



```
#include <stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main()
{
pid_t p;
printf("before fork\n");
p=fork();
if(p==0)
{
printf("I am child having id %d\n",getpid());
printf("My parent's id is %d\n",getpid());
}
else
{
printf("M child's id is %d\n",p);
printf("I am parent having id %d\n",getpid());
}
printf("common\n");
}
```

guest-oxGE51@cn28-HP-ProDesk-400-G1-SFF: ~/Desktop

guest-oxGE51@cn28-HP-ProDesk-400-G1-SFF:~\$ cd Desktop

guest-oxGE51@cn28-HP-ProDesk-400-G1-SFF:~/Desktop\$ cc programas.c

guest-oxGE51@cn28-HP-ProDesk-400-G1-SFF:~/Desktop\$./a.out

before fork

M child's id is 3488

I am parent having id 3487

common

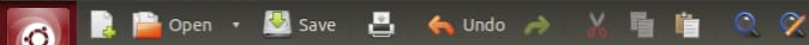
I am child having id 3488

My parent's id is 3488

common

guest-oxGE51@cn28-HP-ProDesk-400-G1-SFF:~/Desktop\$

```
{  
printf("M child's id is %d\n",p);  
printf("I am parent having id %d\n",getpid());  
}  
printf("common\n");  
}
```



Filecopying.c x copyfile.c x hello.c x

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    FILE *fptr1, *fptr2;
    char filename[100], c;
    printf("Enter the filename to open for reading \n");
    scanf("%s", filename);
    fptr1 = fopen(filename, "r");
    if (fptr1 == NULL){
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);
    fptr2 = fopen(filename, "w");
    if (fptr2 == NULL){
        printf("Cannot open file %s \n", filename);
        exit(0);
    }
    c = fgetc(fptr1);
    while (c != EOF){
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }
    printf("\nContents copied to %s", filename);
    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```

```
guest-S2XIFJ@cn28-HP-ProDesk-400-G1-SFF: ~/Desktop
guest-S2XIFJ@cn28-HP-ProDesk-400-G1-SFF:~$ cd Desktop
guest-S2XIFJ@cn28-HP-ProDesk-400-G1-SFF:~/Desktop$ cc copyfile.c
guest-S2XIFJ@cn28-HP-ProDesk-400-G1-SFF:~/Desktop$ ./a.out
Enter the filename to open for reading
file3.txt
Cannot open file file3.txt
guest-S2XIFJ@cn28-HP-ProDesk-400-G1-SFF:~/Desktop$ cc copyfile.c
guest-S2XIFJ@cn28-HP-ProDesk-400-G1-SFF:~/Desktop$ ./a.out
Enter the filename to open for reading
filecopying.c
Enter the filename to open for writing
hello.c
Contents copied to hello.c
```

```
    c = fgetc(fp1);
}
printf("\nContents copied to %s", filename);
fclose(fp1);
fclose(fp2);
return 0;
}
```

(globals)

Project C FCFSOS.C

```
1 #include<stdio.h>
2 int main()
3 {
4     int bt[10]={0},at[10]={0},tat[10]={0},wt[10]={0},ct[10]={0};
5     int n,sum=0;
6     float totalTAT=0,totalWT=0;
7     printf("Enter number of processes: ");
8     scanf("%d",&n);
9     printf("Enter arrival time and burst time for each process\n\n");
10    for(int i=0;i<n;i++)
11    {
12        printf("Arrival time of process[%d] ",i+1);
13        scanf("%d",&at[i]);
14        printf("Burst time of process[%d] ",i+1);
15        scanf("%d",&bt[i]);
16        printf("\n");
17    }
18    for(int j=0;j<n;j++)
19    {
20        sum=bt[j];
21        ct[j]=sum;
22    }
23    for(int k=0;k<n;k++)
24    {
25        tat[k]=ct[k]-at[k];
26        totalTAT+=tat[k];
27    }
28    for(int k=0;k<n;k++)
29    {
30        wt[k]=tat[k]-bt[k];
31        totalWT+=wt[k];
32    }
33    printf("Solution: \n\n");
34    printf("P#\t AT\t BT\t CT\t TAT\t WT\t\n\n");
35    for(int i=0;i<n;i++)
36    {
37        printf("P%d\t %d\t %d\t %d\t %d\t %d\n",i+1,at[i],bt[i],ct[i],tat[i],wt[i]);
38    }
39    printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n);
40    printf("Average WT = %f\n",totalWT/n);
41    return 0;
42 }
```

```
C:\Users\aswin\Documents\FCFSOS.exe
Arrival time of process[1]      0
Burst time of process[1]      23
Arrival time of process[2]     10
Burst time of process[2]     10
Arrival time of process[3]     16
Burst time of process[3]       4
Arrival time of process[4]     19
Burst time of process[4]     16

Solution:

P#      AT      BT      CT      TAT      WT
P1       0      23      23      23       0
P2      10      10      33      23      13
P3      16       4      37      21      17
P4      19      16      53      34      18

Average Turnaround Time = 25.250000
Average WT = 12.000000

-----
Process exited after 43.94 seconds with return value 0
Press any key to continue . . .
```

```
1 #include <stdio.h>
2 int main()
3 {
4     int A[100][4];
5     int i, j, n, total = 0, index, temp; float avg_wt, avg_tat;
6     printf("Enter number of process: "); scanf("%d", &n);
7     printf("Enter Burst Time:\n");
8     for (i = 0; i < n; i++) {
9         printf("P%d: ", i + 1); scanf("%d", &A[i][1]); A[i][0] = i + 1;
10    }
11    for (i = 0; i < n; i++) {
12        index = i;
13        for (j = i + 1; j < n; j++)
14            if (A[j][1] < A[index][1]) index = j;
15        temp = A[i][1]; A[i][1] = A[index][1]; A[index][1] = temp;
16        temp = A[i][0];
17        A[i][0] = A[index][0]; A[index][0] = temp;
18    }
19    A[0][2] = 0;
20    for (i = 1; i < n; i++) {
21        A[i][2] = 0;
22        for (j = 0; j < i; j++)
23            A[i][2] += A[j][1];
24        total += A[i][2];
25    }
26    avg_wt = (float)total / n; total = 0;
27    printf("P BT WT TAT\n"); for (i = 0; i < n; i++) {
28        A[i][3] = A[i][1] + A[i][2];
29        total += A[i][3];
30        printf("P%d %d %d %d\n", A[i][0], A[i][1], A[i][2], A[i][3]);
31    }
32    avg_tat = (float)total / n;
33    printf("Average Waiting Time= %f", avg_wt); printf("\nAverage Turnaround Time= %f", avg_tat);
34 }
```

