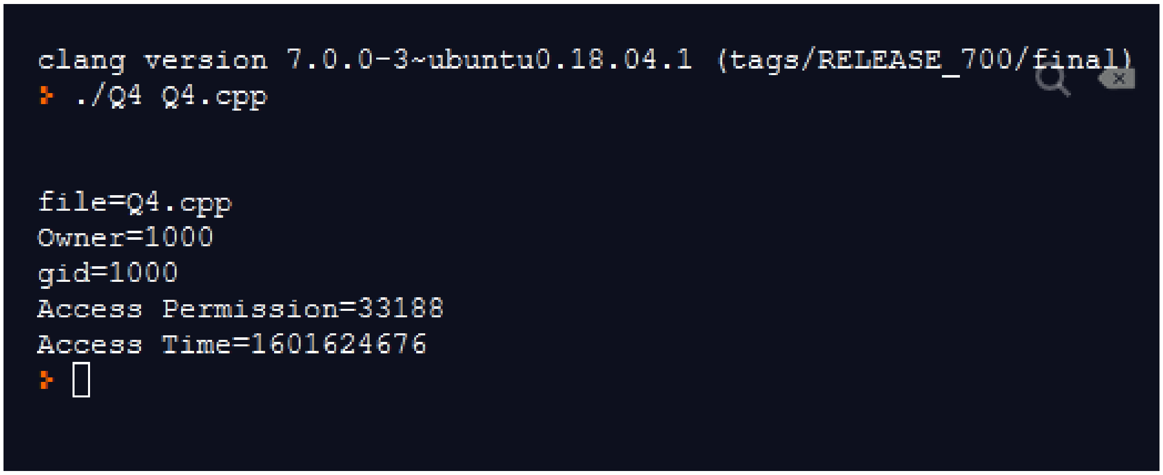
4. Write a program to print file details including owner access permissions, file access time, where file name is given as argument.

A4.

```
#include<stdio.h>
#include<sys/stat.h>
#include<time.h>

//Give file names in command line arguments
int main(int argc,char *argv[])
{
 int i;
 struct stat buffer;
 for(i=1;i<argc;i++)
 {
  printf("\n\nfile=%s\n",argv[i]);
  if(stat(argv[i],&buffer)<0)
   printf("Error in File Started");
  else
  {
   printf("Owner=%d\ngid=%d\n",buffer.st_uid,buffer.st_gid);
   printf("Access Permission=%d\n",buffer.st_mode);
   printf("Access Time=%ld\n", buffer.st_atime);
  // printf(time(&(buffer.st_atime)));
  }
 }
}
```

OUTPUT



```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
❯ ./Q4 Q4.cpp


file=Q4.cpp
Owner=1000
gid=1000
Access Permission=33188
Access Time=1601624676
❯ 
```

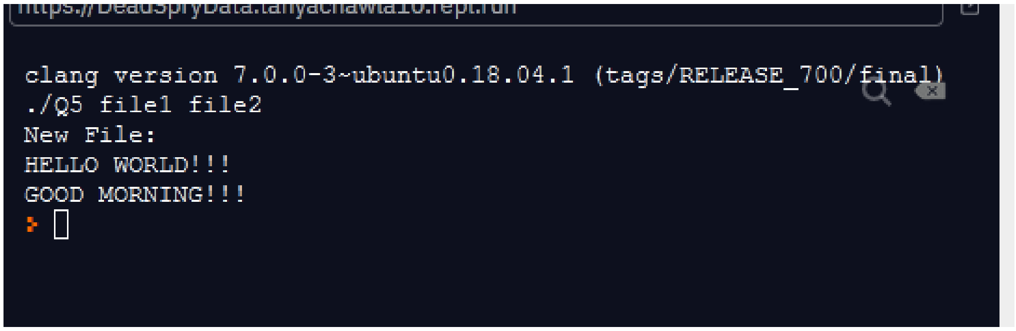
5. Write a program to copy files using system calls.

