Programiz

C Online Compiler

main.c

Share Run

```c
#include<stdio.h>
#include<unistd.h>
int main()
{
printf("Process ID: %d\n", getpid() );
printf("Parent Process ID: %d\n", getpid() );
return 0;
}
```

Output

```
Process ID: 6357
Parent Process ID: 6357


=== Code Execution Successful
```

Online C Compiler - Programiz | Online C Compiler - Programiz | File Copy Using Syscalls | FREE AI Code Generator: Gener

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO

main.c

Share | Run

```c
#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);
}
```

Output

```
Enter the filename to open for reading
mythili
Cannot open file mythili


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

Type here to search | 30°C | 8:49 2/8/

C Online Compiler

main.c

Share Run

```c
#include <stdio.h>
int main()
{
int A[100][4];
int i, j, n, total = 0, index, temp;
float avg_wt, avg_tat;
printf("Enter number of process: ");
scanf("%d", &n);
printf("Enter Burst Time:\n");
for (i= 0; i < n; i++)
{
printf("P%d: ", i + 1);
scanf("%d", &A[i][1]);
A[i][0] = i + 1;
}
for (i = 0; i < n; i++)
{
index = i;
for (j = i + 1; j < n; j++)
if (A[j][1] < A[index][1])
index = j;
temp = A[i][1]; A[i][1] =A[index][1];A[index][1]= temp;
temp = A[i][0]; A[i][0] =A[index][0];A[index][0]= temp;
}
A[0][2] = 0;
for (i = 1; i < n; i++)
```

Output

```
Enter number of process: 3
Enter Burst Time:
P1: 2
P2: 46
P3: 8
P BT WT TAT
P1 2 0 2
P2 4 2 6
P3 8 6 14
Average Waiting Time= 2.666667
AverageTurnaround Time= 7.333333

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

Online C Compiler - Programiz | Online C Compiler - Programiz | File Copy Using Syscalls | FREE AI Code Generator: Gener

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO

main.c

Share | Run

```c
        twt += p[i].wt, ttat += p[i].tat;
    }
    printf("\nAvg WT: %.2f\nAvg TAT: %.2f\n", twt / n, ttat / n);
}

int main() {
    int n;
    printf("Enter no. of processes: ");
    scanf("%d", &n);
    Process p[n];

    for (int i = 0; i < n; i++) {
        p[i].id = i + 1;
        printf("P%d BT: ", i + 1);
        scanf("%d", &p[i].bt);
    }

    sjf(p, n);
    return 0;
}
```

Output

```
Enter no. of processes: 2
P1 BT: 6
P2 BT: 4

P   BT  WT  TAT
P2  4   0   4
P1  6   4   10

Avg WT: 2.00
Avg TAT: 7.00


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

Type here to search

30°C

8:54
2/8/2

Programiz
C Online Compiler

Courses by Programiz

Programiz

main.c

Share Run

```c
#include <stdio.h>
typedef struct {
    int pid;
    int burst_time;
    int remaining_time;
    int turnaround_time;
    int waiting_time;
} Process;

void RoundRobin(Process processes[], int n, int quantum) {
    int time = 0, completed = 0;
    for (int i = 0; i < n; i++) {
        processes[i].remaining_time = processes[i].burst_time;
    }
    // Round Robin Scheduling
    while (completed < n) {
        for (int i = 0; i < n; i++) {
            if (processes[i].remaining_time > 0) {
                if (processes[i].remaining_time > quantum) {
                    time += quantum;
                    processes[i].remaining_time -= quantum;
                } else {
                    time += processes[i].remaining_time;
                    processes[i].waiting_time = time - processes[i]
                        .burst_time;
```

Output

```
Enter number of processes: 3
Enter burst time for each process:
Process 1: 1
Process 2: 2
Process 3: 3
Enter time quantum: 3
Process Burst Time  Waiting Time    Turnarou
1       1       0       1
2       2       1       3
3       3       3       6

Average Waiting Time: 1.33
Average Turnaround Time: 3.33


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

Online C Compiler - Programiz | Online C Compiler - Programiz | File Copy Using Syscalls | Online C Compiler - Programiz

