Reading and Reference

- Glass and Ables, Chapters 4, 5, and 8
- Classic Shell Scripting book (course web site)
- Mac OS X Developer Library (course web site)

Background

There is a utility called `factor` which is available on many Unix systems. For this assignment, you will develop a shell script that behaves in much the same way as this program. Developing this script will give you an opportunity to learn more about scripting using familiar control structures and numerical operations.

We will not be concerned with efficiency. You can, for example, use a simple "brute force" method for factoring: to factor an integer n, first remove all factors of 2, followed by factors of 3, and so forth, stopping when the original number has been reduced to 1.

Exercise 1. The `factor` program is not present on OS X, but it is available on `mathvnc.eiu.edu`, a departmental Linux server you have access to. Using `ssh`, login to this server and give the following commands, noting the response to each:

```
factor 24
factor -24
factor 24 59 82
factor 2a3 24 82
echo "24 59 82" | factor
factor
```

In the last example, notice that `factor` waits for interactive input. Provide some integer values; terminating with a control-D, as usual. In each case above, what is the value of the exit code? In addition to these examples, try some of your own.

Exercise 2. Adjust your `PATH` variable, appending your `bin` directory. To make this permanent, edit the file `.profile`, adding the following line:

```
PATH=$PATH:~/4970/bin
```

(If the file `.profile` doesn't exist, create it.) Close the terminal, then re-open it and inspect the value of `PATH` to ensure that it has the revised value. From a directory *outside* of your `bin` directory, enter the name of one of your scripts and verify it executes. From now on, you can add shell scripts and executables to your `bin` directory and they will appear to be "built-in" Unix commands.

Exercise 3. Develop a shell script named `factor` which behaves like its namesake. In particular, the following requirements must be met:

- Any number of command-line arguments may appear.
- If no command-line arguments appear, input should be read interactively from the standard input.
- All inputs should be validated: only positive integers should be factored; anything else is an error.
- Appropriate exit codes should be generated.

See the recommendations on the following page before starting this exercise.

Recommendations

Work incrementally. Rather than implementing the entire script at once, try to identify smaller subgoals that you can meet. Test your script as you go, adding more functionality along the way. When you aren't completely sure how to write a portion of the shell script, experiment at the command line. Insert `echo` commands throughout your script in order to verify intermediate results. (These can be removed when everything is working to your satisfaction.)

The following scripting capabilities will be useful:

- Comments
- Variable assignments
- Control structures: `while`, `for`, `if`
- Input with `read`
- The Unix `expr` and/or `bc` utilities
- Script functions (not essential, but handy)

What to Submit

Drag your completed `factor` script onto the EIU submit icon. Be sure that your name appears in the script as a comment somewhere near the top of the file.