Practice Labs 3a/3b – Threads in Linux

Operating Systems Comp Sci 3SH3 Term 2, Winter 2018

Dr. Neerja Mhaskar

Practice Labs are not mandatory, they are primarily designed to help you with your Lab tests and assignments.

Lab Format: Practice labs will be posted a day before or on the day of the lab on the course website. You can choose to solve it beforehand and come in with your solutions and check the correctness of your solution with your TA.

The TAs will also be available to answer any questions you might have on your assignments.

Solutions to practice labs will not be posted online.

Outline

Given a list of size 20 consisting of natural numbers, write a multithreaded C program for adding all the numbers in the list as follows: The list of numbers is divided into two smaller lists of equal size. Two separate threads (which we will term as ***summing threads***) add numbers in each sublist.

Because global data are shared across all threads, the easiest way to set up the data is to create a global array. Each ***summing thread*** will work on one half of this array. This lab will require passing parameters to each of the **summing threads**. In particular, it will be necessary to identify the starting index and ending index of the sublist in which each thread is to begin adding numbers. The parent thread will output the sum once all summing threads have exited.

You are to write a C program using Pthreads that contains the entire solution for this question.

In particular your program needs to do the following:

1. To be able to create threads in your C program you need to include the `pthread.h` header file.
2. Each thread has a unique thread ID. To create thread IDs for your threads your program should use the `pthread_t` data type.
3. Thread attributes should be created/modified using `pthread_attr_t` structure.
4. Declare and code the function in which the thread begins control. For an example see the `runner()` function in Fig 4.9 page 173 of the text book.

5. To be able to identify the starting index and ending index of the sublist in which each thread begins adding numbers you can do the following:

    a. Create a structure to store the starting index and ending index of the sublist. For example:

```
typedef struct
{
     int from_index;
     int to_index;
} parameters;
```

6. Since threads can share heap, you can simply create a variable of type `parameters`, allocate memory for it on the heap, and assign values to its members as follows:

```
parameters *data =
          (parameters *) malloc (sizeof(parameters));

data->from_index = 0;
data->to_index = (SIZE/2) - 1;
```

7. To create threads use `pthread_create()` function and pass in the necessary parameters.
8. For the parent thread to output the sum after all summing threads have exited, it is important that you use the `fork_join()` function.
9. Whenever you dynamically allocate memory on the heap, it is important that you deallocate/free this memory when it is not required by the program using the `free()` function.
10. To compile your program, you need to use the `-pthread` option as follows:

```
gcc -pthread -o pl3a3b pl3a3b.c
```

Sample Output for the following list:

```
List={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20}

./pl3a3b
---------------------------------------------------------------------------------------------------
Sum of numbers in the list is: 210
```