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO

main.c

Share | Run

```
    p[0].wt = 0;
    for (int i = 1; i < n; i++) p[i].wt = p[i - 1].wt + p[i - 1].bt;
    for (int i = 0; i < n; i++) p[i].tat = p[i].wt + p[i].bt;

    printf("\nP\tBT\tWT\tTAT\n");
    for (int i = 0; i < n; i++)
        printf("P%d\t%d\t%d\t%d\n", p[i].id, p[i].bt, p[i].wt, p[i]
            .tat);
}

int main() {
    int n;
    printf("Enter no. of processes: ");
    scanf("%d", &n);
    P p[n];

    for (int i = 0; i < n; i++) p[i].id = i + 1, printf("P%d BT: ",
        i + 1), scanf("%d", &p[i].bt);

    sjf(p, n);
}
```

Output

```
Enter no. of processes: 3
P1 BT: 4
5P2 BT: 5
P3 BT: 6

P   BT  WT  TAT
P1  4   0   4
P2  5   4   9
P3  6   9   15


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

Type here to search

Online C Compiler - Progran | Online C Compiler - Progran | File Copy Using Syscalls | Online C Compiler - Progran | Online C Compiler - Progra

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO

main.c | Share | Run

```c
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>

struct msg_buffer {
    long msg_type;
    char msg_text[100];
};

int main() {
    key_t key = ftok("progfile", 65);  // Generate unique key
    int msgid = msgget(key, 0666 | IPC_CREAT);  // Create message
        queue

    struct msg_buffer message;
    message.msg_type = 1;

    printf("Enter message: ");
    fgets(message.msg_text, sizeof(message.msg_text), stdin);
    msgsnd(msgid, &message, sizeof(message.msg_text), 0);  // Send
        message
```

Output

```
Enter message: message
Message sent: message


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

Type here to search

Earn...

