Write a C program in LINUX to implement Process scheduling algorithms and compare.

A. First Come First Serve (FCFS) Scheduling (fcfs.c)

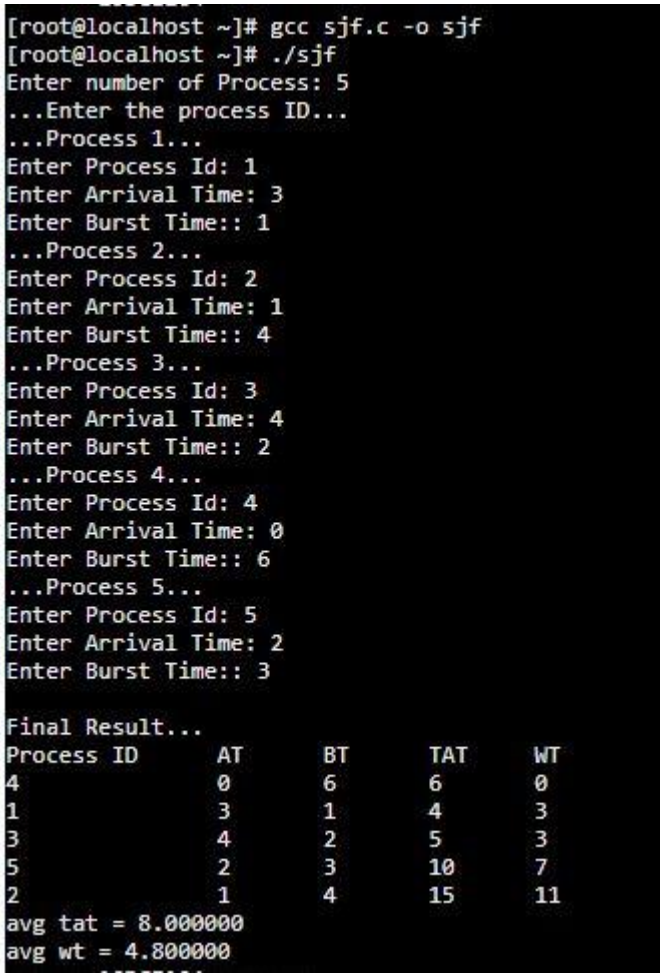
```
"fcfs.c" 43L, 831B written
[root@localhost ~]# gcc fcfs.c -o fcfs
[root@localhost ~]# ./fcfs
Enter total number of processes3

Enter Process Burst TimenP[1]:24
P[2]:3
P[3]:3

Process         BT              WT      TAT
P[1]            24              0       24
P[2]            3               23      26
P[3]            3               25      28

Average Waiting Time:16
Average Turnaround Time:26
```

B. Shortest-Job-First (SJF) Scheduling (sjf.c)



```
[root@localhost ~]# gcc sjf.c -o sjf
[root@localhost ~]# ./sjf
Enter number of Process: 5
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 3
Enter Burst Time:: 1
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 1
Enter Burst Time:: 4
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 4
Enter Burst Time:: 2
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 0
Enter Burst Time:: 6
...Process 5...
Enter Process Id: 5
Enter Arrival Time: 2
Enter Burst Time:: 3

Final Result...
Process ID      AT      BT      TAT     WT
4               0       6       6       0
1               3       1       4       3
3               4       2       5       3
5               2       3       10      7
2               1       4       15      11
avg tat = 8.000000
avg wt = 4.800000
```

C. Priority Scheduling (Non-preemption) after completion extend on Preemption. (priority.c)

```
[root@localhost ~]# gcc priority.c -o priority
[root@localhost ~]# ./priority
Enter the number of process : 5

 Enter process :Enter BT and  priorities

Process no 1 : 4
2

Process no 2 : 3
3

Process no 3 : 1
4

Process no 4 : 5
5

Process no 5 : 2
5


 Job     BT      WT      TAT   Priority

 1       4       0       4      2

 4       5       1       6      5

 5       2       5       7      5

 3       1       9       10     4

 2       3       11      14     3

 Average Turn Around Time : 8.200000

 Average Wait Time : 5.200000
```

D. Round Robin(RR) Scheduling (rr.c)

```
[root@localhost ~]# gcc priority.c -o priority
[root@localhost ~]# ./priority
Enter the number of process : 5

 Enter process :Enter BT and  priorities

Process no 1 : 4
2

Process no 2 : 3
3

Process no 3 : 1
4

Process no 4 : 5
5

Process no 5 : 2
5


 Job     BT      WT      TAT   Priority

 1       4       0       4      2

 4       5       1       6      5

 5       2       5       7      5

 3       1       9       10     4

 2       3       11      14     3

 Average Turn Around Time : 8.200000

 Average Wait Time : 5.200000
```

Simulate Following Page Replacement Algorithms.

A. First In First Out Algorithm (f_in_f_out.c)

```
 ENTER THE NUMBER OF PAGES:
12

 ENTER THE PAGE NUMBER :
1 2 3 4 5 1 3 1 6 3 2 3

 ENTER THE NUMBER OF FRAMES :4
Page Fault Is 9
```

B. Least Recently Used Algorithm (least_recently.cpp)

```
Enter Page reference string(seperate it with comma, no extra space) : 4,3
Enter Frame size : 4
Page Fault : 2
```

C. Optimal Algorithm (optimal.c)

```
Enter number of frames: 12
Enter number of pages: 2
Enter page reference string: 4 9

4       -1      -1      -1      -1      -1      -1      -1      -1      -1      -1      -1
4       9       -1      -1      -1      -1      -1      -1      -1      -1      -1      -1

Total Page Faults = 2
```

Write a C program in LINUX to perform Memory allocation algorithms and Calculate Internal and External Fragmentation.

A. First Fit (first_fit.c)

```
[root@localhost ~]# gcc ff.c -o a
[root@localhost ~]# ./a
Enter no. of blocks: 3

Enter size of each block: 8
10
12

Enter no. of processes: 10

Enter size of each process: 1^C
[root@localhost ~]# ./a
Enter no. of blocks: 3

Enter size of each block: 8
10
12

Enter no. of processes: 3

Enter size of each process: 10
12
14

Block no.       size            process no.             size
1               8               Not allocated
2               10              1                       10
3               12              2                       12[root@localhost ~]#
```

B. Best Fit (best_fit.c)

```
[root@localhost ~]# gcc bestfit.c -o b
[root@localhost ~]# ./b
Enter no of Blocks.
3
Enter the 0st Block size:8
Enter the 1st Block size:10
Enter the 2st Block size:12
Enter no of Process.
3
Enter the size of 0st Process:10
Enter the size of 1st Process:12
Enter the size of 2st Process:14
The Process 0 allocated to 10
The Process 1 allocated to 12
The Process 2 is not allocated
```

C. Worst Fit (worst_fit.c)

```
[root@localhost ~]# gcc wf.c -o a
[root@localhost ~]# ./a

Process No.     Process Size    Block no.
 1              212             5
 2              417             2
 3              112             4
 4              426             Not Allocated
```

Created by Manan Patel.