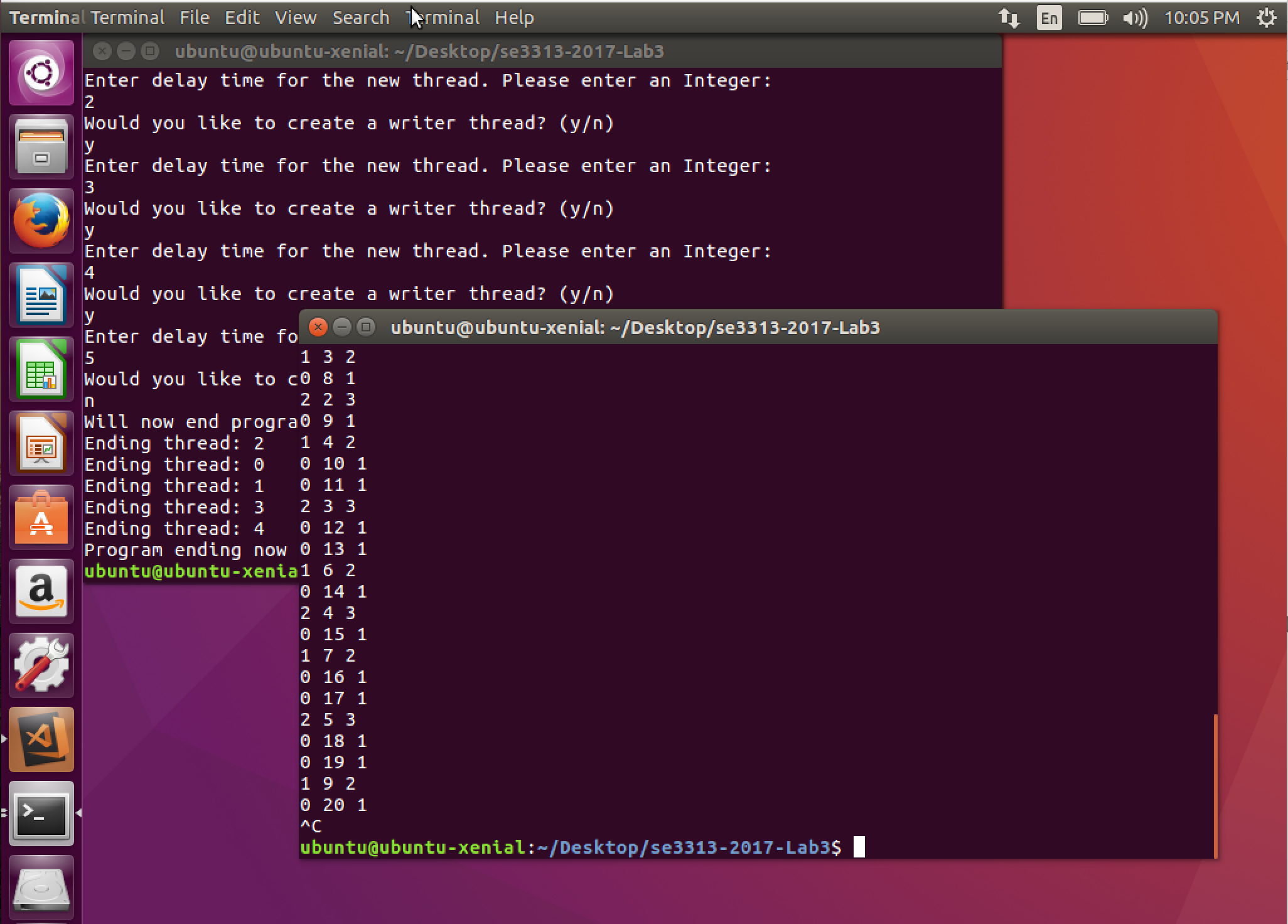
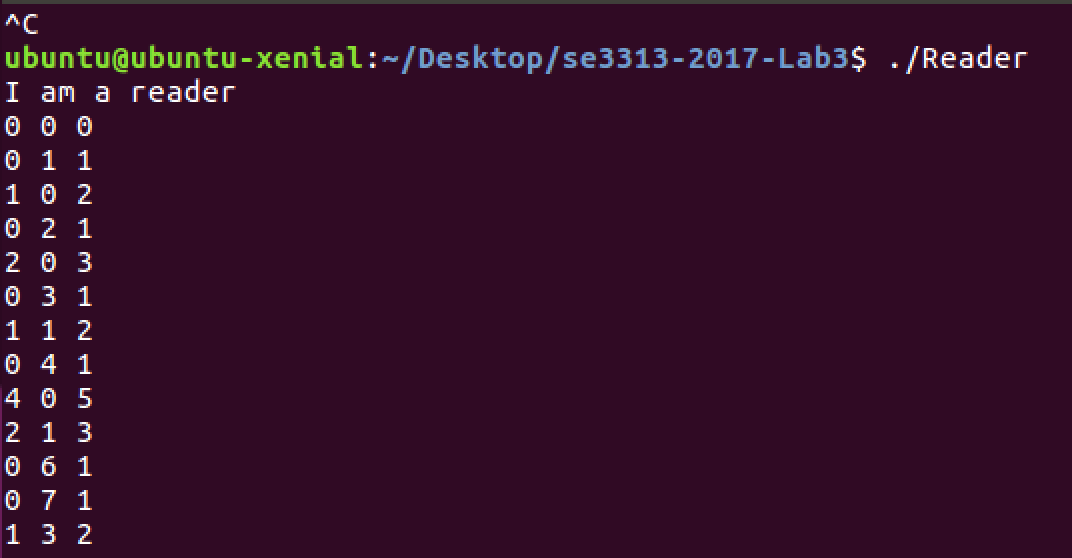
**LAB 3**

