CSE 303, Spring 2005, Assignment 1
Due: Friday 8 April, 9:00AM

Last updated: March 29

You will get experience using the Linux tcsh shell, using emacs, and writing a short script. Be sure to look at the cryptic *Hints* section on the *next page*.

1. (Exploring commands) First run the `script` command. 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. (Exploring command-line editing) Note: Although this problem is very silly, it is helpful to learn how to modify the command-line without having to retype everything. It is also a fun logical puzzle.

   The goal is to have the following interaction with the shell:

   ```
   attu2% gramana
   gramana: Command not found.
   attu2% ragman
   ragman: Command not found.
   ```

   But what you type must obey the following rules:

   - Begin by typing `anagram`
   - Hold the `Ctrl` key down for everything you type afterward, except `Enter`.

   Using `emacs`, create a text file explaining your solution.

   Example: If the goal were to have the interaction:

   ```
   attu2% anag
   anag: Command not found.
   ```

   then a correct solution would be a file containing: "After typing anagram, press C-h C-h C-h Enter. This sequence deletes the last 3 characters and attempts to run the result."

   The sample solution includes 15 keystrokes, including two uses of `Enter`.

3. (Short Alias) Create `tcsh` *alias* `myhistory` that is like `history` except:

   - It prints the most recent commands first.
   - It does not print any command-line that contains the character sequence "history".

   Interesting question: You cannot write a shell script with this functionality: Why not?

4. (Short Script) Write a `tcsh` script `sendme` that takes one filename as an argument and sends the current user the contents of the file in an email.

   - The email's subject line should be `file:  `*`filename`* where *`filename`* is the name of the file.
   - `sendme` should print an appropriate error message if it is passed the wrong number of arguments or the file it is passed does not exist or is a directory.

   Cool fact: This script works on itself, i.e., `./sendme sendme` sends your solution to you in an email.

5. (**Extra Credit**) Write a `tcsh` script `sendme2` that takes any number of filenames as arguments and sends their content to the current user in an email.

   - The email's subject line should be `files:` *`filenames`* where *`filenames`* is the list of files in the email, but *without* their directories.
   - The body of the message should have the file's contents *in order*, i.e., the first file should be at the beginning of the message and so on. Before each file should be a line of the form `=====` *`filename`* `=====` (including a newline) where *`filename`* is the name of the file as entered on the command-line.
   - Extra Extra Credit: Change the previous requirement as follows: Before each file should be a line of the form `=====` *`filename`* `=====` (including a newline) where *`filename`* is the absolute path of the filename, preferably without any uses of "`./`" or "`../`". Warning: Correctly removing *all* such occurrences is remarkably difficult. You will need a loop, `grep`, and `sed`. You do *not* need to do this part to receive extra credit.

   Note: Remember the course policy on extra credit.

Hints: cat, control a, b, f, h, k, p, t, y, echo, grep, history, mail, tac, user
Extra credit hints: mktemp, rm, :gt

Assessment: Your solutions should be:

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

Turn-in Instructions

- For problem one, turn in the `typescript` file you created.
- For problem two, turn in a text file named `problem_two`.
- For problem three, turn in a file named `problem_three` that contains your alias.
- For problem four, turn in a file named `sendme` that is your shell script.
- For the extra credit, (optionally) turn a file named `sendme2`.
- Follow the link on the course website and follow the instructions to submit your files.