Practical-4

Objective: Thread creation and Termination (pthread_create and pthread_join)

What is a Thread?

A thread can be loosely defined as a separate stream of execution that takes place simultaneously with and independently of everything else that might be happening. A thread is like a classic program that starts at point **A** and executes until it reaches point **B**. It does not have an event loop.

A thread runs independently of anything else happening in the computer. Without threads an entire program can be held up by one CPU intensive task or one infinite loop, intentional or otherwise. With threads the other tasks that don't get stuck in the loop can continue processing without waiting for the stuck task to finish.

It turns out that implementing threading is harder than implementing multitasking in an operating system. The reason it's relatively easy to implement multitasking is that individual programs are isolated from each other. Individual threads, however, are not.

Threads vs. Processes

Both threads and processes are methods of parallelizing an application. However, processes are independent execution units that contain their own state information, use their own address spaces, and only interact with each other via interprocess communication mechanisms (generally managed by the operating system).

Applications are typically divided into processes during the design phase, and a master process explicitly spawns sub-processes when it makes sense to logically separate significant application functionality. Processes, in other words, are an architectural construct.

By contrast, a thread is a coding construct that doesn't affect the architecture of an application. A single process might contains multiple threads; all threads within a process share the same state and same memory space, and can communicate with each other directly, because they share the same variables.

Threads typically are spawned for a short-term benefit that is usually visualized as a serial task, but which doesn't have to be performed in a linear manner (such as performing a complex mathematical computation using parallelism, or initializing a large matrix), and then are absorbed when no longer required.

Thread operations include thread creation, termination, synchronization (joins,blocking), scheduling, data management and process interaction.

A thread does not maintain a list of created threads, nor does it know the thread that created it.

All threads within a process share the same address space.

Threads in the same process share:

- Process instructions
- Most data
- open files (descriptors)
- signals and signal handlers
- current working directory
- User and group id

Each thread has a unique:

- Thread ID
- set of registers, stack pointer
- stack for local variables, return addresses
- signal mask
- priority
- Return value: errno

PROGRAM 4.1

AIM: Write a simple program that will create two threads, main thread should wait for both the thread to complete.

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

void *print_message_function( void *ptr );

main()
{
     pthread_t thread1, thread2;
     char *message1 = "I am Thread 1";
     char *message2 = "I am 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);

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

     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);
}
```

Compile:

gcc pthread1.c -o pthread1 -lpthread

OUT PUT 3.2

I am Thread 1
I am Thread 2
Thread 1 returns: 0
Thread 2 returns: 0

EXERCISES – 4

4.1) Write a program to multiply two matrices using threads. Let say first matrix M1 of dimensions A X B and second matrix M2 of dimension B X C. Identify the number of threads required for multiplication.