End Term Project

On

Design of Operating Systems (CSE 4049)

Submitted by

Name	**: Saswat Mohanty**
Reg. No.	**1941012407**
Semester	**: 5th**
Branch	**: CSE**
Section	**: D**
Session	**: 2021-2022**
Admission Batch	**: 2019**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY (ITER)

SIKSHA 'O' ANUSANDHAN DEEMED TO BE UNIVERSITY

BHUBANESWAR, ODISHA – 751030

Objective of this Assignment:

To design a CPU scheduler for simulating a few CPU scheduling policies

Project Description:

Program to 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)

Programs

FCFS

FCFS.h

```
void FCFS();
```

FCFS.c

```
#include <stdio.h>
#include <stdlib.h>
#include "FCFS.h"
typedef struct node
{
    int no;
    float at, bt, pc, tat, wt, rt, rd;
    struct node *next;
} NODE;

void create_insert(NODE **p, int no, float at, float bt, float *fr)
{
    NODE *q, *r = *p;
    q = (NODE *)malloc(sizeof(NODE));
    q->no = no;
    q->at = at;
    q->bt = bt;
    q->rt = *fr - at;
    q->pc = *fr + bt;
    q->tat = q->pc - at;
    q->wt = q->tat - bt;
```

```
    q->rd = q->tat / bt;
    *fr = *fr + bt;
    q->next = NULL;
    if (*p == NULL)
        *p = q;
    else
    {
        while (r->next != NULL)
            r = r->next;
        r->next = q;
    }
}

void gantt_chart(NODE *p, int process)
{
    int i;
    NODE *r =  p;
    printf("\n\nGannt Chart:\n");
    for (i = 1; i <= process; i++)
        printf("-------------------------");
    printf("\n");

    for (i = 1; i <= process; i++)
    {
        printf("|\tP%d\t", p->no);
        p = p->next;
    }
    printf("|\n");

    for (i = 1; i <= process; i++)
        printf("-------------------------");

    printf("\n");
    printf("%.1f      \t", r->at);
    for (i = 1; i <= process; i++)
    {
        printf("%.1f      \t", r->pc);
        r = r->next;
    }
}

void display(NODE *p, int process)
{
    float ttat, twt, trd, trt, tbt;
    ttat = twt = trd = trt = tbt = 0;
```

```
    printf("\n\n\nProcess Details:\n");
    printf("************************************\n");
    printf("Pro\tArTi\tBuTi\tTaTi\tWtTi\tRTi\n");
    printf("************************************\n");
    while (p != NULL)
    {
        printf("%d\t", p->no);
        printf("%.2f\t", p->at);
        printf("%.2f\t", p->bt);
        printf("%.2f\t", p->tat);
        printf("%.2f\t", p->wt);
        printf("%.2f\n", p->rt);
        ttat += p->tat;
        twt += p->wt;
        trd += p->rd;
        trt += p->rt;
        tbt += p->bt;

        p = p->next;
    }
    printf("\n\n\nOverall Details:\n");
    printf("Average Turn Around Time: %.2f\n", ttat / process);
    printf("Average Waiting Time: %.2f\n", twt / process);
    printf("Average Response Time: %.2f\n", trt / process);
}

void FCFS()
{
    NODE *head = NULL;
    int process, i;
    float arrival_time, burst_time, first_response;
    printf("First- Come, First-Served (FCFS) Scheduling\n");
    printf("Enter Number of Processes\n");
    scanf("%d", &process);
    for (i = 1; i <= process; i++)
    {
        printf("\nEnter the Details for Process %d: \n", i);
        printf("Arrival Time: ");
        scanf("%f", &arrival_time);
        printf("Burst Time: ");
        scanf("%f", &burst_time);
        if (i == 1)
            first_response = arrival_time;
        create_insert(&head, i, arrival_time, burst_time, &first_response);
    }
```

```
    printf("\n<----------------------------------------------------------------------------  START ----------
----------------------------->\n");
    gantt_chart(head, process);
    display(head, process);
    printf("\n<----------------------------------------------------------------------------  END ------------------
--------------------------->\n");
}
```

