Slide 1

Administrivia

- Homework 7 (makefiles) on Web. Due next Monday.
- My CSCI 3291 next fall will be Unix system administration. (By request of Dr. Pitts!) Required background is what you've (presumably) learned in this course.

Slide 2

Minute Essay From Last Lecture

- Question: Anything else you want to do from the command-line?
- Many answers — some outside the scope of the course. One request was for regular expressions — today. Several requests for mail — next time. Also as many others as we have time for. I'll probably also talk a little about X, window managers, etc.

Slide 3

Commands, Continued from Last Week

- (See notes from last time also.)
- `script` is still (maybe) useful, but use `cat` or `more` to view results. Can also use as a crude "view what another user is doing" by combining `script -f` and `tail -f`.

Slide 4

Regular Expressions

- Definition from Wikipedia:

  > A regular expression (abbreviated as regexp or regex) is a string that describes a whole set of strings, according to certain syntax rules. These expressions are used by many text editors and utilities (especially in the Unix operating system) to search bodies of text for certain patterns and, for example, replace the found strings with a certain other string.

- Idea has roots in formal theory of languages, where the "languages" (sets of strings) described by regular expressions are exactly the ones accepted by finite state automata.

Slide 5

Regular Expressions and Unix Tools

- Tools that use regular expressions include editors and also text-manipulation commands such as `grep` and `sed`. Also supported in many programming languages, especially ones for scripting (Perl, Python, `bash`, etc.).
- This being Unix, not all the tools accept exactly the same syntax. POSIX defines two standards, "basic" and "extended". Some tools/languages add more. Simple stuff is very similar in all versions, fortunately. Key difference — in basic syntax, must precede many special characters with "escape character" (backslash).

Slide 6

Character Literals and Metacharacters

- Most characters represent themselves.

  `hello` matches what?
- Other characters are "special" (metacharacters):

  `^` matches start of line

  `$` matches end of line

  `.` matches any character (except newline)

  To use these as regular character literals, "escape" with a backslash.

  Example: `\.5`

Slide 7

Character Classes

- Character classes represent “one of these characters”.

  Examples: `[abcd]`, `[0-9]`
- `^` at the start of a list means “any character other than these”:

  Example: `[^abcd]`
- Most tools define some shorthand:

  Example: `\s` for whitespace

  Example: `[:alpha:]` for letter

  Example of use: `[^[:print:]]`

Slide 8

“OR” (Alternation)

- Unix pipe symbol (`|`) separates alternatives. (Must escape in basic syntax.)

  Example: `cat|dog`
- (What about AND? In many contexts, doesn’t mean anything anyway. For `grep`, pipe one `grep` into another.)

Slide 9

Quantifiers

- `*` means "preceding character (or group), zero or more times".

  Example: `.*`
- `+` means "preceding character/group, one or more times". (Must escape in basic syntax.)

  Example: `a+`
- `{N,M}` means "preceding character/group, N to M times". (Must escape curly brackets in basic syntax.)
- Notice that quantifiers are "greedy" — match longest string possible.

Slide 10

Grouping in Regular Expressions

- Use parentheses to group. (Must escape them in basic syntax.)

  Example: `(abc)(def)`

  Example: `(abc)*`
- Can then "backreference" groups, with `\1`, `\2`, etc.

  Example: `(abc)(.*)\1`

Slide 11

A Few More Tricks

- Angle brackets match beginning/end of word. (Must escape in basic syntax.)

  Example: `<hello>`

Slide 12

Usage of Regular Expressions, Revisited

- Can use regular expression to search — `grep`, search in `vi`.
- Can also use them to modify — `sed`, search-and-replace in `vi`.

  Backreferences can be useful here!

  Example: `s/\(^..\)\(.*\)/\2\1`

Slide 13

Where to Learn More

- `man` and/or `info` pages for `sed`, `grep`.
- Online help for `vim`.
- Books and online references/tutorials . . .
- Useful advice from `vim`'s help:

  Which of these should you use? Whichever one you can remember.

Slide 14

Minute Essay

- Try writing a regular expression that would match a "license plate" string of the form "one uppercase letter, then two digits, then three uppercase letters".