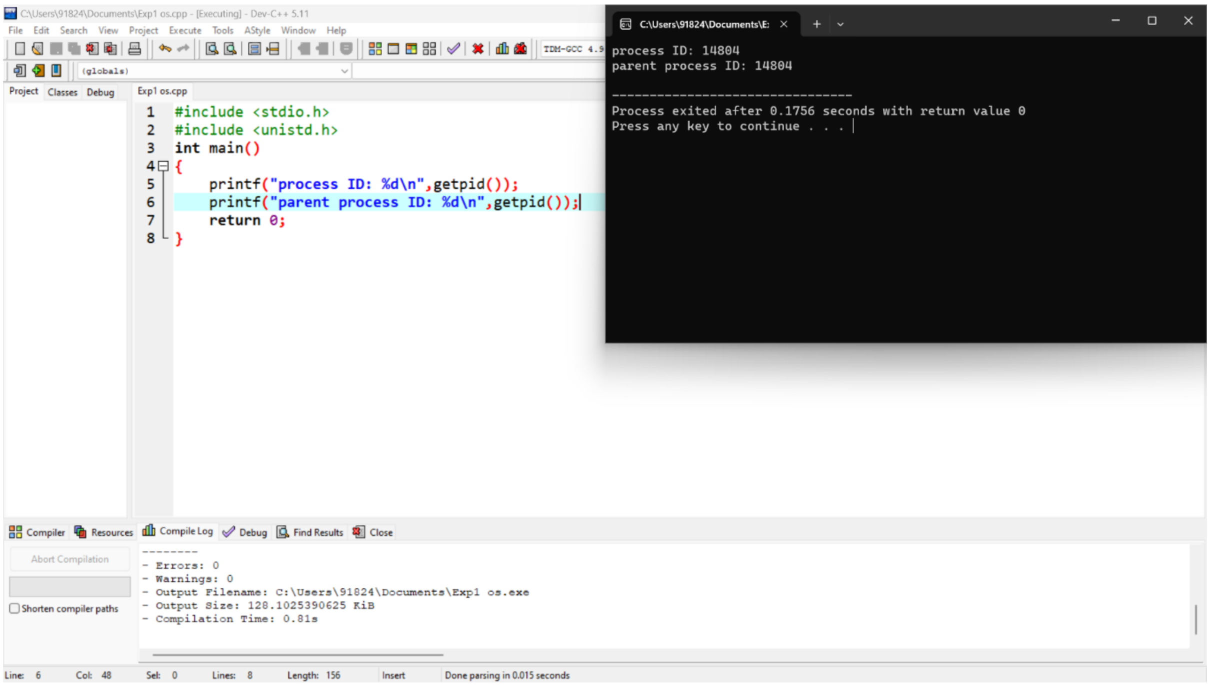
1.Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

Program

```c
#include <stdio.h>

#include <unistd.h>

int main()

{

        printf("process ID: %d\n",getpid());

        printf("parent process ID: %d\n", getpid());

        return 0;

}
```

INPUT and OUTPUT:



C:\Users\91824\Documents\Exp1 os.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9
(globals)
Project Classes Debug
Exp1 os.cpp
C:\Users\91824\Documents\E:
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
□ Shorten compiler paths

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

```
process ID: 14804
parent process ID: 14804

--------------------------------
Process exited after 0.1756 seconds with return value 0
Press any key to continue . . .
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91824\Documents\Exp1 os.exe
- Output Size: 128.1025390625 KiB
- Compilation Time: 0.81s
```

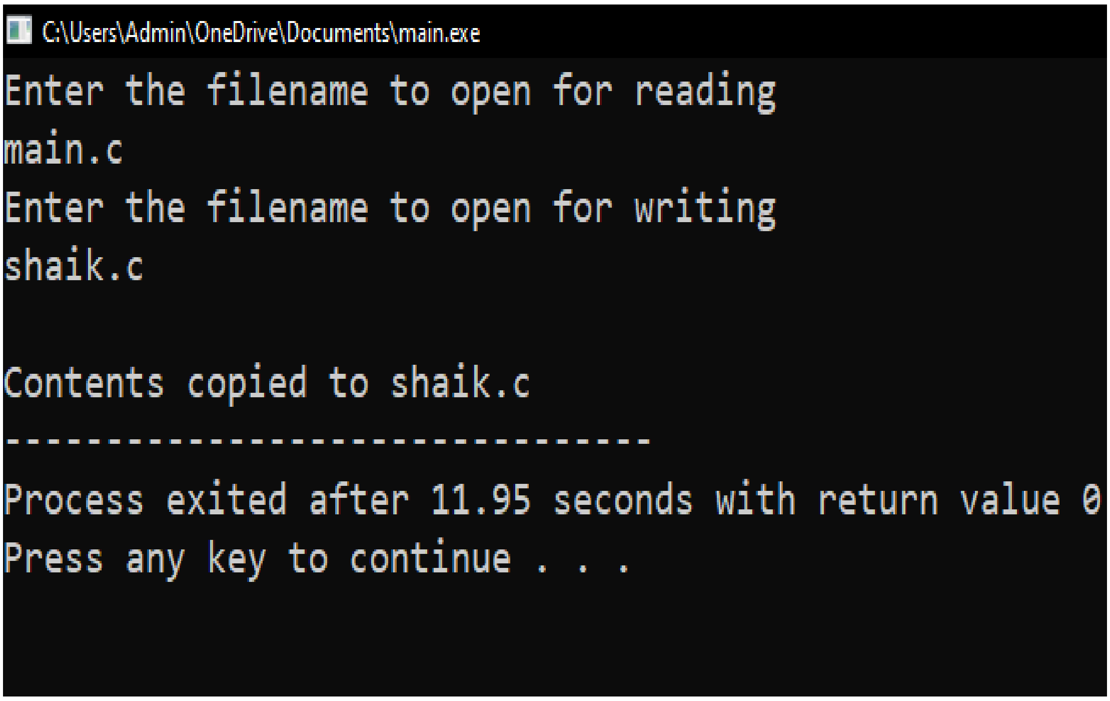
Line: 6 Col: 48 Sel: 0 Lines: 8 Length: 156 Insert Done parsing in 0.015 seconds

