

```
moujanmirjalili@ubuntu:~/Desktop/Lab8$ ./FCFS
Enter total number of processes(maximum 10): 4

Enter Process Burst Time
Process 1: 8
Process 2: 2
Process 3: 5
Process 4: 14

Process          Burst Time      Waiting Time      Turnaround Time
Process 1         8              0                 8
Process 2         2              8                10
Process 3         5              10               15
Process 4        14              15               29

Average Waiting Time:8.25
Average Turnaround Time:15.50
moujanmirjalili@ubuntu:~/Desktop/Lab8$
```

در اینجا به دلیل اینکه فرآیندی با زمان کوتاه ممکن است پشت فرآیندی با زمان طولانی منتظر بماند، الگوریتم بهینه نیست و ممکن است convoy effect رخ دهد.

```

1 #include<stdio.h>
2
3 typedef struct {
4     int pid;
5     int bt;
6     int wt;
7     int tt;
8 }Process;
9
10 int main()
11 {
12     Process process[10];
13     int n, i, j;
14     float avg_wt = 0, avg_tt = 0;
15     printf("Enter total number of processes(maximum 10): ");
16     scanf("%d",&n);
17
18     printf("\nEnter Process Burst Time\n");
19     for(i = 0; i < n;i++)
20     {
21         process[i].pid = i+1;
22         printf("Process %d: ",process[i].pid);
23         scanf("%d",&process[i].bt);
24     }
25
26     process[0].wt = 0; //waiting time for first process is 0
27
28     //calculating waiting time
29     for(i = 1; i < n;i++) {
30         process[i].wt = 0;
31         for(j = 0; j < i;j++)
32             process[i].wt += process[j].bt;
33     }
34
35     printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
36
37     //calculating turnaround time
38     for(i = 0; i < n;i++) {
39         process[i].tt = process[i].bt + process[i].wt;
40         avg_wt += process[i].wt;
41         avg_tt += process[i].tt;
42         printf("\nProcess %d\t\t%d\t\t%d\t\t%d", process[i].pid,process[i].bt,
43             process[i].wt,process[i].tt);
44     }
45
46     avg_wt /= n;
47     avg_tt /= n;
48     printf("\n\nAverage Waiting Time:%0.2f", avg_wt);
49     printf("\n\nAverage Turnaround Time:%0.2f\n", avg_tt);
50     return 0;
51 }

```

```
moujanmirjalili@ubuntu:~/Desktop/Lab8$ gcc -o SJF SJF.c
moujanmirjalili@ubuntu:~/Desktop/Lab8$ ./SJF
Enter number of process:4

Enter Process Burst Time
Process 1: 8
Process 2: 2
Process 3: 5
Process 4: 14

Process      Burst Time      Waiting Time      Turnaround Time
Process 2      2              0              2
Process 3      5              2              7
Process 1      8              7              15
Process 4     14              15              29

Average Waiting Time=6.00
Average Turnaround Time=13.25
```

در اینجا کمترین زمان انتظار را داریم و هر فرآیند با burst time کمتر، زودتر اجرا می‌شود.

```

1 #include<stdio.h>
2
3 typedef struct {
4     int pid;
5     int bt;
6     int wt;
7     int tt;
8 }Process;
9
10 void main()
11 {
12     Process process[10];
13     int n, i, j, total = 0, pos, temp;
14     float avg_wt = 0, avg_tt = 0;
15
16     printf("Enter number of process:");
17     scanf("%d",&n);
18
19     printf("\nEnter Process Burst Time\n");
20     for(i = 0; i < n;i++)
21     {
22         process[i].pid = i+1;
23         printf("Process %d: ",process[i].pid);
24         scanf("%d",&process[i].bt);
25     }
26
27
28     //sorting burst time in ascending order using selection sort
29     for(i = 0; i < n; i++) {
30         pos = i;
31         for(j = i+1; j < n;j++) {
32             if(process[j].bt < process[pos].bt)
33                 pos = j;
34         }
35
36         temp = process[i].bt;
37         process[i].bt = process[pos].bt;
38         process[pos].bt = temp;
39
40         temp = process[i].pid;
41         process[i].pid = process[pos].pid;
42         process[pos].pid = temp;
43     }
44
45     process[0].wt = 0;    //waiting time for first process will be zero
46
47     //calculate waiting time
48     for(i = 1; i < n; i++)
49     {
50         process[i].wt = 0;
51         for(j = 0; j < i; j++)
52             process[i].wt += process[j].bt;
53
54         total += process[i].wt;
55     }
56
57     avg_wt=(float)total/n;    //average waiting time
58     total=0;
59
60     printf("\nProcess\t\tBurst Time \tWaiting Time \tTurnaround Time");
61     for(i = 0; i < n; i++)
62     {
63         //calculate turnaround time
64         process[i].tt = process[i].bt + process[i].wt;
65         total += process[i].tt;
66         printf("\nProcess %d\t\t%d\t\t%d\t\t%d", process[i].pid, process[i].bt,
67             process[i].wt, process[i].tt);
68     }
69
70     avg_tt = (float)total/n;    //average turnaround time
71     printf("\n\nAverage Waiting Time=%0.2f",avg_wt);
72     printf("\n\nAverage Turnaround Time=%0.2f\n",avg_tt);
73 }

