**CSE 325** **Spring 2021**

# Computer Project #6

**Assignment Overview**

This assignment focuses on multi-threaded programming in a Linux environment, and is the final step in a two-part project. You will design and implement the C++ program which simulates a simple producer-consumer system, as described below.

It is worth 50 points (5% of course grade) and must be completed no later than 11:59 PM on Thursday, 2/25.

**Assignment Deliverables**

The deliverables for this assignment are the following files:

**proj06.makefile** – the makefile which produces **proj06**
**proj06.student.c**pp – the source code file for your solution

Be sure to use the specified file names and submit them for grading via the CSE Handin system before the project deadline.

**Assignment Specifications**

A wholesale tree company maintains an inventory of tree seedlings and sells them to its customers (greenhouses, garden centers, etc). The customers can place purchase orders using several different methods (website, toll-free phone number, local phone number, and so on) between the hours of 8:00 AM and 6:00 PM.

The simulation for this system will use concurrent execution of N producer threads (where N is between 1 and 9, inclusive) and a single consumer thread to process a series of customer orders using a bounded buffer. Each producer thread handles the orders placed using a specific method (perhaps producer #1 handles orders placed on a website, producer #2 handles orders placed on a toll-free phone number, producer #3 handles orders placed on a local phone number, and so on).

a) The file named "inventory.old" contains the inventory at the start of the day.

That file will contain zero or more lines, where each line contains four fields: the product ID number (field width of 6, unsigned integer), the price per seedling (field width of 5, dollars and cents), the quantity on hand (field width of 5, unsigned integer), and the product description (up to 30 characters). There is one space between fields. For example:

```
100492  2.50   360 Northern Red Oak
201005 10.17    62 Shagbark Hickory
100305  1.95  1043 Sugar Maple
100491  2.50   803 White Oak
```

b) The file named "ordersN" contains the set of all purchase orders to be processed by producer #N, in temporal order. For example, if there are four producer threads, the files will be named "orders1", "orders2", "orders3", and "orders4".

Each file will contain zero or more lines, where each line contains three fields: the customer ID number (field width of 7, unsigned integer), the product ID number being ordered (field width of 6, unsigned integer), and the number being ordered (field width of 5, unsigned integer). There is one space between fields. For example:

```
0003183 100305   100
9981532 100492    25
0003183 201005     5
0050600 100305    60
```

c) The file named "inventory.new" contains the inventory at the end of the day. That file will have the same format as "inventory.old" (the inventory at the start of the day).

1. The number of producers and the size of the bounded buffer will be available to the program as command-line arguments. Valid executions of the program might appear as:

```
proj06 -p 3 -b 15
proj06 -b 5 -p 4
proj06 -b 20
```

The number of producers will not exceed 9 and will default to 1. The bounded buffer will be circular and will consist of R records, where R will not exceed 30 and will default to 10.

2. The program will input the contents of "inventory.old" and build a data structure representing the current inventory.

After building the current inventory, the program will create the set of producer threads and the consumer thread, all of which will execute concurrently.

After all producer threads and the consumer thread halt, the program will create "inventory.new" and output the current inventory into it.

3. Each producer thread will process the contents of "ordersN" (where N is the thread number). Each producer thread will repeatedly read one order from the appropriate input file and will then insert one record representing that order into the bounded buffer. After all of the orders in the appropriate input file have been processed, that producer thread will insert a special record into the bounded buffer to indicate that it is finished processing orders and will then halt.

4. The consumer thread will create the output file named "log", and will then repeatedly extract one order record from the bounded buffer and process it. After all of the producer threads have halted and the bounded buffer is empty, the consumer thread will halt.

The consumer thread will validate each purchase request. If there are enough seedlings on hand to fill the order, the current inventory will be updated. If the order cannot be filled for any reason, it will be rejected.

The consumer thread will track the results of processing each purchase order by sending one line to the "log" output file. Each line will be no more than 80 characters in length and will contain:

a) customer ID number
b) product ID number
c) product description
d) number ordered
e) transaction amount (number ordered x price per seedling)
f) result (filled or rejected)

The log entries will be appropriately formatted: items will be aligned in columns, and monetary values will be displayed as dollars and cents (for example, $50.00).

5. The program will include appropriate logic to handle exceptional cases and errors.

**Assignment Notes**

1. As stated above, your source code file will be named "proj06.student.cpp" and you must use "g++" to translate your source code file in the CSE Linux environment.

2. You must use the POSIX threads library for this assignment. Information about library functions which might be useful for this project may be viewed using the "man" utility. For example:

```
man 3 pthread_create                    man 3 sem_init
man 3 pthread_join                      man 3 sem_wait
man 3 pthread_exit                      man 3 sem_post
```

3. The producer threads and the consumer thread must execute simultaneously.

4. You will model the bounded buffer using an array and you will model one order in the bounded buffer using a record ("struct" or "class"). You may model the current inventory using any data structure (including data structures from the STL).

5. Each critical section will be kept as small as possible and no I/O operations will be performed inside a critical section.

6. You may assume that the lines in the input file named "inventory.old" (if it exists) are formatted correctly and contain valid information.

7. You may assume that the lines in the input file named "ordersN" (if it exists) are formatted correctly.

8. You may assume that all of the input and output files are in the directory where your program begins execution.

9. Examples of the input files are available in the "/user/cse325/Projects" directory. Note that those examples are intended to illustrate the format of the files and are inadequate to serve as non-trivial test cases. It will be necessary for you to develop a series of input files to test your solution.