LAB ASSESMENT 2

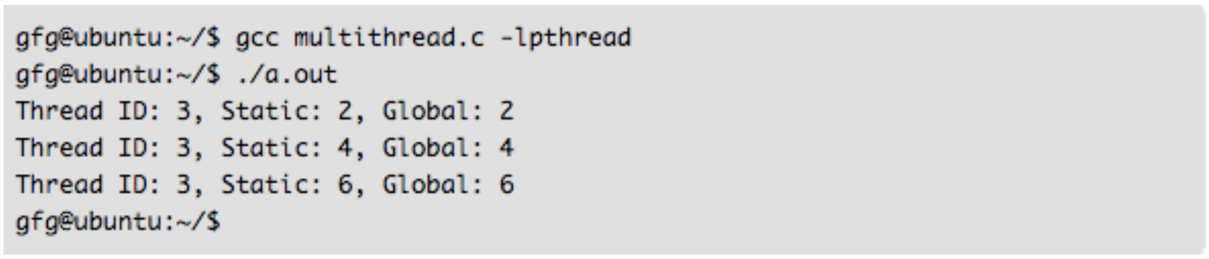
NAME : Siddhi Singh

REG.NO:17BIT0028

a) Write a program to create a thread and perform the following :

Create a thread runner function

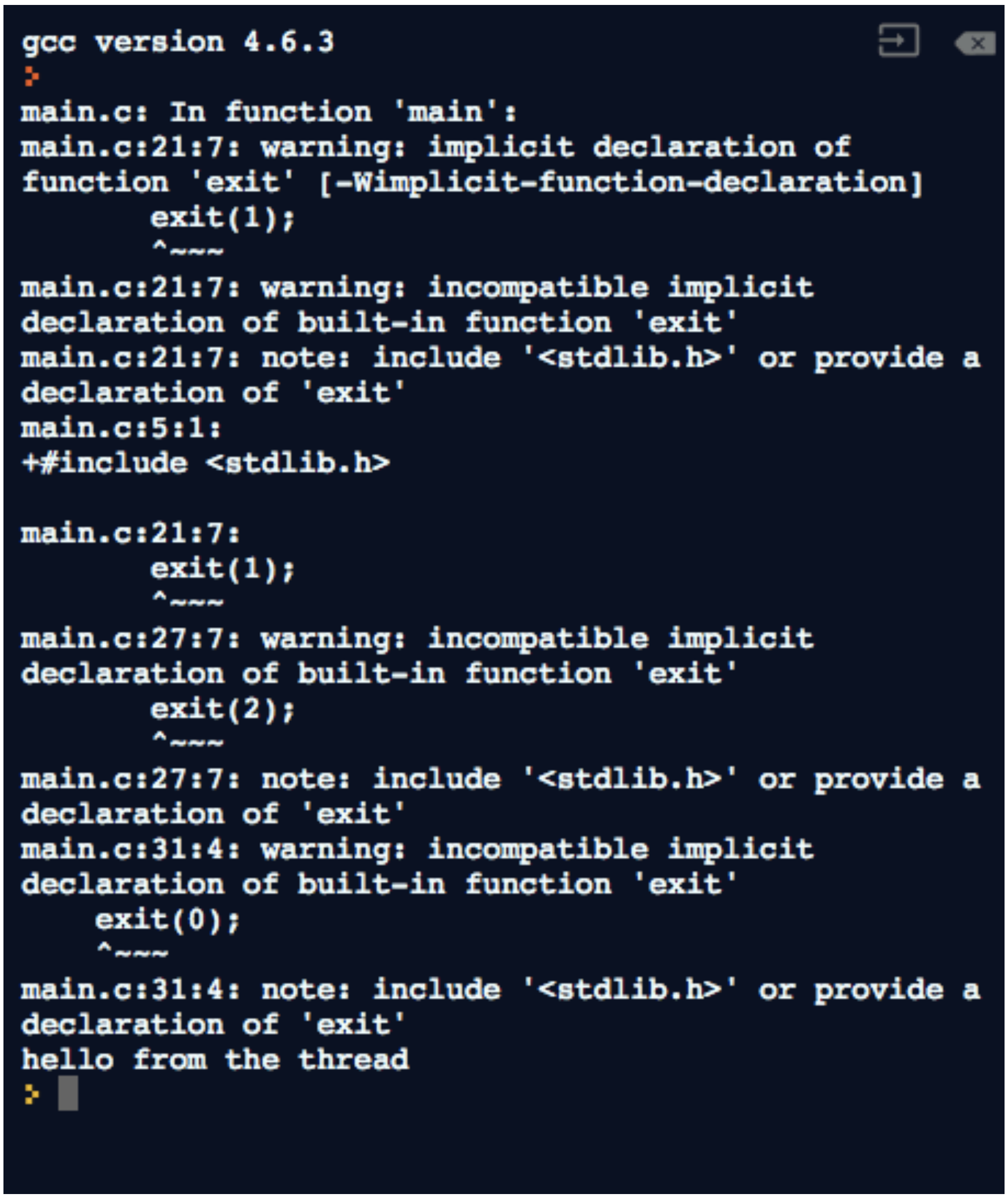
```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
    // Store the value argument passed to this thread
    int *myid = (int *)vargp;

    // Let us create a static variable to observe its changes
    static int s = 0;

    // Change static and global variables
    ++s; ++g;

    // Print the argument, static and global variables
    printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, +
+g);
}

int main()
{
    int i;
    pthread_t tid;

    // Let us create three threads
    for (i = 0; i < 3; i++)
        pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

    pthread_exit(NULL);
    return 0;
}
```



```
gfg@ubuntu:~/$ gcc multithread.c -lpthread
gfg@ubuntu:~/$ ./a.out
Thread ID: 3, Static: 2, Global: 2
Thread ID: 3, Static: 4, Global: 4
Thread ID: 3, Static: 6, Global: 6
gfg@ubuntu:~/$
```

