**Laboratory Record On**

**Design of Operating Systems**

**(CSE 4049)**

**Submitted by**

**Name : Aniket Kumar**

**Reg. No. : 1741012238**

**Branch : CSE**

**Semester : 6th**

**Section : A**

**Session : 2017-21**

**Date : 30/05/2020**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING FACULTY OF ENGINEERING & TECHNOLOGY (ITER)**

**SIKSHA ‘O’ ANUSANDHAN DEEMED TO BE UNIVERSITY**

**BHUBANESWAR, ODISHA – 751030**

**END TERM PROJECT**

**Assignment 5: End term project on CPU Scheduling**

**Objective of this Assignment:**

** To design a CPU scheduler for simulating a few CPU scheduling policies.**

**Overview of the Project:**

**One of the main tasks of an operating system is scheduling processes to run on the CPU. The goal of this programming project is to build a program using C to implement a simulator with different scheduling algorithms discussed in theory. The simulator should select a process to run from the ready queue based on the scheduling algorithm chosen at runtime. Since the assignment intends to simulate a CPU scheduler, it does not require any actual process creation or execution.**

**Project Description: The C program provides an interface to the user to implement the following scheduling policies as per the choice provided:**

**1. First Come First Served (FCFS)**

**2. Shortest Job First (SJF)**

**3. Shortest Remaining Time First (SRTF)**

**4. Round Robin (RR)**

**Appropriate option needs to be chosen from a switch case based menu driven program with an option of “Exit from program” in case 5 and accordingly a scheduling policy will print the Gantt chart and the average waiting time, average turnaround time and average response time. The program will take Process ids, its arrival time, and its CPU burst time as input. For implementing RR scheduling, users also need to specify the time quantum. Assume that the process ids should be unique for all processes. Each process consists of a single CPU burst (no I/O bursts), and processes are listed in order of their arrival time. Further assume that an interrupted process gets placed at the back of the Ready queue, and a newly arrived process gets placed at the back of the Ready queue as well. The output should be displayed in a formatted way for clarity of understanding and visual.**

Sol.

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX 100

//declarations

void fcfs();

void sjf();

void srtf();

void rr();

int input();

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);

void sort\_process\_by\_burst\_time(Process p[], int n);

void calculate\_waiting\_time(Process p[], int n);

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

int main()

{

int choice;

while (true)

{

printf("1. First Come First Serve FCFS\n");

printf("2. Shortest Job SJF\n");

printf("3. Shortest Remaining Time First SRTF\n");

printf("4. Round Robin RR\n");

printf("5. Exit\n");

printf("\nEnter your choice : \n");

choice = input();

switch (choice)

{

case 1:

{

fcfs();

break;

}

case 2:

{

sjf();

break;

}

case 3:

srtf();

break;

case 4:

rr();

break;

case 5:

exit(0);

break;

default:

printf("Please enter valid option.\n");

}

}

}

int input()

{

int number;

scanf("%d", &number);

return (number);

}

//FOR FCFS

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

{

int i;

puts("+-----+------------+--------------+-----------------+");

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

puts("+-----+------------+--------------+-----------------+");

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);

puts("+-----+------------+--------------+-----------------+");

}

}

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

{

int i, j;

// print top bar

printf(" ");

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

{

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

printf("--");

printf(" ");

}

printf("\n|");

// 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 bottom bar

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

{

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

printf("--");

printf(" ");

}

printf("\n");

// printing the timeline

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 fcfs()

{

Process p[MAX];

int i, j, n;

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

printf("First Come First Serve\n");

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

puts(" GANTT CHART ");

puts(" \*\*\*\*\*\*\*\*\*\*\* ");

print\_gantt\_chart(p, n);

}

//FOR SJF

double average\_waiting\_time, average\_turnaround\_time;

int total\_waiting\_time;

int sum\_turnaround\_time;

void calculate\_waiting\_time(Process p[], int n)

{

int i;

total\_waiting\_time = 0;

p[0].waiting\_time = 0;

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

{

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

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

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

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

}

}

void sort\_process\_by\_burst\_time(Process p[], int n)

{

int i, j;

