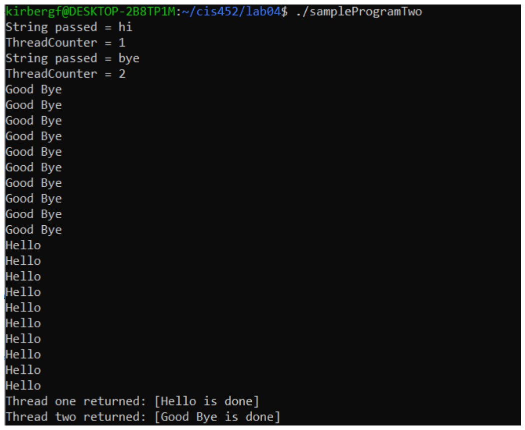
Sebastian Torrejon Alonzo
Fabian Kirberg

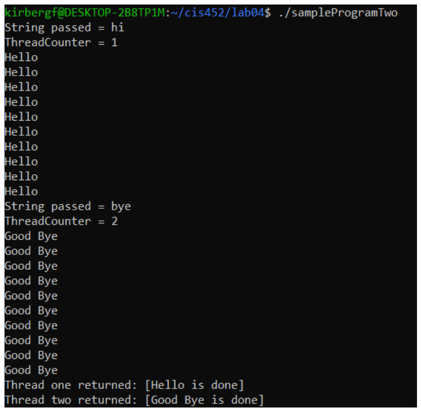
Lab 4

1. **Describe/explain your observations, i.e., what must have happened in the original, unmodified program?**
The original unmodified program only output the line “Running the program”. Once the sleep(2) line was added, the program output an additional “Thread version of Hello, world.” line. This is because the original, unmodified program immediately exited after creating its thread and as a result the doGreeting() thread did not have enough time to print its message.

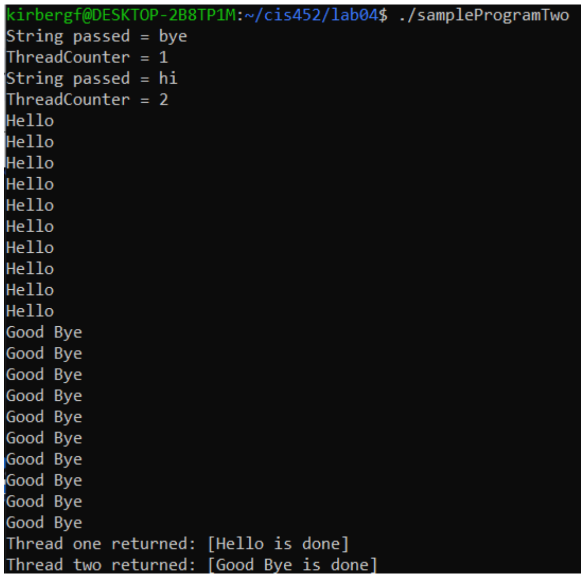
2. **What does sampleProgramTwo output? If you run it a repeated number of times does the output vary? Why?**



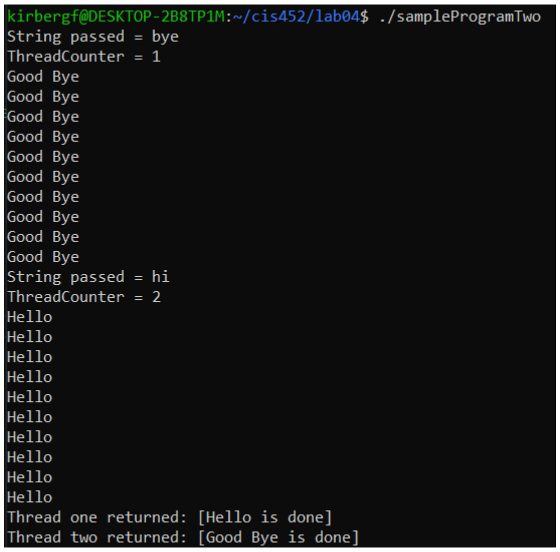
```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab04$ ./sampleProgramTwo
String passed = hi
ThreadCounter = 1
String passed = bye
ThreadCounter = 2
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Thread one returned: [Hello is done]
Thread two returned: [Good Bye is done]
```



```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab04$ ./sampleProgramTwo
String passed = hi
ThreadCounter = 1
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
String passed = bye
ThreadCounter = 2
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Thread one returned: [Hello is done]
Thread two returned: [Good Bye is done]
```



```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab04$ ./sampleProgramTwo
String passed = bye
ThreadCounter = 1
String passed = hi
ThreadCounter = 2
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Thread one returned: [Hello is done]
Thread two returned: [Good Bye is done]
```