2.To identify the system calls to copy the content of one file to another and illustrate the same using a C program.

Program:

```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);
            }
            c = fgetc(fptr1);
            while (c != EOF)
            {
                        fputc(c, fptr2);
                        c = fgetc(fptr1);
}
            printf("\nContents copied to %s", filename);
            fclose(fptr1);
            fclose(fptr2);
            return 0;
}
```

INPUT AND OUTPUT:



C:\Users\Admin\OneDrive\Documents\main.exe

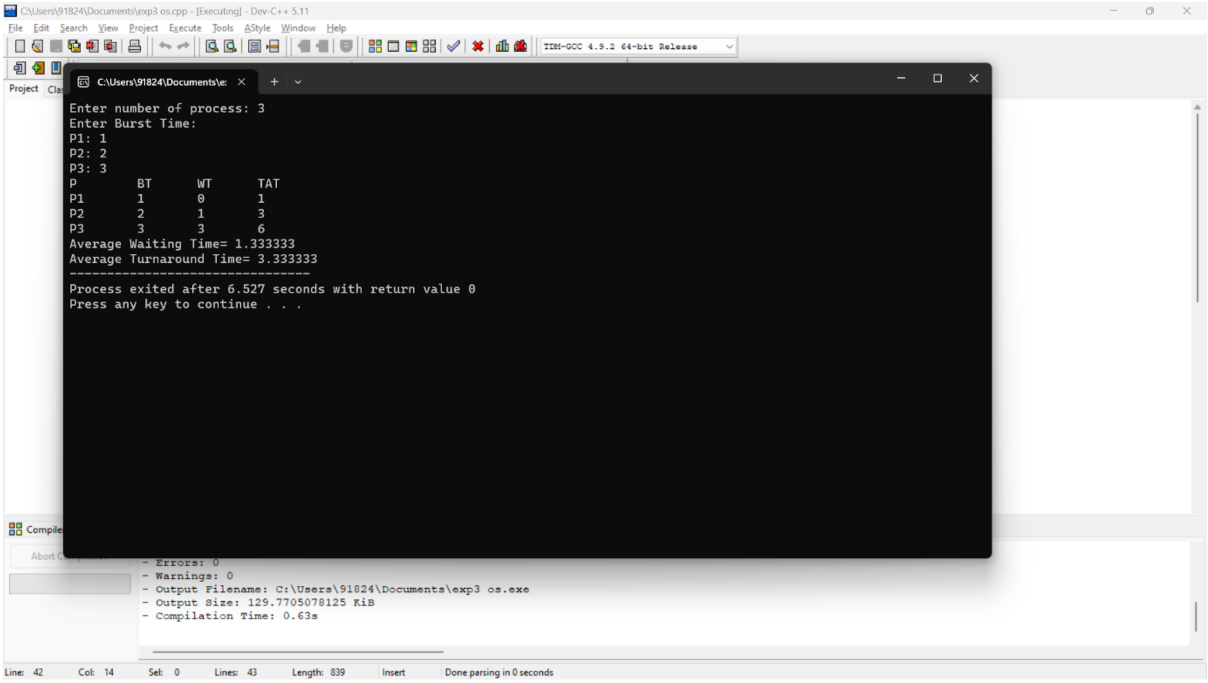
```
Enter the filename to open for reading
main.c
Enter the filename to open for writing
shaik.c

Contents copied to shaik.c
--------------------------------
Process exited after 11.95 seconds with return value 0
Press any key to continue . . .
```

3.ToDesign a CPU scheduling program with C using First Come First Served technique with the following considerations. a. All processes are activated at time 0. b. Assume that no process waits on I/O devices.

