Systems and Device Programming

Laboratory 1

Learning goals: In this laboratory activity, you will learn how to use `system` and `exec` system calls. You will understand how to execute commands in your own programs.

Exercise 1

Write a C program that takes two arguments from the command line, **`n`** and **`k`**, creates a subdirectory **`data`**, and creates in that directory **`n`** files, named **`f00`**, **`f01`**, …, **`fnn`**, respectively, each including a variable number (random in range [**`1-k`**]) of random integers.

Exercise 2

Write a C program that takes as arguments a number **`C`** and a directory name **`dir`**.

The main program, using the system call **`system`**, outputs in a file **`list.txt`** the list of files in directory **`dir`**. Then it reads the content of the file **`list.txt`**, and for each read line (a filename) forks a child process, which must sort the file by executing (through the **`execlp`** system call) the Unix **`sort`** program with the appropriate arguments.

Notice that the command **`sort –n –o fname fname`** sorts in ascending order the content of **`fname`**, and by means of the **`–o`** option rewrites the content of file **`fname`** with the sorted numbers. Option **`–n`** indicates numeric rather than alphabetic ordering

The main process can create a maximum of **`C`** children that sort different files in concurrency, to avoid overloading the system. Then, it has to wait the termination of these children before reading the next filename from file **`list.txt`**.

After all files listed in **`list.txt`** have been sorted, the main process must produce a single file **`all_sorted.txt`**, where all the numbers appearing in all the sorted files are sorted in ascending order. Do this by using again system call **`system`** with the appropriate command.

Take care of dealing with a number of files that is not a multiple of **`C`**, i.e., remember to wait for the last files of the list.

Exercise 3

Implement the solution for Exercise 2 replacing system call **`execlp`** with system call **`execv`**.

Summary
At the end of this laboratory activity, you should have understood how to use **`system`** and **`exec`** system calls.