**Question 1:** Can you find evidence of a thread report getting obliterated? That is to say, some thread should have reported and you can’t find its report? For example, thread #3’s report #4 is just gone? What is causing this? (In this case, I expect you will certainly see such events.)

*Answer:* Yes, this is because of the delay times. Let’s say Thread A has a smaller delay time then Thread B, so because there is no semaphore, there is nothing stopping thread A from accessing and overwriting thread B resources before thread B even gets a chance to finish its work. Thread management isn’t done correctly. There is no synchronizing between the threads, so no insurance is provided for all threads opening or closing correctly. 

**Question 2:** Can you find evidence of a thread’s report getting corrupted? In other words, the output from Reader is clearly partially from one thread and partially from another? What is causing this? (If you can’t find such a case, and you might not be able to, since it depends mostly on luck, hypothesize about what would cause it, and what the corrupt output would look like.)

*Answer:* There is no management or protected access for the critical section of the threads. Corrupt data would be data that is partially written to by BOTH of the threads.



**Question 3:** Explain your semaphore design. How are the semaphores being used? What is their function? How do they affect the various threads and processes?

*Answer:* The semaphore are used to block or unblock access to the shared memory. This functionality is shared between threads in a processes, providing synchronizing of the threads. While a thread lets say A is running, it is blocked and no other thread can access the data while thread A is running. One thread A is finished, it’ll let the other threads know it is done by assigning unblocking call to the semaphore. Because the threads are synchronized now, it allows for mutual exclusion of the threads. The blocking function will block or it’ll return right away if the value of the semaphore was 1 or higher. Unblocking the semaphore simply increases the semaphores value and unblocks the next waiting thread. In our program we make two semaphores, one for protecting access to the data and another for communicating with the reader. The second semaphore blocks when it’s waiting for data to be read, and it signals where there is data to be read. So when thread one is running the first semaphore blocks so no other threads can access into the critical section while that thread is modifying the data, then once its done modifying, it signals the first semaphore and signals the second semaphore too because now that file is done writing and it can be read. Thread is written as a loop, and can be terminated externally. Basically it has a Boolean variable called running and checks that variable each time it enters the loop.

**Question 4:** Do you still see any corrupt reports? Could you see any corrupt reports? Explain.

*Answer:* No, Unless you have more then around 10000’s threads running at once which MIGHT cause corruption of data because of the enormous amount of thread and resource management the processor has to do. Corruption of data is unlikely because a semaphore sole purpose is to prevent and guard the critical section while data is being modified.

**Question 5:** Do you still see any missing reports? Could you see any missing reports? Explain. (HINT: You may wish to consider worst cases here.)

*Answer:* No, because we have a semaphore that notifies the reader MOST of the time there is a file to be read. In worst case if we have an immense amount of threads that get ready to be read back to back, the reader might miss a notification for one of the threads being ready to read, and since the queue size is of size 1, the data will be lost.

**Discussion:** Finally, as always I like to invite you to discuss the experience of the lab. Was it worthwhile? Did you learn anything? Can you see ways to apply what you have learned? Was it too easy? Too difficult?

*Answer:* At first it was overwhelming but once I learned what each semaphore was for and how the functions of the semaphores are used(Wait(), and Signal()), it started to all make sense. The difficulty level of this lab was fair other then learning how the shared memory objects work and how I can call and access them. The provided lab documents really helped in understanding this concept and even the overall lab. In conclusion, this is lab was overall a learning experience.