9:05
2/8/2

main.c

Share Run

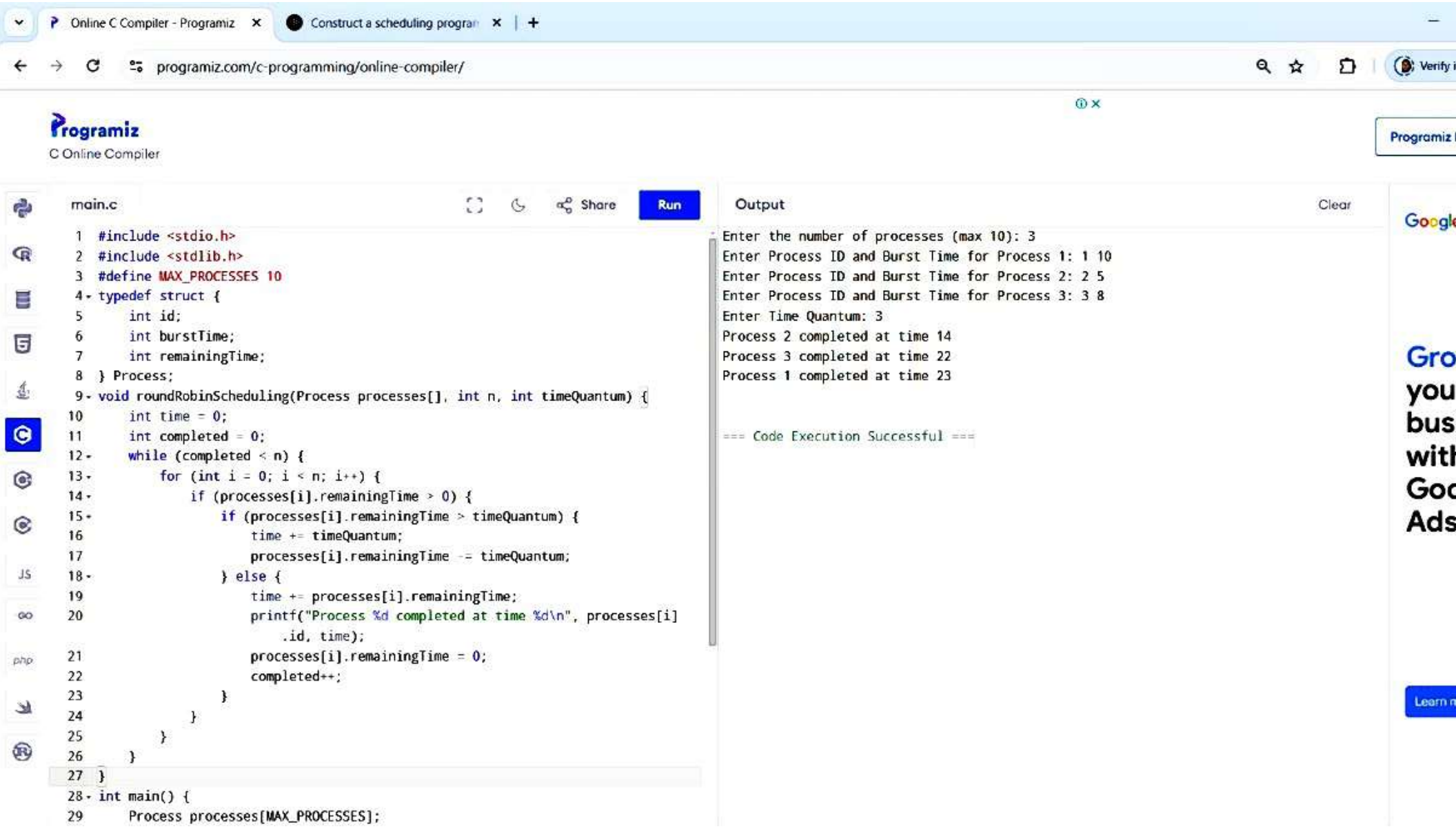
Output

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_PROCESSES 5
typedef struct {
    int id, priority;
} Process;
Process queue[MAX_PROCESSES];
int count = 0;
void addProcess(int id, int priority) {
    if (count < MAX_PROCESSES) {
        queue[count++] = (Process){id, priority};
    } else {
        printf("Queue full!\n");
    }
}
Process getHighestPriority() {
    Process highest = {0, -1};
    for (int i = 0; i < count; i++) {
        if (queue[i].priority > highest.priority) {
            highest = queue[i];
        }
    }
    return highest;
}
void removeProcess(int id) {
    for (int i = 0; i < count; i++) {
        if (queue[i].id == id) {
            queue[i] = queue[--count];
```

```
1. Add Process
2. Execute Highest Priority
3. Exit
Choice: 2
No processes.

1. Add Process
2. Execute Highest Priority
3. Exit
Choice: 3


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


Online C Compiler - Programiz
Construct a scheduling progra
programiz.com/c-programming/online-compiler/
Programiz
C Online Compiler
main.c
Share
Run
Output
Clear
#include <stdio.h>
#include <stdlib.h>
#define MAX_PROCESSES 10
typedef struct {
int id;
int burstTime;
int remainingTime;
} Process;
void roundRobinScheduling(Process processes[], int n, int timeQuantum) {
int time = 0;
int completed = 0;
while (completed < n) {
for (int i = 0; i < n; i++) {
if (processes[i].remainingTime > 0) {
if (processes[i].remainingTime > timeQuantum) {
time += timeQuantum;
processes[i].remainingTime -= timeQuantum;
} else {
time += processes[i].remainingTime;
printf("Process %d completed at time %d\n", processes[i]
.id, time);
processes[i].remainingTime = 0;
completed++;
}
}
}
}
int main() {
Process processes[MAX_PROCESSES];
Enter the number of processes (max 10): 3
Enter Process ID and Burst Time for Process 1: 1 10
Enter Process ID and Burst Time for Process 2: 2 5
Enter Process ID and Burst Time for Process 3: 3 8
Enter Time Quantum: 3
Process 2 completed at time 14
Process 3 completed at time 22
Process 1 completed at time 23
=== Code Execution Successful ===

Online C Compiler - Programiz | Construct a scheduling progra

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Google Ads New to Google Ads? Claim Now

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>

#define SHM_SIZE 1024
int main() {
    int shmid;
    char *shared_memory;
    key_t key = ftok("shmfile", 65);
    shmid = shmget(key, SHM_SIZE, 0666 | IPC_CREAT);
    if (shmid == -1) {
        perror("shmget failed");
        exit(1);
    }
    shared_memory = (char *)shmat(shmid, NULL, 0);
    if (shared_memory == (char *)(-1)) {
        perror("shmat failed");
        exit(1);
    }
    pid_t pid = fork();
    if (pid < 0) {
        perror("Fork failed");
        exit(1);
    }
    if (pid == 0) {
        sleep(1);
        printf("Child Process: Reading from shared memory...\n");
```

Output Clear

```
Enter a message for the child process: Child Process: Reading from shared memory...
Data read from shared memory:
angry bird
Parent Process: Message written to shared memory.
Parent Process: Shared memory segment deleted.


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