

1)C program to create Zombie Process

```

1#include<sys/types.h>
2#include<stdio.h>
3#include<stdlib.h>
4#include<unistd.h>
5
6int main()
7{
8pid_t p;
9p=fork();
10
11printf("\nValue of p : %d",p);
12
13if(p>0) // Parent process --> Sleeping
14{
15    printf("\nInside Parent's process");
16    printf("\nProcess ID : %d",getpid());
17
18    sleep(30);
19    //exit(0); // Parent doesn't exits and waits for the child to return back by leaving the child alive
20}
21
22if(p==0) // Child process --> exiting while the parent is sleeping
23{
24    printf("\nInside Child's process");
25    printf("\nProcess ID of child : %d",getpid());
26    printf("\nProcess ID of parent : %d",getppid());
27    //sleep(30)
28    exit(0);
29}
30return 0;
31}

```

Output

```

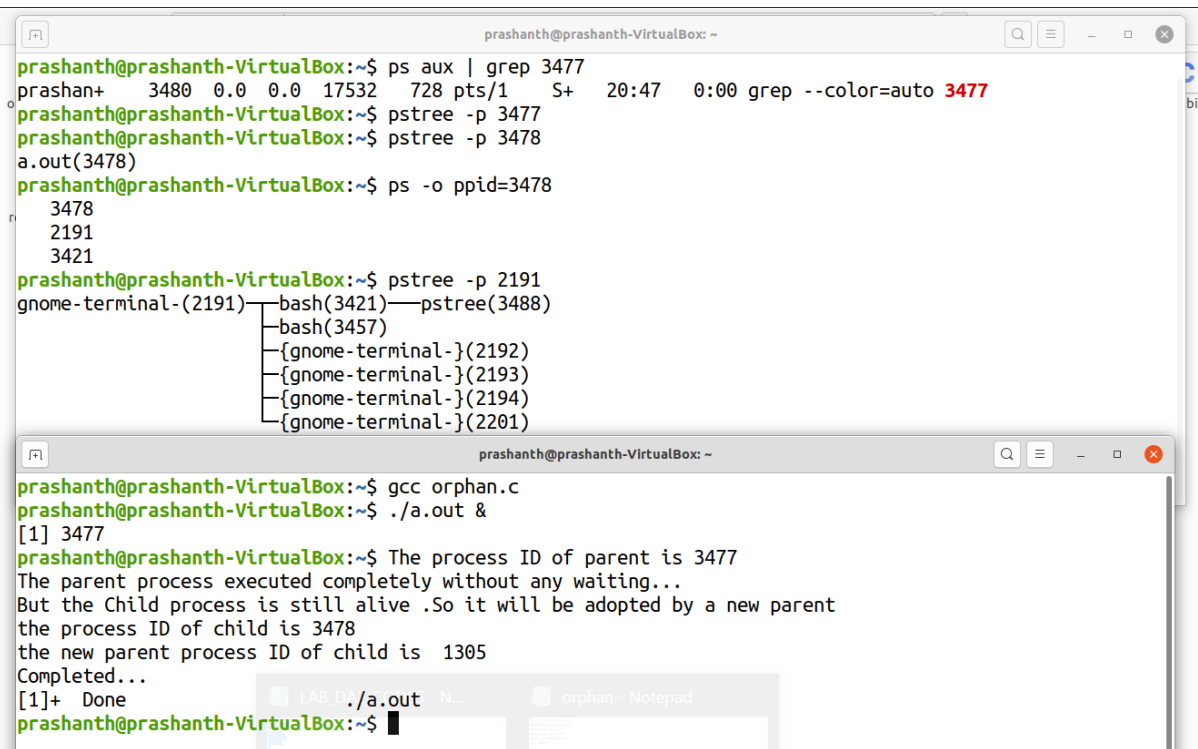
prashanth@prashanth-VirtualBox:~/Process$ ./a.out &
[1] 2465
prashanth@prashanth-VirtualBox:~/Process$ pstree -p 2465
a.out(2465)---a.out(2466)
prashanth@prashanth-VirtualBox:~/Process$ ps aux | grep 2466
prashan+  2466  0.0  0.0    0    0 pts/0    Z   08:30   0:00 [a.out] <defunct>
prashan+  2470  0.0  0.0 17532  664 pts/0    S+  08:30   0:00 grep --color=auto 2466
prashanth@prashanth-VirtualBox:~/Process$ ps aux | grep 2466
prashan+  2466  0.0  0.0    0    0 pts/0    Z   08:30   0:00 [a.out] <defunct>
prashan+  2472  0.0  0.0 17532  736 pts/0    S+  08:30   0:00 grep --color=auto 2466
prashanth@prashanth-VirtualBox:~/Process$ ps aux | grep 2466
prashan+  2466  0.0  0.0    0    0 pts/0    Z   08:30   0:00 [a.out] <defunct>
prashan+  2474  0.0  0.0 17532  736 pts/0    S+  08:31   0:00 grep --color=auto 2466
prashanth@prashanth-VirtualBox:~/Process$ ps aux | grep 2466
prashan+  2466  0.0  0.0    0    0 pts/0    Z   08:30   0:00 [a.out] <defunct>
prashan+  2476  0.0  0.0 17532  664 pts/0    S+  08:31   0:00 grep --color=auto 2466
prashanth@prashanth-VirtualBox:~/Process$ ps aux | grep 2466
prashan+  2478  0.0  0.0 17532  664 pts/0    S+  08:31   0:00 grep --color=auto 2466
[1]+  Done                  ./a.out
prashanth@prashanth-VirtualBox:~/Process$

