Tyler Paquet

Anthony Dowling

CIS 452

Due: 2/9/2017

**Lab 4: Introduction to pthreads and Multi-threaded Programming**

1. **describe your results (which may not be what you expected)**

Nothing happened.

1. **again, describe your results and explain your observations**

After inserting the sleep into the program, it allows the do\_function to print its results. This is because there is a 1 second sleep in the do\_funtion, and without the 2 second sleep in the main, the do\_function was unable to actually run before the main exited.

**3. report your results. Is the output of the program what you would have expected? Describe what causes the observed formatting**

We got a printout of “World” 20 times, ten per thread. This is due to the random number generator creating a number modded by 2 and if that does not = 0, “Hello” is printed, if it is not 0, “World” is printed.

**4. report your results. Is the output of the program what you would have expected? Describe what causes the observed formatting**

The first output was all “Hello” and they all printed at once, the second time we ran the program, the output was “Hello World” repeatedly and they printed line by line. This is because if the buffer does not see a new line character, it prints the output all at once, whereas a new line character will break up the output and print it line by line. Since “Hello” was not followed by a new line character, and “World” was, when “World” printed it would print line by line rather than all at once.

**5. based on your observations: does pthreads under Linux use the 1-to-1 or the many-to-many thread mapping model? Justify your answer**

Linux uses a 1-to-1 thread mapping model because for every one user thread that is created, it is mapped to one kernel thread.

**6. compile the sample program and run it multiple times (you may see some variation between runs). Choose one particular sample run. Describe, trace, and explain the output of the program**

The output of the program starts with the parent initially seeing the number 5, which is what the variable “sharedData” was initialized to. Prior to this in the program the threads are created and they also see 5 initially, but they then sleep for 1 second, giving the parent enough time to print off and then increment sharedData. After this, the children print off what they initially saw, which was 5. After this the sharedData is incremented and the children print off what they see, the first child that received “a” sees 7, the second child “b” sees 8, since there is a small delay between the two children’s processes. The last thing printed is the parent’s final look at the sharedData which is 8.

**7. explain in your own words how the thread-specific (not shared) data is communicated to the child threads**

The non-shared data of the threads, that being the status, is communicated through the join function. After the function runs, the “status” data is communicated back giving a 0 on a successful run, any other return signifies an error.