Slide 1

Administrivia

- Reminder: Homework 4 due today.
- Homework 3 grades mailed this afternoon. Sample solution online.
- As in e-mail — more time on Homework 5. Once a week is probably about right for homework, and we won't move through topics quite as fast.

Slide 2

Minute Essay From Last Lecture

- Most people came fairly close on the regular-expression question.
- A couple of people mentioned liking having two choices on problems (on Homework 3).

Slide 3

Text Editors Revisited

- Some text editors (`vim` among them) allow you to "filter" text through an external program.
- One thing this allows is building on-the-fly scripts — construct in `vim` the lines to execute, then execute them with, e.g., `:%!sh`. (No need to save unless you want to reuse another time.)

Slide 4

On-the-fly Scripts, Continued

- I like this "on-the-fly scripting" for various kinds of file moving/renaming operations — use `r!ls` to get a list of files, "massage" with various editing operations, then execute as above. I find this works well as a way of dealing with filenames containing spaces — relatively easy to add double quotes around names. A useful idiom employs a simple regex and `&` to reference the matched text, e.g.,

  ```
  :%s/.*/mv -v "&" targetdir/
  ```

- (Of course I could also use a `bash` loop, and sometimes I do, but — whatever seems easiest for the particular use case?)

Slide 5

Text Editors Revisited, Continued

- I also use the ability to execute ranges of lines from within `vim` in combination with programs/scripts that do arithmetic — for example, last step in grading assignments is to total points and record, and if you keep scores as text files (as I do) . . . (A little more about this later.)
- (Here too I could do this other ways! A traditional maxim about the scripting language Perl is "there's more than one way to do it" (TMTOWTDI), and — true for the toolkit being discussed in this class as well?)

Slide 6

Text Editors Revisited, Continued

- I also use `vim`'s ability to record and play back "macros" fairly regularly. To do this (on purpose): Start recording with `q` plus a single letter. End with another `q`. Play back with `@` and the single letter.

  (Somewhere sometime I think I remember a comment to the effect that with regard to certain repetitive tasks there were two kinds of people — the ones who write macros and the ones who write a regular expression. I do both, depending on the situation.)
- Examples another time maybe, but as a general comment — it can be tricky to record in a way that will "play back" effectively, but when this works, it works well.

Slide 7

Regular Expressions Revisited

- Regular expressions can be complex — constructs include “character classes”, repetition, “or”, and back references.
- Can be very powerful but also very cryptic.

Slide 8

Regular Expressions — Examples

- As an example, consider taking a “class roster” as produced by TigerPaws (showing student name, ID number, e-mail address, etc.,) and extracting from it just the student’s last name and e-mail address. Here’s a `vim` command to do that:

  ```
  :%s/\(.\+\),.*\d\d\d\d\d\d\d\s\(\S\+\)\s.*/\2 \1/
  ```

  (Admittedly it did take a few tries in class to get right!)
- As another example, consider revising that little script that computes factorials using a recursive shell function. Really would be nice if it rejected invalid input (and as we discovered in class, more than “nice” — it seems to “fork bomb” the computer!).

  (Revised script next time?)

Slide 9

Regular Expressions — Another Example

- One example from my own quirky ways of doing things:
- I use the old program `procmail` to filter incoming e-mail into "folders" based on a variety of criteria, including header markup added by `spamassassin`. (More about it later maybe.) It generates a log describing what it does. I'm sometimes interested in, e.g., how many of the incoming messages were identified as spam.
- So I can massage a copy of the log to get this information — extract the "folder" lines, strip out unneeded fields, then use `sort` and `uniq -c` to get the result I want. (I actually do this in `vim` but really I ought to write a script!)

  (Details next time?)

Slide 10

Regular Expressions — Yet Another Example

- Another example from my own quirky ways of doing things:
- When I grade programs I make a text file for each student with point deductions/additions and comments.
- How to total up these deductions/additions to compute score? A way that occurs to me is to use regular expressions to pick out the lines +num and -num and then construct an expression to pass to `bc`.

  I wrote a script for this that I can invoke from within `vim` on a range of lines. Quirky? Probably, but I like it!

  (Details next time?)

Slide 11

Minute Essay

- I've described some of the ways I use some of the tools discussed. How about you — anything we've talked about thus far that you've been able to put to use to do something that helps you?