```

2) C program to create orphan process

```
1#include<sys/types.h>
2#include<stdio.h>
3#include<stdlib.h>
4#include<unistd.h>
5
6int main()
7{
8pid_t p;
9p=fork();
10
11printf("\nValue of p : %d",p);
12
13if(p>0)    // Parent process --> exiting while the child is sleeping
14{
15{
16    printf("\nInside Parent's process");
17    printf("\nProcess ID : %d",getpid());
18
19    //sleep(30);
20    exit(0);    // Parent exits and doesn't wait for the child to return back by leaving the child orphan
21}
22
23if(p==0)    // Child process --> Sleeping
24{
25    printf("\nInside Child's process");
26    printf("\nProcess ID of child : %d",getpid());
27    printf("\nProcess ID of parent : %d",getppid());
28    sleep(30)
29    //exit(0);
30}
31return 0;
32}
```

Output



The screenshot shows a terminal window with the following commands and output:

```
prashanth@prashanth-VirtualBox: ~$ ps aux | grep 3477
prashanth+ 3480  0.0  0.0 17532  728 pts/1    S+   20:47   0:00 grep --color=auto 3477
prashanth@prashanth-VirtualBox:~$ pstree -p 3477
prashanth@prashanth-VirtualBox:~$ pstree -p 3478
a.out(3478)
prashanth@prashanth-VirtualBox:~$ ps -o ppid=3478
3478
2191
3421
prashanth@prashanth-VirtualBox:~$ pstree -p 2191
gnome-terminal-(2191)─bash(3421)─pstree(3488)
                        │
                        └─bash(3457)
                           │
                           ├──{gnome-terminal-}(2192)
                           ├──{gnome-terminal-}(2193)
                           ├──{gnome-terminal-}(2194)
                           └─{gnome-terminal-}(2201)

prashanth@prashanth-VirtualBox:~$ gcc orphan.c
prashanth@prashanth-VirtualBox:~$ ./a.out &
[1] 3477
prashanth@prashanth-VirtualBox:~$ The process ID of parent is 3477
The parent process executed completely without any waiting...
But the Child process is still alive .So it will be adopted by a new parent
the process ID of child is 3478
the new parent process ID of child is 1305
Completed...
[1]+  Done                  LAB 04: ./a.out - N...
prashanth@prashanth-VirtualBox:~$
```

The output demonstrates that the parent process (PID 3477) exits before the child process (PID 3478) finishes. The child process is then adopted by a new parent (PID 1305), as shown by the change in its ppid value.

3) First come First Serve Scheduling Algorithm

```
1#include<stdio.h>
2
3int main()
4{
5    int n,time_burst[20],time_wait[20],turn_wait[20],avgtime_wait=0,avgturn_wait=0,i,j;
6    printf("Enter total number of processes:");
7    scanf("%d",&n);
8
9    printf("Enter Process Burst Timen for \n");
10   for(i=0;i<n;i++)
11   {
12       printf("P[%d]:",i+1);
13       scanf("%d",&time_burst[i]);
14   }
15
16   time_wait[0]=0;
17
18   for(i=1;i<n;i++)
19   {
20       time_wait[i]=0;
21       for(j=0;j<i;j++)
22           time_wait[i]+=time_burst[j];
23   }
24
25   printf("\nProcess\tBurst_Time\tWaiting_Time\tTurnaround_Time");
26
27   for(i=0;i<n;i++)
28   {
29       turn_wait[i]=time_burst[i]+time_wait[i];
30       avgtime_wait+=time_wait[i];
31       avgturn_wait+=turn_wait[i];
32       printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,time_burst[i],time_wait[i],turn_wait[i]);
33   }
34
35   avgtime_wait/=i;
36   avgturn_wait/=i;
37   printf("\nAverage Waiting Time:%d",avgtime_wait);
38   printf("\nAverage Turnaround Time:%d\n",avgturn_wait);
39
40   return 0;
41 }
```

