MULTI-THREADED PRODUCER-CONSUMER

AN EVALUATION OF C++ AND DESIGN ABILITIES

Thomson Reuters Collections

PROBLEM DESCRIPTION

You will be writing a program which will use threads to receive and process data. Your design may use one or more threads to handle each part of the program with a minimum of two threads (one to receive input data and one to process that data).

It is anticipated that this test will take no more than three hours for a basic implementation. You may spend as much time as you wish to provide a submission you find satisfactory but our expectations are that this will not be a long task.

ENVIRONMENT

Use the Linux server referenced in the email accompanying this test to develop your solution. It provides g++ and the Boost libraries for your use.

If you prefer to develop the program on your own machine and upload the finished code to the server, please make sure it builds and runs on the server before calling it complete.

THE PRODUCER

Your program will read data from a CSV file using a producer thread to send it to the consumer thread.

INPUT DATA FORMAT

The input data is formatted as an ASCII encoded Simple CSV file with 4 fields separated by commas. None of the fields will contain commas and there will be no quotation marks around any of the fields. Multiple test files will be used in the evaluation of your applications. The files will vary in the number of rows and the number of symbols and length of symbols. The file is laid out and described in Table 1.

Table 1 -- Input Data Field Formats

COLUMN	FIELD	MEANING
1	Sequence Number	This is a 32-bit integer monotonically increasing by 1, expressed in Hexadecimal ASCII text.
2	Symbol	This is a character string name representing a financial instrument stored as ASCII text.
3	Price	This is a decimal price expressed in ASCII with 2 digits after the decimal point.
4	Quantity	This is a number of shares written as a positive integer. Its value will never be larger than 32-bits.

DATA FORMAT

Your producer thread will send the data it reads from the input file to the consumer thread via any mechanism or data structure you choose. You may use whatever format you would like for providing the data to the consumer.

USAGE

Your program will take one command line argument, the Input Data File. Upon execution, it should read the Input Data File, send each of the rows of data to the consumer. Please display the number of rows read from the file before exiting. You may display whatever other data you would like.

THE CONSUMER

Your consumer will receive the data from the producer and will write the data to a separate file for each symbol.

OUTPUT DATA FORMAT

For each symbol in the input data, you will output a separate file formatted as simple CSV. Each file should be named *<Symbol>.csv* and stored in the current working directory. Each file should contain the rows for the symbol sorted from lowest price to highest price. For duplicate prices, order the rows by sequence number in decreasing order.

Table 2 -- Output Data Field Formats

COLUMN	FIELD	MEANING
1	Price	This is a decimal price expressed in ASCII with 2 digits after the decimal point.
2	Quantity	This is a number of shared written as a positive integer. Its value will never be larger than 32-bits.
3	Sequence Number	This is the sequence number from the input data, expressed in decimal format.

USAGE

The consumer thread should write each of the Output Data Files. Please display the number of rows received for each symbol before exiting. You may display whatever other data you would like. After all of the input data has been processed and the output files have been written, the program will exit.

Warning We will test your applications with multiple data sets. Do not depend on a specific value or number of rows in the data file.

DELIVERABLES

Please provide the items listed in Table 3 as the results of your implementation. These files should be stored in a zip file archive. You should not include any executables in your archive, as we will be compiling your applications locally.

Table 3 -- Output Data Field Formats

DELIVERABLE	MEANING
Source files	We will rebuild your program from source during the evaluation. Please include a makefile or build script.
Design Summary	This is a freeform document describing your program. It should describe the structure of each of your code. This document does not need to be long, but you may write as much as you like. You may include any diagrams you have created or other information you feel is relevant.
Testing Material	Please include any testing code you used as you developed this project. Provide a free-form document describing a further test plan for the system.
Additional Files (optional)	You may include any additional files you would like considered as part of your evaluation. This can be additional source files, additional documentation, etc.

APPENDIX 1 INPUT / OUTPUT EXAMPLES

FILES

InputData.csv

```
0x000003e8,TRI,54.56,8100
0x000003e9,TRI,61.38,6900
0x000003ea,TRI,40.37,7100
0x000003eb,MSFT,10.05,3600
0x000003ec,IBM,32.63,3800
0x000003ed,AAPL,5.71,400
0x000003ee,TRI,40.37,7900
0x000003ef,MSFT,9.52,3700
0x000003f0,AAPL,4.6,2300
0x000003f1,TRI,23.17,7200
```

AAPL.csv

```
4.60,2300,1008
5.71,400,1005
```

IBM.csv

```
32.63,3800,1004
```

MSFT.csv

```
9.52,3700,1007
10.05,3600,1003
```

TRI.csv

```
23.17,7200,1009
40.37,7900,1006
40.37,7100,1002
54.56,8100,1000
61.38,6900,1001
```

Console Output

```
> program InputData.csv
Total Rows: 10
AAPL Rows: 2
IBM Rows: 1
MSFT Rows: 2
TRI Rows: 5
>
```