SJF

SJF.h

```
void SFJ();
```

SJF.c

```
#include <stdio.h>
#include "SJF.h"
struct time
{
	int p, art, but, wtt, tat, st;
};
int process(struct time a[], int pro, int t)
{
	int i, minpro, mintime = 999;
	for (i = 0; i < pro; i++)
	{
		if (a[i].art <= t && a[i].st == 0)
		{
			if (mintime > a[i].but)
			{
				mintime = a[i].but;
				minpro = i;
			}
		}
	}
	a[minpro].st = 1;
	return minpro;
}
void ganttchart(struct time a[], int gc[], int pro)
{
```

```
        int i, x = 0;
        printf("Gantt Chart\n");
        printf("0");
        for (i = 0; i < pro; i++)
        {
                x = x + a[gc[i]].but;
                printf(" -> [P%d] <- %d", a[gc[i]].p, x);
        }
        printf("\n");
        return;
}
void SFJ()
{
        int i, pro, curpro, t = 0, gc[100];
        struct time a[100];
        float avgwt = 0, avgtt = 0;
        printf("Shortest-Job-First (SJF) Scheduling\n");
        printf("Enter Number of Processes\n");
        scanf("%d", &pro);
        for (i = 0; i < pro; i++)
        {
                printf("Enter Arrival Time & Burst Time for Process P%d\n", i);
                a[i].p = i;
                scanf("%d%d", &a[i].art, &a[i].but);
                a[i].st = 0;
        }
        for (i = 0; i < pro; i++)
        {
                curpro = process(a, pro, t);
                a[curpro].wtt = t - a[curpro].art;
                a[curpro].tat = a[curpro].art + a[curpro].but;
                t = t + a[curpro].but;
                avgwt = avgwt + a[curpro].wtt;
                avgtt = avgtt + a[curpro].tat;
                gc[i] = curpro;
        }
        printf("\n<----------------------------------------------------------------------------  START ----------
-------------------------------->\n");
        printf("****************************************\n");
        ganttchart(a, gc, pro);
        printf("****************************************\n");
        printf("\n\n\nProcess Details:\n");
        printf("****************************************\n");
        printf("Pro\tArTi\tBuTi\tTaTi\tWtTi\n");
        printf("****************************************\n");
```

```
        for (i = 0; i < pro; i++)
        {
                printf("%d\t%d\t%d\t%d\t%d\n", a[i].p, a[i].art, a[i].but,
a[i].tat, a[i].wtt);
        }
        avgwt = avgwt / pro;
        avgtt = avgtt / pro;
        printf("\n\n\nOverall Details:\n");
        printf("Average Waiting Time : %.2f\n", avgwt);
        printf("Average Turnaround Time : %.2f\n", avgtt);
        printf("\n<----------------------------------------------------------------------------  END ----------
----------------------------->\n");
}
```

SRTF

SRTF.h

```
void SRTF();
```

SRTF.c

```
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#include "SRTF.h"
struct process_struct
{
    int pid;
    int at;
    int bt;
    int ct, wt, tat, rt, start_time;
} ps[100];

int findmax(int a, int b)
{
    return a > b ? a : b;
}

int findmin(int a, int b)
{
    return a < b ? a : b;
```

```
}

void SRTF()
{

    int n;
    float bt_remaining[100];
    bool is_completed[100] = {false}, is_first_process = true;
    int current_time = 0;
    int completed = 0;
    ;
    float sum_tat = 0, sum_wt = 0, sum_rt = 0, total_idle_time = 0, length_cycle,
prev = 0;
    float cpu_utilization;

    int max_completion_time, min_arrival_time;
    printf("Shortest-Remaining-Time-First (SRTF) Scheduling\n");
    printf("Enter Number of Processes\n");
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Arrival Time: ", i);
        scanf("%d", &ps[i].at);
        ps[i].pid = i;
    }

    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Burst Time: ", i);
        scanf("%d", &ps[i].bt);
        bt_remaining[i] = ps[i].bt;
    }

    while (completed != n)
    {
        int min_index = -1;
        int minimum = INT_MAX;
        for (int i = 0; i < n; i++)
        {
            if (ps[i].at <= current_time && is_completed[i] == false)
            {
                if (bt_remaining[i] < minimum)
                {
                    minimum = bt_remaining[i];
                    min_index = i;
```

```
                }
                if (bt_remaining[i] == minimum)
                {
                    if (ps[i].at < ps[min_index].at)
                    {
                        minimum = bt_remaining[i];
                        min_index = i;
                    }
                }
            }
        }

        if (min_index == -1)
        {
            current_time++;
        }
        else
        {
            if (bt_remaining[min_index] == ps[min_index].bt)
            {
                ps[min_index].start_time = current_time;
                total_idle_time += (is_first_process == true) ? 0 :
(ps[min_index].start_time - prev);
                is_first_process = false;
            }
            bt_remaining[min_index] -= 1;
            current_time++;
            prev = current_time;
            if (bt_remaining[min_index] == 0)
            {
                ps[min_index].ct = current_time;
                ps[min_index].tat = ps[min_index].ct - ps[min_index].at;
                ps[min_index].wt = ps[min_index].tat - ps[min_index].bt;
                ps[min_index].rt = ps[min_index].start_time - ps[min_index].at;

                sum_tat += ps[min_index].tat;
                sum_wt += ps[min_index].wt;
                sum_rt += ps[min_index].rt;
                completed++;
                is_completed[min_index] = true;
            }
        }
    }
    max_completion_time = INT_MIN;
    min_arrival_time = INT_MAX;
```

```
    for (int i = 0; i < n; i++)
    {
        max_completion_time = findmax(max_completion_time, ps[i].ct);
        min_arrival_time = findmin(min_arrival_time, ps[i].at);
    }
    length_cycle = max_completion_time - min_arrival_time;
    printf("\n<----------------------------------------------------------------------------  START ----------
---------------------------->\n");
    printf("\n\n\nProcess Details:\n");
    printf("*************************************\n");
    printf("Pro\tArTi\tBuTi\tTaTi\tWtTi\tRTi\n");
    printf("*************************************\n");
    for (int i = 0; i < n; i++)
        printf("%d\t%d\t%d\t%d\t%d\t%d\n", ps[i].pid, ps[i].at, ps[i].bt,
ps[i].tat, ps[i].wt, ps[i].rt);
    printf("*************************************\n");
    printf("\n");
    printf("\n\n\nOverall Details:\n");
    printf("\nAverage Turn Around time= %f ", (float)sum_tat / n);
    printf("\nAverage Waiting Time= %f ", (float)sum_wt / n);
    printf("\nAverage Response Time= %f ", (float)sum_rt / n);
    printf("\n<----------------------------------------------------------------------------  END ----------
--------------------------->\n");
}
```