Process temp;

for(i=0; i<n-1; i++) {

for(j=0; j<n-1-i; j++) {

if(p[j].burst\_time > p[j+1].burst\_time) {

temp = p[j];

p[j] = p[j+1];

p[j+1] = temp;

}

}

}

}

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

{

int i, j;

int last = p[n - 1].burst\_time + (n == 1 ? 0 : p[n - 1].waiting\_time);

// printing top bar

printf(" ");

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

{

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

printf("--");

printf(" ");

}

printf("\n|");

// middle position

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 ");

// bottom bar

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

{

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

printf("--");

printf(" ");

}

printf("\n");

// printing waiting time

int minus = 0;

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

{

if (p[i].waiting\_time > 9)

printf(" ");

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

if (p[i + 1].waiting\_time > 9)

{

minus = 1;

}

if (i + 1 == n)

if (last > 9)

minus = 1;

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

printf(" ");

}

if (last > 9)

printf(" ");

printf("%d\n", last);

}

void sjf()

{

Process p[MAX];

int n, i, j;

puts("SHORTEST JOB FIRST SCHEDULING ALGORITHM");

puts("=======================================");

printf("Enter total process: ");

scanf("%d", &n);

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

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

{

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

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

p[i].pid = i + 1;

}

sort\_process\_by\_burst\_time(p, n);

calculate\_waiting\_time(p, n);

average\_waiting\_time = (double)((double)total\_waiting\_time / (double)n);

average\_turnaround\_time = (double)((double) sum\_turnaround\_time / (double)n);

puts("");

printf("Average Waiting Time: %.2lf\n", average\_waiting\_time);

printf("Average TurnAround Time : %.21f\n", average\_turnaround\_time);

printf("Gantt Chart:\n");

print\_gantt\_chart(p, n);

}

//FOR SRTF

struct sjf

{

int bt, at, wt, st, pno, tt, cbt;

};

int get(struct sjf arr[], int t, int n)

{

int imin, min = 9999, i;

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

{

if (arr[i].at <= t && arr[i].st == 0)

if (min > arr[i].bt)

{

min = arr[i].bt;

imin = i;

}

}

return imin;

}

void gantt\_chart(struct sjf arr[], int p[], int n, int nop)

{

int i, a[100], s = 0;

float avgtt = 0, avgwt = 0;

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

printf("GANTT CHART\n");

printf("0");

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

{

while (i < n - 1 && p[i] == p[i + 1])

{

s++;

i++;

}

s++;

printf(" -> [P%d] <- %d", arr[p[i]].pno, s);

arr[p[i]].wt = s - arr[p[i]].at - arr[p[i]].tt;

}

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

{

arr[i].tt += arr[i].wt;

avgwt += arr[i].wt;

avgtt += arr[i].tt;

}

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

printf("Pro\tArTi\tBuTi\tTaTi\tWtTi\n");

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

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

{

printf("[P%d]\t%d\t%d\t%d\t%d\n", arr[i].pno, arr[i].at, arr[i].cbt, arr[i].tt, arr[i].wt);

}

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

avgwt = avgwt / nop;

avgtt = avgtt / nop;

printf("Average Waiting Time : %.2f\n", avgwt);

printf("Average Turnaround Time : %.2f\n", avgtt);

return;

}

int iscomplite(struct sjf arr[], int n)

{

int i;

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

if (arr[i].st == 0)

return 0;

return 1;

}

void srtf()

{

int n, i, a, t = 0;

int p[100];

float avgwt = 0, avgtt = 0;

struct sjf arr[100];

printf("Shortest Remaining Job First\n");

printf("Enter Number of Processes\n");

scanf("%d", &n);

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

{

printf("Enter Arrival Time & Burst Time for Process [P%d]\n", i);

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

arr[i].pno = i;

arr[i].cbt = arr[i].bt;

arr[i].st = 0;

arr[i].tt = arr[i].bt;

arr[i].wt = 0;

}

i = 0;

while (1)

{

if (iscomplite(arr, n))

break;

a = get(arr, t, n);

p[i] = a;

arr[a].bt -= 1;

if (arr[a].bt == 0)

arr[a].st = 1;

t = t + 1;

i++;

}

gantt\_chart(arr, p, i, n);

