**Assignment 7**

**Aim –** Simulate the following scheduling algorithm using c language.

* FCFS (Non-preemptive by default)
* SRTN (Preemptive version of SJF)
* Priority (Non-preemptive)
* Round Robin (Preemptive by default)
* Code :
* FCFS (Non-preemptive by default)

#include<stdio.h>

int main()

{

int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;

printf("Enter total number of processes(maximum 20):");

scanf("%d",&n);

printf("\nEnter Process Burst Time\n");

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

{

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

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

}

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

//calculating waiting time

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

{

wt[i]=0;

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

wt[i]+=bt[j];

}

printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");

//calculating turnaround time

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

{

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

avwt+=wt[i];

avtat+=tat[i];

printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);

}

avwt/=i;

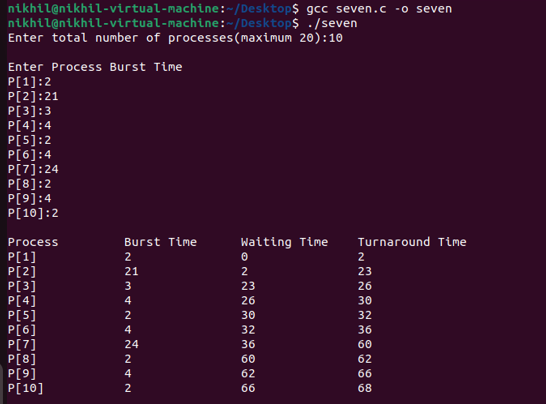
avtat/=i;

printf("\n\nAverage Waiting Time:%d",avwt);

printf("\nAverage Turnaround Time:%d",avtat);

return 0;

}



SRTN (Preemptive version of SJF)

#include <stdio.h>

int main()

{

int A[100][4]; // Matrix for storing Process Id, Burst

// Time, Average Waiting Time & Average

// Turn Around Time.

int i, j, n, total = 0, index, temp;

float avg\_wt, avg\_tat;

printf("Enter number of process: ");

scanf("%d", &n);

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

// User Input Burst Time and alloting Process Id.

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

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

scanf("%d", &A[i][1]);

A[i][0] = i + 1;

}

// Sorting process according to their Burst Time.

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

index = i;

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

if (A[j][1] < A[index][1])

index = j;

temp = A[i][1];

A[i][1] = A[index][1];

A[index][1] = temp;

temp = A[i][0];

A[i][0] = A[index][0];

A[index][0] = temp;

}

A[0][2] = 0;

// Calculation of Waiting Times

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

A[i][2] = 0;

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

A[i][2] += A[j][1];

total += A[i][2];

}

avg\_wt = (float)total / n;

total = 0;

printf("P BT WT TAT\n");

// Calculation of Turn Around Time and printing the

// data.

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

A[i][3] = A[i][1] + A[i][2];

total += A[i][3];

printf("P%d %d %d %d\n", A[i][0],

A[i][1], A[i][2], A[i][3]);

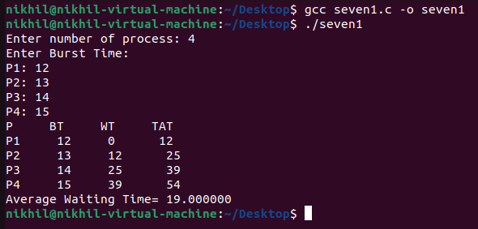
}

avg\_tat = (float)total / n;

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

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

}

****

Priority (Non-preemptive)

#include <stdio.h>

//Function to swap two variables

void swap(int \*a,int \*b)

{

int temp=\*a;

\*a=\*b;

\*b=temp;

}

int main()

{

int n;

printf("Enter Number of Processes: ");

scanf("%d",&n);

// b is array for burst time, p for priority and index for process id

int b[n],p[n],index[n];

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

{

printf("Enter Burst Time and Priority Value for Process %d: ",i+1);

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

index[i]=i+1;

}

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

{

int a=p[i],m=i;

//Finding out highest priority element and placing it at its desired position

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

{

if(p[j] > a)

{

a=p[j];

m=j;

}

}

//Swapping processes

swap(&p[i], &p[m]);

swap(&b[i], &b[m]);

swap(&index[i],&index[m]);

}

// T stores the starting time of process

int t=0;

//Printing scheduled process

printf("Order of process Execution is\n");

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

{

printf("P%d is executed from %d to %d\n",index[i],t,t+b[i]);

t+=b[i];

}

printf("\n");

printf("Process Id Burst Time Wait Time TurnAround Time\n");

int wait\_time=0;

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

{

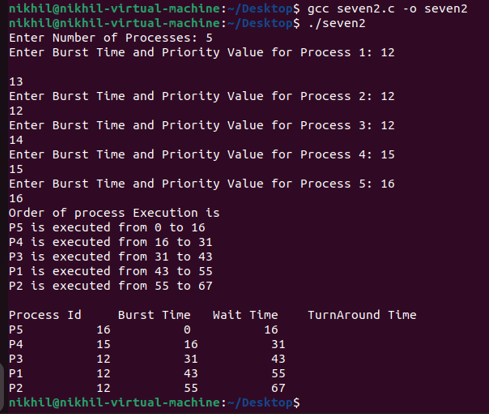
printf("P%d %d %d %d\n",index[i],b[i],wait\_time,wait\_time + b[i]);

wait\_time += b[i];

}

return 0;

}



Round Robin (Preemptive by default) #include<stdio.h>

int main()

{

int cnt,j,n,t,remain,flag=0,tq;

int wt=0,tat=0,at[10],bt[10],rt[10];

printf("Enter Total Process:\t ");

scanf("%d",&n);

remain=n;

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

{

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

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

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

rt[cnt]=bt[cnt];

}

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

scanf("%d",&tq);

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

for(t=0,cnt=0;remain!=0😉

{

if(rt[cnt]<=tq && rt[cnt]>0)

{

t+=rt[cnt];

rt[cnt]=0;

flag=1;

}

else if(rt[cnt]>0)

{

rt[cnt]-=tq;

t+=tq;

}

if(rt[cnt]==0 && flag==1)

{

remain--;

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

wt+=t-at[cnt]-bt[cnt];

tat+=t-at[cnt];

flag=0;

}

if(cnt==n-1)

cnt=0;

else if(at[cnt+1]<=t)

cnt++;

else

cnt=0;

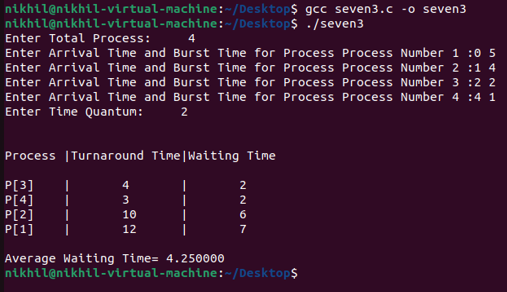
}

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

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

  return 0;

}



Conclusion : hence we have successfully studied and executed the following scheduling process.

**THANK YOU!!!!!!!!!!**