Name : Kishaanth.S

Reg.No : 205001054

Implementation of CPU Scheduling Policies: FCFS and SJF (Non-preemptive)

Aim:

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

Algorithm : (FCFS)

```
    1: Input the number of processes from the user.
    2: Have a structure with pid, waiting, burst, arrival, turn_around and completion as data members.
    3: Using a loop, input all the given details for each process and store it.
    4: Sort the array based on the arrival time of each process.
    5: Have 2 variables, average_weight and avg_ta both initialized to 0.
    6: Initialize the completion time and turn_around of 1st process as burst of the 1st process.
    7: Using a loop, from 1 to n,
        7.1: If the completion time of previous process is greater than or equal to arrival time of current process,
            7.1.1: Assign completion time of that process as sum of completion time of previous and burst of current process.
        7.2 : Else
            7.2.1: completion time of that process is the sum of the burst time and arrival time of that process.
        7.3: Waiting time of current process is equal to the difference between the completion time and sum of arrival and burst of that current process.
        7.4: average weight is incremented by the waiting time of that process.
        7.5 : turn_around of that process is the difference between the completion time and arrival time of that process.
        7.6 : average_ta is incremented by the turn_around of that process.
    8: Print all the details of the processes. Also print the gantt chart along with the average weighting time and average turn_around time.
```

Algorithm : (SJF)

1: Input the number of processes from the user.

2: Have a structure with pid, waiting, burst, arrival, turn_around and completion as data members.

3: Using a loop, input all the given details for each process and store it.

4: Sort the array based on the burst of each process.

5: Have 2 variables, average_weight and avg_ta both initialized to 0.

6: Initialize the completion time and turn_around of 1st process as burst of the 1st process.

7: Using a loop, from 1 to n,

7.1: Assign waiting time of that process as completion time of the previous process.

7.2: Assign completion time of that process with the sum of the completion time of previous process and burst time of current process.

7.3: Assign turn_around of current process as the completion time of that process.

7.4: Increment the average_weight by the waiting time of the current process.

7.5 Increment the average_ta by the turn_around time of current process.

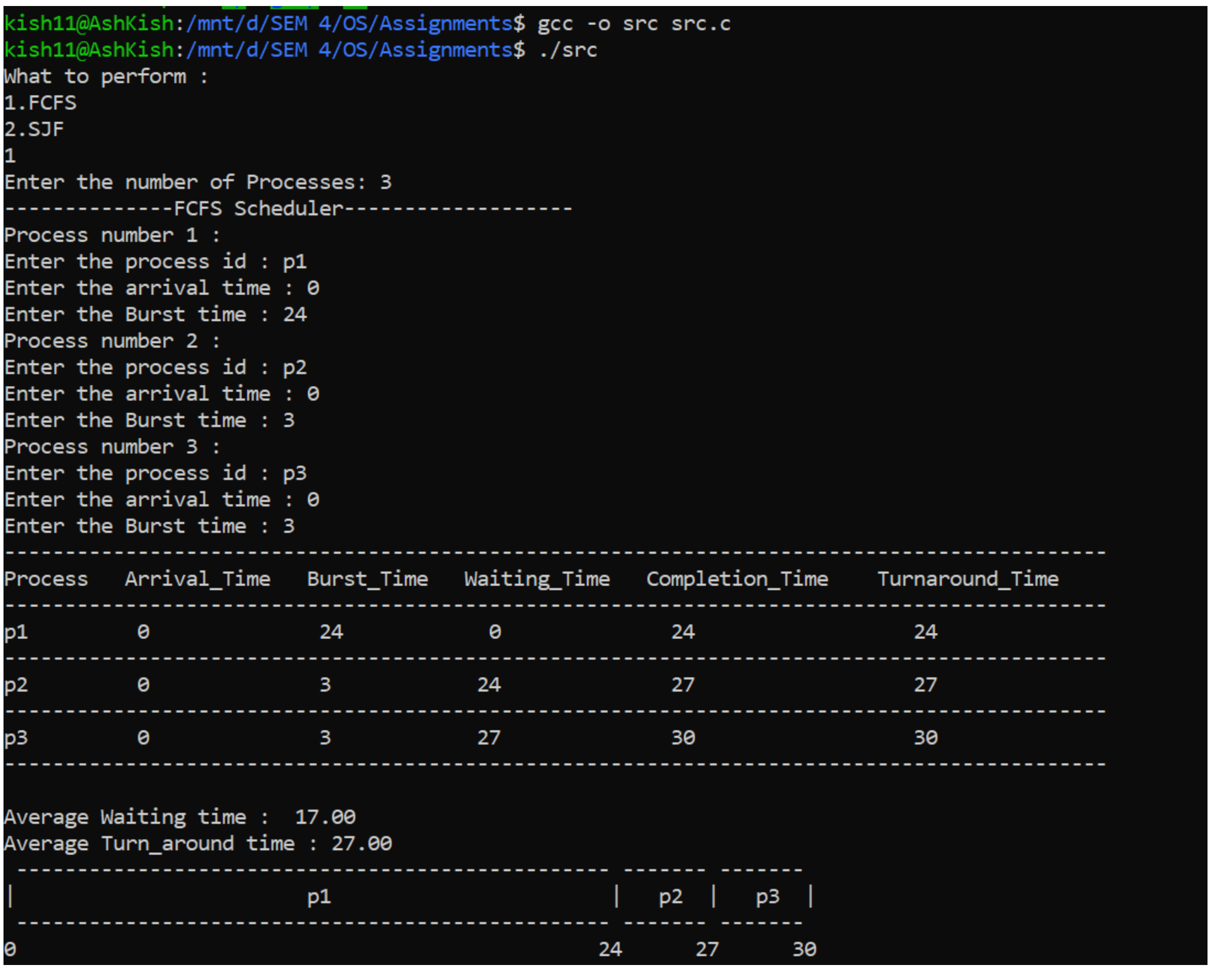
8: Print all the details of the processes.

