SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1411 - OPERATING SYSTEMS LAB

LAB EXERCISE 3

Implementation of CPU Scheduling Policies

Submission Date:23-03-2022

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Develop a menu driven C program to implement the CPU Scheduling Algorithms FCFS and SJF

Algorithm for FCFS:

1) Get total no of process from the user.
2) Get process id, arrival time, burst time for all process.
3) Sort the process based on arrival time.
4) Loop until all process ends
    a) Set waiting time of process as sum of previous process waiting time and burst time by subtracting arrival time of the process
    b) Add current waiting time to total waiting time
    c) Set turnaround time of process as sum of waiting time and burst time
    d) Add current turnaround time to total turnaround time
5) Calculate average waiting time by dividing total waiting time by total no of process
6) Calculate average turnaround time by dividing total turnaround time by total no of process
7) Print process table
8) Print Gantt Chart

Algorithm for SJF - Non-Preemptive:

1) Get total no of process from the user.
2) Get process id, burst time for all process.
3) Sort the process based on burst time.
4) Loop until all process ends
    a) Set waiting time of process as sum of previous process waiting time and burst time by subtracting arrival time of the process
    b) Add current waiting time to total waiting time
    c) Set turnaround time of process as sum of waiting time and burst time
    d) Add current turnaround time to total turnaround time
5) Calculate average waiting time by dividing total waiting time by total no of process

6) Calculate average turnaround time by dividing total turnaround time by total no of process
7) Print process table
8) Print Gantt Chart

