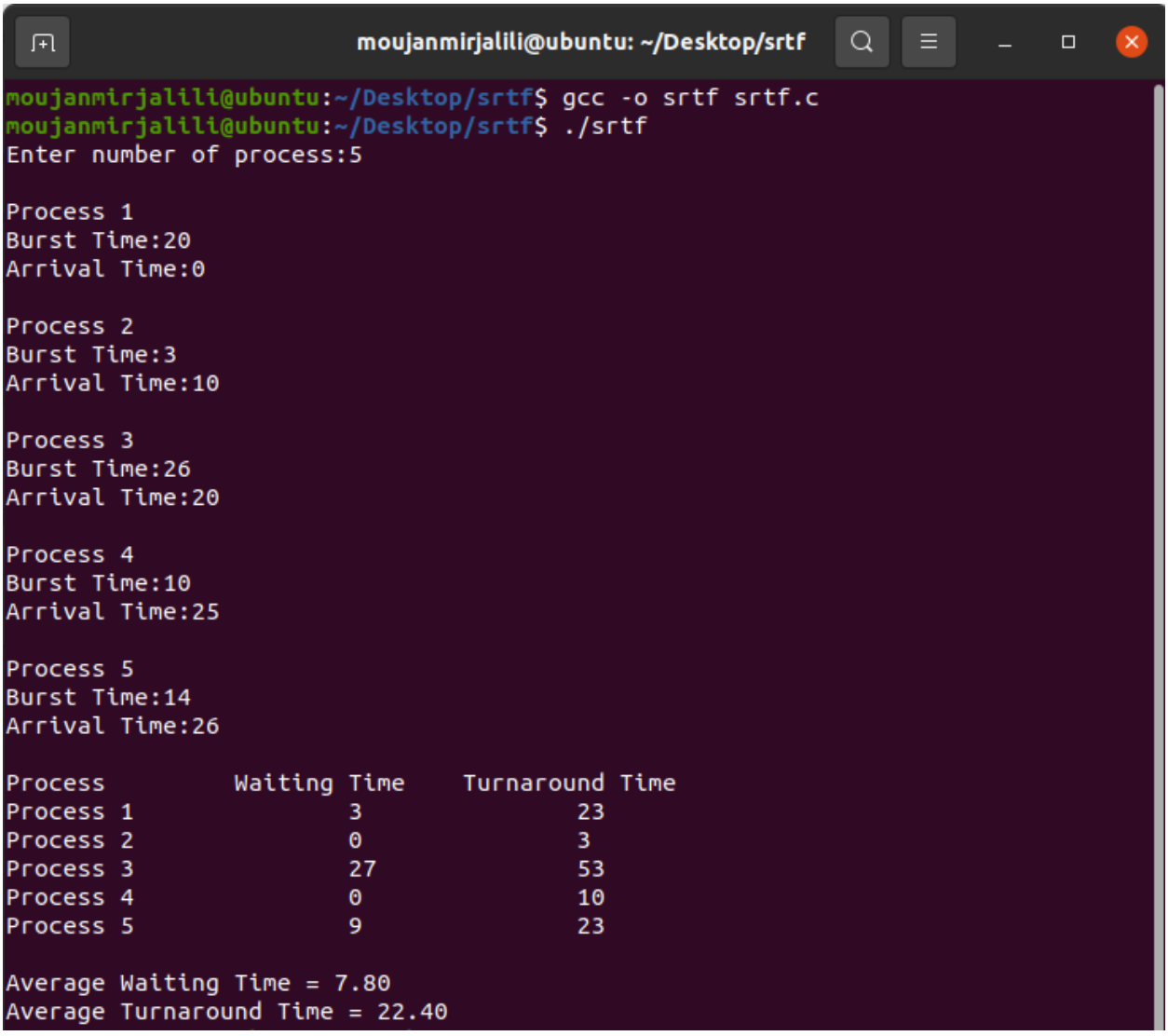
Open srtf.c ~/Desktop/srtf Save

```c
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>

struct process{
        int pid;
        int burst,const_burst;
        int wait,turnaround,arrival;
} p[10];

int n;
int timer = 0;

bool isFinished(){
    int sum = 0;
    for(int i=0; i<n; i++){
        sum += p[i].burst;
    }
    if(sum == 0){
        return true;
    }else{
        return false;
    }
}



int main(){
        int inp;

        printf("Enter number of process:");
        scanf("%d",&n);

        for(int i=0; i<n; i++){
                p[i].pid = i+1;
                printf("\nProcess %d\n",i+1);
                printf("Burst Time:");
                scanf("%d",&p[i].burst);
                p[i].const_burst=p[i].burst;
                printf("Arrival Time:");
                scanf("%d",&p[i].arrival);
        }


        while(!isFinished()){
                int min_burst=INT_MAX;
                int index_min_burst=-1;
                for(int i=0;i<n;i++){
                        if(p[i].arrival <= timer && p[i].burst<min_burst && p[i].burst!=0){
                                min_burst=p[i].burst;
                                index_min_burst=i;
                        }
                }
                timer++;
                if(min_burst!=INT_MAX){
                        p[index_min_burst].burst--;
                        if(p[index_min_burst].burst==0){
                                p[index_min_burst].turnaround=timer-p[index_min_burst].arrival;
                                p[index_min_burst].wait=timer-p[index_min_burst].arrival-p[index_min_burst].const_burst;
                        }
                }
        }

        printf("\nProcess\t\tWaiting Time\tTurnaround Time");
        for(int i=0;i<n;i++){
                printf("\nProcess %d\t\t%d\t\t%d", p[i].pid, p[i].wait, p[i].turnaround);
        }
        double avg_wait=0;
        double avg_turnaround=0;
        for(int i=0;i<n;i++){
                avg_wait+=p[i].wait;
                avg_turnaround+=p[i].turnaround;
        }
        avg_wait/=n;
        avg_turnaround/=n;
        printf("\n\nAverage Waiting Time = %0.2f",avg_wait);
        printf("\nAverage Turnaround Time = %0.2f\n",avg_turnaround);
        return 0;
}
```