Output

```
prashanth@prashanth-VirtualBox:~$ gcc fcfs.c
prashanth@prashanth-VirtualBox:~$ ./a.out
Enter total number of processes:8
Enter Process Burst Timen for
P[1]:9
P[2]:7
P[3]:5
P[4]:3
P[5]:2
P[6]:1
P[7]:6
P[8]:4

Process Burst_Time      Waiting_Time      Turnaround_Time
P[1]          9             0                9
P[2]          7             9               16
P[3]          5            16               21
P[4]          3            21               24
P[5]          2            24               26
P[6]          1            26               27
P[7]          6            27               33
P[8]          4            33               37
Average Waiting Time:19
Average Turnaround Time:24
prashanth@prashanth-VirtualBox:~$
```

4) Shortest Job First scheduling algorithm

Code:

```
1#include<stdio.h>
2int main()
3{
4    int time_burst[20],p[20],wait[20],time_turn[20],i,j,n,total=0,pos,temp;
5    float time_avgwait,time_avg_turn;
6    printf("Enter number of process:");
7    scanf("%d",&n);
8
9    printf("Enter Burst Time:\n");
10   for(i=0;i<n;i++)
11   {
12       printf("p%d:",i+1);
13       scanf("%d",&time_burst[i]);
14       p[i]=i+1;
15   }
16
17   //sorting of time_burst times
18   for(i=0;i<n;i++)
19   {
20       pos=i;
21       for(j=i+1;j<n;j++)
22       {
23           if(time_burst[j]<time_burst[pos])
24               pos=j;
25       }
26
27       temp=time_burst[i];
28       time_burst[i]=time_burst[pos];
29       time_burst[pos]=temp;
30
31       temp=p[i];
32       p[i]=p[pos];
33       p[pos]=temp;
34   }
35
36   wait[0]=0;
37
38
39   for(i=1;i<n;i++)
40   {
41       wait[i]=0;
42       for(j=0;j<i;j++)
43           wait[i]+=time_burst[j];
44
45       total+=wait[i];
46   }
47
48   time_avgwait=(float)total/n;
49   total=0;
50
51   printf("\nProcess\tBurst_Time\tWaiting_Time\tTurnaround_Time");
52   for(i=0;i<n;i++)
53   {
54       time_turn[i]=time_burst[i]+wait[i];
55       total+=time_turn[i];
56       printf("\np%d\t\t%d\t\t%d\t\t%d",p[i],time_burst[i],wait[i],time_turn[i]);
57   }
58
59   time_avg_turn=(float)total/n;
60   printf("\nAverage Waiting Time=%f",time_avgwait);
61   printf("\nAverage Turnaround Time=%f\n",time_avg_turn);
62
63   return 0;
```

Output:

```
prashanth@prashanth-VirtualBox:~$ gcc sjf.c
```

```
prashanth@prashanth-VirtualBox:~$ ./a.out
```

```
Enter number of process:5
```

```
Enter Burst Time:
```

```
p1:8
```

```
p2:6
```

```
p3:4
```

```
p4:2
```

```
p5:10
```

Process	Burst_Time	Waiting_Time	Turnaround_Time
p4	2	0	2
p3	4	2	6
p2	6	6	12
p1	8	12	20
p5	10	20	30

```
Average Waiting Time=8.000000
```

```
Average Turnaround Time=14.000000
```

5) Round Robin Scheduling Algorithm

```
1#include<stdio.h>
2
3int main()
4{
5    int i,n, total = 0, x, counter = 0, time_quantum;
6    int wait = 0, turn = 0, arrival[10], burst[10], temp[10];
7    float avgwait, avgturn;
8    printf("Enter Total Number of Processes:");
9    scanf("%d", &n);
10    x = n;
11    printf("Enter Details of Process :\n");
12    for(i = 0; i <n; i++)
13    {
14        printf("Enter arrival Time for P%d:",i+1);
15        scanf("%d", &arrival[i]);
16        printf("Enter burst Time for P%d:",i+1);
17        scanf("%d", &burst[i]);
18        temp[i] = burst[i];
19    }
20
21    printf("Enter Time Quantum:");
22    scanf("%d", &time_quantum);
23    printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time");
24    for(total = 0, i = 0; x != 0;)
25    {
26        if(temp[i] <= time_quantum && temp[i] > 0)
27        {
28            total = total + temp[i];
29            temp[i] = 0;
30            counter = 1;
31        }
32        else if(temp[i] > 0)
33        {
34            temp[i] = temp[i] - time_quantum;
35            total = total + time_quantum;
36        }
37        if(temp[i] == 0 && counter == 1)
38        {
39            x--;
40            printf("\nProcess[%d]\t\t%d\t\t%d\t\t%d", i + 1, burst[i], total - arrival[i], total - arrival[i] - burst[i]);
41            wait = wait + total - arrival[i] - burst[i];
42            turn = turn + total - arrival[i];
43            counter = 0;
44        }
45        if(i == n - 1)
46        {
47            i = 0;
48        }
49        else if(arrival[i + 1] <= total)
50        {
51            i++;
52        }
53        else
54        {
55            i = 0;
56        }
57    }
58
59    avgwait = wait * 1.0 / n;
60    avgturn = turn * 1.0 / n;
61    printf("\nAverage Waiting Time:%f", avgwait);
62    printf("\nAvg Turnaround Time:%f", avgturn);
63    return 0;
64}
```