9: Print the gantt chart and also print the average waiting and ta time.

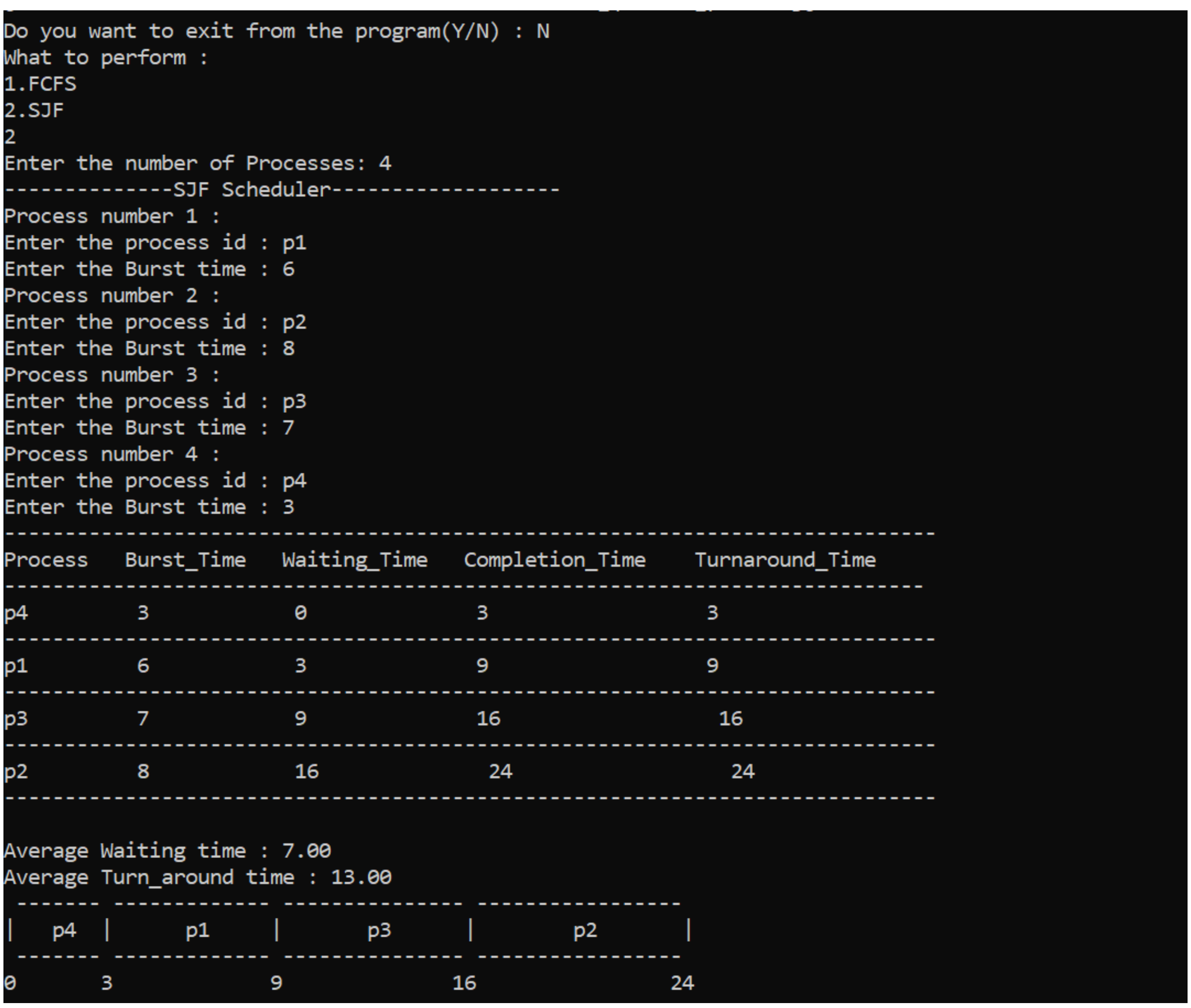
Source Code :

