Slide 1

Administrivia

- Homework 1 was due Friday, but — two problems added, due date moved to next week.
- Readings updated to be more targeted.

Slide 2

Minute Essay From Last Lecture

- "What goes into writing a shell?"

  At a minimum, parsing command and arguments and starting a new program with those arguments. Can be quite simple — first homework in O/S asks students to write one (or fill in blanks anyway).

  Beyond that . . . Possibly a lot! consider the features discussed so far. Enhancing the simple-minded O/S-class shell can be a fun project.

Slide 3

Processes and Job Control, Revisited

- "The" shell (okay, there are several, but all that I know of) starts a new process for each command. Normally runs "in the foreground" (of the login session).
- Or you can start it "in the background" by putting a `&` after the command. You can also suspend the foreground process with ctrl-Z. (Useful if you want to get back to a command prompt.) Restart a suspended process with ctrl-Z, or put it in the background with `bg`.
- Background and suspended processes get a number, which you can show with `jobs`. You can use this number with `fg`, `bg`, or `kill`.

Slide 4

Shell Customizations

- At startup, shell reads in various configuration files (see `man` page for details). At least one will be in your home directory. For `bash`, `.bashrc` is read for all shells and `.bash_profile` when it's a "login shell" (e.g., `ssh` session, but not terminal window).
- In these files, you can
  - Define/redefine environment variables. (e.g., `PATH`, `PS1`). For `bash`, be sure to `export` them. Can define new ones (I find this useful).
  - Set various shell options and variables.
  - Define aliases/functions.
  - Invoke other commands (e.g., `umask` to set default file permissions, or `module load` (later).
- Caution: The default setup on our lab machines is somewhat elaborate. Originally designed to support diverse UNIX-like enviroments (Linux, Mac

Slide 5

OS X, etc.) and still in use although currently only needs to support Linux. Look at `~defaults/system/SYSTEM.bashrc` for details.

Slide 6

Environment Variables

- Some we've mentioned already — e.g, `PATH`. Others we haven't (e.g., `PS1`).
- For `bash`, be sure to `export` them so they're available to called programs.
- Can also define new ones (I find this useful).

Slide 7

Shell Options and Variables

- `set` and `shopt` let you set various shell variables and options.
- Details in `man` page or manual, but some I find useful:

```
set -o noclobber
set -o ignoreeof
shopt -s histappend
```

Slide 8

Shell Aliases and Functions (`bash`)

- Aliases are simple substitution, no parameters. E.g.

```
alias lt='ls -ltF'
alias google='lynx http://www.google.com'
```

- Functions can have positional parameters. E.g.,

```
function cd-and-show() { cd $1 ; pwd ; ls; }
```

Slide 9

Another `bash` Feature — Directory Stack

- `bash` maintains a stack of directories. Use commands `pushd`, `popd`, `dirs` to manipulate it.
- Very useful (I think!) if you want to navigate from one deeply-nested subdirectory to another without losing your place.

Slide 10

I/O Redirection

- In programming classes I talk about "reading from standard input" (`stdin`) rather than "reading from the keyboard", and "writing to standard output" rather than "writing to the screen". Why?

Slide 11

I/O Redirection, Continued

- `stdin` (standard input) can come from keyboard, file, or inline in shell script.
- `stdout` and `stderr` (standard output, error) can go to terminal or file (overwrite or append), separately or together. (Syntax depends in part on which shell you're using.)
- How is this useful? (e.g., in program development? testing?)
- *OR* — remember quotation from first class?

  "Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."

Slide 12

Pipes

- "Pipes" provide one-way communication between programs — output of program A becomes input of program B.
- Key component of "the UNIX philosophy" — emphasis on providing a toolkit of small programs, mechanisms for combining them.
- "Filters" are programs designed to work this way, and there are lots of them (next time). `less` and `more` also useful.

Slide 13

Minute Essay

- What (if anything) has been noteworthy about the first homework?