A5.

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
void copy(int,int);
void display(int);
int main(int argc,char *argv[])
{
 int fold,fnew;
 if(argc!=3)
 {
  printf("Two Arguments Required");
  exit(1);
 }
 fold=open(argv[1],0);
 if(fold==-1)
 {
  printf("Unable to Open the File\n%s",argv[1]);
  exit(1);
 }
 fnew=creat(argv[2],0666);
 if(fnew==-1)
 {
  printf("Unable to Create the File%s\n",argv[2]);
  exit(1);
 }
 copy(fold,fnew);
 //exit(0);
 close(fold);
 close(fnew);
 fnew=open(argv[2],0);
 printf("New File:\n");
 display(fnew);
 close(fnew);
 exit(0);
}

void copy(int old, int newfile)
```

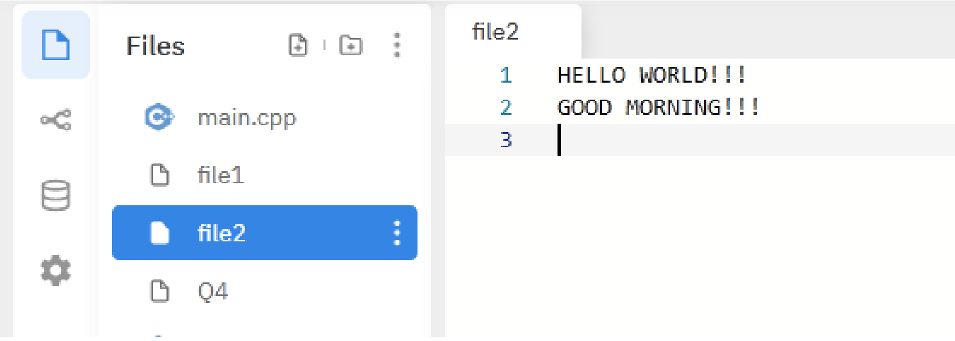
```
{
 int count=0;
 char buffer[512];
 while((count=read(old,buffer,sizeof(buffer)))>0)
 {
  write(newfile,buffer,count);
 }
}

void display(int fnew)
{
 int count=0,i;
 char buffer[512];
 while((count=read(fnew,buffer,sizeof(buffer)))>0)
 {
  for(i=0;i<count;i++)
  {
   printf("%c",buffer[i]);
  }
 }
 for(i=0;i<count;i++)
 {
  printf("%c",buffer[i]);
 }
}
```

OUTPUT



https://DeadSpryData.tanyachawla10.repl.run

```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
./Q5 file1 file2
New File:
HELLO WORLD!!!
GOOD MORNING!!!
```



```
HELLO WORLD!!!
GOOD MORNING!!!
```

7. Write a program to implement Round Robin scheduling algorithm.

A7.

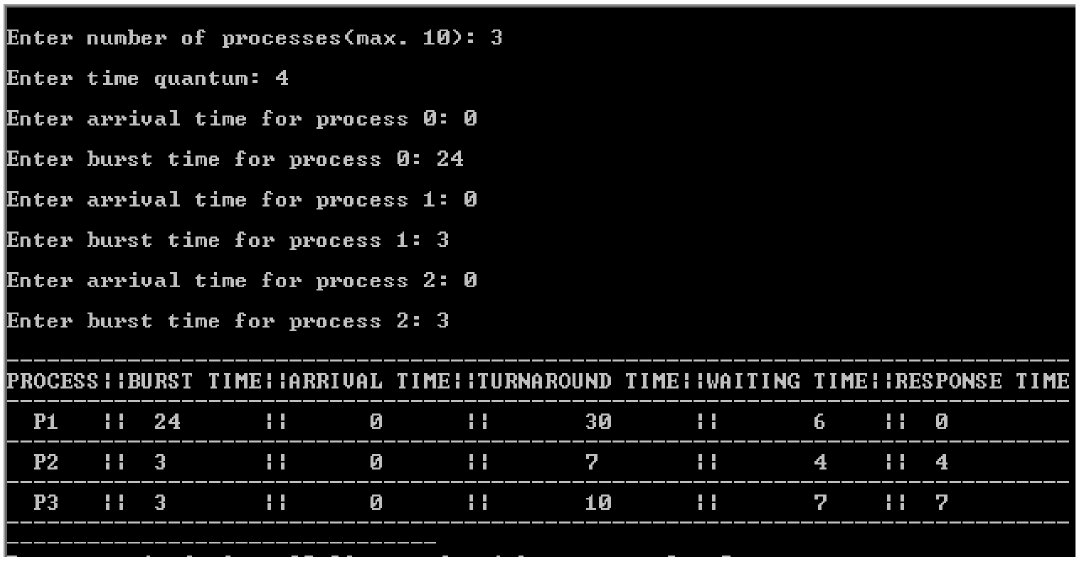
```
#include<iostream>
using namespace std;


