Lab Test 2 – Threads in Linux

Operating Systems CS 3SH3 Term 2, Winter 2018

Dr. Neerja Mhaskar

Thursday, February 1st, 2018

- **Submit your solution on Avenue under assessments -> assignments -> labtest2 submission folder.**
- Make sure to submit a version of your lab test ahead of time to avoid last minute uploading issues.
- Note that students copying each other's solution will get a zero. This includes (but is not limited to) sharing your solution with students taking the test during a different lab hour.
- You cannot share the lab test question with students taking the test in the next lab sections.
- Only one copy of your solution per group should be submitted.
- There is one deliverable for this lab test - labtest2.c

Outline

Write a multithreaded C program using the Pthreads API that takes four command line arguments, each an integer value < 100. Your main program/thread should create three separate threads (Thread 1, Thread 2 and Thread 3) each performing the below outlined task:

1. Thread1 – Take the first two command line arguments passed as parameters to the function called **`doubles`** in which it begins its execution. The **`doubles`** function, simply adds these values and then doubles this sum, and returns this value to the main thread. For example: if the first two command line arguments are 10 and 20, **`doubles`** returns (10+20)*2=60 to the main thread.

2. Thread2 – Takes the last two command line arguments passed as parameters to the function called **`triples`** in which it begins its execution. The **`triples`** function, simply adds these values and then triples this sum, and returns this value to the main thread. For example: if the last two command line arguments are 30 and 40, **`triples`** returns (30+40)*3=210 to the main thread.

3. Thread 3 – Takes the values returned by Thread1 and Thread2 in main as parameters to the function called **`sum`** in which it begins its execution. The **`sum`** function simply adds these values and returns it to the main thread. For example, in the example shown above, **`sum`** returns 60 + 210 = 270 to the main thread.

4. The main thread simply prints the value returned by Thread 3 on the console. For example, in the example shown above, it simply prints 270.

Important:

1. You need to create threads using the `pthread_Create()` function. While creating these threads you need to pass the thread identifier, the attributes for the thread, the function in which the thread starts its execution, and the parameters to this function.
    a. To pass these parameters, you need to create a structure called **`parameters`** that holds this data.
    b. Since threads can share heap, you can simply create an instance of this structure and allocate memory for it on the heap and pass it to thread.
    c. The threads cannot use global variables to return data back to the main thread. Hint: You can simply add a variable to the structure **`parameters`** to hold the return value.
2. Any memory allocated on the heap should be deallocated before exiting the program.

Sample Output:`./labtest2 10 20 30 40`

```
triples returns: 210
doubles returns: 60
sum returns: 270
Value returned by Thread3 in main: 270
```