****

**LAB EXERCISE 3**

**Implementation of CPU Scheduling Policies**

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

**Code:**

*/\*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:**

 



 

 