WENTWORTH INSTITUTE OF TECHNOLOGY

College of Engineering and Technology
Department of Electrical Engineering and Technology

Operating Systems
Spring 2018

Lab 3 Addendum & Notes

In part II:

Insert a random wait time (from 0 to 1 second) between the creation of each element in the Fibonacci sequence. This is mostly to enable proper testing of your code and to ensure you have synchronized the two processes properly.

Hints:

- To insert a random wait time, use `usleep(1000 * (rand()%1000) )`. This allows you to wait from 0 to 1 second, in increments in 1 mS. (`usleep` specifies the number of microseconds you sleep for). The insertion of this random wait exposes any synchronization issues (if any) and thus allows you to test the correctness of your code.
- One way (not the only way) to synchronize the two processes, without using OS synchronization calls, is to follow the example on lecture 4C, slides 11 and 12 (final 2 slides).
- You will thus need to share not only the buffer storing the Fibonacci sequence, but also two additional variables, the read index and the write index (each could be an integer, whose size is sizeof(int) and is 4 bytes for your Linux platform).
- Use the following command as an example of how to compile your code:

  ```
  gcc -lrt lab3.c -o lab3
  ```

  This links the librt library with your program. Use the following to invoke your program and pass it a value of 7 for example:

  ```
  lab3 7
  ```
- A pointer is nothing but an address. To access a variable using a pointer you must dereference the pointer, for example, if `pReadIdx` is a pointer to the `ReadIdx` variable, and you want to clear `RdIdx`, then use:

  ```
  *pReadIdx = 0;
  ```

  If you want to increment it, use:

  ```
  (*pReadIdx)++;
  ```