Code:

```c
/*Algorithm: 1. Read the following a. Number of processes b. Process IDs c. Arrival time for each process d. Burst Time for each process 2. Design a menu with FCFSand SJFoptions 3. Upon selection of menu option apply the corresponding algorithm. 4. Compute the Turnaround Time, Average waiting Time for each of the algorithm. 5. Tabularize the results. 6. Display the Gantt Chart.*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <dirent.h>
#include <ctype.h>

typedef struct process
{
    char pid[3];
    int arrival, burst, turnaround, waiting, completion;
} process;

void print_gantt_chart(process p[], int n)
{
    printf("\n\nGantt-Chart\n");
    int i, j;
    printf(" ");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < p[i].burst; j++)
            printf("--");
        printf(" ");
    }
    printf("\n|");

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < p[i].burst - 1; j++)
            printf(" ");
        printf(" %s", p[i].pid);
        for (j = 0; j < p[i].burst - 1; j++)
            printf(" ");
        printf("|");
    }
    printf("\n ");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < p[i].burst; j++)
```

```
            printf("--");
        printf(" ");
    }
    printf("\n");

    printf("0");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < p[i].burst; j++)
            printf("  ");
        if (p[i].turnaround > 9)
            printf("\b");
        printf("%d", p[i].turnaround);
    }
    printf("\n");
}

int main()
{
    int no_of_process;
    int totalwaitingtime = 0, totalturnaround = 0;
    int pos;
    char ch = 'y';
    process p[100];
    while (ch == 'y' || ch == 'Y')
    {
        int choice;
        printf("\nMenu\n\t1.FCFS\n\t2.SJF-Non Preemptive\n\t3.SJF-Preemptive\nEnter
Choice:");
        scanf(" %d", &choice);
        switch (choice)
        {
        case 1:
        {
            printf("\nFCFS\n");
            int no_of_process;
            printf("\nNumber of Processes :");
            scanf(" %d", &no_of_process);
            for (int i = 0; i < no_of_process; i++)
            {
                printf("\n\nProcess %d\n", i + 1);
                printf("Process ID: ");
                scanf(" %s", p[i].pid);
                printf("Arrival Time :");
                scanf(" %d", &p[i].arrival);
                printf("Burst Time :");
                scanf(" %d", &p[i].burst);
            }

            process temppro;
            for (int i = 0; i < no_of_process; i++)
            {
                pos = i;
                for (int j = i + 1; j < no_of_process; j++)
```

```
            {
                if (p[j].arrival < p[pos].arrival)
                    pos = j;
            }
            temppro = p[i];
            p[i] = p[pos];
            p[pos] = temppro;
        }
        totalwaitingtime = 0, totalturnaround = 0;
        p[0].waiting = 0;
        p[0].turnaround = p[0].burst;
        totalturnaround += p[0].turnaround;
        for (int i = 1; i < no_of_process; i++)
        {
            if (p[i - 1].waiting + p[i - 1].burst - p[i].arrival > 0)
            {
                p[i].waiting = p[i - 1].waiting + p[i - 1].burst - p[i].arrival;
            }
            else
            {
                p[i].waiting = 0;
            }
            totalwaitingtime += p[i].waiting;
            p[i].turnaround = p[i].burst + p[i].waiting;
            totalturnaround += p[i].turnaround;
        }
        printf("\nP_ID\tArrival Time\tBurst Time\tTurnaround Time\t\tWaiting Time\n");
        for (int i = 0; i < no_of_process; i++)
        {
            printf("%s\t\t%d\t\t%d\t\t%d\t\t\t%d\n", p[i].pid, p[i].arrival,
p[i].burst, p[i].turnaround, p[i].waiting);
        }

        float avgwaiting = (float)(totalwaitingtime / no_of_process);
        float avgturnaround = (float)(totalturnaround / no_of_process);
        printf("\n\t\tAVERAGE  AverageTurnaroundTime=%.2f\tAverageWaitingTime=%.2f\n",
avgturnaround, avgwaiting);
        print_gantt_chart(p, no_of_process);
        break;
    }

    case 2:
    {
        printf("\nSJF-Non Preemptive\n");
        int no_of_process;
        printf("\nNumber of Processes :");
        scanf(" %d", &no_of_process);
        int totalwaitingtime = 0, totalturnaround = 0;
        char pid[no_of_process][5];
        for (int i = 0; i < no_of_process; i++)
        {
            printf("\n\nProcess %d\n", i + 1);
            printf("Process ID: ");
            scanf(" %s", p[i].pid);
```

```
            // printf("Arrival Time :");
            p[i].arrival = 0;
            printf("Burst Time :");
            scanf(" %d", &p[i].burst);
        }

        process temppro;
        for (int i = 0; i < no_of_process; i++)
        {
            pos = i;
            for (int j = i + 1; j < no_of_process; j++)
            {
                if ((p[j].arrival < p[pos].arrival) || ((p[j].arrival <=
p[pos].arrival) && (p[j].burst < p[pos].burst)))
                {
                    pos = j;
                }
            }

            temppro = p[i];
            p[i] = p[pos];
            p[pos] = temppro;
        }
        totalwaitingtime = 0, totalturnaround = 0;
        p[0].waiting = 0;
        p[0].turnaround = p[0].burst;
        totalturnaround += p[0].turnaround;
        for (int i = 1; i < no_of_process; i++)
        {
            if (p[i - 1].waiting + p[i - 1].burst - p[i].arrival > 0)
            {
                p[i].waiting = p[i - 1].waiting + p[i - 1].burst - p[i].arrival;
            }
            else
            {
                p[i].waiting = 0;
            }
            totalwaitingtime += p[i].waiting;
            p[i].turnaround = p[i].burst + p[i].waiting;
            totalturnaround += p[i].turnaround;
        }
        printf("\nP_IDs\tBurst Time\tTurnaround Time\t\tWaiting Time\n");
        for (int i = 0; i < no_of_process; i++)
        {
            printf("%s\t\t%d\t\t%d\t\t\t%d\n", p[i].pid, p[i].burst, p[i].turnaround,
p[i].waiting);
        }

        float avgwaiting = (float)(totalwaitingtime / no_of_process);
        float avgturnaround = (float)(totalturnaround / no_of_process);
        printf("\n\t\tAVERAGE  AverageTurnaroundTime=%.2f\tAverageWaitingTime=%.2f\n",
avgturnaround, avgwaiting);
        print_gantt_chart(p, no_of_process);
        break;
```

```
        }
        case 3:
        {
            printf("\nSJF - Preemptive\n");
            int no_of_process;
            printf("\nNumber of Processes :");
            scanf(" %d", &no_of_process);
            for (int i = 0; i < no_of_process; i++)
            {
                printf("\n\nProcess %d\n", i + 1);
                printf("Process ID: ");
                scanf(" %s", p[i].pid);
                printf("Arrival Time :");
                scanf(" %d", &p[i].arrival);
                printf("Burst Time :");
                scanf(" %d", &p[i].burst);
            }

            process temppro;
            for (int i = 0; i < no_of_process; i++)
            {
                pos = i;
                for (int j = i + 1; j < no_of_process; j++)
                {
                    if (p[j].arrival < p[pos].arrival)
                        pos = j;
                }
                temppro = p[i];
                p[i] = p[pos];
                p[pos] = temppro;
            }
            int rem_time[no_of_process];
            for (int i = 0; i < no_of_process; i++)
            {
                rem_time[i] = p[i].burst;
            }
            for (int cur_time = 0, completed = 0; completed < no_of_process; cur_time++)
            {
                int idx = -1;
                for (int i = 0; i < no_of_process; i++)
                {
                    if (p[i].arrival <= cur_time && rem_time[i] > 0 && (idx == -1 ||
rem_time[i] < rem_time[idx]))
                    {
                        idx = i;
                    }
                }
                if (idx != -1)
                {
                    rem_time[idx]--;
                    if (rem_time[idx] == 0)
                    {
                        completed++;
                        p[idx].completion = cur_time;
```

```
                    p[idx].turnaround = p[idx].completion - p[idx].arrival + 1;
                    p[idx].waiting = p[idx].turnaround - p[idx].burst;
                }
            }
        }

        for (int i = 0; i < no_of_process; i++)
        {
            //  p[i].turnaround=p[i].burst+p[i].waiting;
            totalwaitingtime += p[i].waiting;
            totalturnaround += p[i].turnaround;
        }
        printf("\nP_ID\tArrival Time\tBurst Time\tTurnaround Time\t\tWaiting Time\n");
        for (int i = 0; i < no_of_process; i++)
        {
            printf("%s\t\t%d\t\t%d\t\t%d\t\t\t%d\n", p[i].pid, p[i].arrival,
p[i].burst, p[i].turnaround, p[i].waiting);
        }

        float avgwaiting = (float)(totalwaitingtime / no_of_process);
        float avgturnaround = (float)(totalturnaround / no_of_process);
        printf("\n\t\tAVERAGE  AverageTurnaroundTime=%.2f\tAverageWaitingTime=%.2f\n",
avgturnaround, avgwaiting);
        print_gantt_chart(p, no_of_process);
        break;
    }
    case 4:
        printf("Exiting...");
        return 0;
    }
    printf("\nWant to Continue (Y/N):");
    scanf(" %c", &ch);
}
    return 0;
}
```