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct schedule *SCH;
typedef struct schedule{
    char process[3];
    int waiting;
    int arrival;
    int turn_around;
    int burst;
    int completion;
}sch;


void sortarrival(SCH P[],int n){
    for(int i=0;i<n;i++){
        int x=i;
        for(int j=i+1;j<n;j++){
            if(P[x]->arrival>P[j]->arrival){
                x=j;
            }
        }
        if(x!=i){
            SCH temp;
            temp=P[i];
            P[i]=P[x];
            P[x]=temp;
        }
    }
}

void sortburst(SCH P[],int n){
    for(int i=0;i<n;i++){
        int x=i;
        for(int j=i+1;j<n;j++){
            if(P[x]->burst>P[j]->burst){
                x=j;
            }
        }
        if(x!=i){
            SCH temp;
            temp=P[i];
            P[i]=P[x];
            P[x]=temp;
        }
    }
}

void gantt_chart(SCH P[], int 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]->process);
        for(j=0; j<P[i]->burst; j++) printf(" ");
        printf("\b");
        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("  ");
        printf(" ");
        if(P[i]->turn_around > 9) printf("\b");
        printf("%d", P[i]->turn_around);

    }
    printf("\n");
}


int main(){
    int n;
    char ch;
    do{
        printf("What to perform :\n1.FCFS\n2.SJF\n");
        int choice;
        scanf("%d",&choice);
        printf("Enter the number of Processes: ");
        scanf("%d",&n);
        if(choice==1){
            printf("--------------FCFS Scheduler-------------------\n");
            SCH P[n];
            for(int i=0;i<n;i++){
                printf("Process number %d : \n",i+1);
                P[i]=malloc(sizeof(sch));
                printf("Enter the process id : ");
                scanf("%s",P[i]->process);
                printf("Enter the arrival time : ");
                scanf("%d",&P[i]->arrival);
                printf("Enter the Burst time : ");
                scanf("%d",&P[i]->burst);
            }

            double avg_wait,trn_around;
            P[0]->waiting=0;
            avg_wait=P[0]->waiting;
            P[0]->completion=P[0]->burst+P[0]->arrival;
            P[0]->turn_around=P[0]->burst;
            trn_around=P[0]->turn_around;
            for(int i=1;i<n;i++){
                if(P[i-1]->completion>=P[i]->arrival){
                    P[i]->completion=P[i-1]->completion+P[i]->burst;
                }
                else P[i]->completion=P[i]->burst+P[i]->arrival;
                P[i]->waiting=P[i]->completion-(P[i]->burst+P[i]->arrival);
                avg_wait+=P[i]->waiting;
                P[i]->turn_around=P[i]->completion-P[i]->arrival;
                trn_around+=P[i]->turn_around;
            }
            printf("----------------------------------------------------------
---------------------------------\n");
            printf("Process   Arrival_Time   Burst_Time   Waiting_Time   Compl
etion_Time    Turnaround_Time\n");
            printf("----------------------------------------------------------
---------------------------------\n");
            for(int i=0;i<n;i++){
                printf("%s         %d              %d            %d          
   %d                  %d\n",P[i]->process,P[i]->arrival,P[i]->burst,P[i]-
>waiting,P[i]->completion,P[i]->turn_around);
                printf("------------------------------------------------------
-------------------------------------\n");
            }
            printf("\n");
            printf("Average Waiting time :  %.2f\n",avg_wait/n);
            printf("Average Turn_around time : %.2f\n",trn_around/n);
            gantt_chart(P,n);

