Assignment 1. Getting to know your system

[35L home > assignments]

Laboratory: Linux and Emacs 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 in the answers to these questions, turn in a description of your session discovering them. As you do actions, use a Linux-based editor to record your keystrokes and each answer in files `key1.txt` and `ans1.txt` 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 are the `cp` and `wc` programs located in the file system?
3. What executable programs have names that are just one character long, and what do they do?
4. When you execute the command named by the symbolic link `/usr/bin/emacs`, which file actually is executed?
5. The `chmod` program changes permissions on a file. What does the symbolic mode `g+s,o-x` mean, in terms of permissions?
6. What option to `find` lets you search for files that have been modified in the last three weeks?
7. Use the previous answer to find all directories modified in the last three weeks.
8. Of the files in the same directory as `find`, how many of them are symbolic links?
9. What is the oldest regular file in the `/usr/lib` directory?
10. Where does the `locale` command get its data from?
11. In Emacs, what commands have `sort` in their name?
12. Briefly, what do the Emacs keystrokes `C-M-a` through `C-M-h` do? Can you list their actions concisely?
13. In more detail, what does the Emacs keystroke `C-g` do?
14. What does the Emacs `yank` function do?
15. When looking at the directory `/usr/bin`, what's the difference between the output of the `ls -l` command, and the directory listing of the Emacs `dired` command?

Homework: Learning to use Emacs

- Keith Waclena, A Tutorial Introduction to GNU Emacs (2009)
- The Emacs editor, version 25.1 (2016)
- An Introduction to Programming in Emacs Lisp, version 25.1 (2016)

For all the exercises, record the steps taken to accomplish the given tasks. Use intelligent ways of answering the questions. For example, if asked to move to the first occurrence of the word "scrumptious", do not merely use cursor keys to move the cursor by hand; instead, use the builtin search capabilities to find "scrumptious" quickly.

To start, download a copy of the web page you're looking at into a file named `assign1.html`. You can do this with Wget or curl. Use `cp` to make three copies of this file. Call the copies `exer1.html`, `exer2.html`, and `exer3.html`.

Exercise 1.1: Moving around in Emacs

1. Use Emacs to edit the file `exer1.html`.
2. Move the cursor to just after the first occurrence of the word "PUBLIC".
3. Now move the cursor to the start of the first later occurrence of the word "Laboratory".
4. Now move the cursor to the start of the first later occurrence of the word "self-referential".
5. Now move the cursor to the start of the first later occurrence of the word "arrow".
6. Now move the cursor to the end of the current line.
7. Now move the cursor to the beginning of the current line.
8. 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 Emacs?
9. Did you move the cursor using the arrow keys? If so, repeat the above steps, without using the arrow keys.
10. When you are done, exit Emacs.

Exercise 1.2: Deleting text in Emacs

1. Use Emacs to edit the file `exer2.html`. The idea is to delete its HTML comments; the resulting page should display the same text as the original.
2. Delete the 18th line, which is an HTML comment. `<!-- HTML comments look like this. -->`
3. Delete the HTML comment containing the text "`DELETEME DELETEME DELETEME`".
4. Delete the HTML comment containing the text "`https://en.wikipedia.org/wiki/HTML_comment#Comments`".
5. There are two more HTML comments; delete them too.

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 check your work by using a browser to view `exer2.html`. Also, check that you haven't deleted something that you want to keep, by using the following command:

```
diff -u exer1.html exer2.html >exer2.diff
```

The output file `exer2.diff` should describe only text that you wanted to remove. Don't remove `exer2.diff`; you'll need it later.

Exercise 1.3: Inserting text in Emacs

1. Use Emacs to edit the file `exer3.html`.
2. Change the first two instances of "Assignment 1" to "Assignment 37".
3. Change the first instance of "UTF-8" to "US-ASCII".
4. Insert a blank line before the first line containing "`</ol>`".
5. When you finish, save the text file and exit Emacs. As before, use the `diff` command to check your work.

