National University of Computer and Emerging Sciences

Laboratory Manual

for

Operating Systems Lab

(CS 205)

Course Instructor	Mr Raziuddin
Lab Instructor(s)	Mr. Usman Anwer Mr Tahir Mahmood
Section	CS-4C
Semester	Spring 2024

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.**

1.1 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.
```

2) 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.
```

3) Pthread_exit

```
#include <pthread.h>

void pthread_exit(void *retval);

Compile and link with -pthread.
```

1 DESCRIPTION top

```
       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)
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.

New thread inherits parent thread scheduling priority.

inheritsched `PTHREAD_INHERI T_SCHED`

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: (Row - Wise SUM)

Suppose a file (input.txt) do have the following contents:

```
65  12  45  78  98

23  87  56   1  34

99  17   9  88  72

41  67  54  33  20

76  14  92  63  31
```

The matrix can have N Rows, the main process reads the contents of the file and creates N Threads. Each row is passed to a separate thread. Each thread computes the sum of the integer values and returns the computed value to the main thread. Now the job of the main thread is to compute and display the total sum based upon the values returned from N threads.

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
```