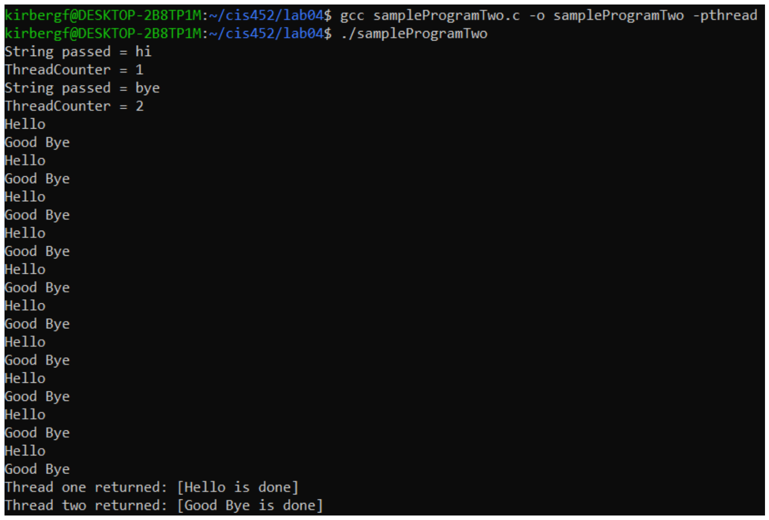


```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab04$ ./sampleProgramTwo
String passed = bye
ThreadCounter = 1
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
String passed = hi
ThreadCounter = 2
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Thread one returned: [Hello is done]
Thread two returned: [Good Bye is done]
```

Yes, the output can vary in a few different ways as shown in the above screenshots. This is because both threads are created at the same time and don’t wait for one or the other to finish running before they process, which can result in the “hi” thread running first in one instance of the program, but potentially second in another.

Sebastian Torrejon Alonzo
Fabian Kirberg

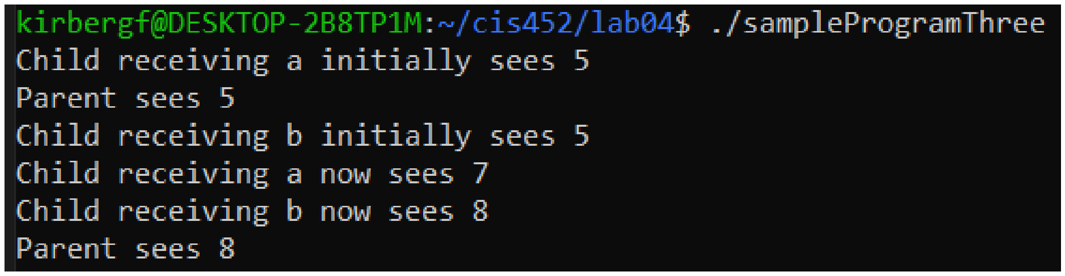
3. **Report your results again. Explain why they are different from the results seen in question 2.**



```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab04$ gcc sampleProgramTwo.c -o sampleProgramTwo -pthread
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab04$ ./sampleProgramTwo
String passed = hi
ThreadCounter = 1
String passed = bye
ThreadCounter = 2
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Thread one returned: [Hello is done]
Thread two returned: [Good Bye is done]
```

The results now differ because the sleep(1) line now essentially causes the two threads to leap frog. After adding sleep(1), We see a more predictable pattern where each thread will print a message, pause for 1 second, and then print the next message instead of the chaotic output from question 2.