}

//FOR RR

void rr()

{

int at[100], bt[100], rt[100], temp[100];

float wait\_time = 0, turn\_time = 0;

int c, j, n, time, r, flag = 0, time\_q, ltt, i, wt = 0;

printf("Round Robin\n");

printf("Enter no.of process:");

scanf("%d", &n);

r = n;

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

{

printf("Enter arrival time of p%d:\t", c + 1);

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

printf("Enter burst time of p%d: \t", c + 1);

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

rt[c] = bt[c];

temp[c] = bt[c];

printf("\n");

}

printf("Enter time quantum:\t");

scanf("%d", &time\_q);

printf("\n\n\tprocess\tAT\tTAT\tWT\n\n");

for (time = 0, c = 0; r != 0;)

{

if (rt[c] <= time\_q && rt[c] > 0)

{

time = time + rt[c];

rt[c] = 0;

flag = 1;

}

else if (rt[c] > 0)

{

rt[c] = rt[c] - time\_q;

time = time + time\_q;

}

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

{

wt = 0;

wt = time - at[c] - bt[c];

r--;

printf("\tP%d\t%d\t%d\t%d\n", c + 1, at[c], time - at[c], wt);

ltt = time - at[c];

wait\_time = wait\_time + time - at[c] - bt[c];

turn\_time = turn\_time + time - at[c];

flag = 0;

}

if (c == n - 1)

c = 0;

else if (at[c + 1] <= time)

c++;

else

c = 0;

}

j = 0;

printf("\n\n\n");

printf("Gantt Chart ");

printf("\n\n\n");

printf("\t");

for (i = at[0]; i < time;)

{

if (bt[j] >= time\_q)

{

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

i += time\_q;

bt[j] = bt[j] - time\_q;

}

else if (bt[j] > 0)

{

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

i += bt[j];

bt[j] = 0;

}

j++;

if (j >= n)

{

j = 0;

}

}

printf("\n");

j = 0;

printf("\t");

for (i = at[0]; i < time;)

{

if (temp[j] >= time\_q)

{

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

i += time\_q;

temp[j] = temp[j] - time\_q;

}

else if (temp[j] > 0)

{

printf("%d\t", i + temp[j]);

i += temp[j];

temp[j] = 0;

}

j++;

if (j >= n)

{

j = 0;

}

}

printf("\n\n\n");

printf("\nAverage\_waiting\_time=%f\n", wait\_time / n);

printf("Average\_turn\_around\_time=%f\n", turn\_time / n);

printf("\n\n");

}

Output:  
**TEST CASE 1**

**1. FCFS**

$ gcc eproject.c

pawankumar@DESKTOP-LG0R4F6:~/USP$ ./a.out

1. First Come First Serve FCFS

2. Shortest Job SJF

3. Shortest Remaining Time First SRTF

4. Round Robin RR

5. Exit

Enter your choice :

1

First Come First Serve

Enter total number of process: 5

Enter burst time for each process:

P[1] : 10

P[2] : 1

P[3] : 2

P[4] : 1

P[5] : 5

+-----+------------+--------------+-----------------+

| PID | Burst Time | Waiting Time | Turnaround Time |

+-----+------------+--------------+-----------------+

| 1 | 10 | 0 | 10 |

+-----+------------+--------------+-----------------+

| 2 | 1 | 10 | 11 |

+-----+------------+--------------+-----------------+

| 3 | 2 | 11 | 13 |

+-----+------------+--------------+-----------------+

| 4 | 1 | 13 | 14 |

+-----+------------+--------------+-----------------+

| 5 | 5 | 14 | 19 |

+-----+------------+--------------+-----------------+

Total Waiting Time : 48

Average Waiting Time : 9.60

Total Turnaround Time : 32589

Average Turnaround Time : 6517.80

GANTT CHART

\*\*\*\*\*\*\*\*\*\*\*

-------------------- -- ---- -- ----------

| P1 |P2| P3 |P4| P5 |

-------------------- -- ---- -- ----------

0 10 11 13 14 19

**2. SJF**

Enter your choice :

2

SHORTEST JOB FIRST SCHEDULING ALGORITHM

=======================================

Enter total process: 5

Enter burst time for each process:

