COP4610 / CGS5765 Operating Systems Project 4
Producer-Consumer Problem (100pts)

Important Notes

1. This is a **single-person** project. Do your own work!
2. The project is based on the Linux operating system, not the Xv6.
3. Make sure your program works on `linprog.cs.fsu.edu` because that is where it will be graded.

Programming Tasks

In this project, you will learn how to use semaphores and mutex to implement the producer-consumer problem as specified by **Chapter 5, Project 3, Producer-Consumer Problem, Page 253** with the following additional requirements:

1. Solve the problem using **Pthreads**, not Win32 API. You may find a good introduction to **Pthreads** programming at `https://computing.llnl.gov/tutorials/pthreads/`
2. Properly handle major error conditions.
3. Make sure you print messages related to produced and consumed items, as specified in the book:

   ```
   printf("producer produced %d\n", item);
   printf("consumer consumed %d\n", item);
   ```

   Do not print other messages in the producing/consuming loop.
4. In the specification, the producer and the consumer will keep executing (**while (TRUE) ...**). To stop the execution, you can type **CTRL-c** on the console.**CTRL-c** is delivered as a signal to the program. Normally, this will cause an asynchronous cancellation of the threads. In this project, you will learn how to implement the deferred cancellation. Specifically, you will install a handler for the **SIGINT** signal. In this handler, you can set a flag to tell the producer and consumer threads to terminate itself. In the loop of the producer and the consumer, they will test the flag, and exit the loop if the flag is set (make sure the lock and the semaphores are in correct states before exiting). The main program will exit when both the producer and the consumer have terminated (use **pthread_join**).

   You can find some information here:

   `http://www.thegeekstuff.com/2012/03/linux-signals-fundamentals/`

   `http://www.thegeekstuff.com/2012/03/catch-signals-sample-c-code/`
5. Provide a write-up to briefly explain your code, and provide an example output of your program, and briefly describe the output.