CS 279 - Homework 11

Deadline:

Due by 11:59 pm on **SUNDAY**, November 30.

How to submit:

Submit your files using `~ah270/279submit` on nrs-labs, with a homework number of `11`, by the deadline shown above.

Purpose

To practice more with `sftp` and some of the other commands discussed in class this past week.

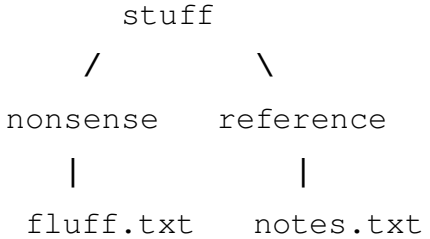
Important notes:

- **Each** bash shell script that you write is expected to include a descriptive opening comment block including your name and the last modified date.
  - each bash shell script FUNCTION you write should also have an descriptive opening comment block including at least descriptions of what it expects and what it produces and/or its side-effects when run
- It is possible that your answers may be collected and posted to the course Moodle site.

The Problems:

Problem 1

Assume you have the following directory structure on your local machine:



Assume you are currently in directory `stuff` on your local machine, and assume that your home directory on `nrs-labs` does not current contain directories named `stuff` or `nonsense`.

In a file `hw11-1.txt`, put your name, and then a sequence of commands you could type to connect to your account on remote machine `nrs-labs.humboldt.edu` with command-line `sftp` and then, WITHIN that SINGLE `sftp` SESSION**,** create directories `nonsense` and `reference` and transfer copies of the `.txt` files shown in each above on your local machine to those respective directories on

`nrs-labs`. Include a command for ending `sftp` when you are done.

Submit your resulting `hw11-1.txt`.

Problem 2

In a file `hw11-2.txt`, include your name and an appropriate command for each of the following.

2 part a

Write an `expr` command that will compute the average of three variables `val1`, `val2`, and `val3`.

2 part b

Write a command using `echo`, a pipe, and `bc` to compute the result of raising variable `val1`'s value to the 3rd power (cube it).

2 part c

You want your shell script to pause for 10 seconds before proceeding to the next command. Write a command that will accomplish this.

2 part d

Write a command that would show you the real, user, and system time used by a `find` command that starts in the current working directory and prints out the names of files whose names end in `.sh`.

2 part e

Write a command that would start a shell script `do_a_bunch.sh` in the background in such a way that it would continue to execute even if you log out of the terminal you start it in (or close it or otherwise disconnect from it).

2 part f

Write a command that would start a shell script `resource_hog.sh` in the background in such a way that its priority would be reduced by `15` from the default priority it would usually receive.

2 part g

Assume that `7683` is the process id of a currently-running background process. Write a command that will reduce the priority of this process by `5` from its current priority.

Submit your resulting `hw11-2.txt`.

Problem 3

In a bash shell script `funct-play.sh`:

- write a function of your choice that accepts at least one command-line argument and uses it in a computation -- making use of either `expr` or `bc` -- that echoes its result to the screen.
- write a second, separate function of your choice that accepts at least one command-line argument and indicates its "result" by either returning a zero exit status (for success) or a non-zero exit status (for failure).

Then, in another shell script `use-functs.sh`,

- (source the functions from `funct-play.sh`)
- call the first function from `funct-play.sh` such that its result is assigned to a local variable, and then print the resulting value of that local variable in a descriptive message
- call the second function from `funct-play.sh` twice, once in a way such that it succeeds and once in a way such that it fails, after each printing out the exit status of that call in a descriptive message.

Submit your resulting `funct-play.sh` and `use-functs.sh`.

Problem 4

Consider function `mantra2` from Homework 10, and shell script `mantra2.sh` from Homework 7.

What kind of time difference might there be in running a shell script version of something versus a bash function version of something?

Write a shell script `compare-mantras.sh` that:

- expects three command line arguments: the string to be printed, the desired number of repetitions of that string, and the desired number of times to *time* calls of `mantra2` and `mantra2.sh` using that string and number of repetitions
  - (exit with a non-zero exit status if you don't get 3 arguments or if the 2nd and 3rd arguments aren't quantities greater than 0)
  - print a descriptive message of your choice to the screen, and then call shell script `mantra2.sh` using those arguments that many times, timing them each time;
  - and print a descriptive message of your choice to the screen, and then call function `mantra2` that many times, timing them each time.
- DON'T copy the functions into `compare-mantras.sh` -- source them!

Are you surprised by the results? (You don't have to answer that -- it is just interesting to consider.)