P[1]: 10

P[2]: 1

P[3]: 2

P[4]: 1

P[5]: 5

Average Waiting Time: 3.20

Average TurnAround Time : 6.799999999999999822364

Gantt Chart:

-- -- ---- ---------- --------------------

|P2|P4| P3 | P5 | P1 |

-- -- ---- ---------- --------------------

0 11 2 4 9 19

1. First Come First Serve FCFS

2. Shortest Job SJF

3. Shortest Remaining Time First SRTF

4. Round Robin RR

5. Exit

**3. SRTF**

Enter your choice :

3

Shortest Remaining Job First

Enter Number of Processes

5

Enter Arrival Time & Burst Time for Process [P0]

0

10

Enter Arrival Time & Burst Time for Process [P1]

0

1

Enter Arrival Time & Burst Time for Process [P2]

0

2

Enter Arrival Time & Burst Time for Process [P3]

0

1

Enter Arrival Time & Burst Time for Process [P4]

0

5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GANTT CHART

0 -> [P1] <- 1 -> [P3] <- 2 -> [P2] <- 4 -> [P4] <- 9 -> [P0] <- 19

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Pro ArTi BuTi TaTi WtTi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[P0] 0 10 19 9

[P1] 0 1 1 0

[P2] 0 2 4 2

[P3] 0 1 2 1

[P4] 0 5 9 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Average Waiting Time : 3.20

Average Turnaround Time : 7.00

1. First Come First Serve FCFS

2. Shortest Job SJF

3. Shortest Remaining Time First SRTF

4. Round Robin RR

5. Exit

**4. RR**

Enter your choice :

4

Round Robin

Enter no.of process:5

Enter arrival time of p1: 0

Enter burst time of p1: 10

Enter arrival time of p2: 0

Enter burst time of p2: 1

Enter arrival time of p3: 0

Enter burst time of p3: 2

Enter arrival time of p4: 0

Enter burst time of p4: 1

Enter arrival time of p5: 0

Enter burst time of p5: 5

Enter time quantum: 4

process AT TAT WT

P2 0 5 4

P3 0 7 5

P4 0 8 7

P5 0 17 12

P1 0 19 9

Gantt Chart

P1 p2 p3 p4 P5 P1 p5 p1

4 5 7 8 12 16 17 19

Average\_waiting\_time=7.400000

Average\_turn\_around\_time=11.200000

**TEST CASE 2**

**1. SRTF**

1. First Come First Serve FCFS

2. Shortest Job SJF

3. Shortest Remaining Time First SRTF

4. Round Robin RR

5. Exit

Enter your choice :

3

Shortest Remaining Job First

Enter Number of Processes

04

Enter Arrival Time & Burst Time for Process [P0]

0

4

Enter Arrival Time & Burst Time for Process [P1]

0

2

Enter Arrival Time & Burst Time for Process [P2]

1

3

Enter Arrival Time & Burst Time for Process [P3]

2

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GANTT CHART

0 -> [P1] <- 2 -> [P3] <- 4 -> [P2] <- 7 -> [P0] <- 11

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Pro ArTi BuTi TaTi WtTi

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[P0] 0 4 11 7

[P1] 0 2 2 0

[P2] 1 3 6 3

[P3] 2 2 2 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Average Waiting Time : 2.50

Average Turnaround Time : 5.25

1. First Come First Serve FCFS

2. Shortest Job SJF

3. Shortest Remaining Time First SRTF

4. Round Robin RR

5. Exit

**2. RR**

Enter your choice :

4

Round Robin

Enter no.of process:4

Enter arrival time of p1: 0

Enter burst time of p1: 4

Enter arrival time of p2: 0

Enter burst time of p2: 2

Enter arrival time of p3: 1

Enter burst time of p3: 3

Enter arrival time of p4: 2

Enter burst time of p4: 2

Enter time quantum: 4

process AT TAT WT

P1 0 4 0

P2 0 6 4

P3 1 8 5

P4 2 9 7

Gantt Chart

P1 p2 p3 p4

4 6 9 11

Average\_waiting\_time=4.000000

Average\_turn\_around\_time=6.750000