# Assignment 2. Shell scripting

## Laboratory: Spell-checking Hawaiian

Keep a log in the file `lab2.log` of what you do in the lab so that you can reproduce the results later. This should not merely be a transcript of what you typed: it should be more like a true lab notebook, in which you briefly note down what you did and what happened.

For this laboratory we assume you're in the standard C or POSIX locale. The shell command `locale` should output `LC_CTYPE="C"` or `LC_CTYPE="POSIX"`. If it doesn't, use the following shell command:

```
export LC_ALL='C'
```

and make sure `locale` outputs the right thing afterwards.

We also assume the file `words` contains a sorted list of English words. Create such a file by sorting the contents of the file `/usr/share/dict/words` on the SEASnet GNU/Linux hosts, and putting the result into a file named `words` in your working directory. To do that, you can use the `sort` command.

Then, take a text file containing the HTML in this assignment's web page, and run the following commands with that text file being standard input. Describe generally what each command outputs (in particular, how its output differs from that of the previous command), and why.

```
tr -c 'A-Za-z' '[\n*]'
tr -cs 'A-Za-z' '[\n*]'
tr -cs 'A-Za-z' '[\n*]' | sort
tr -cs 'A-Za-z' '[\n*]' | sort -u
tr -cs 'A-Za-z' '[\n*]' | sort -u | comm - words
tr -cs 'A-Za-z' '[\n*]' | sort -u | comm -23 - words
```

Let's take the last command as the crude implementation of an English spelling checker. Suppose we want to change it to be a spelling checker for Hawaiian, a language whose traditional orthography has only the following letters (or their capitalized equivalents):

```
p k ' m n w l h a e i o u
```

In this lab for convenience we use ASCII apostrophe (') to represent the Hawaiian ‘okina (‘); it has no capitalized equivalent.

Create in the file `hwords` a simple Hawaiian dictionary containing a copy of all the Hawaiian words in the tables in "English to Hawaiian", an introductory list of words. Use Wget to obtain your copy of that web page. Extract these words systematically from the tables in "English to Hawaiian". Assume that each occurrence of "`<tr> <td>Eword</td> <td>Hword</td>`" contains a Hawaiian word in the `Hword` position. Treat upper case letters as if they were lower case; treat "`<u>a</u>`" as if it were "`a`", and similarly for other letters; and treat `` ` `` (ASCII grave accent) as if it were `'` (ASCII apostrophe, which we use to represent ‘okina). Some entries, for example "`H<u>a</u>lau, kula`", contain spaces or commas; treat them as multiple words (in this case, as "`halau`" and "`kula`"). You may find that some of the entries are improperly formatted and contain English rather than Hawaiian; to fix this problem reject any entries that contain non-Hawaiian letters after the abovementioned substitutions are performed. Sort the resulting list of words, removing any duplicates. Do not attempt to repair any remaining problems by hand; just use the systematic rules mentioned above. Automate the systematic rules into a shell script `buildwords`, which you should copy into your log; it should read the HTML from standard input and write a sorted list of unique words to standard output. For example, we should be able to run this script with a command like this:

If the shell script has bugs and doesn't do all the work, your log should record in detail each bug it has.

Modify the last shell command shown above so that it checks the spelling of Hawaiian rather than English, under the assumption that `hwords` is a Hawaiian dictionary. Input that is upper case should be lower-cased before it is checked against the dictionary, since the dictionary is in all lower case.

Check your work by running your Hawaiian spelling checker on this web page (which you should also fetch with Wget), and on the Hawaiian dictionary `hwords` itself. Count the number of "misspelled" English and Hawaiian words on this web page, using your spelling checkers. Are there any words that are "misspelled" as English, but not as Hawaiian? or "misspelled" as Hawaiian but not as English? If so, give examples.

## Homework: Find duplicate files

Suppose you're working in a project where software (or people) create lots of files, many of them duplicates. You don't want the duplicates: you want just one copy of each, to save disk space. Write a shell script `sameln` that takes a single argument naming a directory D, finds all regular files immediately under D that are duplicates, and replaces the duplicates with hard links. Your script should not recursively examine all files that are in subdirectories of D; it should examine only files that are immediately in D.

If your script finds two or more files that are duplicates, it should keep the file whose name is lexicographically first (for example, if the duplicates are named X, A, and B, it should keep A and replace X and B with hard links to A); however, it should prefer files whose name start with "." to other files (for example, if the duplicates are named .Y, .X, A, and B, it should keep .X and replace the others with hard links to .X).

If your script finds a file in D that is not a regular file, it should silently ignore it; for example, it should silently ignore all symbolic links and directories. If your script has a problem reading a file (for example, if the file not readable to you), it should report the error and not treat it as a duplicate of any file.

You need not worry about the cases where your script is given no arguments, or more than one argument. However, be prepared to handle files whose names contain special characters like spaces, "*", and leading "–".

Your script should be runnable as an ordinary user, and should be portable to any system that supports the standard POSIX shell and utilities; please see its list of utilities for the commands that your script may use. Hint: see the `cmp`, `ln`, and `test` utilities.

When testing your script, it is a good idea to do the testing in a subdirectory devoted just to testing. This will reduce the likelihood of messing up your home directory or main development directory if your script goes haywire.

## Submit

Submit the following files.

- The script `buildwords` as described in the lab.
- The file `lab2.log` as described in the lab.
- The file `sameln` as described in the homework.

All files should be ASCII text files, with no carriage returns, and with no more than 80 columns per line. The shell command:

```
awk '/\r/ || 80 < length' lab2.log sameln
```