class Process
{
        int ar;
        int bt;
        int wt;
        int res;
        int tar;
        int pid;
        int temp;
        public:
                void entry(int n)
                {
                        pid=n+1;
                        cout<<"\nEnter arrival time for process "<<n<<": ";
                        cin>>ar;
                        cout<<"\nEnter burst time for process "<<n<<": ";
                        cin>>bt;
                        temp=bt;
                        res=wt=tar=0;
                }
                void sorter(Process p[], int size)
                {
                        for(int i=0; i<size; ++i)
                        {
                                for(int j=0; j<size-1; ++j)
                                {
                                        if(p[j].ar>p[j+1].ar)
                                        {
                                                Process temp=p[j];
                                                p[j]=p[j+1];
                                                p[j+1]=temp;
                                        }
                                }
                        }
                }
                void final(Process p[], int q, int n)
                {

                        int i=0;
                        int check=0;
                        do
                        {
                                if(p[i].res==0 && i!=0)
                                        p[i].res=p[i].tar;
```

```
if(p[i].temp>0)
{
    if(p[i].temp-q>0)
    {
        p[i].temp-=q;
        for(int j=0; j<n; j++)
        {
            if(p[j].temp!=0)
                p[j].tar+=q;
        }
    }
    else
    {
        for(int j=0; j<n; j++)
        {
            if(p[j].temp!=0)
                p[j].tar+=p[i].temp;
        }
        p[i].temp=0;
        check++;
    }
}
if(i==n-1)
{
    i=0;
}
else
```

```
                    i++;
            }while(check<n);
        }

        void show(Process p[], int sz)
        {
            cout<<"\n--------------------------------------------------------------------------------";
            cout<<"\nPROCESS||BURST TIME||ARRIVAL TIME||TURNAROUND TIME||WAITING TIME||RESPONSE TIME" ;
            cout<<"\n--------------------------------------------------------------------------------";
            for(int i=0; i<sz; i++)
            {
                cout<<"\n  P"<<p[i].pid;
                cout<<"   ||  "<<p[i].bt;
                cout<<"\t   ||     "<<p[i].ar;

                p[i].tar=p[i].tar-p[i].ar;
                p[i].wt=p[i].tar-p[i].bt;

                cout<<"     ||      "<<p[i].tar;
                cout<<"\t   ||      "<<p[i].wt;
                cout<<"\t ||  "<<p[i].res;
                cout<<"\n--------------------------------------------------------------------------------";
            }

        }
};
int main()
```

```
{
        Process pr[10];
        int n, quant;
        cout<<"\nEnter number of processes(max. 10): ";
        cin>>n;
        if(n<0||n>10)
        {
                cout<<"\nWrong input";
                exit(0);
        }
        cout<<"\nEnter time quantum: ";
        cin>>quant;
        for(int i=0; i<n; i++)
        {
                pr[i].entry(i);
        }
        pr[0].sorter(pr,n);
        pr[0].final(pr, quant, n);
        pr[0].show(pr, n);
        return 0;
}
```

OUTPUT



