System and Device Programming

Examination Test – Programming Part
17 July 2015

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

Candidates have to write a Windows 32/64 application computing some statistics on a set of files stored in different directories as described in the following paragraphs.

The application receives three parameters on the command line: a string `S`, and two integers `N` and `M`.

The first parameter `S` is the name of a file. This file stores, in binary format, an undefined number of fixed-length records. Each record includes three strings, each one of 128 characters:

`directoryName inputFileName outputName`

where `directoryName` specifies the name of a directory, `inputFileName` indicates (using wild-cards) one or more input file names, and `outputName` designates a single output file name (without wild-cards but with a path-name). There are at most 64 different output names specified in the file. The following is a correct example of such a file (written in ASCII format **only** for convenience):

```
C:\Users\Documents   *.txt   C:\tmp\doc.txt
C:\Users\Downloads\foo   *.docx   C:\tmp\doc.txt
D:\Pictures   myPicture.jpg   C:\tmp\bin.doc
D:\Templates   tmp*.*   C:\tmp\doc.txt
C:\User\bin   *.*   C:\tmp\bin.doc
...
```

The second parameter `N` indicates that the program has to run `N+1` threads.
The first `N` threads have to read all records of the input file, such that each record is read by only one thread. After reading a record, each thread has to read all input files matching the name `inputFileName` and stored in the `directoryName` directory (meaning a single, flat directory, not a directory tree). For example thread number 3 may read all files `C:\Users\Documents\*.txt`, while thread number 2 may read all files `C:\Users\Downloads\foo\*.docx`, etc. While doing that, each thread has to count the number of files it reads, and for each file the number of characters and the number of lines it contains (to count the number of lines just count the number of newlines stored in the file). At the end of this job, each thread writes those data as a single record at the end of the file `outputName` (e.g., `C:\tmp\doc.txt`). All numbers have to be written as 32-bit integers.
Notice that the same `outputName` (e.g., `C:\tmp\doc.txt`) may appear on more than one record of the input file. As a consequence, threads have to synchronize to write on it.
Conversely, the `outputName` may be different for different input records. As a consequence, threads writing on different output files (e.g., `C:\tmp\doc.txt` and `C:\tmp\bin.doc`) may do that at the same time.
All threads have to stop working when there are no more records (that is, directories to manipulate) in the input file.

The third parameter `M` is the frequency at which the last thread (thread number `N+1`) has to update the user on the status of **all** output files. In other words, thread `N+1` has to check the status of each output file, and it has to print-out (on standard output) the file name and the last `M` records written on each output file (such as `C:\tmp\doc.txt`, `C:\tmp\bin.doc`, etc.) every time `M` records have been written on that file.