CS 114 Unix Tools – Fall 2004

Assignment 4: mvsed

Handed out: Wed, 20 Oct
Due: Fri, 5 Nov, 5pm

This assignment asks you to write one shell script. There's tons of time to do this assignment, but don't procrastinate. I'll be out of town from 23 Oct to 31 Oct, so I'll be unavailable that week to answer questions in person; I'll still check email intermittently.

The task

Write a shell script, called `mvsed`, that takes at least two command-line arguments. The first argument is a `sed` expression. The remaining arguments are file or directory names.

The script renames all files and directories given on the command line by invoking `sed` with the expression on each name to produce the target filename. For example, to append `.bak` to several files, one can use `mvsed` as follows:

```
$ ls
file1 file2 dir3
$ mvsed 's/$/\.bak/' *
$ ls
file1.bak file2.bak dir3.bak
```

Details

1. Be careful that your script correctly handles filenames containing spaces and wildcard characters like "`*`".
2. The script should check that at least two arguments are given on the command line and output to `stderr` the single line:

   ```
   usage: mvsed sedexpr files...
   ```

   if they are not present.
3. The script should check that all files given on the command line exist and output:

   ```
   mvsed: FILE not found
   ```

   where `FILE` is replaced with the name of the file, if the file is not found. **Note:** to test for the existence of a file, "`[ -e` *file* `]`" will not work on Solaris; use "`[ -r` *file* `]`" or even "`[ -f` *file* `] || [ -d` *file* `]`" instead.
4. If `sed` returns an error, print that error to `stderr` (or just let `sed` do it), then output:

   ```
   mvsed: sed failed
   ```

   Hint: if using `sed` in a pipeline, get its exit status using the `$?` variable.
5. The script should not overwrite existing files or directories. If, after applying the `sed` expression to a path, the new path is the name of an existing file or directory, the script should output:

   ```
   mvsed: cannot rename OLD; NEW exists
   ```

   where `OLD` is replaced with the old path and `NEW` is replaced with the new path. The *exception* to this is when the new path is the same as the old path. In this case, no error should be reported. In either case, *do not rename* the file.
6. All error messages should be sent to `stderr` (hint: use `>&2`). The script may exit after an error, or may keep running.

Bonus

This part is optional, but may help your final grade for the course if you didn't do well on, or didn't turn in, one or more of the earlier assignments. Remember: this is an S/U course, so don't sweat over this if you've been doing well so far. Talk to me if you're you're concerned about how you're doing in the course.

1. Extend the `mvsed` script to take an optional argument "`-f`". When "`-f`" is given on the command line *before* the `sed` expression, `mvsed` should rename all files given on the command line, even if they exist. However, if the target path refers to a directory, the script should report an error, as in point 5 above.
2. Extend the `mvsed` script so that when it is invoked as `mvsed` (that is, when the base name of `$0` is `mvsed`) it behaves as above, but when it is invoked as `cpsed`, it should copy the files rather than renaming them. Error messages should be prefixed with "`cpsed:`" instead of "`mvsed`". Both scripts should have *identical* code. You can run "`ln mvsed cpsed`" to ensure this.

Submitting your assignment

Before the deadline, do the following:

1. Ensure your script is executable by you and can be invoked from the command line.
2. Make sure your script works. Create some files and test that they get renamed correctly and that errors are handled correctly.
3. With your script `mvsed` in the current directory, run the script `~cs114/bin/submit-hw4`. Within 24 hours (but probably sooner), you should receive an email acknowledgment that your assignment was submitted. If you do not receive an acknowledgment or if the submission script seems broken, email `cs114@cs.cornell.edu`.

Late submissions

No late submissions will be accepted. You should have plenty of time. Contact the instructor if you need more.

Debugging shell scripts

Here are a few suggestions to help you debug:

1. Start small. Get the basic `sed` processing and file renaming functionality working correctly. Then, add argument and error checking.
2. If you're not sure what a command does, just try it on the command line. If the command expects to read from `stdin`, use `echo` to give it some input.
3. To check if a variable `var` is set as you expect, insert `echo $var` into the script after `var` is set. Be sure to remove these debugging statements before submission.
4. To see what your script is doing as it runs, add `set -x` to the top of the script. This will cause `sh` to print out each command as it is run—with substitutions performed. You can turn this option off by adding `set +x` later. Again, be sure to remove these statements before submission.
5. Use `#` to comment out a statement.

Getting help

You are free to discuss this assignment with others in the class, but your work should be your own. In particular, copying other peoples answers (or portions thereof) is prohibited. If in doubt, err on the side of caution, or ask the instructor.

If you need help, you should:

1. Take a look at the man pages, lecture notes, and suggested reading.
2. If you have more than just a simple question, please consider coming to my office hours.
3. Go to `cornell.class.cs114` newsgroup and see if your question is already answered. If not, post it there. I will be happy to answer any questions posted to the newsgroup, not only homework-related ones.
4. If you want to keep your question private, e-mail me at `cs114@cs.cornell.edu`.