National University of Computer and Emerging Sciences



**Laboratory Manual**

*for*

**Operating Systems Lab**

**(CS 205)**

| Course Instructor | Dr Rana Asif |
| --- | --- |
| Lab Instructor(s) | Mr. Usman Anwer |
| Section | CS-5A |
| Semester | Fall 2022 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Lab Topic:**

* Threads, Multithreading

**Lab Objectives:**

* **Understanding difference between threads and processes**
* **Concurrency in threads.**
* **Using pthread library.**

## Difference between Process and Thread

| **S.N.** | **Process** | **Thread** |
| --- | --- | --- |
| 1 | Process is heavy weight or resource intensive. | Thread is light weight, taking lesser resources than a process. |
| 2 | Process switching needs interaction with operating system. | Thread switching does not need to interact with operating system. |
| 3 | In multiple processing environments, each process executes the same code but has its own memory and file resources. | All threads can share same set of open files, child processes. |
| 4 | If one process is blocked, then no other process can execute until the first process is unblocked. | While one thread is blocked and waiting, a second thread in the same task can run. |
| 5 | Multiple processes without using threads use more resources. | Multiple threaded processes use fewer resources. |
| 6 | In multiple processes each process operates independently of the others. | One thread can read, write or change another thread's data. |

## 

## Advantages of Thread

* Threads minimize the context switching time.
* Use of threads provides concurrency within a process.
* Efficient communication.
* It is more economical to create and context switch threads.
* Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

## Types of Thread

Threads are implemented in following two ways −

* **User Level Threads** − User managed threads.
* **Kernel Level Threads** − Operating System managed threads acting on kernel, an operating system core.

1. **Pthread\_create**

**#include <pthread.h>**

***int pthread\_create(pthread\_t \*****thread****, const pthread\_attr\_t \*****attr****,******void \*(\*****start\_routine****) (void \*), void \*****arg****);***

Compile and link with *-pthread*.

The **pthread\_create**() function starts a new thread in the calling process. The new thread starts execution by invoking

*start\_routine*(); *arg* is passed as the sole argument of

*start\_routine*().

On success, **pthread\_create**() returns 0; on error, it returns an error number, and the contents of *\*thread* are undefined.

1. **Pthread\_join**

**#include <pthread.h>**

**int pthread\_join(pthread\_t** *thread***, void \*\****retval***);**

Compile and link with *-pthread*.

The **pthread\_join**() function waits for the thread specified by *thread* to terminate. If that thread has already terminated, then **pthread\_join**() returns immediately. The thread specified by *thread* must be joinable.

1. **Pthread\_exit**

**#include <pthread.h>**

**void pthread\_exit(void \****retval***);**

Compile and link with *-pthread*.

1. **DESCRIPTION**[**top**](http://man7.org/linux/man-pages/man3/pthread_exit.3.html#top_of_page)

The **pthread\_exit**() function terminates the calling thread and returns

a value via *retval* that (if the thread is joinable) is available to

another thread in the same process that calls [pthread\_join(3)](http://man7.org/linux/man-pages/man3/pthread_join.3.html)

This call always succeeds and does not return anything in the calling process.

* Maximum number of thread that system allows : cat /proc/sys/kernel/threads-max
* In a Multithreaded process all processes have the same pid, how to uniquely identify thread?
* Compiling multithreaded programs with-lpthread
* How to uniquely identify threads in a multithreaded program.
* Threads executed concurrently

Command to see thread ID:

Cd /proc/pid

Ls task

Every thread has a set of attributes.

Default set of attributes: Table 3-1 Default Attribute Values for *tattr*

| **Attribute** | **Value** | **Result** |
| --- | --- | --- |
| *scope* | PTHREAD\_SCOPE\_PROCESS | New thread is unbound - not permanently attached to LWP. |
| *detachstate* | PTHREAD\_CREATE\_JOINABLE | Exit status and thread are preserved after the thread terminates. |
| *stackaddr* | NULL | New thread has system-allocated stack address. |
| *stacksize* | 1 megabyte | New thread has system-defined stack size. |
| *priority* |  | New thread inherits parent thread priority. |
| *inheritsched* | PTHREAD\_INHERIT\_SCHED | New thread inherits parent thread scheduling priority. |
| *schedpolicy* | SCHED\_OTHER | New thread uses Solaris-defined fixed priority scheduling; threads run until preempted by a higher-priority thread or until they block or yield. |

**Points to Ponder:**

1) if a thread executes fork system call (can a thread executes fork call)

2) if a thread executes exec system call (execute execlp system call)

**Task 1:**

Write a program that creates two synchronous threads and pass the file names f1.txt and f2.txt as a parameter to each thread function and it will memory map the file and will remove all the duplicate and negative values from the file and then return the unique and non-negative values it to the main thread and will not display it. The main thread will then compute the average of the unique integers and will display it on the screen.

**Task 2:**

Write a program twoMany.c that will create a number N of threads specified in the command line, each of which prints out a hello message and its own thread ID. To see how the execution of the threads interleaves, make the main thread sleep for 1 second for every 4 or 5 threads it creates. The output of your code should be similar to: (hint: *pthread\_self()* function to get thread id and *pthread\_exit()* to exit the thread execution)

I am thread 1. Created new thread (4) in iteration 0...

Hello from thread 4 - I was created in iteration 0

I am thread 1. Created new thread (6) in iteration 1...

I am thread 1. Created new thread (7) in iteration 2...

I am thread 1. Created new thread (8) in iteration 3...

I am thread 1. Created new thread (9) in iteration 4...

I am thread 1. Created new thread (10) in iteration 5...

Hello from thread 6 - I was created in iteration 1

Hello from thread 7 - I was created in iteration 2

Hello from thread 8 - I was created in iteration 3

Hello from thread 9 - I was created in iteration 4

Hello from thread 10 - I was created in iteration 5

I am thread 1. Created new thread (11) in iteration 6...

I am thread 1. Created new thread (12) in iteration 7...

Hello from thread 11 - I was created in iteration 6

Hello from thread 12 - I was created in iteration 7