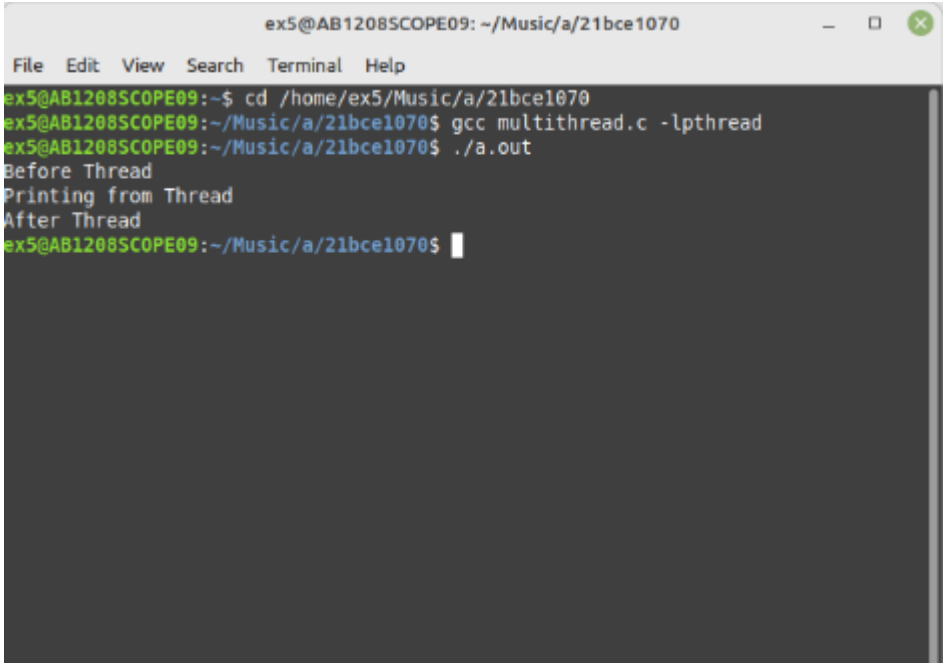
Threads

Code

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>  //Header file for sleep().

#include <pthread.h> //thread functions


// A normal C function that is executed as a thread

// when its name is specified in pthread_create()


void *myThreadFun(void *vargp) //thread creation function; will be given as argument for
pthread_create()

{

   sleep(1);

   printf("Printing from Thread \n");

   return NULL;

}



int main()

{

   pthread_t thread_id;  //thread variable creation //int can be also be used


   printf("Before Thread\n");


   pthread_create(&thread_id, NULL, myThreadFun,NULL); //creating thread and assigning to the
variable; menmonic -&variable,null,func,null


   pthread_join(thread_id, NULL); //wait till thread with thread_id terminates; second argument can
be char * variable that ca store exit statement from the thread function (later used)
```

```
    printf("After Thread\n");


    exit(0);
}
```

Output

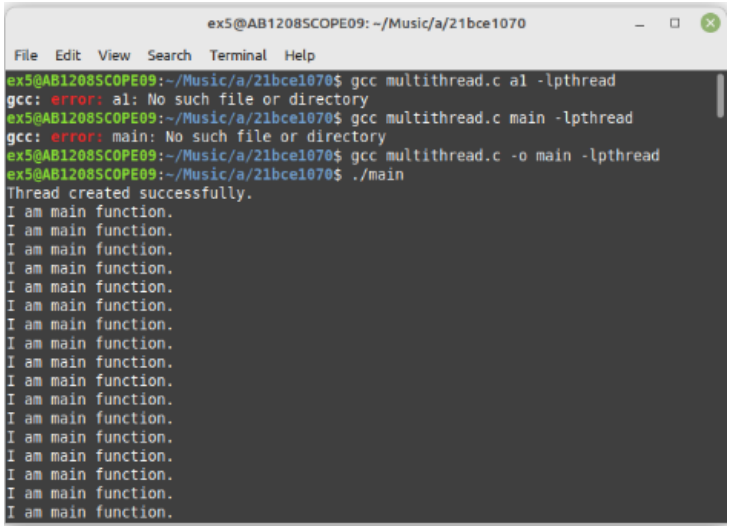


Code

```
#include <stdio.h>

#include <pthread.h>


/*thread function definition*/

void* threadFunction(void* args)

