CSE 303, Spring 2007, Assignment 1
Due: Friday 6 April, 9:00AM

Last updated: March 28

You will get experience using the Linux bash shell, using emacs, and writing very short scripts.

1. (Commands) First run the command `script problem1`. Then run at least 75 *different* commands using at least 12 different programs. Then run the `exit` command.

   - Only commands that succeed (do not print an error) count.
   - For this problem, two commands are *different* if they use different programs and/or different options, but *not* just different filenames. (Examples: `ls` and `ls -a` are different but `ls foo` and `ls bar` are the same.)

2. (Command-line editing) Suppose you type `my flea has dogs` on the bash command-line, leaving the cursor to the right of the "`s`". Your job is to turn the command line into `my dog has fleas` in a small number of keystrokes where *every* keystroke involves holding down either the Meta (often Esc) or Ctrl key. Use emacs to create a text file called `problem2` that describes your solution, including the state of the command-line after each step.

   Notes:

   - This question is silly, but it should help you learn useful things.
   - Your instructor was able to do it in 8 keystrokes, the first of which was Ctrl-b. A few more is fine for full credit; dozens is not.

3. (Job-control) Your instructor has put two annoying-but-harmless programs on `attu`: `~djg/inf1` and `~djg/inf2` both print something out once every second forever. Suppose you have a shell where someone has typed `~djg/inf1 &` and `~djg/inf2 &`, but you do not know which was typed first. Describe two different ways in this shell to make `~djg/inf2` stop running without making `~djg/inf1` stop. Use emacs to create a text file called `problem3` that describes your solutions.

   Hints:

   - You can make a program stop whether it is in the foreground or the background.
   - It is fine for your two ways to start the same way.

4. (An 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 its files and subdirectories recursively if it is a directory) has its permissions changed as follows:

   - The user's permissions are unchanged.
   - The group and world permissions are set to no access of any sort.

   Put your alias in a file `problem4` such that running `source problem4` would make `private` available in the shell.

5. (Script) Create a bash script called `acro` that works as follows:

   - If it is not given exactly one argument, it prints an appropriate error and exits.
   - Assume the argument is a filename, possibly including an abosolute or relative path.
   - If the filename ends with the four characters `.pdf` then run `/usr/local/bin/acroread` in the background passing it the filename. (This will launch Adobe Reader, displaying the pdf file.)
   - Otherwise, run `/usr/local/bin/acroread` in the background passing it the filename with `.pdf` added to the end.

In other words, `acro` is a little "wrapper" that makes the `.pdf` part of the file name *optional*; it works whether or not it is present.

Hints: `dirname`, `basename`, backquotes. There is no need to use conditionals (except for argument-checking). Sample solution is 9 lines including everything (even 2 blank lines).

6. (Script) Create a bash script called `datedwordcount` that works as follows:
   - If it is given fewer than two arguments, it prints an appropriate error and exits.
   - Assume all the arguments are filenames for text files; you do not need to check for this.
   - Append to the file indicated by the second argument the following information:
     - The time and date
     - One line for each of the third-through-last arguments, containing the number of words in the file and then the name of the file
     - One line with the total number of words in all the files and then the word "total"

   For example, executing: `./datedwordcount log foo bar; ./datedwordcount log foo*; cat log` might produce something like:

```
Mon Mar 26 20:42:16 PDT 2007
   4 foo
  17 bar
  21 total
Mon Mar 26 20:42:16 PDT 2007
  4 foo
  3 food
  7 total
```

   (The dates are the same only because the two invocations of the script happened in the same second.)

   Hints: shift, date, wc, `$@`. Sample solution is 11 lines including everything.

7. **(Extra Credit)** Note: Remember the course policy on extra credit. You may do any or all of the following.
   - For problem 3, also describe how to do it from *a different shell*. You may assume there is only one process running `~djg/inf2`.
   - For problem 5, write a variant `acro2` that also allows the filename given to end with a period (`.`) in which case it adds the three characters `pdf`.
   - For problem 6, write a variant `datedwordcount2` with either or both of these changes:
     - Do *not* include the total line.
     - For each filename, if the filename already occurs in the output file *and* the most recent occurrence of it has the same number of words as you would output, then suppress the output for that file.

Assessment: Your solutions should be:

- Correct scripts, etc. that run on `attu.cs.washington.edu`
- In good style, including indentation and line breaks
- Of reasonable size

Turn-in Instructions Use the `turnin` command (man turnin) for course cse303 and project hw1. In particular, type:

```
turnin -ccse303 -phw1 problem1 problem2 problem3 problem4 acro datedwordcount
```

from a directory containing your solution. If you use one late-day (see the syllabus) use the project hw1late1 instead of hw1 and similarly hw1late2 for two late days. If you do the extra credit, turn in additional files with appropriate names.