PROGRAM:

```
#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;

        }

        A[0][2] = 0;

        for (i = 1; i < n; i++) {

                A[i][2] = 0;

                for (j = 0; j < i; j++)

                        A[i][2] += A[j][1];

                total += A[i][2];

        }

       avg_wt = (float)total / n;

        total = 0;

        printf("P        BT       WT       TAT\n");

          for (i = 0; i < n; i++) {

                A[i][3] = A[i][1] + A[i][2];
                total += A[i][3];
                printf("P%d       %d       %d       %d\n", A[i][0],A[i][1], A[i][2], A[i][3]);
        }
         avg_tat = (float)total / n;
        printf("Average Waiting Time= %f", avg_wt);
        printf("\nAverage Turnaround Time= %f", avg_tat);


        return 0;
}
```

INPUT AND OUTPUT:



C:\Users\91824\Documents\exp3 os.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Project
C:\Users\91824\Documents\e:
Abort Compilation
Line: 42 Col: 14 Sel: 0 Lines: 43 Length: 839 Insert Done parsing in 0 seconds

```
Enter number of process: 3
Enter Burst Time:
P1: 1
P2: 2
P3: 3
P       BT      WT      TAT
P1      1       0       1
P2      2       1       3
P3      3       3       6
Average Waiting Time= 1.333333
Average Turnaround Time= 3.333333
--------------------------------
Process exited after 6.527 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91824\Documents\exp3 os.exe
- Output Size: 129.7705078125 KiB
- Compilation Time: 0.63s
```

4.Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.

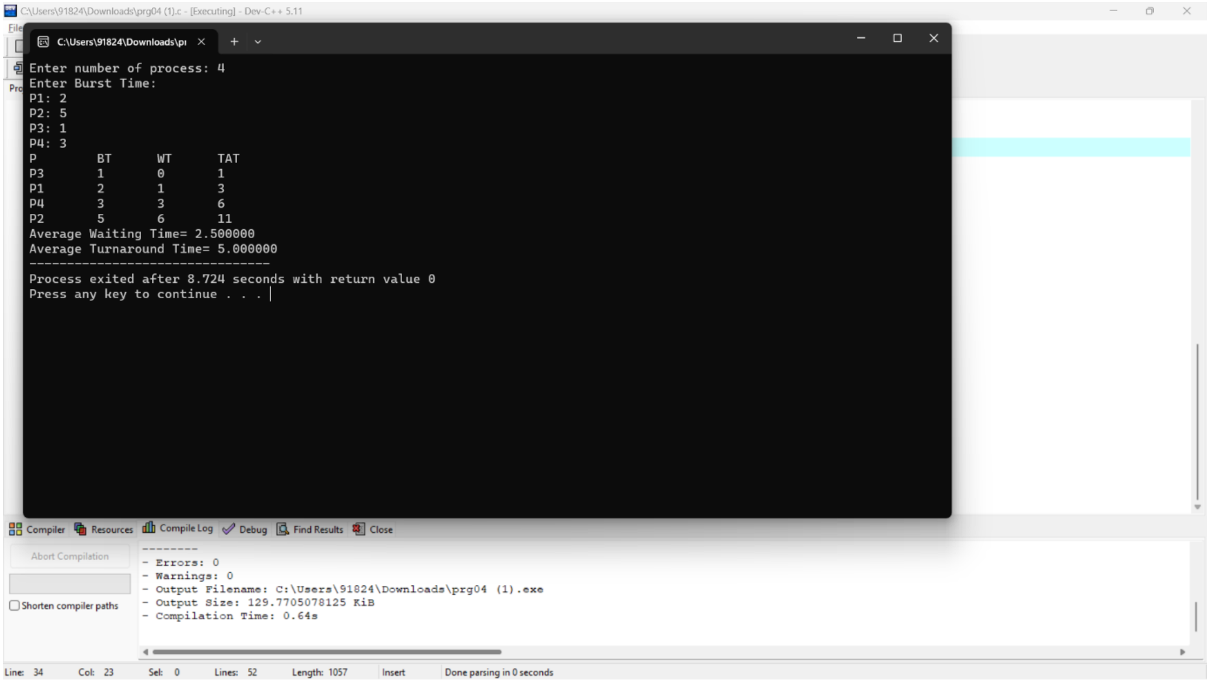
PROGRAM:

```
#include <stdio.h>
int main()
{
        int A[100][4];
        int i, j, n, wt=0, tat=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;
A[0][3] = A[0][1];
for (i = 1; i < n; i++) {
        A[i][2] = A[i-1][3];
        A[i][3] = A[i][1] + A[i][2];
        wt += A[i][2];
        tat += A[i][3];
}

printf("P       BT      WT      TAT\n");
for (i = 0; i < n; i++)
        printf("P%d     %d      %d      %d\n", A[i][0],A[i][1], A[i][2], A[i][3]);

avg_wt = (float)wt / n;
avg_tat = (float)tat / n;
```

```
	printf("Average Waiting Time= %f", avg_wt);
	printf("\nAverage Turnaround Time= %f", avg_tat);

	return 0;
}
```

INPUT AND OUTPUT:



```
Enter number of process: 4
Enter Burst Time:
P1: 2
P2: 5
P3: 1
P4: 3
P        BT      WT      TAT
P3       1       0       1
P1       2       1       3
P4       3       3       6
P2       5       6       11
Average Waiting Time= 2.500000
Average Turnaround Time= 5.000000
--------------------------------
Process exited after 8.724 seconds with return value 0
Press any key to continue . . .
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91824\Downloads\prg04 (1).exe
- Output Size: 129.7705078125 KiB
- Compilation Time: 0.64s
```