{

    while(1)

    {

        printf("I am threadFunction.\n");

    }

}
```

```
int main()
{

    pthread_t id; //variable
    int ret;

    /*creating thread*/
    ret=pthread_create(&id,NULL,&threadFunction,NULL); //thread creation, assigning to therad
variavle; return value
    if(ret==0){
        printf("Thread created successfully.\n");
    }
    else{
        printf("Thread not created.\n");
        return 0; /*return from main*/
    }

    while(1)
    {
        printf("I am main function.\n");
    }

    return 0;
}
```

Output

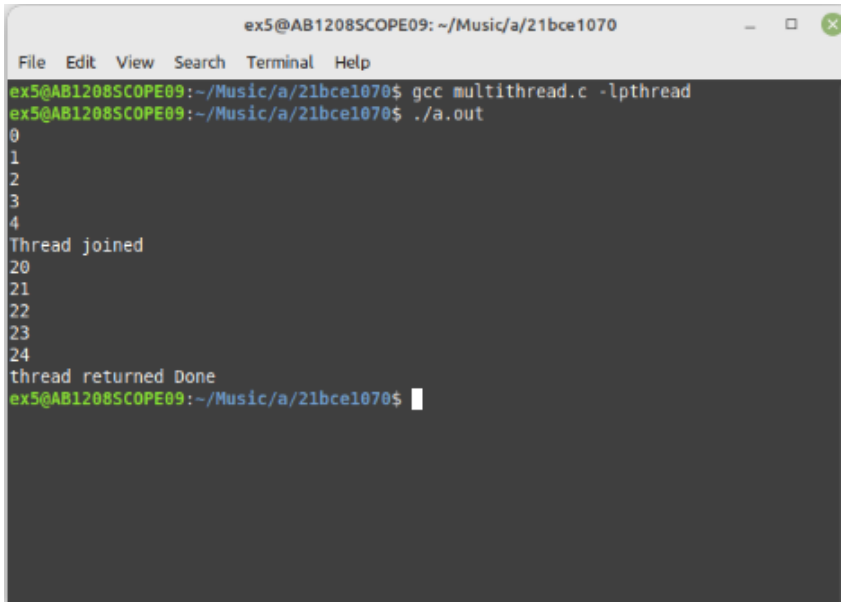


Code

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<pthread.h>

#include<string.h>



void *thread_function(void *arg);

 int i,n,j;



 int main() {

 char *m="5";

 pthread_t a_thread;  //thread declaration


 void *result;


 pthread_create(&a_thread, NULL, thread_function, m); //thread is created; last argument has to be
string
```

```
pthread_join(a_thread, &result); //pthread_exit statemnet will stored in result varibale

printf("Thread joined\n");
for(j=20;j<25;j++)
{
printf("%d\n",j);
sleep(1);
}
printf("thread returned %s\n",(char *)result);
}


void *thread_function(void *arg) {
int sum=0;
n=atoi(arg);  //atoi converts string of char in to integer
for(i=0;i<n;i++)
{
printf("%d\n",i);
sleep(1);
}
pthread_exit("Done"); // pthtread_exit(), in other programs, thread function terminated on its own
}
```

