CS342 Operating Systems – Spring 2017
Project 1: Concurrent Processes, IPC, Threads

Assigned: Feb 17, 2017
Due date: Mar 3, 2017, 23:55

Note that looking to the solutions from previous years or Internet or having the project done by somebody else with or without paying money is not allowed and any such action can be penalized by a disciplinary action that may result with 1 or 2 semesters suspension from the school. Even though there is a solution for a project, which might have been assigned in previous semesters, you are not allowed to look at it and you are not allowed to use it. You should come up with your own solution and implementation. This is the only way that you can gain knowledge and acquire skills. This is the purpose of Education.

Objective: *Practicing process creation and process communication thread creation and thread communication. multi-process / multi-threaded application development.*

In this project you will develop a program that will match words in a set of input files to a given keyword and will output the matching lines. The program will be called **pmatch** and it will take the following command line parameters: a keyword of alphanumeric characters, number of input files (N), input filenames, and an output filename. N will be >= 1. Each input file is an ascii text file; a sequence of lines. Each line may contain one or more words of printable characters. A line of an input file is considered to be matching if the keyword appears in the line as a separate word. For example, the keyword *hello* matches to a line "*I say hello to world*", but does not match to a line "*this is hellohello*". The output file will be a sequence of matching lines including the name of the input file containing the matching line and the line number of the matching line in the following format:

```
<input filename>, <line number>: <line>
```

The output should be in sorted order according to first <input filename> and then <line number>. An example invocation of your program can be:

```
pmatch hello 2 in1.txt in2.txt out.txt
```

Program 1: Implement the program using multiple processes created by the parent main program. Call the program as pmatch1. Children will be created with the fork() system call. For N input files, N children will be created. Each child process will process one input file and will write the matching lines together with respective line numbers to an intermediate output file. When all children finish, the parent will read the intermediate files and will generate an output file in the specified format above. In this program, files will be used to pass information from children processes to the parent process.

Program 2: Implement the same program, called pmatch2, but this time use unnamed *pipes* to pass matching line info from children to the parent. No intermediate files.

Program 3: Implement the same program, called pmatch3, using threads this time. Instead of creating one child per input file, now create one thread per input file. Each thread will add the matching line information to a linked list. There will N threads and N linked lists. When all threads finish their job, the initial main thread will process the linked lists and will generate the output in the specified format. You will use POSIX Pthreads library.

Report: You have now 3 programs. Do the following tests and experiments with those 3 programs. Use various input files of various sizes. For each input file set and program combination, measure the time it takes to run the program. Then obtain some plots. For example, given a fix number of input files, how does the completion time of your program change when the input size (average number of lines in an input file) changes? How does the completion time change according to number of matching lines? How does the completion time change depending on the number of input files? Do these experiments for each program. Plot graphs. Do comparisons. Try to interpret the results and try to draw conclusions. Put all into a report.

Submission:

Put the following files into a project directory named with your ID, tar the directory (using **tar xvf**), zip it (using **gzip**) and upload it to Moodle. For example, a student with ID 20140013 will create a directory named 20140013, will put the files there, tar and gzip the directory and upload the file. The uploaded file will be 20140013.tar.gz. In group projects, one of the students will upload and his ID will be used (group members will be included in a README file)

- pmatch1.c: Program 1 C file.
- pmatch2.c: Program 2 C file.
- pmatch3.c: Program 3 C file.
- Makefile: A makefile to compile your programs. We will just type "make" and your programs will be compiled and executables pmatch1, pmatch2, and pmatch3 will be obtained.
- Report.pdf: Your report. Final report should be in PDF form.
- README: Your name and ID and any additional information that you want to put.

Additional Information and Clarifications:

- *Suggestion: work incrementally; step by step; implement something, test it, and when you are sure it is working move on to the next thing.*
- More **clarifications**, additional information and explanations that can be useful for you may be put to the **course website**, just near this project PDF. Check it regularly.
- MAX line length is 256 including the newline character and NULL character at the end.
- MAX filename is 64 characters including the NULL characters at the end.
- MAX word length is 64 characters including the NULL characters at the end.
- MAX number of input files is 20.
- There is no limit on the number of lines in a file.

References:

[1]. *The C Programming Language*. B. Kernighan and D. Ritchie. Second Edition. Prentice Hall. 1998. *A must have book; very useful.*
[2]. Any Book on C, available in Meteksan Bookstore.
[3]. *Operating System Concepts*, Silberschatz et al., 9th edition, Wiley, 2014.