RR

RR.h

```
void RR();
```

RR.c

```
#include <stdio.h>
#include "RR.h"
struct times
{
    int p, art, but, wtt, tat, rnt;
};
void sortart(struct times a[], int pro)
{
    int i, j;
    struct times temp;
```

```
    for (i = 0; i < pro; i++)
    {
        for (j = i + 1; j < pro; j++)
        {
            if (a[i].art > a[j].art)
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    return;
}
void RR()
{
    int i, j, pro, time, remain, flag = 0, ts;
    struct times a[100];
    float avgwt = 0, avgtt = 0;
    printf("Round Robin (RR) Scheduling\n");
    printf("Enter Number of Processes\n");
    scanf("%d", &pro);
    remain = pro;
    for (i = 0; i < pro; i++)
    {
        printf("Enter arrival time and Burst time for Process P%d : ", i);
        scanf("%d%d", &a[i].art, &a[i].but);
        a[i].p  =  i;
        a[i].rnt = a[i].but;
    }
    sortart(a, pro);
    printf("Enter Time Quantum Number : ");
    scanf("%d", &ts);
    printf("\n<----------------------------------------------------------------------------  START ----------
----------------------------->\n");
    printf("\n**************************************\n");
    printf("Gantt Chart\n");
    printf("0");
    for (time = 0, i = 0; remain != 0;)
    {
        if (a[i].rnt <= ts && a[i].rnt > 0)
        {
            time = time + a[i].rnt;
            printf(" -> [P%d] <- %d", a[i].p, time);
            a[i].rnt = 0;
```

```
            flag = 1;
        }
        else if (a[i].rnt > 0)
        {
            a[i].rnt = a[i].rnt - ts;
            time = time + ts;
            printf(" -> [P%d] <- %d", a[i].p, time);
        }
        if (a[i].rnt == 0 && flag == 1)
        {
            remain--;
            a[i].tat = time - a[i].art;
            a[i].wtt = time - a[i].art - a[i].but;
            avgwt = avgwt + time - a[i].art - a[i].but;
            avgtt = avgtt + time - a[i].art;
            flag = 0;
        }
        if (i == pro - 1)
            i = 0;
        else if (a[i + 1].art <= time)
            i++;
        else
            i = 0;
    }
    printf("\n\n");
    printf("\n\n\nProcess Details:\n");
    printf("***************************************\n");
    printf("Pro\tArTi\tBuTi\tTaTi\tWtTi\n");
    printf("***************************************\n");
    for (i = 0; i < pro; i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\n", a[i].p, a[i].art, a[i].but, a[i].tat,
a[i].wtt);
    }
    printf("***************************************\n");
    avgwt = avgwt / pro;
    avgtt = avgtt / pro;
    printf("\n\n\nOverall Details:\n");
    printf("Average Waiting Time : %.2f\n", avgwt);
    printf("Average Turnaround Time : %.2f\n", avgtt);
    printf("\n<----------------------------------------------------------------------------  END ----------
-------------------------->\n");
}
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "FCFS.h"
#include "SJF.h"
#include "SRTF.h"
#include  "RR.h"
int main()
{
    int choice;
    printf("CPU scheduler for simulating a few CPU scheduling policies.\n");
    while (1)
    {
        printf("\nChosse Option\n1->FCFS\n2->SJF\n3->SRTF\n4->RR\n5->Exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            FCFS();
            break;
        case 2:
            SFJ();
            break;
        case 3:
            SRTF();
            break;
        case 4:
            RR();
            break;
        case 5:
            exit(0);
        default:
            printf("Wrong Input\n");
            break;
        }
    }
}
```