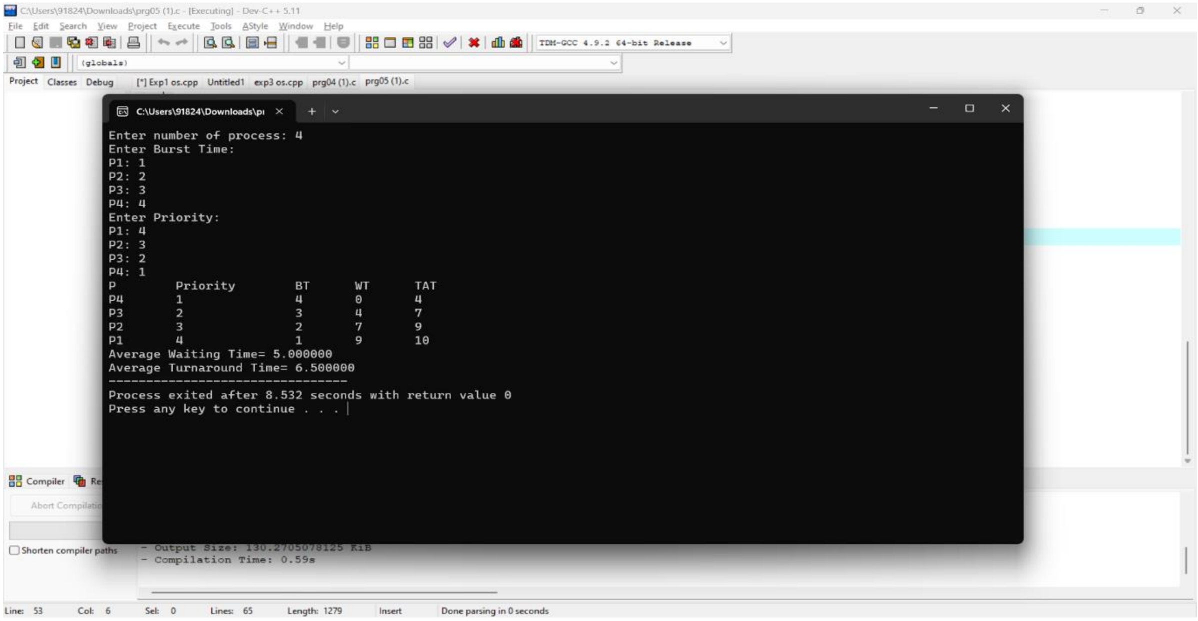
5. Construct a scheduling program with C that selects the waiting process with the highest priority to execute next.

Program:

```
#include <stdio.h>

int main()

{

	int A[100][5];

	int i, j, n, wt=0, tat=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;
}

printf("Enter Priority:\n");
for (i = 0; i < n; i++) {
    printf("P%d: ", i + 1);
    scanf("%d", &A[i][4]);
}


for (i = 0; i < n; i++) {
    index = i;
    for (j = i + 1; j < n; j++)
        if (A[j][4] < A[index][4])
            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;

        temp = A[i][4];
        A[i][4] = A[index][4];
        A[index][4] = temp;
}

        A[0][2] = 0;
        A[0][3] = A[0][1];
        for (i = 1; i < n; i++) {
                A[i][2] = A[i-1][3];
                A[i][3] = A[i][1] + A[i][2];
                wt += A[i][2];
                tat += A[i][3];
        }
          printf("P         Priority BT        WT       TAT\n");
        for (i = 0; i < n; i++)
                printf("P%d       %d\t     %d        %d        %d\n", A[i][0],A[i][4],A[i][1], A[i][2], A[i][3]);
            avg_wt = (float)wt / n;
        avg_tat = (float)tat / n;
        printf("Average Waiting Time= %f", avg_wt);
        printf("\nAverage Turnaround Time= %f", avg_tat);

        return 0;
}
```

INPUT AND OUTPUT:



C:\Users\91824\Downloads\prg05 (1).c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] Exp1 os.cpp
Untitled1
exp3 os.cpp
prg04 (1).c
prg05 (1).c
C:\Users\91824\Downloads\pı
Compiler
Re
Abort Compilatio
□ Shorten compiler paths
Line: 53 Col: 6 Sel: 0 Lines: 65 Length: 1279 Insert Done parsing in 0 seconds

```
Enter number of process: 4
Enter Burst Time:
P1: 1
P2: 2
P3: 3
P4: 4
Enter Priority:
P1: 4
P2: 3
P3: 2
P4: 1
P         Priority         BT      WT      TAT
P4        1                4       0       4
P3        2                3       4       7
P2        3                2       7       9
P1        4                1       9       10
Average Waiting Time= 5.000000
Average Turnaround Time= 6.500000
--------------------------------
Process exited after 8.532 seconds with return value 0
Press any key to continue . . .
```

