

main.c

Share

Run

Output

Clear

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4
5 int main() {
6     pid_t pid = fork();
7
8     if (pid == 0) {
9         // Child process
10        printf("Child Process:\n");
11        printf("PID: %d\n", getpid());
12        printf("PPID: %d\n", getppid());
13    } else if (pid > 0) {
14        // Parent process
15        printf("Parent Process:\n");
16        printf("PID: %d\n", getpid());
17        printf("PPID: %d\n", getppid());
18    } else {
19        // Fork failed
20        printf("Fork failed\n");
21    }
22 }
```

```
Parent Process:
PID: 29921
PPID: 29910
Child Process:
PID: 29922
PPID: 29921

=== Code Execution Successful ===
```

main.c

Share

Run

```
1 #include <fcntl.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 int main() {
6     int src = open("source.txt", O_RDONLY);
7     int dest = open("destination.txt", O_WRONLY | O_CREAT | O_TRUNC,
8         0644);
9     char buffer[1024];
10    int bytes;
11
12    while ((bytes = read(src, buffer, sizeof(buffer))) > 0) {
13        write(dest, buffer, bytes);
14    }
15
16    close(src);
17    close(dest);
18
19    printf("File copied successfully.\n");
20
21    return 0;
22 }
```

Output

Clear

File copied successfully.

=== Code Execution Successful ===

main.c

main.c

Share

Run

```
1 #include <stdio.h>
2
3 int main() {
4     int n, i;
5     float avg_wt = 0, avg_tat = 0;
6     printf("Enter number of processes: ");
7     scanf("%d", &n);
8
9     int bt[n], wt[n], tat[n];
10
11     // Input burst times
12     for(i = 0; i < n; i++) {
13         printf("Enter burst time for process %d: ", i + 1);
14         scanf("%d", &bt[i]);
15     }
16
17     // Waiting time for first process is 0
18     wt[0] = 0;
19
20     // Calculate waiting time for each process
21     for(i = 1; i < n; i++) {
```

Output

Clear

Enter number of processes: 4
Enter burst time for process 1: 5
Enter burst time for process 2: 3
Enter burst time for process 3: 8
Enter burst time for process 4: 6

Process	Burst Time	Waiting Time	Turnaround Time
P1	5	0	5
P2	3	5	8
P3	8	8	16
P4	6	16	22

Average Waiting Time: 7.25
Average Turnaround Time: 12.75

=== Code Execution Successful ===

main.c

Share

Run

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

```
// Calculate waiting time for each process
for(i = 1; i < n; i++) {
    wt[i] = wt[i - 1] + bt[i - 1];
}
// Calculate turnaround time and average times
for(i = 0; i < n; i++) {
    tat[i] = wt[i] + bt[i];
    avg_wt += wt[i];
    avg_tat += tat[i];
}
avg_wt /= n;
avg_tat /= n;
// Display results
printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
for(i = 0; i < n; i++) {
    printf("P%d\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
}
printf("\nAverage Waiting Time: %.2f", avg_wt);
printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
return 0;
}
```

Output

Clear

Enter number of processes: 4
Enter burst time for process 1: 5
Enter burst time for process 2: 3
Enter burst time for process 3: 8
Enter burst time for process 4: 6

Process Burst Time Waiting Time Turnaround Time
P1 5 0 5
P2 3 5 8
P3 8 8 16
P4 6 16 22

Average Waiting Time: 7.25
Average Turnaround Time: 12.75

=== Code Execution Successful ===

main.c

Share

Run

Output

Clear

```
1 #include <stdio.h>
2
3- int main() {
4     int n, i, j, bt[10], wt[10] = {0}, tat[10], p[10], temp;
5     float avg_wt = 0, avg_tat = 0;
6
7     printf("Enter number of processes: ");
8     scanf("%d", &n);
9-     for(i = 0; i < n; i++) {
10         printf("Burst time for P%d: ", i+1);
11         scanf("%d", &bt[i]);
12         p[i] = i + 1;
13     }
14
15     // Sort by burst time
16     for(i = 0; i < n-1; i++)
17         for(j = i+1; j < n; j++)
18-             if(bt[i] > bt[j]) {
19                 temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
20                 temp = p[i]; p[i] = p[j]; p[j] = temp;
21             }
22 }
```

Enter number of processes: 4
Burst time for P1: 6
Burst time for P2: 8
Burst time for P3: 7
Burst time for P4: 3

P	BT	WT	TAT
P4	3	0	3
P1	6	3	9
P3	7	9	16
P2	8	16	24

Avg WT=7.00, Avg TAT=13.00

=== Code Execution Successful ===

main.c

Share

Run

Output

Clear

