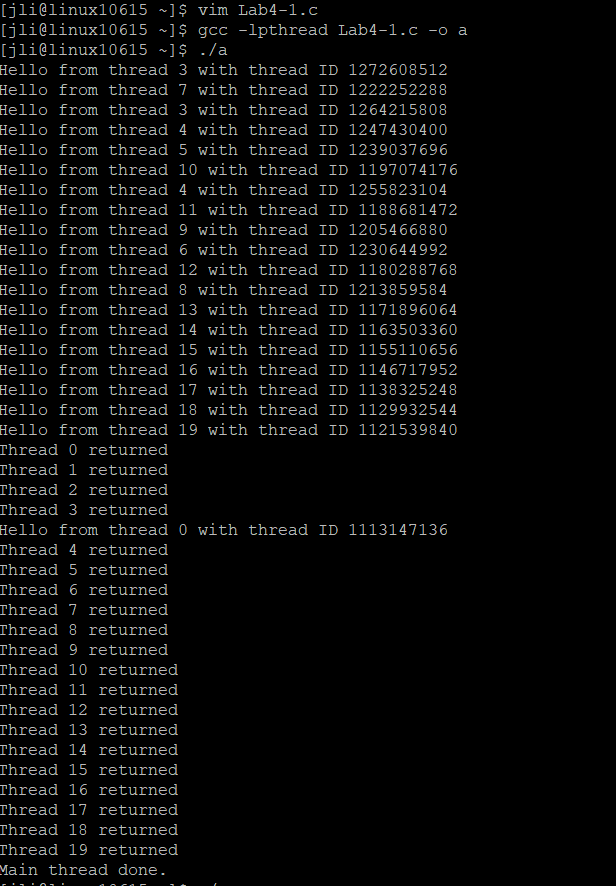
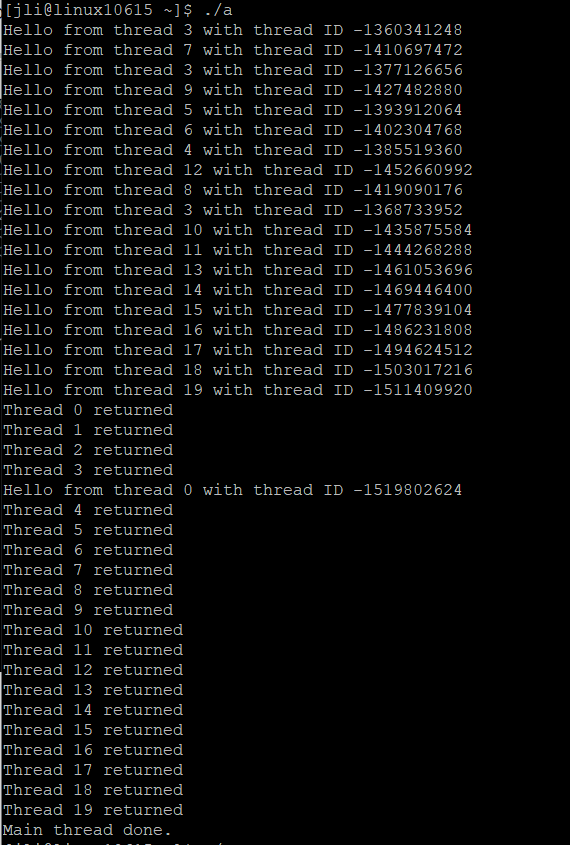
Justin Li

2/4/21

# Lab 4 - Developing multi-threaded applications

Step 1:





Explain what happens when you run the threadHello.c program? Do you get the same result if you run it multiple times? What if you are also running some other demanding processes (e.g., compiling a big program, playing a Flash game on a website, or watching streaming video) when you run this program?

When I ran the threadHello.c program, the output I received was a list of “Hello from thread…” that appeared in seemingly random order with some duplicate threads, followed by “Thread # returned” in numerical order and no duplicates. A trend that I noticed was that the thread IDs decreased as the thread # increased for all cases except for thread 0. And another strange thing that I noticed was that “Hello from thread 0” printed after thread 3 returned. I believe this happened because running another demanding process, aka turning a streaming video in the background, causing the CPU to lag and print some things out of order.