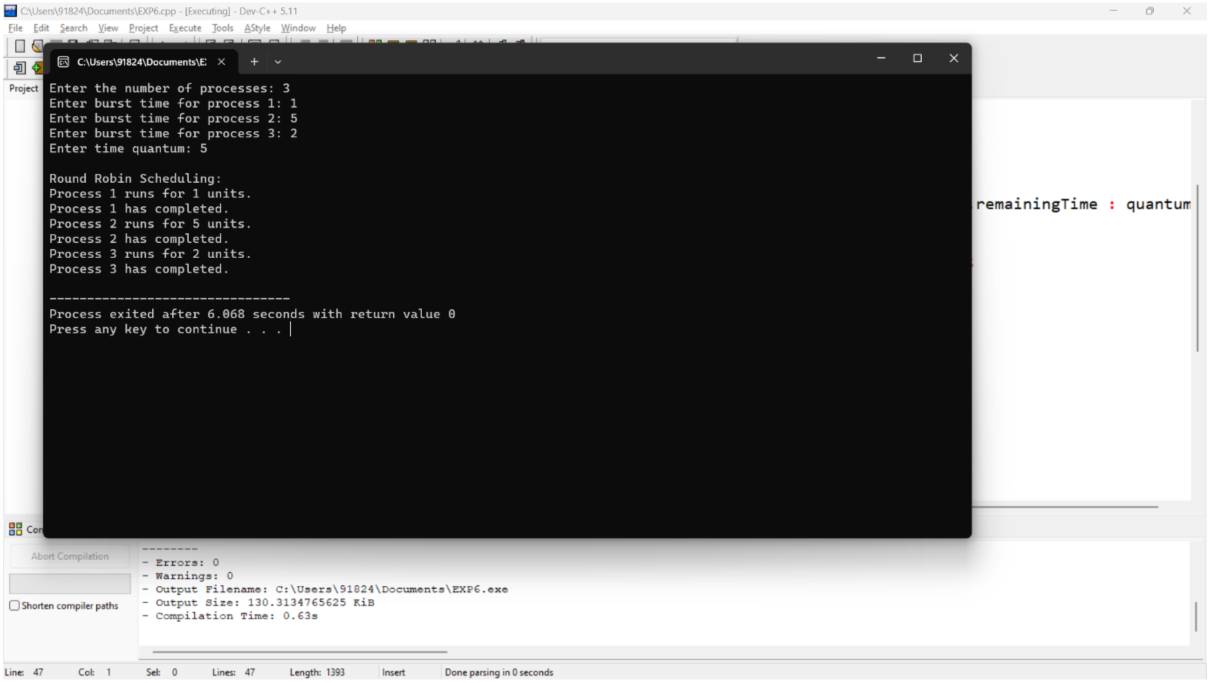
```
- Output Size: 130.2705078125 KiB
- Compilation Time: 0.59s
```

6. Construct a C program to simulate Round Robin scheduling algorithm with C

PROGRAM:

```
#include <stdio.h>

struct Process {
  int id, burstTime, remainingTime;
};
void roundRobin(struct Process processes[], int n, int quantum)
 {
  int time = 0, completed = 0;
  while (completed < n)
 {
    for (int i = 0; i < n; i++)
 {
      if (processes[i].remainingTime > 0)
 {
        int execTime = (processes[i].remainingTime < quantum) ?
processes[i].remainingTime : quantum;
        processes[i].remainingTime -= execTime;
        time += execTime;
        printf("Process %d runs for %d units.\n", processes[i].id, execTime);
        if (processes[i].remainingTime == 0) {
          completed++;
          printf("Process %d has completed.\n", processes[i].id);
        }
      }
    }
  }
}
int main()
 {
  int n, quantum;
  printf("Enter the number of processes: ");
  scanf("%d", &n);
  struct Process processes[n];
  for (int i = 0; i < n; i++)
 {
    processes[i].id = i + 1;
    printf("Enter burst time for process %d: ", processes[i].id);
    scanf("%d", &processes[i].burstTime);
    processes[i].remainingTime = processes[i].burstTime;
  }
  printf("Enter time quantum: ");
  scanf("%d", &quantum);
  printf("\nRound Robin Scheduling:\n");
  roundRobin(processes, n, quantum);
  return 0;
}
```

INPUT AND OUTPUT:


C:\Users\91824\Documents\EXP6.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\91824\Documents\E
Project
remainingTime : quantum
Abort Compilation
Shorten compiler paths
Line: 47 Col: 1 Sel: 0 Lines: 47 Length: 1393 Insert Done parsing in 0 seconds

```
Enter the number of processes: 3
Enter burst time for process 1: 1
Enter burst time for process 2: 5
Enter burst time for process 3: 2
Enter time quantum: 5

Round Robin Scheduling:
Process 1 runs for 1 units.
Process 1 has completed.
Process 2 runs for 5 units.
Process 2 has completed.
Process 3 runs for 2 units.
Process 3 has completed.

--------------------------------
Process exited after 6.068 seconds with return value 0
Press any key to continue . . .
```

