**PROGRAM**: #include<stdio.h>

void main()

{

// initlialize the variable name

int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];

float avg\_wt, avg\_tat;

printf(" Total number of process in the system: ");

scanf("%d", &NOP);

y = NOP; // Assign the number of process to variable y

// Use for loop to enter the details of the process like Arrival time and the Burst Time

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

{

printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);

printf(" Arrival time is: \t"); // Accept arrival time

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

printf(" \nBurst time is: \t"); // Accept the Burst time

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

temp[i] = bt[i]; // store the burst time in temp array

}

// Accept the Time qunat

printf("Enter the Time Quantum for the process: \t");

scanf("%d", &quant);

// Display the process No, burst time, Turn Around Time and the waiting time

**EXPERIMENT NO:**

**CPU SCHEDULING ALGORITHMS**

ROUND ROBIN SCHEDULING

**AIM:**

To write a C program for implementation of Round Robin scheduling algorithms.

**ALGORITHM**:

Step 1: Inside the structure declare the variables.

Step 2: Declare the variable i,j as integer, totwtime and totttime is equal to zero.

Step 3: Get the value of 'n' assign p and allocate the memory.

Step 4: Inside the for loop get the value of burst time and priority and read the time quantum.

Step 5: Assign wtime as zero.

Step 6: Check p[i].pri is greater than pljl.pri.

Step 7: Calculate the total of burst time and waiting time and assign as turnaround time.

Step 8: Stop the program.

printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");

for(sum=0, i = 0; y!=0; )

{

if(temp[i] <= quant && temp[i] > 0) // define the conditions

{

sum = sum + temp[i];

temp[i] = 0;

count=1;

}

else if(temp[i] > 0)

{

temp[i] = temp[i] - quant;

sum = sum + quant;

}

if(temp[i]==0 && count==1)

{

y--; //decrement the process no.

printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);

wt = wt+sum-at[i]-bt[i];

tat = tat+sum-at[i];

count =0;

}

if(i==NOP-1)

{

i=0;

}

else if(at[i+1]<=sum)

{

i++;

}

else

{

i=0;

}

}

// represents the average waiting time and Turn Around time

avg\_wt = wt \* 1.0/NOP;

avg\_tat = tat \* 1.0/NOP;

printf("\n Average Turn Around Time: \t%f", avg\_wt);

printf("\n Average Waiting Time: \t%f", avg\_tat);

}

**OUTPUT**

**Total number of process in the system:4**

**Enter Arrival time and Burst time of the process[1]**

**Arrival time is: 0**

**Burst time is :8**

**Enter Arrival time and Burst time of the process[2]**

**Arrival time is: 1**

**Burst time is :5**

**Enter Arrival time and Burst time of the process[3]**

**Arrival time is: 2**

**Burst time is :10**

**Enter Arrival time and Burst time of the process[4]**

**Arrival time is: 3**

**Burst time is :11**

**Enter the Time Quantum for the process: 6**

|  |  |  |  |
| --- | --- | --- | --- |
| **Process No** | **Brust Time** | **TAT** | **Waiting Time** |
| **Process No[2]** | **5** | **10** | **5** |
| **Process No[1]** | **8** | **25** | **17** |
| **Process No[3]** | **10** | **27** | **17** |
| **Process No[4]** | **11** | **31** | **20** |

**Average turn around time is :1 4.750000**

**Average waiting time is :23.250000**

**RESULT:**

The program is success fully executed.