System and device programming
02 July 2012
Theory part

Time: 45 min. No textbooks and/or course material allowed.

(15 marks) The final mark is the sum of the 1st > 8 and the 2nd part > 10

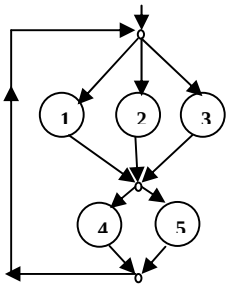
The final mark cannot be refused, it will be registered (no retry for marks >= 18)

1. (3.0 marks) Explain the difference between the **`canonical`**, **`raw`** and **`cbreak`** modes of the terminal line disciplines.

2. (3.0 marks) What is the effect of these bash commands?

```
dd if=/dev/zero of=hd.img bs=512 count=1 seek=$((256*1024))
losetup /dev/loop1 hd.img
```

3. (3.0 marks) Implement this precedence graph with the minimum number of semaphores. The processes are 5, and they are cyclic. Notice that the small dots in the graph are synchronization points, which cannot be substituted by a dummy process.



4. (3.0 marks) Discuss characteristics, advantages and disadvantages of synchronization object in the Windows systems with particular attention to critical sections, mutexes, semaphores, and events.

5. (3.0 marks) Discuss how to perform exception handling in the Windows systems.

System and device programming
02 July 2012
Programming part

Time: 1h 45min. Textbooks and/or course material allowed.

(18 marks) The final mark is the sum of the 1st > 8 and the 2nd part > 10

The final mark cannot be refused, it will be registered (no retry for marks >= 18)

Write a multi-threaded application in the Windows environment, which is able to perform a file system virus scan with the following specifications.

The application receives three parameters on the command line:

```
inputFileName inputDirName outputFileName
```

where:

- `inputFileName` specifies a file including the virus pattern
- `inputDirName` specifies the root of a file system where the virus search has to be performed
- `outputFileName` is the file generated by the application. It contains all file path-names, within the `inputDirNname` file system, whose content is equivalent to the virus pattern.

More specifically, the application has to scan the `inputDirName` filesystem in a recursive way (main directory, files, and sub-directories), and for each:

- file, it has to compare the content of the file (independently from its name or extension) with the virus pattern included in file `inputFileName`.
- sub-directory, it has to run another thread performing the same job on that sub-directory.

In other words, the first thread scans the main directory, it compares all included file with the virus pattern, it runs another thread for each sub-directory, and it waits for all threads he has run.

For each file whose content is equivalent to the one included in the virus file `inputFileName`, the discovering thread has to write on the `outputFileName` a single line, specifying the path and the name of the file. The path is a relative one, starting from `inputDirName` (e.g., `inputDirName\dirA1\dirB2\myFile.txt`).

Path and file names are supposed to be written as strings, in binary form on single lines with fixed length (e.g., of 128 characters). Access to the output file has to be performed in mutual exclusion by all scanning threads. At the end of the entire visit, when all scannning threads have finished but the main one, the main thread prints-out the `outputFileName` on standard output on a line-by-line basis.