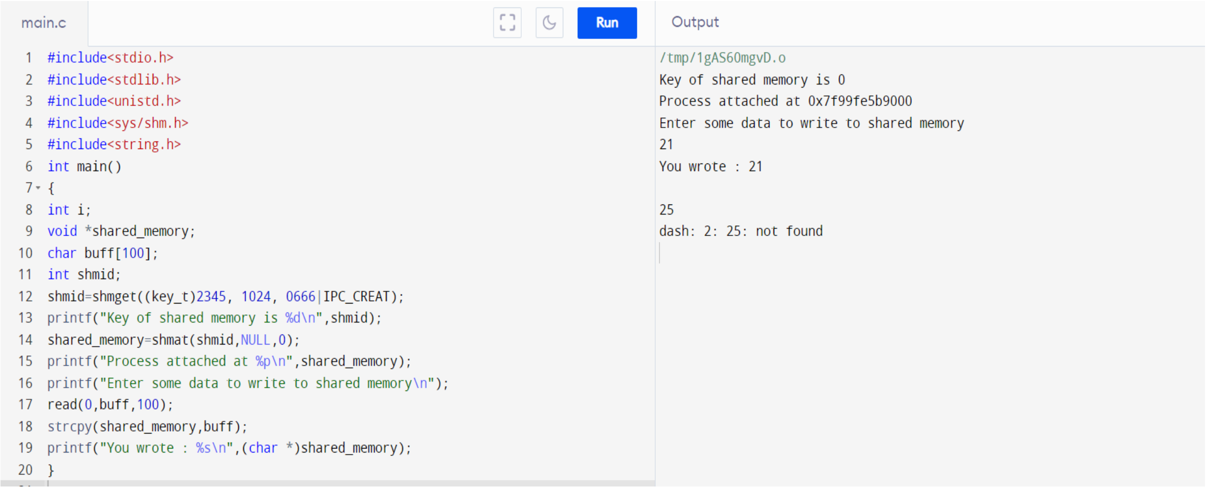
```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91824\Documents\EXP6.exe
- Output Size: 130.3134765625 KiB
- Compilation Time: 0.63s
```

7. Illustrate the concept of inter-process communication using shared memory with a C program.

Program:

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/shm.h>

#include<string.h>

int main()

{

int i;

void *shared_memory;

char buff[100];

int shmid;

shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);

printf("Key of shared memory is %d\n",shmid);

shared_memory=shmat(shmid,NULL,0);
printf("Process attached at %p\n",shared_memory);

printf("Enter some data to write to shared memory\n");

read(0,buff,100);

strcpy(shared_memory,buff);

printf("You wrote : %s\n",(char *)shared_memory);

}
```

INPUT AND OUTPUT:



```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
#include<string.h>
int main()
{
int i;
void *shared_memory;
char buff[100];
int shmid;
shmid=shmget((key_t)2345, 1024, 0666|IPC_CREAT);
printf("Key of shared memory is %d\n",shmid);
shared_memory=shmat(shmid,NULL,0);
printf("Process attached at %p\n",shared_memory);
printf("Enter some data to write to shared memory\n");
read(0,buff,100);
strcpy(shared_memory,buff);
printf("You wrote : %s\n",(char *)shared_memory);
}
```

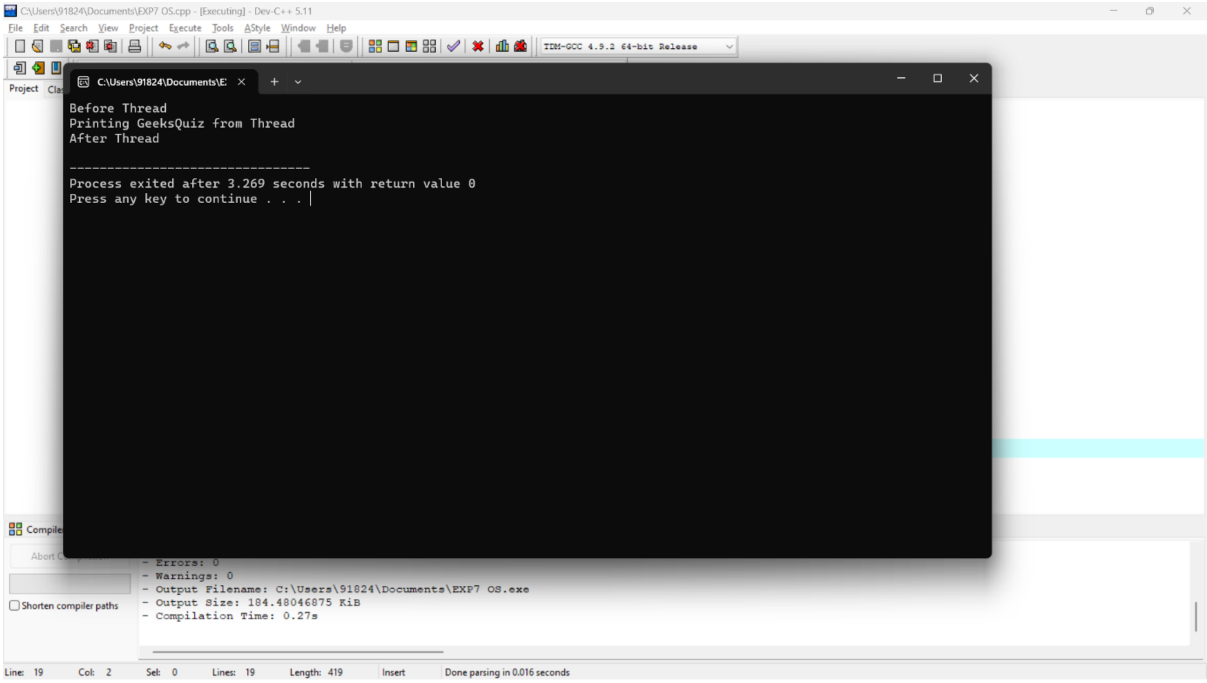
```
/tmp/1gAS60mgvD.o
Key of shared memory is 0
Process attached at 0x7f99fe5b9000
Enter some data to write to shared memory
21
You wrote : 21