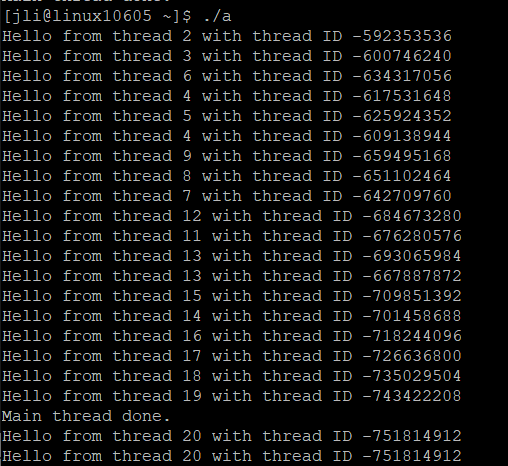
The function go() has the parameter arg passed a local variable. Are these variables per-thread or shared state? Where does the compiler store these variables’ states?

Variable arg is per-thread. The compiler stores these variable’s states in the thread stack.

The main() has local variable i. Is this variable per-thread or shared state? Where does the compiler store this variable?

Variable i is global and shared state. The compiler stores this variable in the process stack.

Step 2:

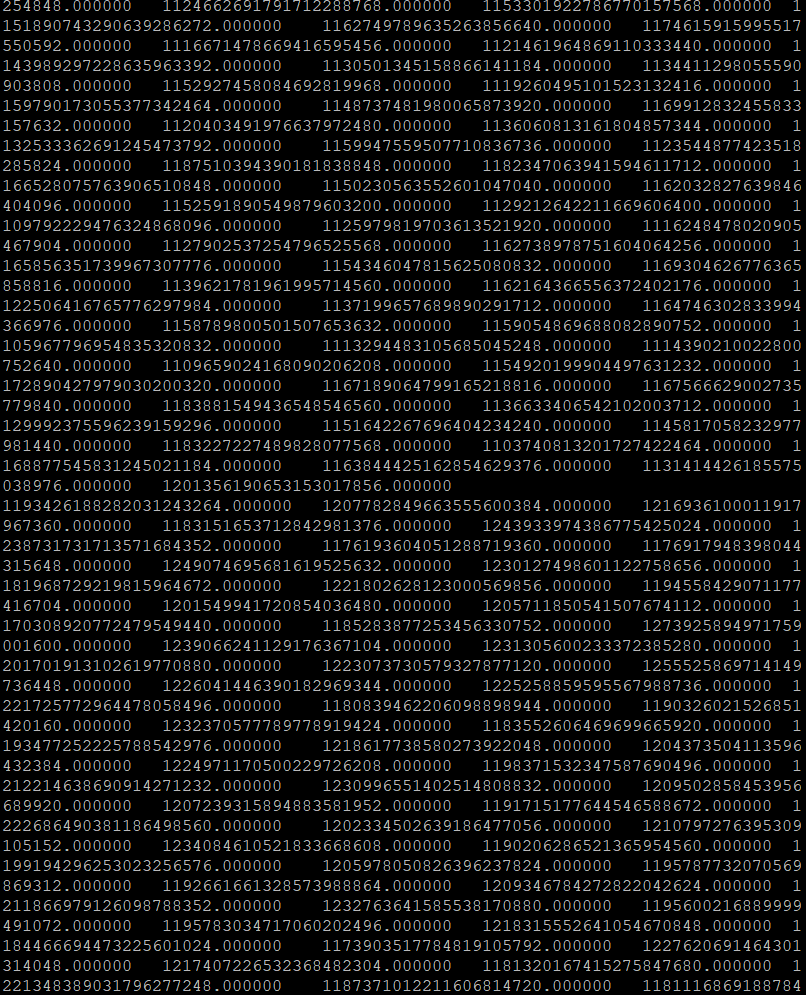


Delete the second for loop in threadHello.c program so that the main routine simply creates NTHREADS threads and then prints “Main thread done.” What are the possible outputs of the program now. Explain.

One main observation that I notice is that sometimes, as in the picture above, “Main thread done.” sometimes prints in the middle of the thread “Hello”s. This happens because in the code, the main print occurs after the threads are created, but when the CPU runs many things in the background, the processes can lag and sometimes print out of order, causing this occurrence. One other small difference is that I notice from before is that thread 20 is now printed. Every other observation seems to be similar as before.

Step 3:

1024 x 1024:



2 x 2:

