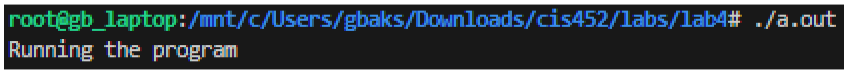
CIS 452 02 – Lab 4 – Gabrielle Munson and Gabe Baksa

Sample Program 1

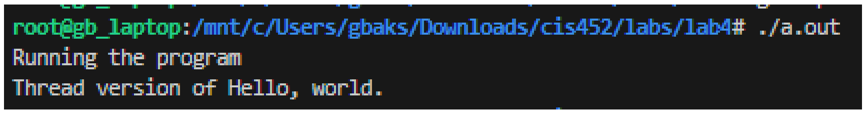
1. **Describe/explain your observations, i.e., what must have happened in the original, unmodified program?**

   Original:



```
root@gb_laptop:/mnt/c/Users/gbaks/Downloads/cis452/labs/lab4# ./a.out
Running the program
```

   Modified:

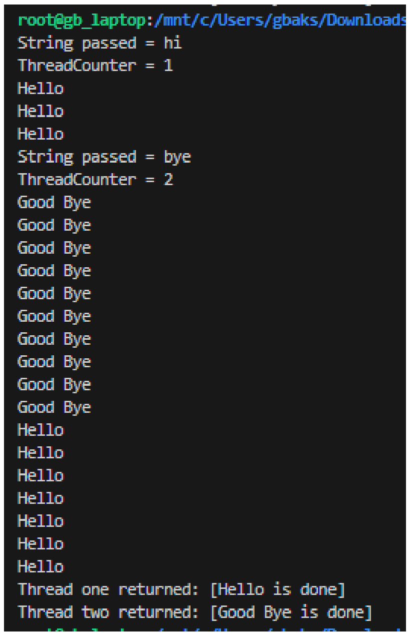


```
root@gb_laptop:/mnt/c/Users/gbaks/Downloads/cis452/labs/lab4# ./a.out
Running the program
Thread version of Hello, world.
```

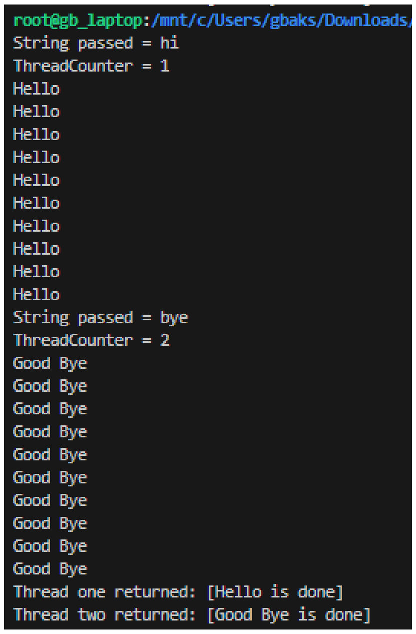
   In the original, the main thread terminates, so the child thread is also terminated because the entire process ends. So the child did not get a chance to run the printf command.

Sample Program 2

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



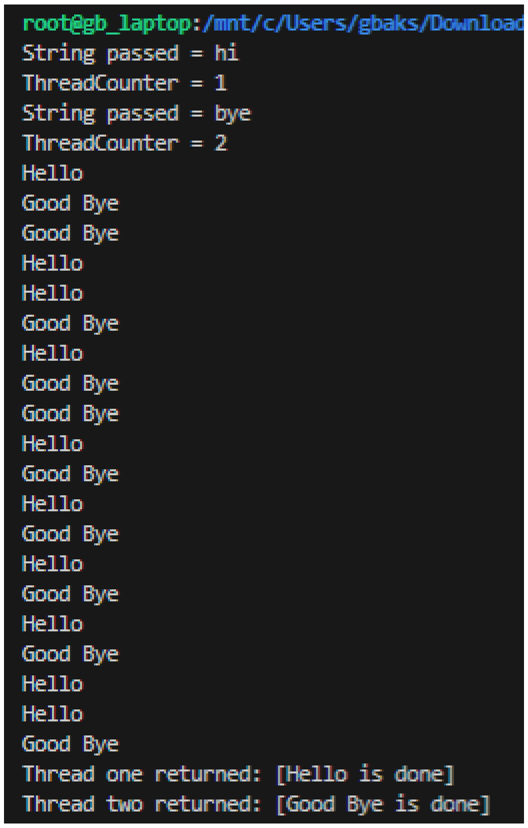
```
root@gb_laptop:/mnt/c/Users/gbaks/Downloads
String passed = hi
ThreadCounter = 1
Hello
Hello
Hello
String passed = bye
ThreadCounter = 2
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Good Bye
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Thread one returned: [Hello is done]
Thread two returned: [Good Bye is done]
```



```
root@gb_laptop:/mnt/c/Users/gbaks/Downloads/
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]
```

Yes, the output varies when the program is run a repeated number of times because at first the threads are running concurrently in the doGreeting functions. This results in a print statement from whatever thread reaches the CPU first. After the threads are created, they are then joined, thread one first and thread two second. With thread one being joined first, it has to wait for the thread to return from the function before thread two can be joined. This results in the print statements of “Thread one returned: []” and “Thread two returned: []” to be printed in that order when the functions return.

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

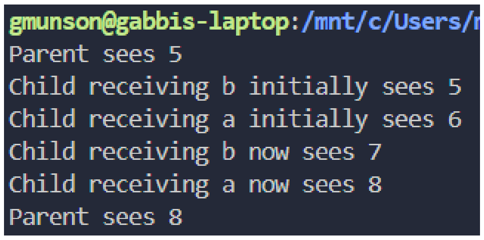


```
root@gb_laptop:/mnt/c/Users/gbaks/Download
String passed = hi
ThreadCounter = 1
String passed = bye
ThreadCounter = 2
Hello
Good Bye
Good Bye
Hello
Hello
Good Bye
Hello
Good Bye
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Good Bye
Hello
Hello
Good Bye
Thread one returned: [Hello is done]
Thread two returned: [Good Bye is done]
```

By adding a sleep at the beginning of the for loop, both threads are delayed before printing, which reduces the speed difference between them. This makes their output more synchronized.

Sample Program 3

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.**

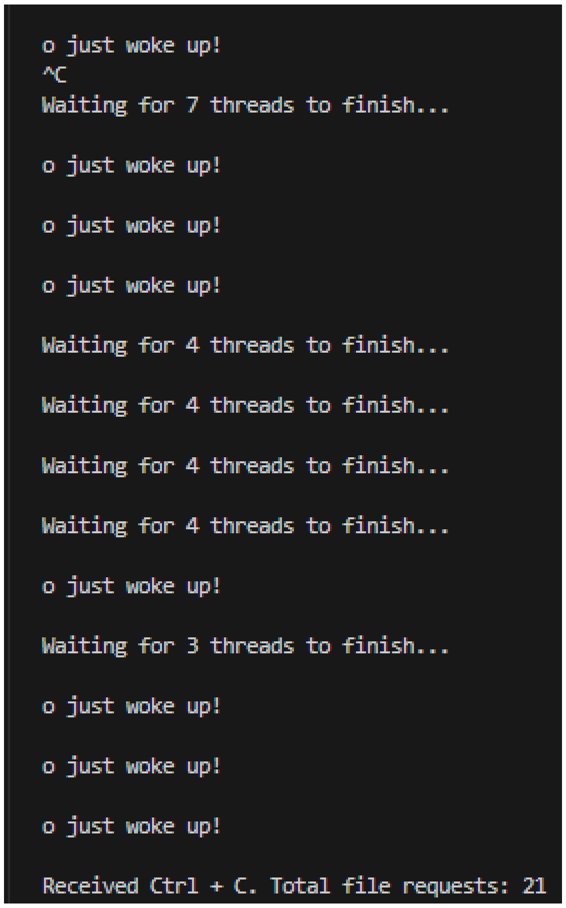


```
gmunson@gabbis-laptop:/mnt/c/Users/r
Parent sees 5
Child receiving b initially sees 5
Child receiving a initially sees 6
Child receiving b now sees 7
Child receiving a now sees 8
Parent sees 8
```