25
dash: 2: 25: not found
```

8. Illustrate the concept of multithreading using a C program.

Program**:**

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<pthread.h>

void *myThreadFun(void *vargp)

{

    sleep(1);

    printf("Printing GeeksQuiz from Thread \n");

    return NULL;

}

int main()

{

    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread\n");
    exit(0);
}
```

INPUT AND OUTPUT:


C:\Users\91824\Documents\EXP7 OS.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
Project Clas
C:\Users\91824\Documents\E
Compile
Abort C
☐ Shorten compiler paths
Line: 19 Col: 2 Sel: 0 Lines: 19 Length: 419 Insert Done parsing in 0.016 seconds

```
Before Thread
Printing GeeksQuiz from Thread
After Thread

--------------------------------
Process exited after 3.269 seconds with return value 0
Press any key to continue . . .
```

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\91824\Documents\EXP7 OS.exe
- Output Size: 184.48046875 KiB
- Compilation Time: 0.27s
```

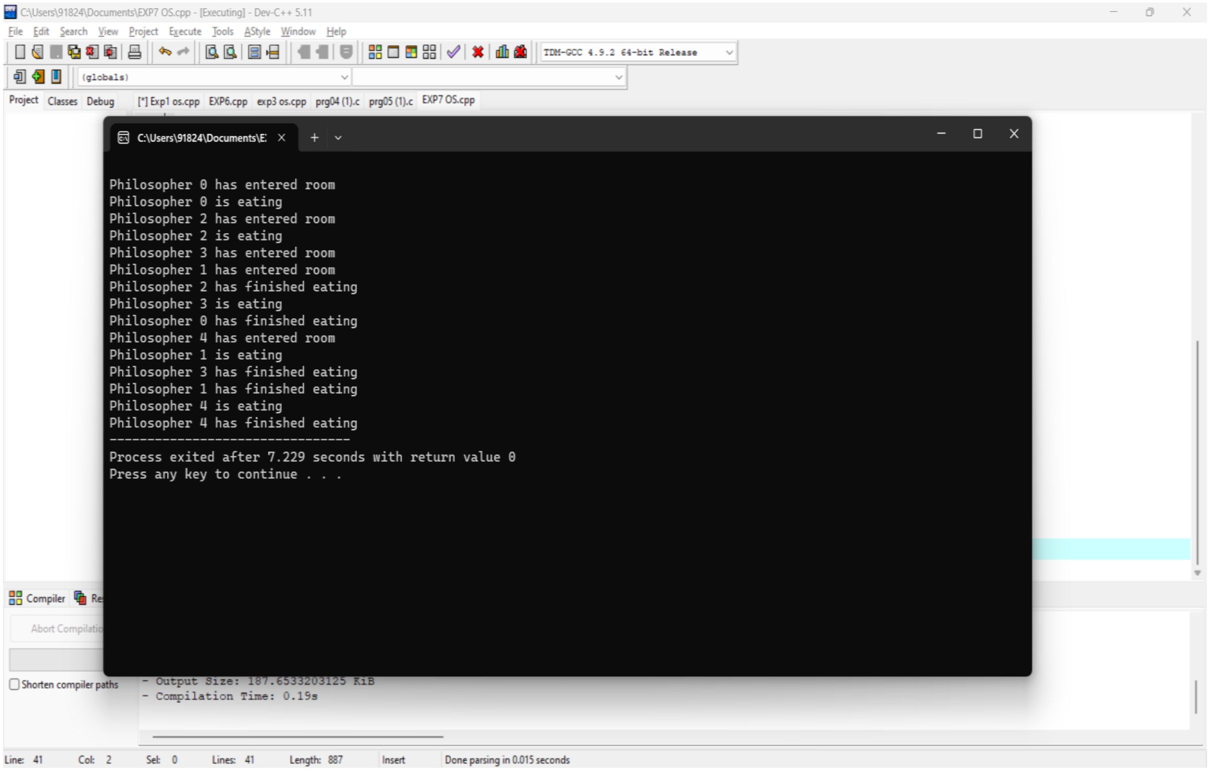
9. Design a C program to simulate the concept of Dining-Philosophers problem

Program**:**

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<unistd.h>
sem_t room;
sem_t chopstick[5];
void * philosopher(void *);
void eat(int);
```

```
int main()
{
        int i,a[5];
        pthread_t tid[5];
        sem_init(&room,0,4);
        for(i=0;i<5;i++)
                sem_init(&chopstick[i],0,1);
        for(i=0;i<5;i++){
                a[i]=i;
                pthread_create(&tid[i],NULL,philosopher,(void *)&a[i]);
        }
        for(i=0;i<5;i++)
                pthread_join(tid[i],NULL);
}
void * philosopher(void * num)
{
        int phil=*(int *)num;
        sem_wait(&room);
        printf("\nPhilosopher %d has entered room",phil);
        sem_wait(&chopstick[phil]);
        sem_wait(&chopstick[(phil+1)%5]);
        eat(phil);
        sleep(2);
        printf("\nPhilosopher %d has finished eating",phil);
        sem_post(&chopstick[(phil+1)%5]);
        sem_post(&chopstick[phil]);
        sem_post(&room);
}
void eat(int phil)
{
        printf("\nPhilosopher %d is eating",phil);}
```