Set the thread attributes

```
/* CELEBP10 */
#define _OPEN_THREADS
#include <stdio.h>
#include <pthread.h>

void *thread1(void *arg)
{
   printf("hello from the thread\n");
   pthread_exit(NULL);
}

int main()
{
   int            rc, stat;
   pthread_attr_t attr;
   pthread_t      thid;

   rc = pthread_attr_init(&attr);
   if (rc == -1) {
      perror("error in pthread_attr_init");
      exit(1);
   }

   rc = pthread_create(&thid, &attr, thread1, NULL);
   if (rc == -1) {
      perror("error in pthread_create");
      exit(2);
   }

   rc = pthread_join(thid, (void *)&stat);
   exit(0);
}
```

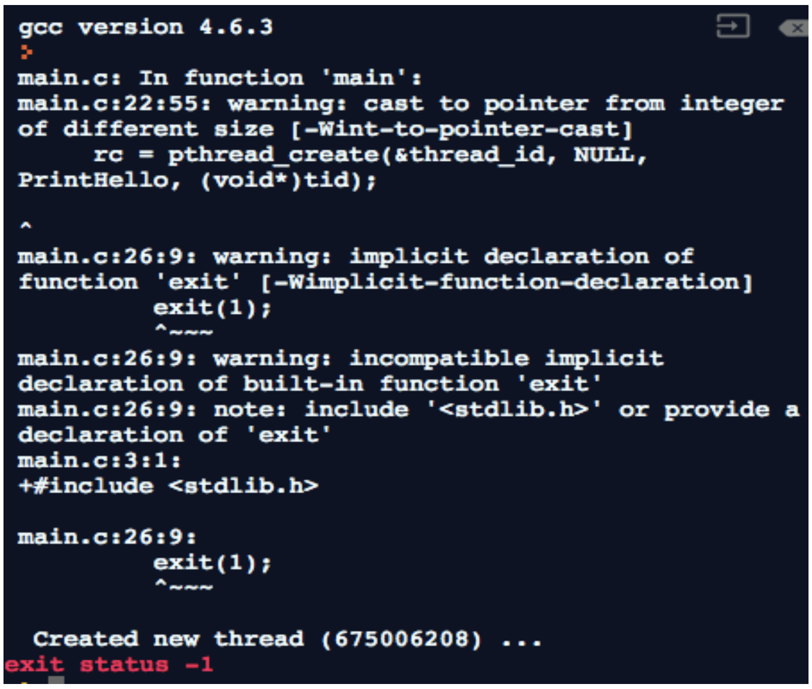


```
gcc version 4.6.3
>
main.c: In function 'main':
main.c:21:7: warning: implicit declaration of
function 'exit' [-Wimplicit-function-declaration]
       exit(1);
       ^~~~
main.c:21:7: warning: incompatible implicit
declaration of built-in function 'exit'
main.c:21:7: note: include '<stdlib.h>' or provide a
declaration of 'exit'
main.c:5:1:
+#include <stdlib.h>

main.c:21:7:
       exit(1);
       ^~~~
main.c:27:7: warning: incompatible implicit
declaration of built-in function 'exit'
       exit(2);
       ^~~~
main.c:27:7: note: include '<stdlib.h>' or provide a
declaration of 'exit'
main.c:31:4: warning: incompatible implicit
declaration of built-in function 'exit'
    exit(0);
    ^~~~
main.c:31:4: note: include '<stdlib.h>' or provide a
declaration of 'exit'
hello from the thread
>
```

Join the parent and thread

```
#include <stdio.h>       /* standard I/O routines                */
#include <pthread.h>     /* pthread functions and data structures */

