EAST WEST UNIVERSITY

Operating Systems Lab

Department of Computer Science and Engineering

Student's Name : **H M Ahsan Uddin Abir**

Student's ID : **2018-3-60-003**

Course Name : Operating Systems

Course code : CSE 325

Section NO : 03

Name of the Experiment : **CPU Scheduling Software**

Submitted To: **Prof. Dr. Md. Motaharul Islam**

Date of allocation: 21.03.22

Date of submission: 17.04.22

Operating Systems Lab

CSE-325 LAB

Name of Experiment:

CPU Scheduling

Abstract:

CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution.

CPU scheduling algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

Objective:

1. Implement First Come First Serve Scheduling Algorithm (FCFS).
2. Implement Shortest Job First Scheduling Algorithm (SJF).
3. Round Robin Scheduling Algorithm (RR).
4. Make a choice based system.

H M Ahsan Uddin Abir (2018-3-60-003)

Implementations:

Output:

Menu:

```
                                ======= Welcome to Process Scheduling Algorithm Software =======



                 =====Menu=====

1. First Come First Serve Scheduling Algorithm (FCFS)
2. Shortest Job First Scheduling Algorithm (SJF)
3. Round Robin Scheduling Algorithm (RR)
0. Exit

Enter a Number to Proceed Further :
```

If invalid number is given

```
                                ======= Welcome to Process Scheduling Algorithm Software =======



                 =====Menu=====

1. First Come First Serve Scheduling Algorithm (FCFS)
2. Shortest Job First Scheduling Algorithm (SJF)
3. Round Robin Scheduling Algorithm (RR)
0. Exit

Enter a Number to Proceed Further :5


Error! Enter a valid number
```

First Come First Serve Scheduling Algorithm (FCFS):

```
                                ======= Welcome to Process Scheduling Algorithm Software =======



                 =====Menu=====

1. First Come First Serve Scheduling Algorithm (FCFS)
2. Shortest Job First Scheduling Algorithm (SJF)
3. Round Robin Scheduling Algorithm (RR)
0. Exit

Enter a Number to Proceed Further :1


                         ====First Come First Serve Scheduling Algorithm (FCFS)====

Enter Total Number of Process: 3

Enter burst time for each process:

P[1] : 24
P[2] : 3
P[3] : 3

+-----+------------+---------------+
| PID | Burst Time | Waiting Time |
+-----+------------+---------------+
|  1  |     24     |       0       |
+-----+------------+---------------+
|  2  |      3     |      24       |
+-----+------------+---------------+
|  3  |      3     |      27       |
+-----+------------+---------------+

Total Waiting Time      : 51
Average Waiting Time    : 17.00

          GANTT CHART
          ***********
 ------------------------------------------------- ------ ------
|                        P1                         |  P2  |  P3  |
 ------------------------------------------------- ------ ------
0                                                  24     27     30

Process returned 0 (0x0)   execution time : 124.288 s
Press any key to continue.
```

Shortest Job First Scheduling Algorithm (SJF):

```
                              ======= Welcome to Process Scheduling Algorithm Software =======


                 =====Menu=====

1. First Come First Serve Scheduling Algorithm (FCFS)
2. Shortest Job First Scheduling Algorithm (SJF)
3. Round Robin Scheduling Algorithm (RR)
0. Exit

Enter a Number to Proceed Further :2


                         ====Shortest Job First Scheduling Algorithm (SJF)====

Enter Total Number of Process: 4

Enter burst time for each process:

p1:6
p2:8
p3:7
p4:3

Process     Burst Time          Waiting Time
p4                3                 0
p1                6                 3
p3                7                 9
p2                8                 16

Average Waiting Time=7.000000

Process returned 0 (0x0)   execution time : 27.181 s
Press any key to continue.
```

