Homework 1 - Intro to Bash

Due: Thursday October 14th at 11:59pm PST
Lock: Sunday October 17th at 11:59pm PST

Intro

In this assignment you'll practice with shell commands covered in lecture and write short shell scripts.

General hints:

- Before trying to write and debug a script, work out the necessary commands by experimenting in a shell window.
- Use man pages and help documentation to fully explore commands. There are a lot of options we didn't have time to cover in lecture.

Resources:

- Lectures 1-4
- Bash Cheat Sheet
- Bash Manual
- Bash Style Guide

Instructions

For this assignment you will be creating three different bash script files:

1. defprivate.sh
2. sortpeople.sh
3. spellcheck.sh

At the top of each file you submit for HW1 please include the following comment:

```
# Your full name
# your uwnetid
# HW1: Problem #X
```

Problem 1 - alias

Create a bash alias `private` such that when you run `private foo`, the entire subtree of the file-system starting at `foo` (so just `foo` if it is a file, but `foo` and all of its files and subdirectories recursively if it is a directory) has its permissions changed as follows:

- The user's (owner's) permissions are unchanged.

- The group permissions are changed to match the owner's.
- The world (others) permissions are changed to remove read and write permissions, but leave the execute permissions unchanged.

Put your alias in a file `defprivate` such that entering `source defprivate` would make `private` available in the current shell (i.e., `source defprivate` makes the `private` command available for use in the current shell, but does not actually execute it immediately). The alias should also work on multiple arguments.

Hint: `g=u`.

Problem 2 - sort

Create a script called `sortpeople.sh` that sorts a list of people by their last names. For example, when we run the script `./sortpeople.sh names.txt`, where `names.txt` contains the following:

- `Ada Lovelace`
- `Rachel Carson`
- `Hypatia Alexandria`

it prints

- `Alexandria`
- `Carson`
- `Lovelace`

You should use three commands, piped together using the `|` symbol. You should use `cat` and `sort` and well as either `cut` or `awk`. Use man pages and/or Google to look up how to use these commands and what options you might need.

You can find a list of names here.

- For background on these names, check this out (scroll down).

A fully completed script will include error checking for the input argument, and be used as an executable. However, it might be useful to test the commands at the command line first.

Problem 3 - spell check

Create a bash script called `spellcheck` that works as follows:

- If it is given anything other than one argument, it prints an appropriate error message to stderr and exits with a return code of 1.
- If the argument supplied is not a valid file (`[ ! -f $1 ]`), spellcheck prints an appropriate error message to stderr and exits with a return code of 1.
- Your script should read each word in the supplied file and use `grep` to compare that word to the dictionary that comes default with all Unix systems (which is assumed to exist at `/usr/share/dict/words`) .
    - If the word is *not* found in the dictionary it should be added to the end of a file called `<FILE>.spelling`. You may consider any word not found in the dictionary to be misspelled, including acronyms or numbers. You may add repeated words to the word list in the initial step.
- After processing the text file, print a statement that specifies how many misspelled words were found.
- Your script should exit with a return code of 0 after completing the final print statement.

For example, executing: `./spellcheck linus.txt` will create a file called `linus.txt.spelling`, which contains a list of words such a 'Linux' and 'Torvalds'.

The output of `spellcheck` with all the extra credit complete:

```
./spellcheck replacing linus.txt.spelling file
./spellcheck found 32 spelling errors, output to file
linus.txt.spelling
```

Hints:

- You should use shorttext and linus.txt to test your code. You can also find an example output in linus.txt.spelling.
- Nested loops allow you to read lines, and words within lines. This is not the only way to solve the problem, but it works.
- In Lecture 6 we discovered how to find words in the words file. Look for options that allow case insensitive and quiet operation.
- There are a couple of slides from Lecture 3 that talk about redirecting your output.
- As always, get one small part of your script working at a time.