C:\Users\aswin\Documents\SJFOS.exe

Enter number of process: 4

Enter Burst Time:

P1: 12

P2: 32

P3: 8

P4: 22

P BT WT TAT

P3 8 0 8

P1 12 8 20

P4 22 20 42

P2 32 42 74

Average Waiting Time= 17.500000

Average Turnaround Time= 36.000000

Process exited after 16.44 seconds with return value 0

Press any key to continue . . .

(globals)

Project C < > PRIORITYOS.C

```
1 #include<stdio.h>
2 struct priority_scheduling {
3     char process_name;
4     int burst_time;
5     int waiting_time;
6     int turn_around_time;
7     int priority;
8 };
9 int main() {
10     int number_of_process;
11     int total = 0;
12     struct priority_scheduling temp_process;
13     int ASCII_number = 45;
14     int position;
15     float average_waiting_time;
16     float average_turnaround_time;
17     printf("Enter the total number of Processes: ");
18     scanf("%d", & number_of_process);
19     struct priority_scheduling process[number_of_process];
20     printf("\nPlease Enter the Burst Time and Priority of each process:\n");
21     for (int i = 0; i < number_of_process; i++) {
22         process[i].process_name = (char) ASCII_number;
23         printf("Enter the details of the process %c\n", process[i].process_name);
24         printf("Enter the burst time: ");
25         scanf("%d", & process[i].burst_time);
26         printf("Enter the priority: ");
27         scanf("%d", & process[i].priority);
28         ASCII_number++;
29     }
30     for (int i = 0; i < number_of_process; i++) {
31         position = i;
32         for (int j = i + 1; j < number_of_process; j++) {
33             if (process[j].priority > process[position].priority)
34                 position = j;
35         }
36         temp_process = process[i];
37         process[i] = process[position];
38         process[position] = temp_process;
39     }
40     process[0].waiting_time = 0;
41     for (int i = 1; i < number_of_process; i++) {
42         process[i].waiting_time = 0;
43         for (int j = 0; j < i; j++) {
44             process[i].waiting_time += process[j].burst_time;
45         }
46         total += process[i].waiting_time;
47     }
48     average_waiting_time = (float) total / (float) number_of_process;
49     total = 0;
50     printf("\n\nProcess Name \t Burst Time \t Waiting Time \t Turnaround Time\n");
```

```

51 printf(".....\n");
52 for (int i = 0; i < number_of_process; i++) {
53     process[i].turn_around_time = process[i].burst_time + process[i].waiting_time;
54     total += process[i].turn_around_time;
55     printf("\t %c \t\t %d \t\t %d \t\t %d", process[i].process_name, process[i].burst_time, process[i].waiting_time, process[i].turn_around_time);
56     printf("\n.....\n");
57 }
58 average_turnaround_time = (float) total / (float) number_of_process;
59 printf("\n\n Average Waiting Time : %f", average_waiting_time);
60 printf("\n Average Turnaround Time: %f\n", average_turnaround_time);
61 return 0;
62 }

```

 Compiler
  Resources
  Compile Log
  Debug
  Find Results
  Console

Line: 62 Col: 2 Sel: 0 Lines: 62 Length: 2435 Insert Done parsing in 0.031 seconds

Enter the total number of Processes: 4

Please Enter the Burst Time and Priority of each process:

Enter the details of the process A

Enter the burst time: 31

Enter the priority: 4

Enter the details of the process B

Enter the burst time: 11

Enter the priority: 2

Enter the details of the process C

Enter the burst time: 5

Enter the priority: 1

Enter the details of the process D

Enter the burst time: 15

Enter the priority: 3

Process_name	Burst Time	Waiting Time	Turnaround Time
--------------	------------	--------------	-----------------

A	31	0	31
---	----	---	----

D	15	31	46
---	----	----	----

B	11	46	57
---	----	----	----

C	5	57	62
---	---	----	----

Average Waiting Time : 33.500000

Average Turnaround Time: 49.000000

Process exited after 37.59 seconds with return value 0

Press any key to continue . . .