        }
        else if(choice==2){
            SCH P[n];
            printf("--------------SJF Scheduler-------------------\n");
            for(int i=0;i<n;i++){
                printf("Process number %d : \n",i+1);
                P[i]=malloc(sizeof(sch));
                printf("Enter the process id : ");
                scanf("%s",P[i]->process);
                printf("Enter the Burst time : ");
                scanf("%d",&P[i]->burst);
            }
            sortburst(P,n);
            double avg_wait,trn_around;
            P[0]->waiting=0;
            avg_wait=P[0]->waiting;
            P[0]->completion=P[0]->burst;
            P[0]->turn_around=P[0]->completion;
            trn_around=P[0]->completion;
            for(int i=1;i<n;i++){
                P[i]->waiting=P[i-1]->completion;
                P[i]->completion=P[i-1]->completion+P[i]->burst;
                P[i]->turn_around=P[i]->completion;
                avg_wait+=P[i]->waiting;
                trn_around+=P[i]->turn_around;
            }
            printf("----------------------------------------------------------
-------------------\n");
            printf("Process   Burst_Time   Waiting_Time   Completion_Time    T
urnaround_Time\n");
             printf("---------------------------------------------------------
-------------------\n");
            for(int i=0;i<n;i++){
                printf("%s         %d            %d              %d
       %d\n",P[i]->process,P[i]->burst,P[i]->waiting,P[i]->completion,P[i]-
>turn_around);
                printf("------------------------------------------------------
-----------------------\n");

            }
            printf("\n");
            printf("Average Waiting time : %.2f\n",avg_wait/n);
            printf("Average Turn_around time : %.2f\n",trn_around/n);
            gantt_chart(P,n);

        }
        printf("Do you want to exit from the program(Y/N) : ");
        scanf("%s",&ch);
    }while(ch=='N');
}
```

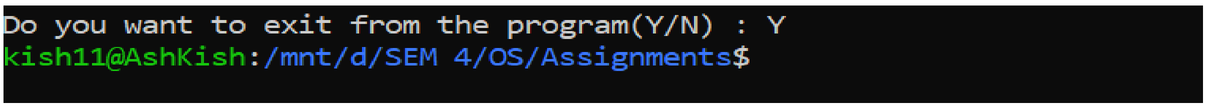
Output :



```
kish11@AshKish:/mnt/d/SEM 4/OS/Assignments$ gcc -o src src.c
kish11@AshKish:/mnt/d/SEM 4/OS/Assignments$ ./src
What to perform :
1.FCFS
2.SJF
1
Enter the number of Processes: 3
--------------FCFS Scheduler-------------------
Process number 1 :
Enter the process id : p1
Enter the arrival time : 0
Enter the Burst time : 24
Process number 2 :
Enter the process id : p2
Enter the arrival time : 0
Enter the Burst time : 3
Process number 3 :
Enter the process id : p3
Enter the arrival time : 0
Enter the Burst time : 3
------------------------------------------------------------------------------------------
Process   Arrival_Time   Burst_Time   Waiting_Time   Completion_Time    Turnaround_Time
------------------------------------------------------------------------------------------
p1         0              24             0              24                  24
------------------------------------------------------------------------------------------
p2         0              3             24              27                  27
------------------------------------------------------------------------------------------
p3         0              3             27              30                  30
------------------------------------------------------------------------------------------

Average Waiting time :  17.00
Average Turn_around time : 27.00
 ------------------------------------------------- ------- -------
|                        p1                       |  p2   |  p3   |
 ------------------------------------------------- ------- -------
0                                                24      27      30
```



```
Do you want to exit from the program(Y/N) : N
What to perform :
1.FCFS
2.SJF
2
Enter the number of Processes: 4
--------------SJF Scheduler------------------
Process number 1 :
Enter the process id : p1
Enter the Burst time : 6
Process number 2 :
Enter the process id : p2
Enter the Burst time : 8
Process number 3 :
Enter the process id : p3
Enter the Burst time : 7
Process number 4 :
Enter the process id : p4
Enter the Burst time : 3
-----------------------------------------------------------------------------
Process    Burst_Time    Waiting_Time    Completion_Time     Turnaround_Time
----------------------------------------------------------------------------
p4          3             0               3                   3
-----------------------------------------------------------------------------
p1          6             3               9                   9
-----------------------------------------------------------------------------
p3          7             9               16                  16
-----------------------------------------------------------------------------
p2          8             16              24                  24
-----------------------------------------------------------------------------

Average Waiting time : 7.00
Average Turn_around time : 13.00
 ------- ------------- -------------- -----------------
|   p4  |      p1      |       p3      |        p2        |
 ------- ------------- -------------- -----------------
0       3              9               16                 24
```



```
Do you want to exit from the program(Y/N) : Y
kish11@AshKish:/mnt/d/SEM 4/OS/Assignments$
```