Output Console:

```
CPU scheduler for simulating a few CPU scheduling policies.

Chosse Option
1->FCFS
2->SJF
3->SRTF
4->RR
5->Exit
```

Test Cases:

1. Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds), and time quantum = 4ms as shown below.

Process	Arrival time	Burst Time
P1	0	10
P2	0	1
P3	0	2
P4	0	1
P5	0	5

Input Choice 1:

```
1
First- Come, First-Served (FCFS) Scheduling
Enter Number of Processes
5

Enter the Details for Process 1:
Arrival Time: 0
Burst Time: 10

Enter the Details for Process 2:
Arrival Time: 0
Burst Time: 1

Enter the Details for Process 3:
Arrival Time: 0
Burst Time: 2

Enter the Details for Process 4:
Arrival Time: 0
Burst Time: 1

Enter the Details for Process 5:
Arrival Time: 0
Burst Time: 5
```

```
<------------------------------------------------  START ------------------------------------->

Gannt Chart:
-------------------------------------------------------------------------------
|       P1      |       P2      |       P3      |       P4      | P5      |
-------------------------------------------------------------------------------
0.0             10.0            11.0            13.0            14.0            19.0

Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi    RTi
**************************************
1       0.00    10.00   10.00   0.00    0.00
2       0.00    1.00    11.00   10.00   10.00
3       0.00    2.00    13.00   11.00   11.00
4       0.00    1.00    14.00   13.00   13.00
5       0.00    5.00    19.00   14.00   14.00



Overall Details:
Average Turn Around Time: 13.40
Average Waiting Time: 9.60
Average Response Time: 9.60

<------------------------------------------------  END --------------------------------------->
```

Input Choice 2:

```
2
Shortest-Job-First (SJF) Scheduling
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
```

```
<-------------------------------------------------  START ------------------------------------->
**************************************
Gantt Chart
0 -> [P1] <- 1 -> [P3] <- 2 -> [P2] <- 4 -> [P4] <- 9 -> [P0] <- 19
**************************************



Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi
**************************************
0       0       10      10      9
1       0       1       1       0
2       0       2       2       2
3       0       1       1       1
4       0       5       5       4



Overall Details:
Average Waiting Time : 3.20
Average Turnaround Time : 3.80

<-------------------------------------------------  END --------------------------------------->
```

Input Choice 3:

```
3
Shortest-Remaining-Time-First (SRTF) Scheduling
Enter Number of Processes
5

Enter Process 0 Arrival Time: 0

Enter Process 1 Arrival Time: 0

Enter Process 2 Arrival Time: 0

Enter Process 3 Arrival Time: 0

Enter Process 4 Arrival Time: 0

Enter Process 0 Burst Time: 10

Enter Process 1 Burst Time: 1

Enter Process 2 Burst Time: 2

Enter Process 3 Burst Time: 1

Enter Process 4 Burst Time: 5
```

```
<-----------------------------------------------  START ----------------------------------------->



Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi    RTi
**************************************
0       0       10      19      9       9
1       0       1       1       0       0
2       0       2       4       2       2
3       0       1       2       1       1
4       0       5       9       4       4
**************************************




Overall Details:

Average Turn Around time= 7.000000
Average Waiting Time= 3.200000
Average Response Time= 3.200000
<-----------------------------------------------  END ------------------------------------------->
```

Input Choice 4:

```
4
Round Robin (RR) Scheduling
Enter Number of Processes
5
Enter arrival time and Burst time for Process P0 : 0 10
Enter arrival time and Burst time for Process P1 : 0 1
Enter arrival time and Burst time for Process P2 : 0 2
Enter arrival time and Burst time for Process P3 : 0 1
Enter arrival time and Burst time for Process P4 : 0 5
Enter Time Quantum Number : 4
```

```
<-------------------------------------------  START ------------------------------------->

**************************************
Gantt Chart
0 -> [P0] <- 4 -> [P1] <- 5 -> [P2] <- 7 -> [P3] <- 8 -> [P4] <- 12 -> [P0] <- 16 -> [P4] <- 17 -> [P0] <- 19



Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi
**************************************
P0      0       10      19      9
P1      0       1       5       4
P2      0       2       7       5
P3      0       1       8       7
P4      0       5       17      12
**************************************


Overall Details:
Average Waiting Time : 7.40
Average Turnaround Time : 11.20

<-------------------------------------------  END ------------------------------------->
```

- ⇨ On analysing the results of the algorithm, the minimum average waiting time is Shortest Job First (SJF) and shortest Remaining Time First (SRTF).

2. Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds), and time quantum =2ms as shown below.

Process	Arrival time	Burst Time
P1	0	4
P2	0	2
P3	1	3
P4	2	2

Input Choice 1:

```
1
First- Come, First-Served (FCFS) Scheduling
Enter Number of Processes
4

Enter the Details for Process 1:
Arrival Time: 0
Burst Time: 4

Enter the Details for Process 2:
Arrival Time: 0
Burst Time: 2

Enter the Details for Process 3:
Arrival Time: 1
Burst Time: 3

Enter the Details for Process 4:
Arrival Time: 2
Burst Time: 2
```

```
<-----------------------------------------------  START ----------------------------------------->


Gannt Chart:
-------------------------------------------------------------------
|       P1      |       P2      |       P3      |       P4      |
-------------------------------------------------------------------
0.0             4.0             6.0             9.0             11.0


Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi    RTi
**************************************
1       0.00    4.00    4.00    0.00    0.00
2       0.00    2.00    6.00    4.00    4.00
3       1.00    3.00    8.00    5.00    5.00
4       2.00    2.00    9.00    7.00    7.00



Overall Details:
Average Turn Around Time: 6.75
Average Waiting Time: 4.00
Average Response Time: 4.00

<-----------------------------------------------  END ------------------------------------------->
```

Input Choice 2:

```
2
Shortest-Job-First (SJF) Scheduling
Enter Number of Processes
4
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
```

```
<------------------------------------------------  START ------------------------------------->
**************************************
Gantt Chart
0 -> [P1] <- 2 -> [P3] <- 4 -> [P2] <- 7 -> [P0] <- 11
**************************************



Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi
**************************************
0       0       4       4       7
1       0       2       2       0
2       1       3       4       3
3       2       2       4       0



Overall Details:
Average Waiting Time : 2.50
Average Turnaround Time : 3.50

<------------------------------------------------  END --------------------------------------->
```

Input Choice 3:

```
3
Shortest-Remaining-Time-First (SRTF) Scheduling
Enter Number of Processes
4

Enter Process 0 Arrival Time: 0

Enter Process 1 Arrival Time: 0

Enter Process 2 Arrival Time: 1

Enter Process 3 Arrival Time: 2

Enter Process 0 Burst Time: 4

Enter Process 1 Burst Time: 2

Enter Process 2 Burst Time: 3

Enter Process 3 Burst Time: 2
```

```
<-------------------------------------------------  START ------------------------------------->



Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi    RTi
**************************************
0       0       4       11      7       7
1       0       2       2       0       0
2       1       3       6       3       3
3       2       2       2       0       0
**************************************




Overall Details:

Average Turn Around time= 5.250000
Average Waiting Time= 2.500000
Average Response Time= 2.500000
<-------------------------------------------------  END ------------------------------------->
```

Input Choice 4:

```
4
Round Robin (RR) Scheduling
Enter Number of Processes
4
Enter arrival time and Burst time for Process P0 : 0 4
Enter arrival time and Burst time for Process P1 : 0 2
Enter arrival time and Burst time for Process P2 : 1 3
Enter arrival time and Burst time for Process P3 : 2 2
Enter Time Quantum Number : 2
```

```
<-------------------------------------------------  START ------------------------------------->

**************************************
Gantt Chart
0 -> [P0] <- 2 -> [P1] <- 4 -> [P2] <- 6 -> [P3] <- 8 -> [P0] <- 10 -> [P2] <- 11




Process Details:
**************************************
Pro     ArTi    BuTi    TaTi    WtTi
**************************************
P0      0       4       10      6
P1      0       2       4       2
P2      1       3       10      7
P3      2       2       6       4
**************************************



Overall Details:
Average Waiting Time : 4.75
Average Turnaround Time : 7.50

<-------------------------------------------------  END ------------------------------------->
```

- ⇨ On analysing the results of the algorithm, the minimum average waiting time is Shortest Job First (SJF) and shortest Remaining Time First (SRTF).