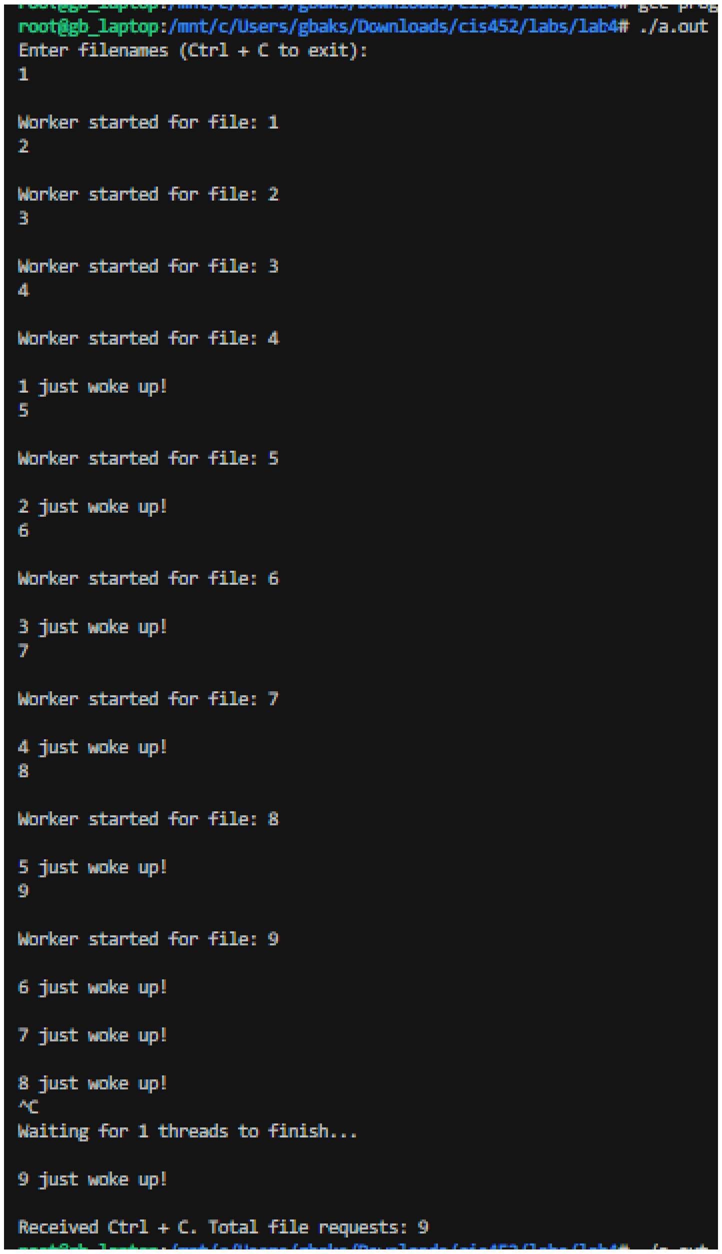
The main thread starts executing first, and creates two other threads. In this example, the Parent accesses the shared data first and sees 5. Then the program context switches to child receiving b which accesses the shared data also seeing 5. The main thread (or one of the threads in the doGreeting() function, unable to tell) then adds 1 to the shared data value. Then, child receiving a is switched to and it accesses the shared data to see 6 now. Shared data is then incremented by 1 again. The program switches back to child receiving b which then accesses the data and sees 7. Shared data is then incremented again, by either the main thread or thread1, unable to tell. The program then switches back to child receiving a which accesses the shared data and sees 8. Once the 2 created threads are returned from their functions, the main thread runs again for the parent to access the shared data, seeing 8 as well. The program then exits.

5. **Explain in your own words how the thread-specific (not shared) data is communicated to the child threads.**
   The address of an element in sampleArray is passed to each child thread as an argument to pthread_create. Each thread receives a pointer to its specific element, allowing it to work with thread-specific data ('a' or 'b') independently.

Programming Assignment



```
o just woke up!
^C
Waiting for 7 threads to finish...

o just woke up!

o just woke up!

o just woke up!

Waiting for 4 threads to finish...

Waiting for 4 threads to finish...

Waiting for 4 threads to finish...

Waiting for 4 threads to finish...

o just woke up!

Waiting for 3 threads to finish...

o just woke up!

o just woke up!

o just woke up!

Received Ctrl + C. Total file requests: 21
```



```
root@gb_laptop:/mnt/c/Users/gbaks/Downloads/cis452/labs/lab4# ./a.out
Enter filenames (Ctrl + C to exit):
1

Worker started for file: 1
2

Worker started for file: 2
3

Worker started for file: 3
4

Worker started for file: 4

1 just woke up!
5

Worker started for file: 5

2 just woke up!
6

Worker started for file: 6

3 just woke up!
7

Worker started for file: 7

4 just woke up!
8

Worker started for file: 8

5 just woke up!
9

Worker started for file: 9

6 just woke up!

7 just woke up!

8 just woke up!
^C
Waiting for 1 threads to finish...

9 just woke up!

Received Ctrl + C. Total file requests: 9
```