```

```

moujanmirjalili@ubuntu:~/Desktop/88$ gcc -o priority priority.c
moujanmirjalili@ubuntu:~/Desktop/88$ ./priority
Enter Total Number of Process:4

Enter Burst Time and Priority

Process 1
Burst Time:21
Priority:2

Process 2
Burst Time:3
Priority:1

Process 3
Burst Time:6
Priority:4

Process 4
Burst Time:2
Priority:3

Process      Burst Time      Waiting Time      Turnaround Time
Process 2      3              0                3
Process 1     21              3               24
Process 4      2              24              26
Process 3      6              26              32

Average Waiting Time=13.25
Average Turnaround Time=21.25

```

در اینجا فرآیند با الویت بالاتر، زودتر انجام می‌شود.

```

1 #include<stdio.h>
2
3 typedef struct {
4     int pid;
5     int bt;
6     int pr; // priority
7     int wt;
8     int tt;
9 }Process;
10
11 int main()
12 {
13     Process process[10];
14     int n, i, j, total = 0, pos, temp;
15     float avg_wt = 0, avg_tt = 0;
16     printf("Enter Total Number of Process:");
17     scanf("%d",&n);
18
19     printf("\nEnter Burst Time and Priority\n");
20     for(i = 0; i < n; i++)
21     {
22         process[i].pid = i+1;
23         printf("\nProcess %d\n",i+1);
24         printf("Burst Time:");
25         scanf("%d",&process[i].bt);
26         printf("Priority:");
27         scanf("%d",&process[i].pr);
28     }
29
30     //sorting burst time, priority and process number in ascending order using
    selection sort
31     for(i=0;i<n;i++)
32     {
33         pos=i;
34         for(j = i+1; j < n; j++)
35         {
36             if(process[j].pr < process[pos].pr)
37                 pos = j;
38         }
39
40         temp=process[i].pr;
41         process[i].pr = process[pos].pr;
42         process[pos].pr = temp;
43
44         temp = process[i].bt;
45         process[i].bt = process[pos].bt;
46         process[pos].bt = temp;
47
48         temp = process[i].pid;
49         process[i].pid = process[pos].pid;
50         process[pos].pid = temp;
51     }
52
53     process[0].wt = 0; //waiting time for first process is zero
54
55     //calculate waiting time
56     for(i = 1; i < n; i++)
57     {
58         process[i].wt = 0;
59         for(j = 0; j < i; j++)
60             process[i].wt += process[j].bt;
61
62         total += process[i].wt;
63     }
64
65     avg_wt = (float)total/n; //average waiting time
66     total=0;
67
68     printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
69     for(i = 0; i < n; i++)
70     {
71         //calculate turnaround time
72         process[i].tt = process[i].bt + process[i].wt;
73         total += process[i].tt;
74         printf("\nProcess %d\t\t%d\t\t%d\t\t%d", process[i].pid, process[i].bt,
        process[i].wt, process[i].tt);
75     }
76
77     avg_tt = (float)total/n; //average turnaround time
78
79     printf("\n\nAverage Waiting Time=%0.2f",avg_wt);
80     printf("\n\nAverage Turnaround Time=%0.2f\n",avg_tt);
81
82     return 0;
83 }

```

```
moujanmirjalili@ubuntu:~/Desktop/Lab8$ ./RoundRobin
Enter number of process:4
Enter q:3

Enter Process Burst Time
Process 1: 6
Process 2: 3
Process 3: 7
Process 4: 1

Process          Waiting Time    Turnaround Time
Process 1          7              13
Process 2          3              6
Process 3         10              17
Process 4          9              10
avg wait time is 7.250000.
avg run time is 11.500000.
```

در این فرآیند پردازشها محدودیت زمانی دارند و همه برنامه‌ها تا حد خوبی پیش می‌روند و زمان انتظار بهینه تر است.

```

1 #include<stdio.h>
2
3 typedef struct {
4     int pid;
5     int bt;
6     int wt;
7     int tt;
8 }Process;
9
10 int n, q, cur, total_rn, total_wt;
11 int tmp[100], l, r;
12
13 int main() {
14     Process process[10];
15     printf("Enter number of process:");
16     scanf("%d",&n);
17     printf("Enter q:");
18     scanf("%d",&q);
19
20     printf("\nEnter Process Burst Time\n");
21     for(int i = 0; i < n; i++) {
22         printf("Process %d: ",i+1);
23         scanf("%d", &process[i].bt);
24         tmp[r++] = i;
25         process[i].wt = -process[i].bt;
26     }
27
28     for(; l < r; l++) {
29         int i = tmp[l];
30         if(process[i].bt <= q) {
31             cur += process[i].bt;
32             process[i].tt = cur;
33             process[i].wt += process[i].tt;
34         }
35         else {
36             cur += q;
37             process[i].bt -= q;
38             tmp[r++] = i;
39         }
40     }
41     printf("\nProcess\t\tWaiting Time\tTurnaround Time");
42     for(int i = 0; i < n; i++) {
43         printf("\nProcess %d\t\t%d\t\t%d", i+1, process[i].wt, process[i].tt);
44         total_wt += process[i].wt;
45         total_rn += process[i].tt;
46     }
47     printf("\n");
48     float avg_wait = 1.0 * total_wt / n;
49     float avg_run = 1.0 * total_rn / n;
50     printf("avg wait time is %f.\n", avg_wait);
51     printf("avg run time is %f.\n", avg_run);
52 }

```