Round Robin Scheduling Algorithm (RR):

```
Enter a Number to Proceed Further :3


                         ====Round Robin Scheduling Algorithm (RR)====

Enter Total Number of Process: 5

Enter an Arrival Time of the Process P1: 0
Enter an Arrival Time of the Process P2: 1
Enter an Arrival Time of the Process P3: 2
Enter an Arrival Time of the Process P4: 3
Enter an Arrival Time of the Process P5: 4

Enter a Burst Time of the Process P1: 6
Enter a Burst Time of the Process P2: 3
Enter a Burst Time of the Process P3: 5
Enter a Burst Time of the Process P4: 1
Enter a Burst Time of the Process P5: 4



Enter the Time Quantum Number: 2



Process         Arrival Time    Burst Time      FT      TAT                     Waiting Time
1               0               6               16      16.000000               10.000000
2               1               3               12      11.000000               8.000000
3               2               5               19      17.000000               12.000000
4               3               1               9       6.000000                5.000000
5               4               4               18      14.000000               10.000000


Average Waiting Time: 9.000000
Average Turn Around Time: 12.800000

Process returned 0 (0x0)   execution time : 45.389 s
Press any key to continue.
```

H M Ahsan Uddin Abir (2018-3-60-003)

System code full:

```
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
#include<cstdlib>
#include<queue>
#include<cstdio>
#include<conio.h>


#define MAX 100

using namespace std;

typedef struct process
{
    int id,at,bt,st,ft,pr;
    float wt,tat;
} process;
process p[10],p1[10],temp;
queue<int> q1;

typedef struct
{
    int pid;
    int burst_time;
    int waiting_time;
    int turnaround_time;
} Process;
```

```
void FCFS();
void sjf();
void rr();


void print_table(Process p[], int n);
void print_gantt_chart(Process p[], int n);
int accept(int ch);
void turnwait(int n);
void display(int n);
void ganttrr(int n);




int main()
{
    int number,y;


    printf("\t\t\t\t======= Welcome to Process Scheduling Algorithm
Software =======\n\n\n\n");


    printf("\t\t=====Menu=====\n\n");
    printf("1. First Come First Serve Scheduling Algorithm
(FCFS)\n");
    printf("2. Shortest Job First Scheduling Algorithm (SJF)\n");
    printf("3. Round Robin Scheduling Algorithm (RR)\n");
    printf("0. Exit\n\n");
    printf("Enter a Number to Proceed Further :");


    scanf("%d", &y);



    number = y;
```

```
    switch(number)
        {
        case 0:
            exit(0);

        case 1:
            printf("\n\n\t\t\t====First Come First Serve Scheduling
Algorithm (FCFS)====\n\n");
            FCFS();
            break;

        case 2:
            printf("\n\n\t\t\t====Shortest Job First Scheduling
Algorithm (SJF)====\n\n");
            sjf();
            break;

        case 3:
            printf("\n\n\t\t\t====Round Robin Scheduling Algorithm
(RR)====\n\n");
            rr();
            break;

        default:
            printf("\n\nError! Enter a valid number\n\n");
            break;

        }

    getch();
    return 0;
}



//...First Come First Serve Scheduling Algorithm (FCFS)
void FCFS()
```

```
{
    Process p[MAX];
    int i, j, n;
    int sum_waiting_time = 0, sum_turnaround_time;
    printf("Enter Total Number of Process: ");
    scanf("%d", &n);
    printf("\nEnter burst time for each process:\n\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;
        }


    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;
        }


    for(i=0; i<n; i++)
        {
        sum_waiting_time += p[i].waiting_time;
        sum_turnaround_time += p[i].turnaround_time;
        }


    puts("");
    print_table(p, n);
    puts("");
```

H M Ahsan Uddin Abir (2018-3-60-003)

```
    printf("Total Waiting Time      : %-2d\n", sum_waiting_time);
    printf("Average Waiting Time    : %-2.2lf\n",
(double)sum_waiting_time / (double) n);


    puts("");
    puts("          GANTT CHART          ");
    puts("          ***********          ");
    print_gantt_chart(p, n);

    exit(0);

}

void print_table(Process p[], int n)
{
    int i;

    puts("+-----+-----------+-------------+");
    puts("| PID | Burst Time | Waiting Time |");
    puts("+-----+-----------+-------------+");

    for(i=0; i<n; i++)
        {
        printf("| %2d  |     %2d     |      %2d      |\n"
               , p[i].pid, p[i].burst_time, p[i].waiting_time );
        puts("+-----+-----------+-------------+");
        }

}



void print_gantt_chart(Process p[], int n)
{
    int i, j;
```

```
printf(" ");
for(i=0; i<n; i++)
    {
    for(j=0; j<p[i].burst_time; j++)
        printf("--");
    printf(" ");
    }
printf("\n|");


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 ");

for(i=0; i<n; i++)
    {
    for(j=0; j<p[i].burst_time; j++)
        printf("--");
    printf(" ");
    }
printf("\n");


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");
        printf("%d", p[i].turnaround_time);

        }
    printf("\n");


}

//............




//...Shortest Job First Scheduling Algorithm (SJF)

void sjf()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter Total Number of Process: ");
    scanf("%d",&n);
    printf("\nEnter burst time for each process:\n\n");
    for(i=0; i<n; i++)
        {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
        }

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

