System and Device Programming

Examination Test – Programming Part
21 January 2015

Examination Time: 1h 45min. Evaluation. 18 marks.
Textbooks and/or course material allowed.

The account history of a set of bank accounts belonging to the same bank customer is stored in a set of file with the following format.

The customer has N bank accounts. The monthly statement of each bank account is recorded in a file. Each monthly statement file records all deposits and withdrawals performed on one bank account during one month. Each deposit (positive value) and each withdrawal (negative value) are preceded by the date in which the operation has been performed. Those files are stored in binary format, with each string field (10 characters) followed by a 32-bit real number. They are named `accountXmonthY.bin` where `X` is the account number, varying from 1 to N, and `Y` is the month number, varying from 1 to M. See the example for more details on the format.

Write a multi-threaded application, within the Windows environment, which is able to store in a file the final balance on each bank account at the end of each month. More specifically the application:

- Receive three parameters on the command line: N, M, and a string. As previously described, N is the number of bank accounts, M is the number of months, and `accountXmonthY.bin` are the names of the $(N \cdot M)$ monthly statement files. The string is the name of the output file storing all monthly balances, and it has to be generated by the application.
- Run N threads such that each thread will be in charge of a bank account. Each thread has to:
  1. Read a monthly statement file of one bank account (starting from the first one, i.e., the one corresponding to the first recorded month).
  2. Compute the final balance for that bank account for that month. Notice that, the initial balance for a month coincides with the final balance for the same bank account computed for the previous month. Moreover, the initial balance for each bank account, i.e., the one before reading the first monthly statement file, is equal to 0.
  3. Synchronize with all other threads to store in the output file the final monthly balance for all bank accounts. The output file must be written in binary format, and must include the month number, and the final monthly balance for all bank accounts stored with the same order in which they are specified in the input files (i.e., from 1 to N). See the example for more details on the output format.
  4. Once all monthly balances for a single month are stored in the output file, all threads have to restart their activity to manage the statements of the next month, starting their behavior from point 1 above.
- When all monthly statement files have been read, all threads have to be stopped, the output file has to be closed, and the main program must terminate.

Notice that the `N` threads have to be **activated only once** at the beginning of the application, and they have to be stopped only when the process is terminated. Moreover, they have to properly synchronize to write their balance in the **correct position** of the output file before going on to manage the next monthly statement file.

Example

Let us suppose that $N = 2$ (2 bank accounts) and $M = 2$ (2 monthly statement files for bank account).
The following are correct ASCII (i.e., readable) version of all files mentioned above:

- Monthly statement files:

`account1month1.bin`	`account2month1.bin`	`account1month2.bin`	`account2month2.bin`
01.01.2014 12345.50	11.01.2014 250.00	08.02.2014 +45.50	07.02.2014 2000.50
20.01.2014 -345.50	20.01.2014 +1250.50	20.02.2014 -5.50	11.02.2014 2800.00
25.01.2014 -2000.00		23.02.2014 1000.00	23.02.2014 -4500.50

- Output file:

month1	10000.00	1500.50
month2	11040.00	1800.50