Project PL1

Due Date: September 21

Purpose

In this small(ish) project, you will get a taste of being an IT professional by writing a few short Linux scripts using Bash.

Problem

You are an IT engineer working for MacroStiff, and as such, you have to supply software and operating system tools for the various software engineers and computer scientists. Many of them don't want to be bothered writing scripts to take care of relatively simple tasks, so the job falls to you. You must write a small suite of Linux Bash scripts for various tasks, the names for which follow Linux conventions of being all lower-case and which will be run without file extensions (the file names for the scripts themselves should have the `.sh` extension):

1. `myfind [OPTION]` *filename* - search for *filename*.
2. `myoccur` *word filename* - find the number of occurrences of *word* in *filename*. This should be case insensitive.
3. `mymeanf [OPTION]` *filename* - find the floating point average of a list of numbers stored in the file named *filename*.
4. `mymeani` *filename* - find the floating point average **rounded** to the nearest integer of a list of integers stored in the file named *filename*.
5. `mymedian` *filename* - find the median of a list of integers stored in the file named *filename*.
6. `myspell` *filename1 filename2* - spell check the words in the file *filename1*, where *filename2* contains a list of the found misspelled words, one per line and without duplicates.
7. `mystore [OPTION]` - display **all** of the file names and file size for the current user, no matter what directory the script is run in.

   The output should be sorted by file size, and display 10 lines at a time. Pressing the space bar should then show the next 10 lines. The output should be formatted as follows:

   ```
   2.3M      /home/student/mydir/afile.doc
   1.1M      /home/student/otherdir/anotherdir/foo.txt
   3K        /home/student/Desktop/one.cpp
   ```

8. `setup.sh` - automatically set up the scripts so that they can be run and done so without the `.sh` extension. This will be run first, and then all of the above should work.

Input

The following uses the number of the script as defined above.

1. The option is `-a` which means search "all"; search the entire user's file space. With no option, just search the current directory. The file name is a valid string with no spaces. Underscores and periods are valid. This is the case with all file names below as well.

2. The file contains any number of lines and words of plain ASCII text. The word to find is a legal string with no spaces or punctuation.

3. The option is -*number*, where *number* is an integer from 1-5 indicating the number of decimal points to **round** to. If no option is given, then the default is two decimal points. The file contains a list of integers, one per line. There is no other data in the file.

4. The file contains a list of integers, one per line. There is no other data in the file.

5. The file contains a list of integers, one per line. There is no other data in the file.

6. The option is `-c` where all the words in the output list should be in lower case. The file *filename1* is simply an ASCII text file; it may contain any valid ASCII characters.

7. The option is `-s` where the output will be in smallest to largest order. By default, the output should be largest to smallest.

8. No input.

If some error condition occurs, then the script's behavior should be consistent with Linux commands. This means that error messages, or lack thereof, are up to you.

Output

- Output should be as defined for the problems above and/or how you think Linux would handle it.

- Additional output for `myspell`:

  - the list of words is alphabetized
  - every word is shown only once; if two words are the same but one is capitalized and the other is not, then they count as two different words; however, see below for another option.
  - if the command is given as:

    `myspell -c` *filename1 filename2*

    then the script should work exactly as above **except** that all the words in the output list should be in lower case. That means that if two words in the input are the same but one was capitalized and the other was not, only the lower case word should be in the output list. If there is a misspelled word that is only used with capitalization in the input, it should be stored in lower case in the output list.

- Following Linux command conventions, labels and explanatory text is **not** necessary or even desirable (in case you want to pipe output to another script, for example). If a function returns no data (for example, there are no occurrences of a word in a file), the output is up to you, but should follow Linux conventions (e.g., do you want to output 0 or nothing at all?).

Notes

- **Do not use awk in any of your scripts.**
- There is a link to a Bash "cheatsheet" on the course web page. There is, of course, a wealth of information on the Interwebs and our new buddy AI, as well.
- Put a comment at the top of each script with:
  - your name
  - the name of the script
  - a description of its function
  - an exact description of valid input
  - an exact description of what is output
- Be sure to name and implement all of your scripts as described above. I will write my own script to test yours, and if something goes wrong because you have misnamed a file or put the arguments in an order not specified, your grade will suffer.
- Being a good IT professional, you want to be as succinct as possible in all of your scripts. This means that you want to use as many Linux and/or Bash built-in tools as is feasible. Try to make each script as short as possible.
- Bundle all of your scripts into one tarball:

  `tar cvf` *yourLastName*`PL1.tar` *file1 file2 file3 ...*

  Upload the tar file (called a "tar ball") view canvas before 11:59:59 PM on the due date. (By the way, you might want to test that your tar ball before turning it in.) Turn in a printed copy of your scripts at the beginning of class the next day. This printout should be as short as possible; combine all of your scripts into one file for printing, with two blank lines between each script. Write and sign the Honor Code on what you turn in.

This is a feature, not a bug.
– Schwartz and Christiansen, *Learning Perl*, p. 65