INPUT AND OUTPUT:



C:\Users\91824\Documents\EXP7 OS.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] Exp1 os.cpp EXP6.cpp exp3 os.cpp prg04 (1).c prg05 (1).c EXP7 OS.cpp
C:\Users\91824\Documents\E
Compiler
Abort Compilation
Shorten compiler paths
Line: 41 Col: 2 Sel: 0 Lines: 41 Length: 887 Insert Done parsing in 0.015 seconds

```
Philosopher 0 has entered room
Philosopher 0 is eating
Philosopher 2 has entered room
Philosopher 2 is eating
Philosopher 3 has entered room
Philosopher 1 has entered room
Philosopher 2 has finished eating
Philosopher 3 is eating
Philosopher 0 has finished eating
Philosopher 4 has entered room
Philosopher 1 is eating
Philosopher 3 has finished eating
Philosopher 1 has finished eating
Philosopher 4 is eating
Philosopher 4 has finished eating
--------------------------------
Process exited after 7.229 seconds with return value 0
Press any key to continue . . .
```

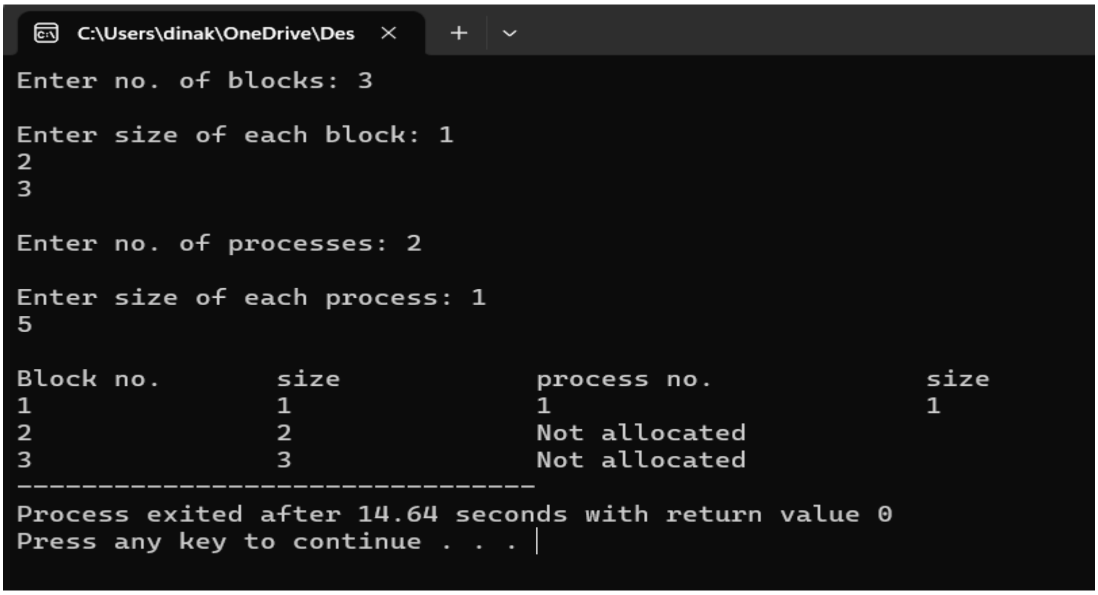
```
- Output Size: 187.6533203125 KiB
- Compilation Time: 0.19s
```

10. Construct a C program for implementation of memory allocation using first fit strategy.

Program:

```
#include<stdio.h>

int main()

{

        int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;

        for(i = 0; i < 10; i++)

        {

                flags[i] = 0;

                allocation[i] = -1;

        }

        printf("Enter no. of blocks: ");

        scanf("%d", &bno);

        printf("\nEnter size of each block: ");

        for(i = 0; i < bno; i++)

                scanf("%d", &bsize[i]);

        printf("\nEnter no. of processes: ");

        scanf("%d", &pno);
        printf("\nEnter size of each process: ");
        for(i = 0; i < pno; i++)
                scanf("%d", &psize[i]);
        for(i = 0; i < pno; i++)
                for(j = 0; j < bno; j++)
                        if(flags[j] == 0 && bsize[j] >= psize[i])
                        {
                                allocation[j] = i;
                                flags[j] = 1;
                                break;
                        }
        printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
        for(i = 0; i < bno; i++)
        {
                printf("\n%d\t\t%d\t\t", i+1, bsize[i]);
                if(flags[i] == 1)

        printf("%d\t\t\t%d",aZZZZZZZZZZZZZZZZZZZZAXwqwqe2d22wwwwwwwwwwwwwwwwwwww
wwwwwwwwwwwllocation[i]+1,psize[allocation[i]]);
                else
                        printf("Not allocated");
        }
}
```

INPUT AND OUTPUT:



```
Enter no. of blocks: 3

Enter size of each block: 1
2
3

Enter no. of processes: 2

Enter size of each process: 1
5

Block no.       size            process no.             size
1               1               1                       1
2               2               Not allocated
3               3               Not allocated
--------------------------------
Process exited after 14.64 seconds with return value 0
Press any key to continue . . .
```