National University of Computer and Emerging Sciences



Laboratory Manual

for

Operating Systems Lab

(CL-220)

|  |  |
| --- | --- |
|  | Mr. Ibtahim Nadir |
| Lab Instructor(s) | Zumirrah Khalid,  Mohammad Amir Waseem |
| Section | 4E |
| Semester | Spring 2021 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Threads**

**OBJECTIVE:**

∙ To understand and learn about threads and their implementation in programs **BACKGROUND:**

**Threads:**

pthread\_create(pthread\_t\* , NULL, void\*, void\*)

∙ First Parameter is pointer of thread ID it should be different for all threads. ∙ Second Parameter is used to change stack size of thread. Null means use default size.  ∙ Third perimeter is address of function which we are going to use as thread.  ∙ Forth parameter is argument to function.

pthread\_join(i pthread\_t , void\*\*)

Pthread join is used in main program to wait for the end of a particular thread.

∙ First parameter is Thread ID of particular thread.

∙ Second Parameter is used to catch return value from thread.

**Thread Library:**

∙ POSIX Pthreads

∙ Two general stratergies for creating multiple threads.

a) Asynchronous threading:

▪ Parent and child threads run independently of each other

▪ Typically little data sharing between threads

b) Synchronous threading:

▪ Parent thread waits for all of its children to terminate

▪ Children threads run concurrently

▪ Significant data sharing

**Pthreads:**

∙ **pthread.h**

∙ Each thread has a set of attributes, including stack size and scheduling information ∙ In a Pthreads program, separate threads begin execution in a specified function //runner()

∙ When a program begins

⮚ A single thread of control begins in main()

⮚ main() creates a second thread that begins control in the runner() function ⮚ Both threads share the global data

**Example:**

Design a multi-threaded program that performs the summation of a non-negative integer in a  separate thread using the summation function:

For example, if N were 5, this function would represent the summation of integers from 0 to  5, which is 15.

|  |
| --- |
| #include <pthread.h> #include <stdio.h> #include <stdlib.h> int sum; //this data is shared by the thread(s) //The thread will begin control in this funtion void\* runner(void \*parameters) { int i, upper= atoi(parameters); if(upper>0) { for(i=1;i<=upper;i++) sum=sum+i; } pthread\_exit(0); }//End runner int main(int argc, char\*argv[]) { //thread identifier pthread\_t threadID; //set attributes for the thread pthread\_attr\_t attributes; //get the default attributes pthread\_attr\_init(&attributes); //create the thread pthread\_create(&threadID, &attributes, runner,   argv[1]); //now wait for the thread to exit pthread\_join(threadID, NULL); printf("sum=%d\n",sum); } |

**Question 1:**

Write a program which takes some positive integers (let’s say **N** number of positive integers)  as command line parameters, creates **N** synchronous threads, and send s the corresponding  integer as parameter to the thread function fibonacciGenerator. The function returns the  generated series to the main thread. The main thread will then print the thread number and the  series generated by that thread. The output will be like:

Thread 1: 0 1 1 2 3 5 8 13

**Example:**

If you pass as command line argument the following numbers: 3 13 34 89

Then the program will create 4 threads. The first thread will find Fibonacci terms until 3 is  generated, the second Fibonacci term will find Fibonacci terms until the term generated is 13  so on and so forth. All generated terms will be output on the screen by the main thread as  follows:

Thread 0: 0, 1, 1, 2, 3

Thread 1: 0, 1, 1, 2, 3, 5, 8, 13

Thread 2: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Thread 3: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

It is possible that the number passed to the thread is not a Fibonacci number. In this case the  thread will generate numbers until the term generated is greater than the passed number. For  example, if 7 is passed as parameter to a thread, then the thread will return the following  series:

 0, 1, 1, 2, 3, 5, 8

**Question 2:**

Write a multithreaded program that calculates various statistical values for a list of numbers.  This program will be passed a series of numbers on the command line and will then create  three separate worker threads. One thread will determine the average of the numbers, the  second will determine the maximum value, and the third will determine the minimum value.  For example, suppose your program is passed the integers. (The array of numbers must be  passed as parameter to threads, and the thread must return the calculated value to main  thread).

90 81 78 95 79 72 85

The main thread will print:

The average value is 82

The minimum value is 72

The maximum value is 95