Slide 1

Administrivia

- Reminder: Homework 3 due Monday.

Slide 2

Minute Essay From Last Lecture

- Most people were fairly happy with the pace and workload. Seems to vary depending on background — no surprise. I'll probably continue as I have been, but if you're in the "this is too fast" group, feel free to interrupt me or ask questions outside class.
- One person sent me one more way to find broken links:

```
find $(pwd) -type l | file -f - | \
        grep "broken symbolic link" | awk { print $6 } | less
```

Minute Essay From Last Lecture, Continued

Slide 3

- One person asked about something to check for errors in shell scripts. I didn't find anything that would avoid the problem I had in class (using a variable not previously defined), and indeed I don't know that it's possible — could be defined externally — but I found a program called `shellcheck` (now installed on the `dias` cluster) . . .

Things I Learned from `ShellCheck`

Slide 4

- This program complained about a lot of things I do in scripts. Some of them surprised me. I checked some of the complaints against the POSIX definition for shell command languages and discovered . . .
- You don't need `function` to define a function, and indeed it's nonstandard.
- Command substitution with backquotes is considered more or less obsolete; the newer syntax I thought was specific to `bash` is standard in all but the oldest shells.
- The syntax for arithmetic expansion that I thought was specific to `bash` is also apparently standard.
- I've revised (most of) my notes and examples accordingly.

Slide 5

“Here” Documents

- We talked about redirecting input and output. One more option for input, useful in scripts, is to get it from the script itself — “here” document. Example:

```
#!/bin/sh
# simple example of sending mail from command line
mutt -s "a subject" bmassing << EOF
hello
I am here
who are you?
is this fun?
EOF
```

Slide 6

Return Codes, Revisited

- As previously mentioned: Conditional execution and `while` loops use as their test a command; if it returns zero, then the condition is true, else false.
- Shell variable `$?` holds the return code from of the last command, if you want to examine it more directly.

Slide 7

Other Useful Things

- Shell option `-x` can be helpful in debugging (`set -x` in script, or `bash -x myscript`).
- `getopt` — process command-line options (to help you write scripts that accept options in any order, in the same way most UNIX commands do).
- Remember `pushd` and `popd`, for temporarily changing to another directory and coming back.
- Sometimes you want to just discard some output of a command; you can do this by redirecting to `/dev/null`.
- Can chain commands with `;` to do in sequence, or with `&&` to only run second if first succeeds, or with `||` to only run second if first fails.

Slide 8

Shells and Subshells

- In a typical shell, each command or shell script runs as a separate process. (Why? Consider what you want to happen if the command crashes.)
- One result of this is that commands and scripts can't generally make changes in the shell (e.g., setting environment variables, or changing the current directory).
- This is why, e.g., some functions such as `cd` are shell built-ins rather than commands. (Try `man cd`.) (Yes, there's also a `/usr/bin/cd`, but its purpose seems a little obscure).
- But you *can* use scripts to make such changes, if instead of executing them you "source" them (with `source` or `.`): This executes the commands in the script in the current shell.

Shells and Subshells, Continued

- Can group sequence of commands with parentheses to run in subshell. Why? could be useful to set environment variables local to the subshell.
- Can group sequence of commands with curly braces. Why? e.g., to run a sequence of commands in the background.

Slide 9

Shell Scripts — “the Ugly”, Revisited

- If variables are set in a subshell their values disappear when it exits. An example is piping something into a `while read` loop.
- How to fix? simplest way is just to find an alternative to piping (“here” documents, maybe, or other input redirection).

  The Advanced Scripting Guide has more about this in section 34.

Slide 10

Minute Essay

- Anything else we should discuss about shell scripts?