Output

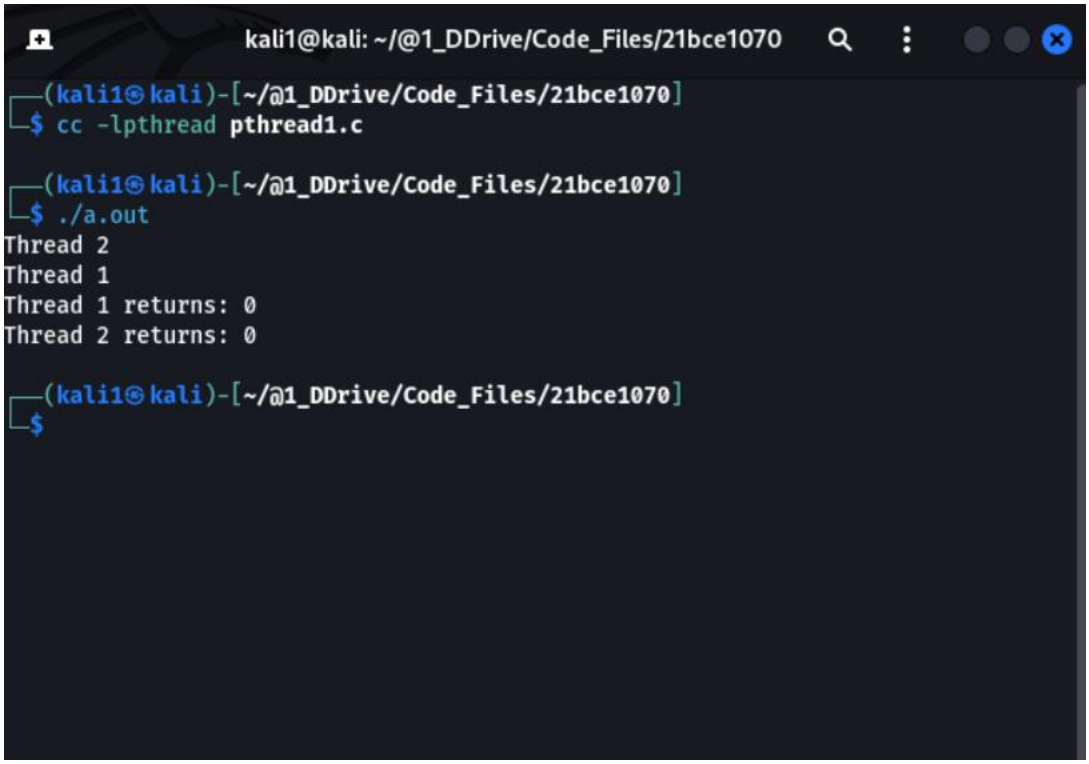


Code

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>


void *print_message_function( void *ptr );


int main()

{

    pthread_t thread1, thread2;

    char *message1 = "Thread 1";

    char *message2 = "Thread 2";

    int  iret1, iret2;


   /* Create independent threads each of which will execute function */


    iret1 = pthread_create( &thread1, NULL, print_message_function, (void*) message1);


    iret2 = pthread_create( &thread2, NULL, print_message_function, (void*) message2);
```

```
    /* Wait till threads are complete before main continues.
            Unless we  */
    /* wait we run the risk of executing an exit which will
            terminate   */
    /* the process and all threads before the threads
           have completed.   */

    pthread_join( thread1, NULL);
    pthread_join( thread2, NULL);

 /* The pthread_join() function shall suspend execution
 of the calling thread until the target thread terminates,
           unless the target thread has already terminated. */

    printf("Thread 1 returns: %d\n",iret1);
    printf("Thread 2 returns: %d\n",iret2);
    exit(0);
}

void *print_message_function( void *ptr )
{
    char *message;
    message = (char *) ptr;
    printf("%s \n", message);
}
```

Output



Code

```
#include <stdio.h>

#include <stdlib.h>

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>


using namespace std;


#define NUM_THREADS 5


void *PrintHello(void *threadid)

 {

   long tid;

   tid = (long)threadid;

   printf("Hello World! Thread ID, %d\n", tid);
```

```
  pthread_exit(NULL);
            }

int main ()
 {
  pthread_t threads[NUM_THREADS];
  int rc;
  long i; //int i will not work
  for( i = 0; i < NUM_THREADS; i++ )
{
     printf ( "main() : creating thread, %ld\n ",i );

rc = pthread_create(&threads[i], NULL, PrintHello, (void *)i);

     if (rc)
 {
       printf("Error:unable to create thread, %d\n", rc);
       exit(-1);
     }
  }
  pthread_exit(NULL);
}
```

Output

```
kali1@kali: ~/@1_DDrive/Code_Files/21bce1070

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ gcc thread1.cpp

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ ./a.out
main() : creating thread, 0
 main() : creating thread, 1
 main() : creating thread, 2
 main() : creating thread, 3
 main() : creating thread, 4
 Hello World! Thread ID, 2
Hello World! Thread ID, 0
Hello World! Thread ID, 4
Hello World! Thread ID, 1
Hello World! Thread ID, 3

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$
```