Exercise 1.4: Other editing tasks in Emacs

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

1. Execute the command "`cat exer2.html exer2.diff >exer4.html`" to create a file `exer4.html` that contains a copy of `exer2.html` followed by a copy of `exer2.diff`.
2. Use Emacs to edit the file `exer4.html`. The idea is to edit the file so that it looks identical to `exer1.html` on a browser, but the file itself is a little bit different internally.
3. Go to the end of the file. Copy the new lines in the last chunk of diff output, and paste them into the correct location earlier in the file.
4. Repeat the process, until the earlier part of the file is identical to what was in the original.
5. Delete the last part of the file, which contains the diff output.
6. … except we didn't really want to do that, so undo the deletion.
7. Turn the diff output into a comment, by surrounding it with "`<!--`" and "`-->`".
8. Now let's try some search and replaces. Search the text document for the pattern "`<ol>`". How many instances did you find? Use the search and replace function to replace them all with the initial-caps equivalent "`<Ol>`".
9. Check your work with viewing `exer4.html` with an HTML browser, and by running the shell command "`diff -u exer1.html exer4.html >exer4.diff`". The only differences should be changes from "`<ol>`" to "`<Ol>`", and a long HTML comment at the end.

Exercise 1.5: Doing commands in Emacs

Do these tasks all within Emacs. Don't use a shell subcommand if you can avoid it.

1. Create a new directory named "`junk`" that's right under your home directory.
2. In that directory, create a C source file `hello.c` that contains the following text. Take care to get the text exactly right, with no trailing spaces or empty lines, with the initial `#` in the leftmost column of the first line, and with all other lines indented to match exactly as shown:

```
#include <stdio.h>
int
main (void)
{
  char n = '\n';
  char b = '\\';
  char q = '"';
  char const *p = "#include <stdio.h>%cint%cmain (void)%c{%c  char n = '%cn';%c  char b = '%c%c';%c  char q = '%c';%c  char const *p = %c%s%c;%c  printf (p, n, n, n, n, b, n, b, b, n, q, n, q, p, q, n, n, n, n);%c  return 0;%c}%c";
  printf (p, n, n, n, n, b, n, b, b, n, q, n, q, p, q, n, n, n, n);
  return 0;
}
```

3. Compile this file, using the Emacs `M-x compile` command.
4. Run the compiled program, and put its output into a new Emacs buffer named `hello-out`.
5. Copy this buffer's contents directly into the log that you're maintaining for this exercise. (You *are* using Emacs to maintain the log, aren't you?)

Exercise 1.6: Running Elisp code

1. Visit Emacs's `*scratch*` buffer.
2. In the buffer, compute a random integer by invoking the `random` function. Use `C-j` (`eval-print-last-sexp`) to invoke the `random` function.
3. In the buffer, assign two random integers to the global variables `x` and `y`. You can start by executing `(setq x (random))`. Again, use `C-j`.
4. What is the product of the two variables? You can find this out by executing `(* x y)`. What do you observe about the result? If the answer is the correct mathematical answer, keep trying again with a different pair of random integers until you get an answer that is not mathematically correct.

mathematical answer, keep trying again with a different pair of random integers until you get an answer that is not mathematically correct.
5. Try evaluating `(* x y)` again, but this time with `M-:` (`eval-expression`). What difference do you observe in the output?
6. Are the two random integers truly random in the mathematical sense? If not, what's not random about them?
7. Assuming `(random)` is truly random, what is the probability that the two-variable product mentioned above is mathematically incorrect? Explain how you calculated this.

Submit

Submit the following files.

`key1.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., "`e m a c s SP e x e r 1 . h t m l Enter`") or to type in the complicated C program of Exercise 1.5. 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 or a newline 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 or newlines in your key name.

`ans1.txt`
: Answers to each lab question.

The `.txt` files should be ASCII text files, with no carriage returns, and with no more than 80 columns per line except when logging program output longer than that. The shell command:

```
awk '/\r/ || 80 < length' key1.txt ans1.txt
```

should output only a small number of log lines.

$Id: assign1.html,v 1.39 2017/01/04 00:45:52 eggert Exp $