این الگوریتم نسخه preemptive، SJF است. درSRTF، اجرای فرآیند را می توان پس از مدت زمان معینی (آمدن پردازه با burst time کم‌تر) متوقف کرد. با ورود هر فرآیند، این الگوریتم فرآیند با کمترین زمان باقی‌مانده burst timeاش از بین فهرست فرآیندهای موجود و فرآیند در حال اجرا زمان‌بندی می‌کند. هنگامی که تمام فرآیندها در صف آماده در دسترس هستند، هیچ قبضه‌ای انجام نمی‌شود و الگوریتم به عنوان SJF کار می‌کند.

```
moujanmirjalili@ubuntu:~/Desktop/srtf$ ./srtf
Enter number of process:5

Process 1
Burst Time:5
Arrival Time:2

Process 2
Burst Time:1
Arrival Time:3

Process 3
Burst Time:8
Arrival Time:3

Process 4
Burst Time:5
Arrival Time:1

Process 5
Burst Time:6
Arrival Time:4

Process         Waiting Time    Turnaround Time
Process 1               5               10
Process 2               0               1
Process 3               15              23
Process 4               1               6
Process 5               8               14

Average Waiting Time = 5.80
Average Turnaround Time = 10.80
```

process	Burst	Arrival
A	5	2
B	1	3
C	8	3
D	5	1
E	6	4

Shortest Remaining Time First - Preemptive Scheduling

Turnaround Time = Exit time – Arrival time ⇨ A = 12-2 = 10, B = 4-3 = 1, C = 26-3 = 23, D = 7-1 = 6, E = 18-4 = 14

⇨ Average Turnaround = (10+1+23+6+14)/5 = 10.8

Waiting time = Turn Around time – Burst time ⇨ A = 10-5 = 5, B = 1-1 = 0, C = 23-8 = 15, D = 6-5 = 1, E = 14-6 = 8

⇨ Average waiting time = (5+0+15+1+8)/5 = 5.8