Assignment 1. Getting to know your system

Laboratory: Linux scavenger hunt

Instructions: Use the commands that you learned in class to find answers to these questions. Don't use a search engine like Google, and don't ask your neighbor. If you need a hint, ask the TA. When you find a new command, run it so you can see exactly how it works. In addition to turning the answers to these questions, turn in a transcript of your session discovering them. As you do actions, use a Linux-based editor to record each action in a file `lab1.log` that you will submit as part of the assignment.

1. How can you get `man` to print all the commands that have a specific word in their man page (or at least the description part of the man page)? (hint: `man man`)
2. Where is are the `diff` and `tar` programs located in the file system?
3. What executable programs have names that are just one character long, and what do they do?
4. Suppose a command is scheduled by `cron` to execute at 02:30 every day. When daylight saving time begins, and the clock jumps forward from 02:00 to 03:00, when (if at all) is the command run? How about when daylight saving time ends, and the clock jumps backward from 03:00 to 02:00?
5. When you execute the command named by the symbolic link `/usr/bin/cc`, which file actually is executed?
6. The `chmod` program changes permissions on a file. What does the symbolic mode `+t` mean, in terms of permissions?
7. The `find` program lets you search for files. What program, related to `find`, lets you do the same thing faster?
8. What option to `find` lets you search for files that have been modified in the last 3 days?
9. Use the previous answer to find all regular files modified in the last 3 days.
10. Of the files in the same directory as `find`, how many of them are symbolic links?
11. Who wrote the `cat` program?
12. What is the newest regular file in the `/usr/lib` directory?
13. Several useful commands start with `wh`. Which one will tell me my name? Which one will tell me what the previous one does? Which one will tell me the location of the program that will run if where to type the program name at the prompt?
14. In Vim, what does the `dw` command (in command mode) do?
15. Where does the `locale` command get its data from?

Homework: Learning to use Vim and Emacs

- Ben Yoshino, Mastering the VI editor (2002) (`vi` is Vim's predecessor)
- Keith Waclena, A Tutorial Introduction to GNU Emacs (2009)

For all the exercises, record the steps taken to accomplish the given tasks.

To start, download the gzipped tarball `hw1.tgz` with the files needed for the exercise.

To extract the file from the tarball, you can run the command:

```
gzip -cd hw1.tgz | tar xvf -
```

Exercise 1.1v: Moving around in Vim

1. Use Vim to edit the file `exer1.txt`.
2. Move the cursor to somewhere in the line that contains the text "Marina del Rey".
3. Move the cursor to somewhere in the line that contains the text "September 1981".
4. Move the cursor to the first letter of the word "Information".
5. Move the cursor to the first letter of the word "Techniques".
6. Move the cursor to the end of the current line.
7. Move the cursor to the beginning of the current line.
8. Move the cursor back to somewhere in the first line of the file.
9. Have you been moving the cursor using the arrow keys? If so, repeat the above, without using the arrow keys.
10. Doing the above tasks with the arrow keys takes many keystrokes, or it involves holding down keys for a long time. Can you think of a way to do it with fewer keystrokes by using some of the commands available in Vim?
11. When you are done, exit Vim.

Exercise 1.2v: Deleting text in Vim

1. Use Vim to edit the file `exer2.txt`. This file is similar to `exer1.txt`, except with some extra text added. The idea is to delete the extra text so the two files become identical.
2. Delete the 21st line.
3. Delete the words " DELETEME DELETEME DELETEME" from the 15th line.
4. Delete the word "DELETEME " from the 13th line.
5. Delete "ASDF" from the first line.

Once again, try to accomplish the tasks using a small number of keystrokes. When you are done, save the file and exit back to the command line. You can make sure you finished the exercise by comparing the two files, using the following command:

```
diff exer1.txt exer1.txt
```

If there's no text output except the command prompt, then the two files are identical.

Exercise 1.3v: Inserting text in Vim

1. Use Vim to edit the file `exer3.txt`. This file is similar to `exer1.txt`, except some text is missing. The idea is to add the missing text so that the two files become identical.
2. Insert the word "Advanced " on line 13 after the word "Defense".
3. Insert the word " Way" to the end of line 22.
4. Insert a blank line after line 23.
5. When you finish, save the text file and exit Vim. As before, use the `diff` command to check your work.

Exercise 1.4v: Other common tasks in Vim

In additional to inserting and deleting text, there are other common tasks that you should know, like copy and paste, search and replace, and undo.

1. Use Vim to edit the file `exer4.txt`. Once again, the idea is to edit the file to be identical to `exer1.txt`.
2. Cut the words "Processing Techniques Office" and paste it to line 14.
3. Copy the words "September 1981" and paste it to line 25.
4. Paste the words to line 2.
5. … except we didn't really want to do that, undo the paste command.
6. Now let's try some search and replaces. Search the text document for the pattern "er". How many instances did you find? You might have noticed the word "California" is misspelled. Use the search and replace function to replace all misspelled instances at once.

Exercises 1.1e through 1.4e

These exercises are the same as Exercises 1.1v through 1.4v, except using Emacs rather than Vim. Start with fresh copies of all the input files by removing the modified versions of the files and re-extracting them from the gzipped tarball.

Submit

Submit the following files.

`lab1.log`
: as described above.

`lab1.txt`
: Answers to each lab question, along with a very brief description of how you can easily find such the answer from a new Ubuntu system, the next time you need the answer.

`hw1.txt`
: For each homework exercise, the set of keystrokes needed to do the exercise. Attempt to use as few keystrokes as possible. Do not bother to write down the keystrokes needed to start the editor (e.g., `v i SP e x e r 1 . t x t Enter`). Write down the label of the key for each keystroke, e.g., "a", "A" (if you type "a" while holding down the shift key), "Tab", "Enter", "Esc". Use SP for space. Use prefix "C-" and "M-" for control and meta characters: e.g., C-f represents Control-F, and M-f represents Meta-F. Put a space between each pair of keystroke representations. For example: `e m a c s < v i Backspace Backspace Backspace > v i`. If you use some key not described above, invent your own ASCII name for the key and explain what key you mean, but don't put spaces in your key name.

The `.txt` files should be ASCII text files, with no carriage returns, and with no more than 80 columns per line. The shell command:

```
awk '/\r/ || 80 < length' lab1.txt hw1.txt
```

should output nothing.
