## 

**DEPARTMENT OF COMPUTER SCIENCE & TECHNOLOGY**

**Lab-5 - Lab-7**

**HARSH MITTAL**

**2K19CSUN01082**

**CSE4B**

**Laboratory Objective:** To implement scheduling algorithms in C.

**Learning Outcome:** Understanding scheduling algorithms.

Course Outcome: CO3

Blooms Taxonomy: BT1, BT2, BT3, BT4

1. Implement the following scheduling algorithms in C on the given scenario:

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| P1 | 0 | 8 |
| P2 | 1 | 4 |
| P3 | 2 | 9 |
| p4 | 3 | 5 |

1. FCFS

CODE:

#include <stdio.h>

int i; *// global variable*

void find\_waiting\_time(int processes*[]*, int n, int bt*[]*, int wt*[]*)

{

wt[0] = 0;

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

{

wt[i] = bt[i - 1] + wt[i - 1];

}

}

void find\_turnAround\_time(int processes*[]*, int n, int bt*[]*, int wt*[]*, int tat*[]*)

{

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

{

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

}

}

void findAverage\_time(int processes*[]*, int n, int bt*[]*)

{

int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

float x, y;

find\_waiting\_time(processes, n, bt, wt);

find\_turnAround\_time(processes, n, bt, wt, tat);

printf("Process Burst time Waiting time Turn around time \n");

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

printf(" %d\t", (i + 1));

printf(" %d\t", bt[i]);

printf(" %d\t", wt[i]);

printf(" %d\n", tat[i]);

}

x = (float)total\_wt / (float)n;

y = (float)total\_tat / (float)n;

printf("Average waiting time = %f \n", x);

printf("\nAverage turn around time = %f \n", y);

}

*// main function*

int main()

{

int processes*[]* = {0, 1, 2, 3};

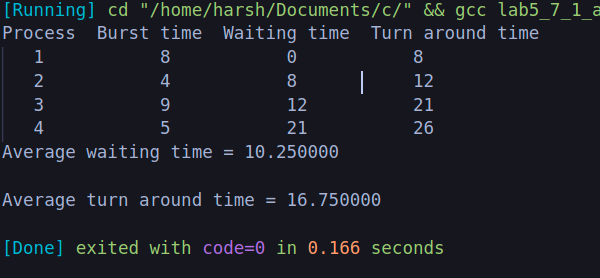
int n = sizeof processes / sizeof processes[0];

int burst\_time*[]* = {8, 4, 9, 5};

findAverage\_time(processes, n, burst\_time);

return 0;

}



1. SJF

code:

#include <stdio.h>

int main()

{

int i, n, p[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, min, k = 1, btime = 0;

int bt[10], temp, j, at[10], wt[10], tt[10], ta = 0, sum = 0;

float wavg = 0, tavg = 0, tsum = 0, wsum = 0;

printf(" -------Shortest Job First Scheduling ( NP )-------\n");

printf("\nEnter the No. of processes :");

scanf("%d", &n);

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

{

printf("\tEnter the burst time of %d process :", i + 1);

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

printf("\tEnter the arrival time of %d process :", i + 1);

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

}

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

{

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

{

if (at[i] < at[j])

{

temp = p[j];

p[j] = p[i];

p[i] = temp;

temp = at[j];

at[j] = at[i];

at[i] = temp;

temp = bt[j];

bt[j] = bt[i];

bt[i] = temp;

}

}

}

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

{

btime = btime + bt[j];

min = bt[k];

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

{

if (btime >= at[i] && bt[i] < min)

{

temp = p[k];

p[k] = p[i];

p[i] = temp;

temp = at[k];

at[k] = at[i];

at[i] = temp;

temp = bt[k];

bt[k] = bt[i];

bt[i] = temp;

}

}

k++;

}

wt[0] = 0;

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

{

sum = sum + bt[i - 1];

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

wsum = wsum + wt[i];

}

wavg = (wsum / n);

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

{

ta = ta + bt[i];

tt[i] = ta - at[i];

tsum = tsum + tt[i];

}

tavg = (tsum / n);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n RESULT:-");

printf("\nProcess\t Burst\t Arrival\t Waiting\t Turn-around\n");

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

{

printf("p%d\t\t %d\t\t %d\t\t %d\t\t %d\n", p[i], bt[i], at[i], wt[i], tt[i]);

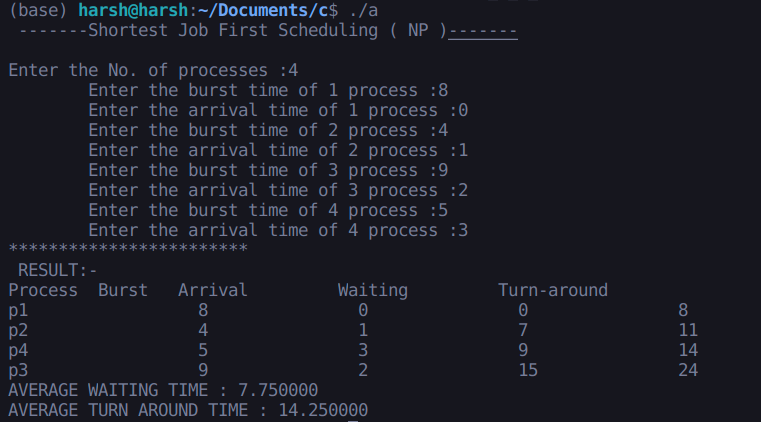
}

printf("AVERAGE WAITING TIME : %f\n", wavg);

printf("AVERAGE TURN AROUND TIME : %f\n", tavg);

return 0;

}



1. SRTF

code:

#include <stdio.h>

int main()

{

int arrival\_time[10], burst\_time[10], temp[10];

int i, smallest, count = 0, time, limit;

double wait\_time = 0, turnaround\_time = 0, end;

float average\_waiting\_time, average\_turnaround\_time;

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

scanf("%d", &limit);

printf("\nEnter Details of %d Processesn", limit);

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

{

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

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

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

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

temp[i] = burst\_time[i];

}

burst\_time[9] = 9999;

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

{

smallest = 9;

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

{

if (arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] && burst\_time[i] > 0)

{

smallest = i;

}

}

burst\_time[smallest]--;

if (burst\_time[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest];

turnaround\_time = turnaround\_time + end - arrival\_time[smallest];

}

}

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

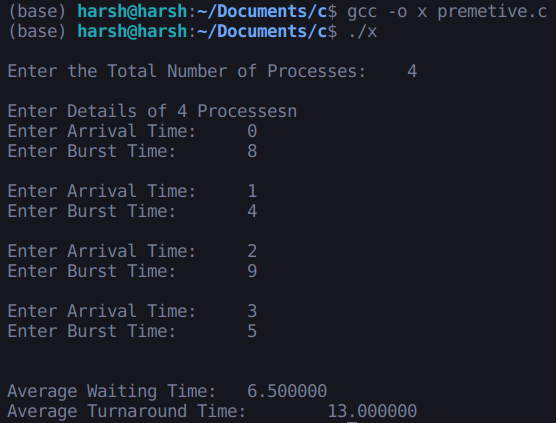
average\_turnaround\_time = turnaround\_time / limit;

printf("\n\nAverage Waiting Time:\t%lf\n", average\_waiting\_time);

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

return 0;

}



1. RR

code:

#include <stdio.h>

void main()

{

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;

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");

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

printf(" \nBurst time is: \t");

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

temp[i] = bt[i];

}

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

scanf("%d", &quant);

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)

{

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--;

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;

}

}

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);

}

