## 

**DEPARTMENT OF COMPUTER SCIENCE & TECHNOLOGY**

**PT-2**

**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;

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};

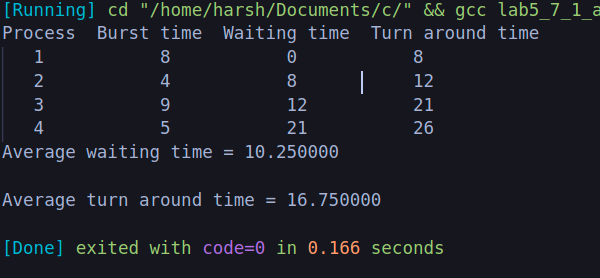
int n = sizeof processes / sizeof processes[0];

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

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

return 0;

}



1. 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;

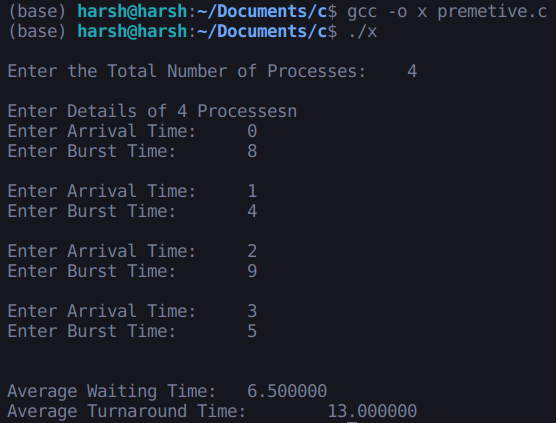
average\_turnaround\_time = turnaround\_time / limit;

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;

}

}

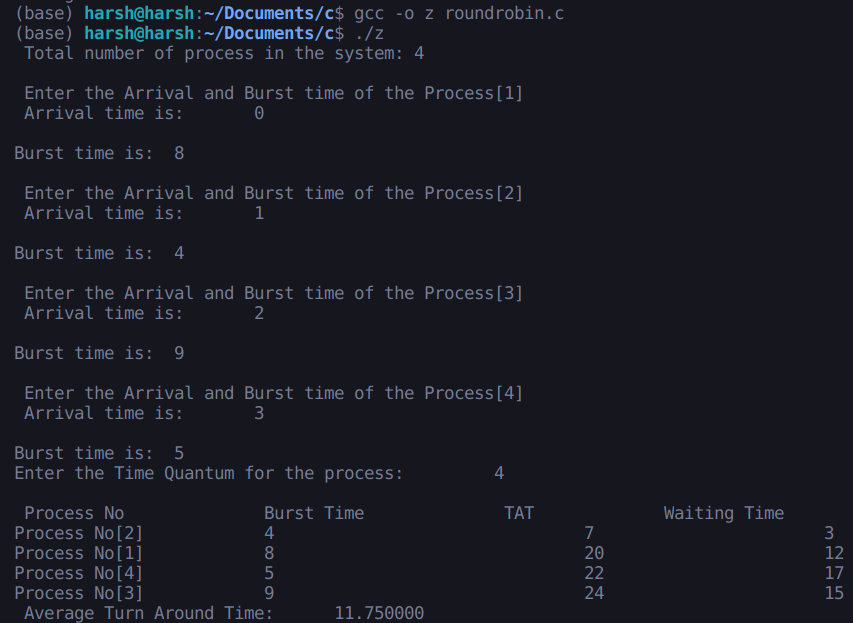
avg\_wt = wt \* 1.0 / NOP;

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);

}



4.

Priorty shduling algorithm:

totalprocess = 5

proc = []

*for* i in range(5):

l = []

*for* j in range(4):

l.append(0)

proc.append(l)

def get\_wt\_time(wt):

service = [0] \* 5

service[0] = 0

wt[0] = 0

*for* i in range(1, totalprocess):

service[i] = proc[i - 1][1] + service[i - 1]

wt[i] = service[i] - proc[i][0] + 1

*if*(wt[i] < 0):

wt[i] = 0

def get\_tat\_time(tat, wt):

*for* i in range(totalprocess):

tat[i] = proc[i][1] + wt[i]

def findgc():

wt = [0] \* 5

tat = [0] \* 5

wavg = 0

tavg = 0

get\_wt\_time(wt)

get\_tat\_time(tat, wt)

stime = [0] \* 5

ctime = [0] \* 5

stime[0] = 1

ctime[0] = stime[0] + tat[0]

*for* i in range(1, totalprocess):

stime[i] = ctime[i - 1]

ctime[i] = stime[i] + tat[i] - wt[i]

print("Process\_no\tStart\_time\tComplete\_time",

"\tTurn\_Around\_Time\tWaiting\_Time")

*for* i in range(totalprocess):

wavg += wt[i]

tavg += tat[i]

print(proc[i][3], "\t\t", stime[i],

"\t\t", end=" ")

print(ctime[i], "\t\t", tat[i], "\t\t\t", wt[i])

print("Average waiting time is : ", end=" ")

print(wavg / totalprocess)

print("average turnaround time : ", end=" ")

print(tavg / totalprocess)

*if* \_\_name\_\_ == "\_\_main\_\_":

arrivaltime = [1, 2, 3, 4, 5]

bursttime = [3, 5, 1, 7, 4]

priority = [3, 4, 1, 7, 8]

*for* i in range(totalprocess):

proc[i][0] = arrivaltime[i]

proc[i][1] = bursttime[i]

proc[i][2] = priority[i]

proc[i][3] = i + 1

proc = sorted(proc, key=lambda x: x[2])

proc = sorted(proc)

findgc()

**OUTPUT:**

