Unix

Why study Unix?

- Contrast with insulating point-and-click OSs, like Windows
  - Understand and manage your own environment
- See a different kind of programming than Java or C programming
- See how simple (and sophisticated) tools can be combined to get interesting effects
  - pipes
  - scripts

Unix is widely available

- Machines running Linux (and other Unix variants)
  - E.g. attu is "the" instruction Linux server
- From Windows machines:
  - Can remotely log in to instructional Linux machines, e.g. using Ssh Secure Shell
  - Can install Cygwin!
    - (How can you find out about Cygwin?)

Let's try it!

- See handout for core Unix concepts and commands

An advanced command: grep

- `grep` *regularExpression fileName*...
  - search the named file(s) for all lines that match (anywhere) the given *regular expression*, and print them out
  - `egrep`, `fgrep` are variations that have slightly different regexp languages
  - `grep -v` *regexp fileName*...
    - prints lines that *don't* match
- Regular expressions are like filename patterns, but more powerful
  - Several Unix commands have similar regular expression sublanguages, so good to know

Regular expressions

- Like filename patterns, except different special characters
- `.` matches any character (like `?`)
- *re*`*` matches zero or more occurrences of the *previous regular expression re*
  - can use `\(...\)` to bracket a regexp to repeat
  - `.*` regexp is same as `*` filename pattern
- (What does `a\(b.c\)*d` match?)

More regular expressions

- [...] notation is similar to filename meaning
  - But also have [^...] to match anything *except* [...]
- \(*re1*\|*re2*\|...\) is similar to filename set patterns
- *c* matches *c*
  - \ disables any special meaning of *c*

Matching start or end of line

- ^ at the front of a regexp means that the regexp must start matching at the start of a line
- $ at the end ... at the end of a line

Another adv. command: sed

- sed –e *command fileName*...
  - sed can be used to perform edits to the input file(s), printing out the result
  - *command* is a special sed command
    - can have as many –e *command* arguments as desired
    - can omit –e if only one command
- lots of possible script commands
  - [how to find out?]
  - we'll look at one: the s command

String replace using sed

- sed 's/*re*/*replacement*/g' *fileName*
  - finds all occurrences of phrases matching regexp *re* in input *fileName*
  - replaces each with *replacement*
  - if leave g off, then only replace first match
  - / can be any character

Bound substrings

- Can remember parts of phrase matching *re*, reuse them in *replacement*
  - & refers to whole matched phrase
  - \1 ... \9 refer to corresponding matching subphrases inside parentheses