CSE 303, Autumn 2007, Assignment 1
Due: Monday, October 8 at Noon

Last updated: Sept. 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 60 *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.)

   Hint: look in the *Unix Pocket Guide* for ideas.

2. (Command-line editing) Suppose you type `how now brown cow` on the bash command-line, leaving the cursor to the right of the final "`w`". Your job is to turn the command line into `don't have a cow now` in a small number of keystrokes where you *may not* retype any words that appear in both phrases, or use the delete or backspace keys to individually delete letters in words that need to be removed. You will need to type in new words, but you must use keystrokes involving holding down either the Meta (often Alt or Esc) or Ctrl keys to rearrange or delete words. Use emacs to create a text file called `problem2` that describes your solution, including the state of the command-line after each step.

   Note: This question is silly, but it should help you learn useful things.

3. (Commands and output) Use each of the following commands such that "xyzzy" (without the quotes and nothing more) is printed on standard out, and nothing is printed on standard error. You can precede your commands with other commands (e.g., to create a file) and/or pass options to your commands.

   ```
   echo, cat, ls, grep, !!
   ```

   Hint: The last one is tricky. Put a script that does nothing in a file with a particular name.

   In a text file called `problem3` describe your solution, including each command you use and a *very brief* explanation of it.

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 `combine` that takes 2 or more arguments, call them $f1$, $f2$, ..., fn. Script `combine` should work as follows:

   - All arguments are treated as filenames.
   - If fewer than two arguments are given, print a suitable error message on `stderr` and exit.
   - If a file or directory $f1$ already exists, print "Error: first file exists" on `stderr` and exit.
   - Otherwise concatenate the contents of $f2$, ..., fn and copy them to `stdout`. Do not print any error messages from this (for example if some file does not exist or is a directory). Instead, any such error messages should be written to $f1$.

Hint: Put filenames in double-quotes in case they contain "funny characters".

Hints: `shift`, `$@`, `-lt`, `-a`.

6. (Script) Create a bash script called `datedlinecount` 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 first argument the following information:
     - The time and date
     - One line for each of the second-through-last arguments, containing the number of lines in the file and then the name of the file
     - One line with the total number of lines in all the files and then the word "total"

   For example, executing: `./datedlinecount log foo bar; ./datedlinecount 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 20:42:17 PDT 2007
 4 foo
 3 food
 7 total
```

   Hints: shift, date, wc, `$@`.

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 combine datedlinecount
```

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.