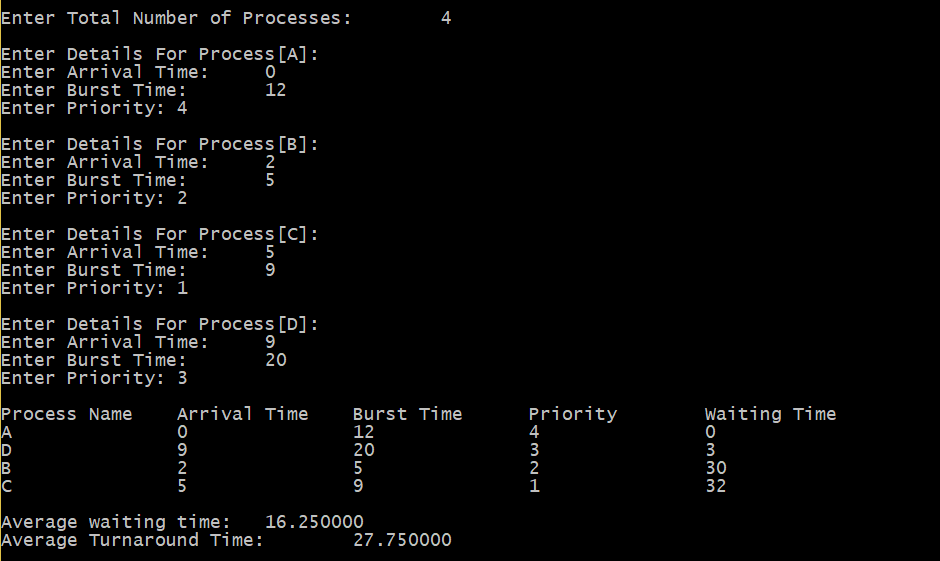
average\_waiting\_time = wait\_time / limit;

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

printf("\n\nAverage waiting time:\t%f\n", average\_waiting\_time);

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

}



**Round Robin:**

#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;

}