Output:

```
Number of Processes :4


Process 1
Process ID: P1
Arrival Time :0
Burst Time :10


Process 2
Process ID: P2
Arrival Time :3
Burst Time :2


Process 3
Process ID: P3
Arrival Time :1
Burst Time :6


Process 4
Process ID: P
Arrival Time :5
Burst Time :4
```

```
Menu
    1.FCFS
    2.SJF-Non Preemptive
    3.SJF-Preemptive
Enter Choice:1

FCFS
```

```
P_ID    Arrival Time    Burst Time  Turnaround Time     Waiting Time
P1      0       10      10          0
P3      1       6       15          9
P2      3       2       14          12
P       5       4       13          9

        AVERAGE  AverageTurnaroundTime=13.00    AverageWaitingTime=7.00


Gantt-Chart
 ------------------- ------------ ---- --------
|         P1        |      P3     |  P2 |    P   |
 ------------------- ------------ ---- --------
0                  10           15   14       13
```

```
SJF-Non Preemptive

Number of Processes :4


Process 1
Process ID: 1
Burst Time :6


Process 2
Process ID: 2
Burst Time :8


Process 3
Process ID: 3
Burst Time :7


Process 4
Process ID: 4
Burst Time :3
```

```
P_IDs   Burst Time  Turnaround Time     Waiting Time
4       3       3           0
1       6       9           3
3       7       16          9
2       8       24          16

        AVERAGE  AverageTurnaroundTime=13.00    AverageWaitingTime=7.00


Gantt-Chart
 ------ ------------ -------------- ----------------
|   4  |      1     |       3      |        2       |
 ------ ------------ -------------- ----------------
0      3            9              16               24
```

```
Process 1
Process ID: 1
Arrival Time :2
Burst Time :6


Process 2
Process ID: 2
Arrival Time :5
Burst Time :2


Process 3
Process ID: 3
Arrival Time :1
Burst Time :8


Process 4
Process ID: 4
Arrival Time :0
Burst Time :3


Process 5
Process ID: 5
Arrival Time :4
Burst Time :4
```

```
P_ID    Arrival Time    Burst Time  Turnaround Time        Waiting Time
4       0       3       3           0
3       1       8       22          14
1       2       6       13          7
5       4       4       6           2
2       5       2       2           0

        AVERAGE  AverageTurnaroundTime=28.00   AverageWaitingTime=15.00
```

Learning Outcome:

- Implemented FCFS Scheduling and SJF Scheduling in C program
- Displayed Gantt Chart for the above scheduling methods