```
Enter number of processes(max. 10): 3

Enter time quantum: 4

Enter arrival time for process 0: 0

Enter burst time for process 0: 24

Enter arrival time for process 1: 0

Enter burst time for process 1: 3

Enter arrival time for process 2: 0

Enter burst time for process 2: 3
-------------------------------------------------------------------------------
PROCESS||BURST TIME||ARRIVAL TIME||TURNAROUND TIME||WAITING TIME||RESPONSE TIME
-------------------------------------------------------------------------------
  P1   ||  24      ||      0      ||       30      ||       6    ||  0
-------------------------------------------------------------------------------
  P2   ||  3       ||      0      ||       7       ||       4    ||  4
-------------------------------------------------------------------------------
  P3   ||  3       ||      0      ||       10      ||       7    ||  7
-------------------------------------------------------------------------------
--------------------------------
```

9. Write a program to implement non-preemptive priority based scheduling algorithm.

A9.

```
#include<iostream>

using namespace std;

class Process

{

        int at;

        int bt;

        int res;

        int tar;

        int wt;

        int priority;

        int pid;

        float avgtt;

        float avgwt;

        public:

                void entry(int n)

                {
```

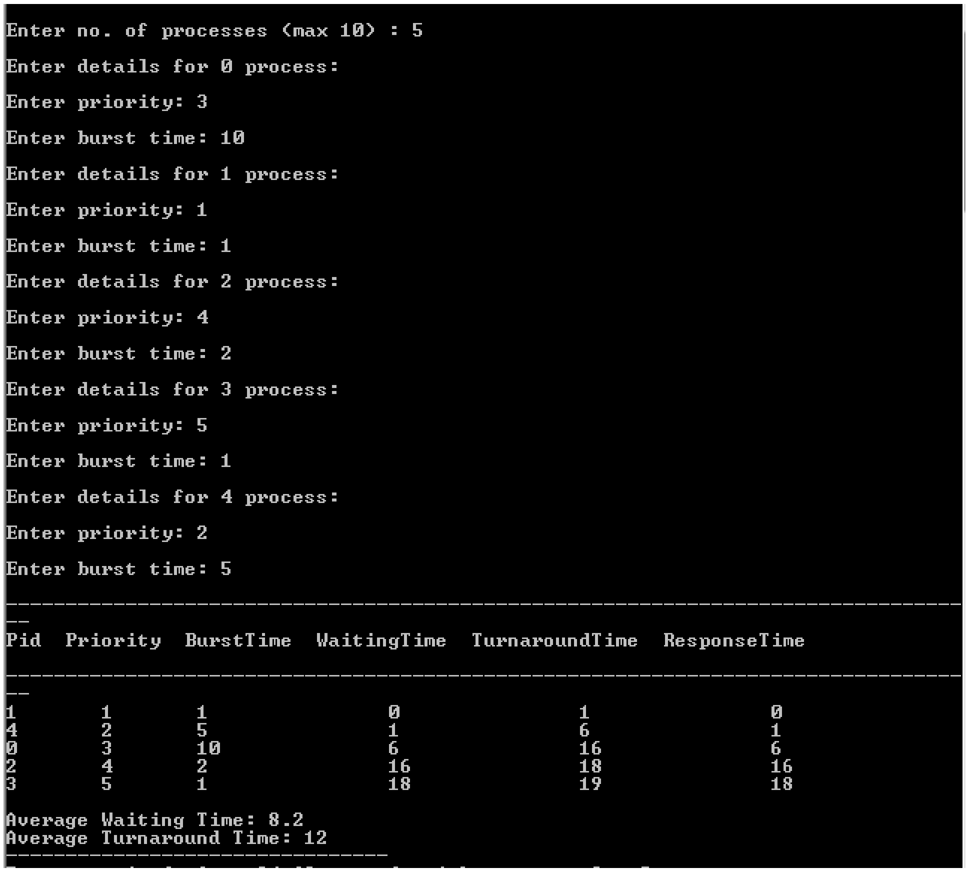
```
    pid=n;
    cout<<"\nEnter priority: ";
    cin>>priority;
    cout<<"\nEnter burst time: ";
    cin>>bt;
    at=res=wt=tar=0;
}
void sorter(Process p[], int size)
{
    for(int i=0; i<size; i++)
    {
        for(int j=0; j<size-1; ++j)
        {
            if(p[j].priority>p[j+1].priority)
            {
                Process temp=p[j];
                p[j]=p[j+1];
                p[j+1]=temp;
            }
        }
    }
}