```
16 for(i = 0; i < n-1; i++)
17     for(j = i+1; j < n; j++)
18         if(bt[i] > bt[j]) {
19             temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
20             temp = p[i]; p[i] = p[j]; p[j] = temp;
21         }
22
23 for(i = 1; i < n; i++)
24     wt[i] = wt[i-1] + bt[i-1];
25
26 printf("\nP\tBT\tWT\tTAT\n");
27 for(i = 0; i < n; i++) {
28     tat[i] = wt[i] + bt[i];
29     avg_wt += wt[i];
30     avg_tat += tat[i];
31     printf("P%d\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);
32 }
33
34 printf("\nAvg WT=%.2f, Avg TAT=%.2f\n", avg_wt/n, avg_tat/n);
35 return 0;
36 }
```

Enter number of processes: 4
Burst time for P1: 6
Burst time for P2: 8
Burst time for P3: 7
Burst time for P4: 3

P	BT	WT	TAT
P4	3	0	3
P1	6	3	9
P3	7	9	16
P2	8	16	24

Avg WT=7.00, Avg TAT=13.00

=== Code Execution Successful ===

main.c

Share

Run

```
1 #include <stdio.h>
2
3 int main() {
4     int n, i, j, temp;
5     int bt[10], wt[10] = {0}, tat[10], pr[10], p[10];
6     float avg_wt = 0, avg_tat = 0;
7
8     printf("Enter number of processes: ");
9     scanf("%d", &n);
10
11     for(i = 0; i < n; i++) {
12         printf("Enter burst time for P%d: ", i+1);
13         scanf("%d", &bt[i]);
14         printf("Enter priority for P%d (lower number = higher priority): ", i+1);
15         scanf("%d", &pr[i]);
16         p[i] = i + 1;
17     }
18
19     // Sort by priority
20     for(i = 0; i < n-1; i++) {
21         for(j = i+1; j < n; j++) {
22             if(pr[i] > pr[j]) {
23                 temp = bt[i];
24                 bt[i] = bt[j];
25                 bt[j] = temp;
26                 temp = pr[i];
27                 pr[i] = pr[j];
28                 pr[j] = temp;
29                 temp = p[i];
30                 p[i] = p[j];
31                 p[j] = temp;
32             }
33         }
34     }
35
36     // Calculate waiting and turnaround times
37     wt[0] = 0;
38     for(i = 1; i < n; i++) {
39         wt[i] = wt[i-1] + bt[i-1];
40     }
41
42     // Calculate average waiting time and average turnaround time
43     for(i = 0; i < n; i++) {
44         avg_wt += wt[i];
45         avg_tat += wt[i] + bt[i];
46     }
47
48     avg_wt /= n;
49     avg_tat /= n;
50
51     printf("Average Waiting Time = %.2f\n", avg_wt);
52     printf("Average Turnaround Time = %.2f\n", avg_tat);
53 }
```

Output

Clear

Enter number of processes: 3
Enter burst time for P1: 10
Enter priority for P1 (lower number = higher priority): 2
Enter burst time for P2: 5
Enter priority for P2 (lower number = higher priority): 1
Enter burst time for P3: 8
Enter priority for P3 (lower number = higher priority): 3

P	BT	PR	WT	TAT
P2	5	1	0	5
P1	10	2	5	15
P3	8	3	15	23

Average Waiting Time = 6.67
Average Turnaround Time = 14.33

=== Code Execution Successful ===

main.c

Share

Run

21+
22+
23
24
25
26
27
28
29
30
31
32+
33
34
35
36
37
38
39
40
41 }

```
for(j = i+1; j < n; j++) {  
    if(pr[i] > pr[j]) {  
        temp = pr[i]; pr[i] = pr[j]; pr[j] = temp;  
        temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;  
        temp = p[i]; p[i] = p[j]; p[j] = temp;  
    }  
}  
  
for(i = 1; i < n; i++)  
    wt[i] = wt[i-1] + bt[i-1];  
printf("\nP\tBT\tPR\tWT\tTAT\n");  
for(i = 0; i < n; i++) {  
    tat[i] = wt[i] + bt[i];  
    avg_wt += wt[i];  
    avg_tat += tat[i];  
    printf("P%d\t%d\t%d\t%d\t%d\n", p[i], bt[i], pr[i], wt[i],  
        tat[i]);  
}  
printf("\nAverage Waiting Time = %.2f", avg_wt / n);  
printf("\nAverage Turnaround Time = %.2f\n", avg_tat / n);  
return 0;
```

Output

Cle

Enter number of processes: 3
Enter burst time for P1: 10
Enter priority for P1 (lower number = higher priority): 2
Enter burst time for P2: 5
Enter priority for P2 (lower number = higher priority): 1
Enter burst time for P3: 8
Enter priority for P3 (lower number = higher priority): 3

P BT PR WT TAT
P2 5 1 0 5
P1 10 2 5 15
P3 8 3 15 23

Average Waiting Time = 6.67
Average Turnaround Time = 14.33

=== Code Execution Successful ===