```
            {
            if(bt[j]<bt[pos])
                pos=j;
            }
        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;
        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
        }
    wt[0]=0;
    for(i=1; i<n; i++)
        {
        wt[i]=0;
        for(j=0; j<i; j++)
            wt[i]+=bt[j];
        total+=wt[i];
        }
    avg_wt=(float)total/n;
    total=0;
    printf("\nProcess\t    Burst Time    \tWaiting Time");
    for(i=0; i<n; i++)
        {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\np%d\t\t  %d\t\t    %d\t\t\t",p[i],bt[i],wt[i]);
        }
    avg_tat=(float)total/n;
    printf("\n\nAverage Waiting Time=%f\n",avg_wt);
    exit(0);
}

//............
//...Round Robin Scheduling Algorithm (RR)
```

```
void rr()
{
    int i,n,ts,ch,j,x;

    p[0].tat=0;
    p[0].wt=0;

    n=accept(ch);
    ganttrr(n);
    turnwait(n);

    display(n);
    exit(0);

}

int accept(int ch)
{
    int i,n;

    printf("Enter Total Number of Process: ");
    scanf("%d",&n);

    if(n==0)
        {
        printf("Invalid");
        exit(1);
        }
    cout<<endl;

    for(i=1; i<=n; i++)
        {
        printf("Enter an Arrival Time of the Process P%d: ",i);
        scanf("%d",&p[i].at);
```

```
        p[i].id=i;
        }

    cout<<endl;

    for(i=1; i<=n; i++)
        {
        printf("Enter a Burst Time of the Process P%d: ",i);
        scanf("%d",&p[i].bt);
        }

    for(i=1; i<=n; i++)
        {
        p1[i]=p[i];
        }
    return n;
}

void ganttrr(int n)
{
    int i,ts,m,nextval,nextarr;

    nextval=p1[1].at;
    i=1;
    cout<<"\n\n\nEnter the Time Quantum Number: ";
    cin>>ts;

    for(i=1; i<=n && p1[i].at<=nextval; i++)
        {
        q1.push(p1[i].id);
        }

    while(!q1.empty())
        {
        m=q1.front();
```

```
            q1.pop();

            if(p1[m].bt>=ts)
                {
                nextval=nextval+ts;
                }
            else
                {
                nextval=nextval+p1[m].bt;
                }
            if(p1[m].bt>=ts)
                {
                p1[m].bt=p1[m].bt-ts;
                }
            else
                {
                p1[m].bt=0;
                }

            while(i<=n&&p1[i].at<=nextval)
                {
                q1.push(p1[i].id);
                i++;
                }

            if(p1[m].bt>0)
                {
                q1.push(m);
                }
            if(p1[m].bt<=0)
                {
                p[m].ft=nextval;
                }
            }
}
```

H M Ahsan Uddin Abir (2018-3-60-003)

```
void turnwait(int n)
{
    int i;

    for(i=1; i<=n; i++)
        {
        p[i].tat=p[i].ft-p[i].at;
        p[i].wt=p[i].tat-p[i].bt;
        p[0].tat=p[0].tat+p[i].tat;
        p[0].wt=p[0].wt+p[i].wt;
        }
    p[0].tat=p[0].tat/n;
    p[0].wt=p[0].wt/n;
}
void display(int n)
{
    int i;
    printf("\n\n\nProcess\t\tArrival Time\tBurst
Time\tFT\tTAT\t\t\tWaiting Time");
    for(i=1; i<=n; i++)
        {

printf("\n%d\t\t%d\t\t%d\t\t%d\t%f\t\t%f",p[i].id,p[i].at,p[i].bt,p[
i].ft,p[i].tat,p[i].wt); }


    printf("\n\n\nAverage Waiting Time: %f\n",p[0].wt);
    printf("Average Turn Around Time: %f\n",p[0].tat);


}
```

Conclusion:

From this experiment, we come to know how to do CPU Scheduling algorithms Implementation and make a choice based system.