Lab 2: Shell Commands

Nathan Jarus

June 7, 2017

Introduction

This lab exercise will give you some experience with navigating around the shell and using IO redirection.

You will need to sign in to https://git-classes.mst.edu and clone the repository for this lab.

Your repository will be named something along the lines of `2017SS-A-lab02-<username>`. Make sure to clone with the HTTPS URL (unless youve set up SSH keys).

Feel free to consult with your favorite search engine, relevant `man` pages, the lab instructor and assistants, and your fellow labmates when you need help. Just make sure that what you turn in is your own work!

Problem 0: Preliminaries

1. If you haven't already made yourself a folder for this assignment, go ahead and do that, then change to that directory.
2. Make yourself a text file called `answers.txt` for writing lab question answers in.
3. Put your name at the top of your answers text file so I know who you are.
4. Set a reasonable editor for `git`: `git config --global core.editor jpico` (or another terminal editor of your choice)
5. Stage and commit the file you just created:
   - Stage your file with `git add`
   - Commit your changes with `git commit`

Problem 1: Output redirection

1. Run `echo foo > file.txt`. What does file.txt contain?

2. Run `echo bar > file.txt`. What does file.txt contain now? What do you conclude?

3. Run `echo baz >> file.txt`. What does file.txt contain? What does `>>` do?

(In case you're wondering, there is a reason I put this first in the lab.)

Problem 2: `ls` and friends

1. Run `ls -al *`. Describe the output. What do you conclude about what `*` matches by default?

2. Run `ls -al .*`. What does it output, and why?

3. Write your answers in your `answers.txt` file. Be sure to save, stage, and commit your changes.

Problem 3: Intermediate `man` usage

1. Consult the man page for `ls`. How would you sort a listing by file modification time?

2. How would you tell `cat` to show line numbers when outputting a file?

3. What is the `whatis` command?

4. Write your answers in your `answers.txt` file. Be sure to save, stage, and commit your changes.

Problem 4: Output Redirection

1. Run `echo "apple" > file.txt`. What does `file.txt` contain?

2. Run `echo "banana" > file.txt`. What does `file.txt` contain now? What do you conclude about `>`?

3. Run `echo "carrot" >> file.txt`. What does `file.txt` contain? What does `>>` do? Be sure to note how is that different from `>`.

4. Run `echo "a c b e d g f" | wc`. What is the output? In your own words, what does `|` do?
(Hint: Use `man` to figure out what `wc` does.)

5. Write your answers down in the `answers.txt` file. Be sure to save, stage, and commit your changes.
Note that you should not commit `file.txt`. You can delete it.

Problem 5: A series of tubes

Clayton Price has given you a text file with a story he wrote in it so that you could edit it. While you were editing, you put in a bunch of comments about his writing skills. You later decide to spare his feelings, and you want to remove those comments. Fortunately, you put each comment on its own line, starting with a `#`:

```
My Cow Story
by Clayton Price
# okay, let's see what this nerd has to say
Homer was in a beautiful field.
# what's this, a Simpsons reference already?
There was a cow.
Homer hungrily eyed the cow, drooling excessively at the thought of steak.
# welcome to adverb city, my friend!
The end.
# That's it!?
```

You could just open the file up in an editor and remove the comments by hand, but you are an enlightened programmer! You decide there is A Better Way.

1. Write a short program, `filter.cpp`, that accepts a story (text) over STDIN and outputs the story (without the comment lines!) over STDOUT. (Hint 1: remember, piping puts command output into STDIN. You also probably want to output some stuff to STDOUT.)
   (Hint 2: use `getline()`, either for character arrays or for strings.)

   (Note: no cheating with `grep`, `sed`, or other tools we haven't talked about yet!)

2. How do you use your program to filter out the comments? In other words, write the shell command that uses your program to filter comments from a story file.

3. Run your program without piping anything into it and type some stuff in. What happens? (Hint: use Ctrl+d to send an EOF character.)

4. Write your answers down in the `answers.txt` file. Be sure to save, stage, and commit your changes.

5. Dont forget to stage and commit `filter.cpp` as well. Do not commit your compiled file... we only want the `.cpp` file.

After completing this problem, bask in your newfound ability to shave yaks.

Problem 5: Submitting your homework

1. Use `git status` and/or `git log` to make sure all your changes are committed.
2. Use `git push` to push your changes to your remote.
3. Log into https://git-classes.mst.edu and make sure your files look correct.

Your repository should contain the following files:

- `README.md` (This file is unchanged we wrote it.)
- `.gitignore` (You can change it if you want to, but its not necessary.)
- `answers.txt`
- `filter.cpp`