CMPT 214: Programming Principles and Practice
Term 1 2016-17
Subversion and LINUX/UNIX pipes

Solve or complete each of the tasks or problems below. For all problems involving the use of UNIX/LINUX commands, place the command or pipeline you used along with the resulting output (i.e. copy-and-paste from your terminal window) in a file called `lab9.txt`. However, do not include extraneous or superfluous commands or output; only include content relevant and essential to the specified task. Then, with a text editor, add to `lab9.txt` identifying information to clearly distinguish which commands/output/description correspond to each task/question. When done, hand in `lab9.txt` through the moodle page for the lab. This lab is out of a total of 18 marks, with the marks for each question as indicated. Marks may be docked for extraneous, irrelevant, or superfluous content or for not following directions. Your submission is due at 11:55 p.m. on Thursday, November 17.

You may use either the local ismac machine (Apple Macintosh in S311 or S315), or you can use tuxworld for completing this laboratory exercise. Identify which architecture you used at the beginning of `lab9.txt` for the benefit of the marker.

1. (3 marks) Construction a UNIX pipeline to produce, on the standard output, the first 20 prime numbers between 30000000 and 31000000 inclusively, one per line. The pipeline must involve the `factor` program you downloaded and built in last week's lab. The other components of the pipeline must be standard UNIX/LINUX commands. The pipeline should be as concise as possible. Remember that documentation on operation of `factor` is available at http://www.acme.com/software/factor/.

2. (1 mark for each step) In this question, you will practise using `svn` for version control. Place a log of all the commands used in this question, as well as the resulting output, in `lab9.txt`. Each student has a personal repository at https://svn.cs.usask.ca/svn/student/214/*nsid* where *nsid* is their NSID. Locate your "working copy" directory under your home directory.

   As shown in class, one can make use of shell variables to reduce the amount of information that has to be typed in commands to `svn`. Also note that if you do not wish to provide the log message via a `-m` option to `svn` commands that want such a message, make sure to have the `EDITOR` environment variable defined to be a reasonable value before issuing the command.

   Perform the following actions or tasks in the order specified.

   (a) Use the "`svn info`" command to get information about your repository.

   (b) Create a subdirectory in your repository for a new project called Lab09. Then within that new project directory, create subdirectories called `trunk`, `branches`, and `tags`. Make sure to provide informative log messages for each directory creation in the repository.

   (c) Using the "`svn list -R`" command, get a listing of all the (files and) directories in your repository within the Lab09 project. If you wish you can use `--recursive` rather than `-R`.

   (d) Create a directory for your working copy of the software in the Lab09 software. Then change your current working directory to this new directory. Finally, using the "`svn checkout`" command, check out the current version of the trunk of the `Lab09` project. That is, you will want to check out `Lab09/trunk` from your repository. Note that currently that trunk is empty.

(e) The previous command should have created a subdirectory called `trunk` in your current working directory. Change your current working directory to this subdirectory. Perform an `ls` command to demonstrate that the directory is, in fact, empty. Finally, using the `echo` command, create a file named `file1.txt` that contains the string "This is file 1".

(f) Perform an "`svn add`" command to inform `svn` that you intend to add `file1.txt` to the software in the repository. Then use `ls` and "`svn list`" commands to show that `file1.txt` is in the working copy directory but hasn't actually been copied into the repository.

(g) Using the "`svn commit`" command cause `file1.txt` to be "pushed" to the repository. Make sure to supply an informative log message to the commit operation. Then using an "`svn list`" command show that `file1.txt` is now present in the repository.

(h) Using the "`svn status -v`" command, show the status of the files in your current working directory. Then, create the file `file2.txt` using `echo`. Have the content of the file be the single line "This is file 2".

(i) Perform another "`svn status -v`" command. In your `lab9.txt` file comment on the meaning of the difference in status output at this step versus the previous step. What is the status output telling you? Place your answer in `lab9.txt`.

(j) Perform an "`svn add`" command to inform `svn` that you intend to add `file2.txt` to the software in the repository. Then perform another "`svn status -v`" command. Finally, comment on the meaning of the difference in status output at this step versus the previous step. What is the status output indicating to you? Place your answer in `lab9.txt`.

(k) Using the "`svn commit`" command cause `file2.txt` to be "pushed" to the repository. Make sure to provide an informative log message. Then perform another "`svn status -v`" command. Finally, comment on the meaning of the difference in status output at this step versus the previous step. Place your comment in `lab9.txt`.

(l) With the `echo` command add the sentence "This is the second line of file 1" to the end of `file1.txt`. Then perform another "`svn status -v`" command. Finally, comment on the meaning of the difference in status output at this step versus the previous step. Place your comment in `lab9.txt`.

(m) At this point the changes to `file1.txt` need to be committed to the repository. However, before doing this, it is good practice to do an "`svn update`" command first to make sure any changes to the repository are propagated to the working copy. Do this by invoking a "`svn update`" command. Finally commit the changes to `file1.txt` to the repository. Again make sure to provide a meaningful log message.

(n) Using "`svn log`" get a verbose log of the evolution of the content of the file `file1.txt` in the repository. Part of the mark for this question will be based on the informativeness and appropriateness of messages stored in the log.

(o) Peform a second "`svn log`" to again get a verbose log, but this time obtain it for the entire trunk of the `Lab09` project in the repository. Again part of the mark for this question will be based on the informativeness and appropriateness of messages stored in the log.

End of the lab.