void* PrintHello(void* data)
{
    pthread_t tid = (pthread_t)data;    /* data received by thread
*/

    pthread_join(tid, NULL);            /* wait for thread tid
*/
```

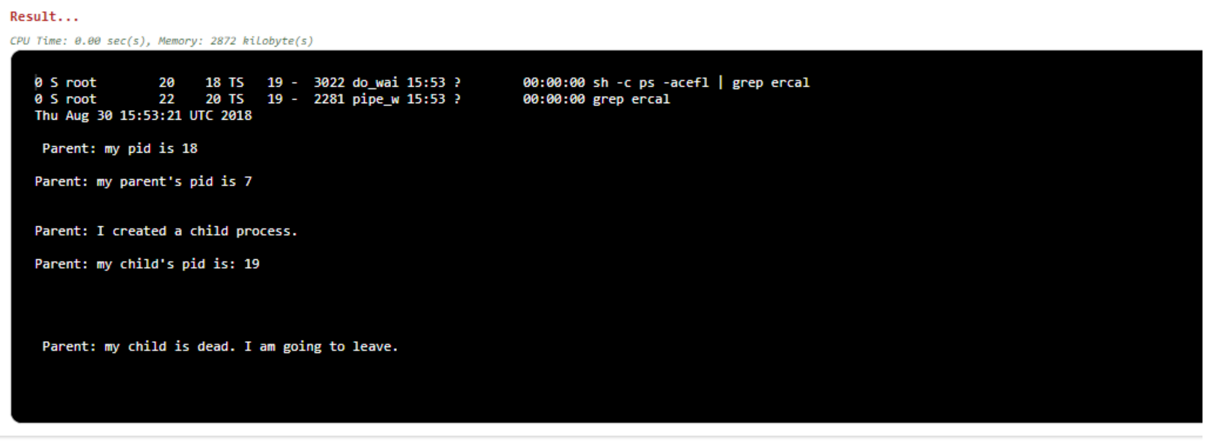
```
    printf("Hello from new thread %u - got %u\n", pthread_self(),
data);
    pthread_exit(NULL);                     /* terminate the thread
*/
}

/* like any C program, program's execution begins in main */
int main(int argc, char* argv[])
{
    int         rc;                  /* return value
*/
    pthread_t   thread_id;                /* thread's ID (just an
integer) */
    int         tid;

    tid = pthread_self();

    rc = pthread_create(&thread_id, NULL, PrintHello, (void*)tid);
    if(rc)                                 /* could not create thread */
    {
        printf("\n ERROR: return code from pthread_create is %d \n",
rc);
        exit(1);
    }
    sleep(1);
    printf("\n Created new thread (%u) ... \n", thread_id);
    pthread_exit(NULL);
}
```



```
gcc version 4.6.3

main.c: In function 'main':
main.c:22:55: warning: cast to pointer from integer
of different size [-Wint-to-pointer-cast]
     rc = pthread_create(&thread_id, NULL,
PrintHello, (void*)tid);

^
main.c:26:9: warning: implicit declaration of
function 'exit' [-Wimplicit-function-declaration]
         exit(1);
         ^~~~
main.c:26:9: warning: incompatible implicit
declaration of built-in function 'exit'
main.c:26:9: note: include '<stdlib.h>' or provide a
declaration of 'exit'
main.c:3:1:
+#include <stdlib.h>

main.c:26:9:
         exit(1);
         ^~~~

 Created new thread (675006208) ...
exit status -1
```

