CS 279 - Week 13 Lab Exercise

Deadline

Due by the end of lab on 2022-11-17.

How to submit

Submit the files specified below on https://canvas.humboldt.edu.

Purpose

To practice a bit with how to re-run previous commands from a Bash shell's history, with hard and symbolic links, and with Bash functions.

Important notes

- Work in PAIRS for this lab exercise:
  - two people at one computer,
  - one typing (driver),
  - one saying what to type (navigator),
  - both discussing along the way!

  When done, the driver should e-mail the files to the navigator, so BOTH of you can EACH submit them.
- Assume, for all `bash` scripts in this course, that the following are required:
  - Start each script *(EXCEPT for a script containing JUST Bash functions)* with the line that is considered good style (and is a CS 279 course requirement), that specifies that this script should be executed using the `bash` shell
  - After a blank line, put in one or more **comments** including at least the name of the shell script, your names, and its last modified date
  - And follow these comments with a blank line.

Lab Exercise setup

- use `ssh` to connect to the one of your accounts on `nrs-projects.humboldt.edu`
- make and protect a directory `279lab13` using the commands:

```
mkdir 279lab13
chmod 700 279lab13
```

- go into that directory using:

```
cd 279lab13
```

Problem 1

A few additional fun `history` command bits I'm not sure were mentioned in class this week:

- If you call `history` with an integer argument `n`, it shows the last `n` commands executed.
- If you type control-r (^R), you can then start typing a previous command, and the shell will attempt to search through the history and complete it.

In a file `lab13-1.txt`:

- put your names
- put your answers for each of the following

1 part a

What *pair* of characters can you type, at any point, to simply re-execute the previous command?

1 part b

What can you type to redo the command with number 46 in the command history?

1 part c

What command can you type to just see the last 7 lines of the command history?

1 part d

What can you type to redo the most recent command starting with `grep`?

1 part e

What can you type to redo the most recent command including `grep` anywhere *within* it? (It might be part of a piped command, for example.)

1 part f

What can you type to redo the command 6 commands ago?

1 part g

What can you type to redo the immediately preceding command *except* replacing the first instance of `297` in that command with `279`?

1 part h

What can you type to redo the immediately preceding command except replacing *all* instances of `297` in that command with `279`?

Submit your resulting `lab13-1.txt`

Problem 2

You'll play with links a bit in this problem.

- Create a subdirectory `lab13-2`, make its permissions 700, and `cd` to it -- I'd like to keep the output files for this problem uncluttered.

- Create a short-but-non-empty text file of your choice in this subdirectory.
- In a file `lab13-2.txt`:
  - put your names
  - put the name of the short-but-non-empty text file you just created
  - write a command to create a hard link to this short-but-non-empty text file in this subdirectory, and also run it in this subdirectory
  - write a command to create a symbolic/soft link to this short-but-non-empty text file in this subdirectory, and also run it in this subdirectory
  - (this file `lab13-2.txt` is now ready to submit, along with the files resulting from the steps below)
- Use `cat` with the name of your short-but-non-empty text file, redirecting the result into a file `lab13-2-part1-orig.txt`, so I'll be able to see the original state of the original file you started with.
- Look at the output of `ls -li` -- see how the hard link and the symbolic link compare to the text file you originally linked to. Then do:

  ```
  ls -li > lab13-2-part2-links.txt
  ```

  ...so I can see that you created these links.
- Use `nano` with the name of your **original short-but-non-empty text file**, and noticeably change it in some fashion.
  - Use `cat` or `more` to then look at your text file, your hard link, and your symbolic link.
  - Then use `cat` with the names of your text file, your hard link, and your symbolic link as its three arguments, redirecting the result to `lab13-2-part3-chg1.txt`
- Use `nano` with the name of your **hard link**, and noticeably change it in some fashion.
  - Use `cat` or `more` to then look at your text file, your hard link, and your symbolic link.
  - Then use `cat` with the names of your text file, your hard link, and your symbolic link as its three arguments, redirecting the result to `lab13-2-part4-chg2.txt`
- Use `nano` with the name of your **symbolic link**, and noticeably change it in some fashion.
  - Use `cat` or `more` to then look at your text file, your hard link, and your symbolic link.
  - Then use `cat` with the names of your text file, your hard link, and your symbolic link as its three arguments, redirecting the result to `lab13-2-part5-chg3.txt`
- Now, use the `rm` command to remove your original short-but-non-empty text file.
- Do the command:

  ```
  ls -li > lab13-2-part6-rm.txt
  ```

  ...so I can see that you removed the original text file.
- Now do the `cat` command with the name of your hard link, redirecting the result into `lab13-2-part7-hard.txt`

  And, do the `cat` command with the name of your symbolic link, redirecting the result **using `2>`** into

`lab13-2-part8-soft.txt`

(Rhetorical question, that you don't have to turn in an answer for: can you figure out why I asked you to use `2>` for the command involving the symbolic link here? 8-))

- Hmm, that's quite a few files, isn't it?

  To more conveniently submit your files for this problem, use `tar` to create an archive of this subdirectory `lab13-2`, use `gzip` to compress it, and submit your resulting file `lab13-2.tar.gz .`

Problem 3

Create a shell script `lab13-3-functions.sh` that contains the following two Bash functions.

- Because it will be included in other shell scripts using `source`, do not start it with the usual `#!/bin/bash` -- but do still include the usual shell-script comment(s)!
- Create a function `make_line` that expects a string to repeat and a number of repetitions, and echoes to standard output a **single** line containing that string repeated that many times. (For today, this can be very trusting, and assume it is called with two reasonable arguments.)
  - For example,

    ```
    make_line Moo 4
    ```

    ...would cause the following to be echoed to the screen:

    ```
    MooMooMooMoo
    ```
- Fun reminder: For variable `$myVar`, you can obtain its length using `${#myVar}`. (And yes, this does work with command-line arguments and parameters, fortunately!)

  Create a function `highlight` that expects a string to echo to the screen in an eye-catching way, and, with the help of `make_line`, echoes to standard output three lines:
  - uses `make_line` to echo a line of = characters whose length is equal to the length of `highlight`'s string argument plus 4
  - echoes an =, a blank, `highlight`'s string argument, a blank, and an =
  - uses `make_line` to echo another line of = characters whose length is equal to the length of `highlight`'s string argument plus 4

  (For today, this also can be very trusting, and assume it is called with one reasonable argument.)
  - For example,

    ```
    highlight "CS 279"
    ```

    ...would cause the following to be echoed to the screen:

    ```
    ==========
    = CS 279 =
    ==========
    ```

Now create another Bash shell script `lab13-3-use.sh` that:

- uses the `source` command with `lab13-3-functions.sh`
- calls `make_line` at least three times, each time with a different-but-reasonable pair of arguments

- calls `highlight` once, with a string of your choice
- asks the user to enter a desired string, and reads in what they enter
- calls `highlight` with that entered string
- (and you can do more with these if you wish, also)

Submit your resulting files `lab13-3-functions.sh` and `lab13-3-use.sh`.

Submit these files to Canvas:

- `lab13-1.txt`
- `lab13-2.tar.gz`
- `lab13-3-functions.sh`
- `lab13-3-use.sh`