

## More about the shell

---

### Reading

Chapter 5 and 6 have more information on shell wildcards, job control, aliases, and pipes. There is also much material buried in the tcsh man page if you have the patience to read through it.

### Commands for today

We will go through more in-depth details about shell commands, use of wildcards, more sophisticated aliases, job control, and input/output pipes. Here is a listing of the material we will try to go through today. Check with your text or read up on the on-line manual pages for more information about any of these topics.

Shell metacharacters: \*, ?, [], preventing interpretation with \, ', and ", shell variable noglob

Inline-expansion: Ctrl-x \* for filenames, Ctrl-\$ for variables, ESC-space for history

Command-substitution using backquote

Aliases with arguments: !\*, !\$, !:1, using backslash to escape the !

Control flow: \$< to get user input, comparison operators, if/then/endif

Job control: jobs, &, fg, kill, ps

Input/output and pipes: <, >, >>, >!, |, shell variable noclobber

Miscellaneous: calendar, fortune, thought, yow, smiley, biff

### Exercise 4, due next Thursday

As in previous exercises, we mostly just want you to go on a free-form adventure trying out some of these new tools and figuring out how to put them to work to make your UNIX life more comfortable and fun. Try creating some aliases that are a little more sophisticated than last week's, that use arguments or do multiple things, for example. Set up some more customizations with these new tools. Get used to the commands to suspend, resume, and manage your jobs. Try stringing together some filter commands in a pipeline to build a more complex operation. You may not have an immediate use for all of the fancier commands, but try them out to get a sense of how they work and what their capabilities are.

Here are a few tasks you might want to put to yourself as possible problems to solve:

- Edit your .login so that each time you login the contents of your .project or .signature are replaced with a random saying from thought or yow or fortune (or output from the cs107 RSG!)
- Experiment with the shell wildcards to learn how to identify different sets of files using \* ? and [] so that you can easily group them for listing, moving, copying, deleting, etc. What does it take to list all files that have a .txt extension? To delete all files who have a digit somewhere in their name? To copy all files whose names begin with a capital letter to a new directory?
- Try creating and manipulating files that have literal wildcards as part of the filename (i.e. files with spaces or \* or [ in the filename). Experiment with the use of backslash or single/double

quotes to escape the special meaning in the shell. How can you create a file with a space in its name? How can you then copy or move it? Here's an especially tricky one-- how can you create a file whose name begins with a dash and how can you then delete it?

- Write an alias that takes files as arguments and will move those files to a trash directory that you can periodically empty on demand. Add other aliases to view the contents of the trash and empty it when desired. You may find this a safer way to remove files than using `rm`.
- Create a new alias called "cleanup" that will remove all auto-saved backup files (e.g. files whose names end with `~` or `#` as `.login~` or `#recipe#`) from the current directory
- Add an entry to your `.cshrc` that asks you whether to have mail notification active for this shell. You can control mail notification with the `biff` command. You may want to look through your `.login` file for similar examples that you can plagiarize for getting input from the user and doing a simple test on the value
- Try out using the job control features of the shell, running jobs in the background, suspending, resuming, and killing jobs, examining your list of jobs, etc.
- Create an alias that uses I/O redirection or pipes such as a "mailmorse" command that given a file and recipient will encode the file using the `morse` command send it the encoded version on to the recipient using the `mail` command.

For your exercise this week, you just need to provide a few examples of the interesting things you were able to accomplish with your new skills. They can be the solutions to a few of the above problems or just other tasks that you found useful to solve. These homework exercises aren't intended to be intensely time-consuming or stressful but because the only way to learn UNIX is by doing, we want to be sure you try things out and learn how they work so you can later use the commands on your own. As you did last week, type up your information in a short file and use the `/usr/class/cs1u/bin/submit` command to send it to us for grading by next Thursday's class. Happy exploring!