Slide 1

Administrivia

- Reminder: Homework 2 due today.
- Homework 3 on the Web. Programming problems, so turn in by e-mail. If you're working remotely you may find the "mail files" script on the "sample programs" page useful.

Slide 2

Minute Essay From Last Lecture

- The question was poorly chosen (sorry about that), but many people came fairly close.

Slide 3

Shell Scripts — Recap

- What you type at the command prompt is statements in a programming language — with variables, conditional execution, loops, and functions; can also collect these into files ("shell scripts") — much like Scala's REPL and scripts.
- (Briefly review examples from last time.)

Slide 4

More Examples

- Rename all `.htm` files in the current directory to `.html` (`-v` isn't really necessary but does show you what's being done):

```
for f in `ls *.htm`
do
  mv -v $f $(basename .htm).html
done
```

(But this fails if names contain spaces. See `rename-files` example.)

Slide 5

More Examples

- Descend into each of several subdirectories and launch a subshell (`exit` to move on):

```
for d in d1 d2
do
  pushd $d ; pwd ; ls ; bash ; popd
done
```

- (`find-broken-links-1` example. But this also does not cope well with names with spaces.)

Slide 6

Arithmetic

- Shell supports simple *integer* arithmetic.

  Most basic/portable way probably `expr`. Example:

  `n=`` `expr $n + 1` ``.

  In `bash`, can also use double parentheses. Example:

  `n=$((n + 1))`.

  `factorial-1`, `factorial-2` examples.

- But if you're doing significant calculations, you should probably be using some other tool — `awk`, `bc`, `dc`, or a program in a "real" programming language.

Slide 7

dc and bc

- Both are simple text-mode calculator programs. dc uses reverse Polish notation, bc the more familiar algebraic notion.
- Both are "arbitrary-precision", which can be useful. Both support non-integer values, but how to set "precision" can be tricky. Details in their man pages.
- Used interactively, bc may be more useful, since you can use variables within it.
- Both are useful in shell scripting, e.g.,

```
echo "2 + 3" | bc
echo "2^10" | bc
```

- (powers-of-two example.)

Slide 8

Reading from Standard Input

- To read from shell's / script's standard input: read.
- Example:

```
echo "Do you really want to do this?  (y/n)"
read ans
if [ ".$ans" = ".y" ] ....
```

(Why the dots? if nothing is read, $ans may be empty, with possibly awkward results. May be okay to omit, but a lot of shell scripts use them.)

- Also useful as a way of coping with names with spaces. (rename-files, find-broken-links-2 examples.)

Slide 9

Minute Essay

- How's the pace of the class so far? Homeworks about the right amount of outside-class time, too much, too little ("as if"?) ?