Output

```
prashanth@prashanth-VirtualBox:~$ gcc roundrobin.c
```

```
prashanth@prashanth-VirtualBox:~$ ./a.out
```

```
Enter Total Number of Processes:5
```

```
Enter Details of Process :
```

```
Enter arrival Time for P1:2
```

```
Enter burst Time for P1:4
```

```
Enter arrival Time for P2:1
```

```
Enter burst Time for P2:6
```

```
Enter arrival Time for P3:0
```

```
Enter burst Time for P3:4
```

```
Enter arrival Time for P4:6
```

```
Enter burst Time for P4:2
```

```
Enter arrival Time for P5:9
```

```
Enter burst Time for P5:5
```

```
Enter Time Quantum:4
```

Process ID	Burst_Time	Turnaround_Time	Waiting_Time
Process[1]	4	2	-2
Process[3]	4	12	8
Process[4]	2	8	6
Process[2]	6	19	13
Process[5]	5	12	7

```
Average Waiting Time:6.400000
```

```
Avg Turnaround Time:10.600000prashanth@prashanth-VirtualBox:~$
```

```

1#include<stdio.h>
2int main()
3{
4    int time_burst[20],p[20],time_wait[20],time_turn[20],pr[20],i,j,n,total=0,pos,temp,avgttime_wait,avgttime_turn;
5    printf("Enter Total Number of Process:");
6    scanf("%d",&n);
7
8    printf("Enter Burst Time and Priority\n");
9    for(i=0;i<n;i++)
10    {
11        printf("P[%d]\n",i+1);
12        printf("Burst Time:");
13        scanf("%d",&time_burst[i]);
14        printf("Priority:");
15        scanf("%d",&pr[i]);
16        p[i]=i+1;
17    }
18
19    for(i=0;i<n;i++)
20    {
21        pos=i;
22        for(j=i+1;j<n;j++)
23        {
24            if(pr[j]<pr[pos])
25                pos=j;
26        }
27
28        temp=pr[i];
29        pr[i]=pr[pos];
30        pr[pos]=temp;
31        temp=time_burst[i];
32        time_burst[i]=time_burst[pos];
33        time_burst[pos]=temp;
34        temp=p[i];
35        p[i]=p[pos];
36        p[pos]=temp;
37    }
38
39    time_wait[0]=0;
40    for(i=1;i<n;i++)
41    {
42        time_wait[i]=0;
43        for(j=0;j<i;j++)
44            time_wait[i]+=time_burst[j];
45
46        total+=time_wait[i];
47    }
48
49    avgttime_wait=total/n;
50    total=0;
51
52    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time");
53    for(i=0;i<n;i++)
54    {
55        time_turn[i]=time_burst[i]+time_wait[i];
56        total+=time_turn[i];
57        printf("\nP[%d]\t\t%d\t\t%d\t\t%d\t\t%d",p[i],time_burst[i],time_wait[i],time_turn[i]);
58    }
59
60    avgttime_turn=total/n;
61    printf("\nAverage Waiting Time=%d",avgttime_wait);
62    printf("\nAverage Turnaround Time=%d\n",avgttime_turn);
63
64    return 0;
65}

```


Output

```
prashanth@prashanth-VirtualBox:~$ gcc psa.c
prashanth@prashanth-VirtualBox:~$ ./a.out
Enter Total Number of Process:5
Enter Burst Time and Priority
P[1]
Burst Time:10
Priority:2
P[2]
Burst Time:4
Priority:1
P[3]
Burst Time:6
Priority:4
P[4]
Burst Time:7
Priority:3
P[5]
Burst Time:8
Priority:5

Process Burst Time      Waiting Time      Turnaround Time
P[2]          4          0          4
P[1]         10          4         14
P[4]          7         14         21
P[3]          6         21         27
P[5]          8         27         35
Average Waiting Time=13
Average Turnaround Time=20
prashanth@prashanth-VirtualBox:~$ █
```