Wait for the thread to complete

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>

int main( int argc, char *argv[], char *env[] )
{
   pid_t my_pid, parent_pid, child_pid;
   int status;

/* get and print my pid and my parent's pid. */

   my_pid = getpid();    parent_pid = getppid();
   printf("\n Parent: my pid is %d\n\n", my_pid);
   printf("Parent: my parent's pid is %d\n\n", parent_pid);

/* print error message if fork() fails */
   if((child_pid = fork()) < 0 )
   {
      perror("fork failure");
      exit(1);
   }

/* fork() == 0 for child process */

   if(child_pid == 0)
   {  printf("\nChild: I am a new-born process!\n\n");
      my_pid = getpid();    parent_pid = getppid();
      printf("Child: my pid is: %d\n\n", my_pid);
      printf("Child: my parent's pid is: %d\n\n", parent_pid);
      printf("Child: I will sleep 3 seconds and then execute - date
- command \n\n");

      sleep(3);
      printf("Child: Now, I woke up and am executing date command
\n\n");
      execl("/bin/date", "date", 0, 0);
      perror("execl() failure!\n\n");

      printf("This print is after execl() and should not have been
executed if execl were successful! \n\n");

      _exit(1);
   }
/*
 * parent process
 */
   else
   {
      printf("\nParent: I created a child process.\n\n");
      printf("Parent: my child's pid is: %d\n\n", child_pid);
      system("ps -acefl | grep ercal");  printf("\n \n");
      wait(&status); /* can use wait(NULL) since exit status
                        from child is not used. */
      printf("\n Parent: my child is dead. I am going to leave.\n \n
");
   }

   return 0;
}
```



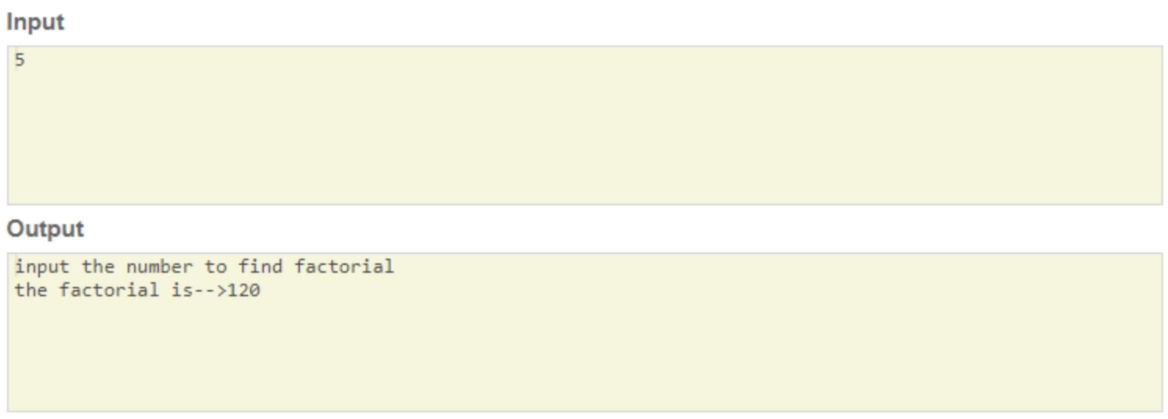
Result...

CPU Time: 0.00 sec(s), Memory: 2872 kilobyte(s)