4. **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.**



```
kirbergf@DESKTOP-2B8TP1M:~/cis452/lab04$ ./sampleProgramThree
Child receiving a initially sees 5
Parent sees 5
Child receiving b initially sees 5
Child receiving a now sees 7
Child receiving b now sees 8
Parent sees 8
```

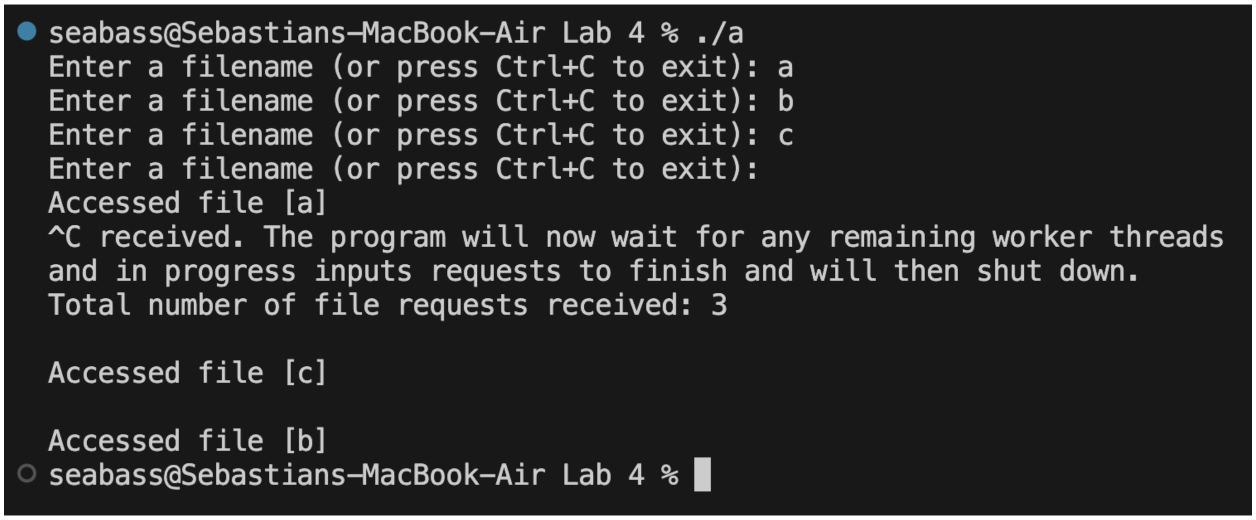
- The parent creates 2 threads. And they all run simultaneously.
- In this output case, thread1 prints "Child receiving a initially sees 5", and then parent prints “Parent sees 5” and then, thread2 prints "Child receiving b initially sees 5". At this point, all threads see sharedData as 5.
- The parent thread increments sharedData to 6 and then waits for both child threads to finish.
- While the parent was executing its code, both child threads sleep for 1 second, allowing the parent to do its execution.
- After the sleep, both child threads increment sharedData by 1 and print the new value of sharedData.

Sebastian Torrejon Alonzo
Fabian Kirberg

- In this output case. Thread1 happens first and makes sharedData 7 and prints “Child receiving a now sees 7”, and then, thread2 increments it one more time making it 8 and printing “Child receiving b now sees 8”.
- Finally, the parent thread prints "Parent sees 8", after both child executions were exited.

5. **Explain in your own words how the thread-specific (not shared) data is communicated to the child threads.**
   Thread-specific data is communicated to the child threads by passing that data as an argument when creating the thread. For example, in sampleProgramThree the line “`threadStatus1 = pthread_create (&thread1, NULL, doGreeting, &sampleArray[0]);`“ creates a thread and passes the address of the respective elements of the sampleArray. Each child thread then accesses its specific data using the myPtr pointer in the doGreeting function.

Lab Programming Assignment (Blocking Multi-threaded Server) Screenshot



```
seabass@Sebastians-MacBook-Air Lab 4 % ./a
Enter a filename (or press Ctrl+C to exit): a
Enter a filename (or press Ctrl+C to exit): b
Enter a filename (or press Ctrl+C to exit): c
Enter a filename (or press Ctrl+C to exit):
Accessed file [a]
^C received. The program will now wait for any remaining worker threads
and in progress inputs requests to finish and will then shut down.
Total number of file requests received: 3

Accessed file [c]

Accessed file [b]
seabass@Sebastians-MacBook-Air Lab 4 %
```