void cal_wait(Process p[], int sz)
{
    avgwt=0;
    p[0].wt=0;
    for(int a=1; a<sz; ++a)
```

```
        {
                p[a].wt=0;
                for(int b=0; b<a; ++b)
                        p[a].wt+=p[b].bt;
                p[a].wt-=p[a].at;
        }
        for(int i=0; i<sz; i++)
                avgwt+=p[i].wt;
        avgwt=avgwt/sz;
}

void cal_trn(Process p[], int s)
{
        avgtt=0;
        for(int i=0; i<s; ++i)
                p[i].tar=p[i].wt+p[i].bt;
        for(int i=0; i<s; i++)
                avgtt+=p[i].tar;
        avgtt=avgtt/s;
}

void cal_res(Process p[], int y)
{
        p[0].res=0;
        for(int i=1; i<y; ++i)
        {
                p[i].res = p[i-1].tar;
        }
```

```
                }

                void show_data(Process pr[], int x)
                {
                        cout<<"\n-----------------------------------------------------------------------------------";
                        cout<<"\nPid     Priority     BurstTime     WaitingTime     TurnaroundTime
ResponseTime\n";
                        cout<<"\n---------------------------------------------------------------------------------
\n";
                        for(int c=0; c<x; ++c)

        cout<<pr[c].pid<<"\t"<<pr[c].priority<<"\t"<<pr[c].bt<<"\t\t"<<pr[c].wt<<"\t\t"<
<pr[c].tar<<"\t\t"<<pr[c].res<<endl;
                        cout<<"\nAverage Waiting Time: "<<avgwt;
                        cout<<"\nAverage Turnaround Time: "<<avgtt;
                }
};
int main()
{
        Process pro[10];
        int n;
        cout<<"\nEnter no. of processes (max 10) : ";
        cin>>n;
        do
        {
                if(n<0||n>10)
                        {
                                cout<<"\nEnter again : ";
                                cin>>n;
                        }
```

```
	}while(n<0||n>10);

	for(int i=0; i<n; ++i)
	{
		cout<<"\nEnter details for "<<i<<" process: \n";
		pro[i].entry(i);
	}
	pro[0].sorter(pro, n);
	pro[0].cal_wait(pro, n);
	pro[0].cal_trn(pro, n);
	pro[0].cal_res(pro, n);
	pro[0].show_data(pro, n);

	return 0;
}
```

OUTPUT



```
Enter no. of processes (max 10) : 5

Enter details for 0 process:

Enter priority: 3

Enter burst time: 10

Enter details for 1 process:

Enter priority: 1

Enter burst time: 1

Enter details for 2 process:

Enter priority: 4

Enter burst time: 2

Enter details for 3 process:

Enter priority: 5

Enter burst time: 1

Enter details for 4 process:

Enter priority: 2

Enter burst time: 5

--------------------------------------------------------------------------------
--
Pid  Priority  BurstTime  WaitingTime  TurnaroundTime  ResponseTime

--------------------------------------------------------------------------------
--
1       1       1               0               1               0
4       2       5               1               6               1
0       3       10              6               16              6
2       4       2               16              18              16
3       5       1               18              19              18

Average Waiting Time: 8.2
Average Turnaround Time: 12
--------------------------------
```