```
0 S root       20    18 TS   19 -  3022 do_wai 15:53 ?        00:00:00 sh -c ps -acefl | grep ercal
0 S root       22    20 TS   19 -  2281 pipe_w 15:53 ?        00:00:00 grep ercal
Thu Aug 30 15:53:21 UTC 2018

 Parent: my pid is 18

Parent: my parent's pid is 7


Parent: I created a child process.

Parent: my child's pid is: 19




 Parent: my child is dead. I am going to leave.
```

b) Write a program to create a thread to find the factorial of a natural number ‘n’.

```
#include<iostream>

#include<Thread>

using namespace std;

void fac(int n)

{

    int res=1;

while(n!=1)

{

    res=res*n;

    n--;

}

cout<<"the factorial is-->"<<res<<endl;
}
int main(){
    cout<<"input the number to find factorial"<<endl;
    int n;
    cin>>n;
thread t(fac,n);


t.join();


return 0;
}
```



```
5
```

```
input the number to find factorial
the factorial is-->120
```

c) Assume that two processes named client and server running in the system. It is required that these two processes should communicate with each other using shared memory concept. The server writes alphabets from a..z to the shared memory . The client should read the alphabets from the shared memory and convert it to A…Z. Write a program to demonstrate the above mentioned scenario

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
Struct country
{
Char name[30];
Char capital_city [30];
Char currency[30];
Int population;
};
Int main(int argc,char*argv[])
{
Int shm_id;
Char*shm_addr;
Int*countries_num;
Struct country*countries;
Struct shmid_ds shm_desc;
Shm_id=shmget(100,2048,IPC_CREAT|IPC_EXCL\0600);
If(shm_id==-1){
Perror("main:shmget:");
Exit(1);
}
Shm_addr=shmat(shm_id,NULL,0);
If(!shm_addr){
Perror("main:shmat:");
Exit(1);
}
Countries_num=(int*)shm_addr;
*countries_num=0;
Countries=(struct country*)((void*)shm_addr sizeof(int));
Strcpy(countries[0],name,"U.S.A");
Strcpy(countries[0],capital_city,"WASHINGTON");
Strcpy(countries[0],currency,"U.S.DOLLAR");
Countries[0].population=250000000;
( countries_num)  ;
```

```
Strcpy(countries[1].name,"israel");
Strcpy(countries[1].capital_city,"jerushalem");
Strcpy(countries[1].currency,"NEW ISRAEL SHEKED");
Countries[1].population=6000000;
(*countries_num)  ;
Strcpy(countries[2].name,"France");
Strcpy(countries[2].capital_city,"paris");
Strcpy(countries[2].currency,"Frank");
Countries[2].population=60000000;
(*countries_num)  ;
For(i=0;i<(*countries_num);i  )
{
Printf("country%d:\n",i 1);
Printf("name:%d:\n",i 1);
Printf("currency:%s:\n",countries[i].currency);
Printf("population:%d:\n",countries[i].population);
}
If(shmdt(shm_addr)==-1){
Perror("main:shmdt:");
}
If(shmctl(shm_id,IPC_RMID,&SHM_DESC)==-1)
{
Perror("main:shmctl:");
}
return 0;
}
```

Output:

```
Student@ubuntu:~$gcc shm.c
Student@ubuntu:~$ ./a.out
Shared memory ID=65537 child pointer 3086680064
Child value =1
Shared memory ID=65537 child pointer 3086680064
Parent value=1
Parent value=42
Child value=42
```