CS114: Lecture 6
sed & awk

HW2 due tonight

No lecture Monday (Fall Break)

Redirection

- `echo something > outputfile`
  - If `outputfile` existed before, old file is deleted!
  - New file created; contents =
    `something`
- `echo anything >> outputfile`
  - `anything` appended to outputfile
  - outputfile is now
    `something`
    `anything`
- `tr 'a-z' 'A-Z' < inputfile`
  - `inputfile` needs to exist

Shell variables

- When do variables need $ and when not?
  - `$foo` -> replaced by shell with value of `foo` variable
    - `echo "My favorite color is $foo"`
    - `ls $PWD/bar`
  - `set foo = ... / setenv foo ...`
- Shell variables aren't declared
  - When you set the value, it doesn't matter whether or not had value before

Automatic text processing

- What if I want to ...?
  - Strip directory prefixes from paths
  - Print column 2, 4, and 7 of a file
  - Remove comments from a shell script
  - Convert from a DOS to UNIX text file
- Answer: write a Java program
  - NO!
  - sed, awk (or perl, python, ...)
  - Right tool for the right job.

sed (Stream Editor)

- `sed 's/`*`regex`*`/`*`text`*`/' `*`file`*
- `echo “roses are red” > poem`
- `sed 's/red/blue/' poem`
  - `roses are blue`
- `echo “roses are red” | sed 's/red/blue/'`
  - `roses are blue`

More useful sed examples

- Strip directory prefixes from paths
  - `sed 's/.*\///'`
- Convert from DOS to UNIX text file
  - `sed 's/^M//'`
    - This is ***not*** ^ followed by M; you press Ctrl-V then Ctrl-M
- Make a MANPATH
  - `setenv MANPATH `echo $PATH | sed 's/bin/man/g'``

A sed script

- Any text file that begins with #! is a script
- `cat trim.sed`

```
#!/usr/bin/sed -f
s/^ *//
s/ *$//
```

- `echo "   lots of extra space   " | trim.sed`
  - `lots of extra space`

awk

- Actually a programming language
- Oriented towards database-like text files

- Print the second and fourth columns
  - `echo “This is a test” | awk '{print $2, $4}'`
    - `is test`

awk: guards

- `awk '`*`guard {command}`*`'`
- Print third column of lines containing 'blue'
  - `awk '/blue/ {print $3}'`
- Print lines where second column is “red”
  - `awk '$2==”red” {print}'`
- Print lines between “#START” and “#FINISH”
  - `awk '/#START/,/#FINISH/ {print}'`

awk: variables

- `$0` – entire current line
- `$1, $2, $3`, ... - Field 1, field 2, field 3
- `NF` = number of fields in current record
- `FS` = field separator
- `foo` – user-declared variable `foo`

awk: special guards

- Sum second column
  - awk '{sum += $1} END {print sum}'
- END happens once at the end
- BEGIN happens once at the beginning

Other scripting languages

- **Python**
- **Perl**
- Tcl
- Ruby
- Embedded scripting languages
  - PHP (server-side webpages)
  - Lua (e.g. World of Warcraft)
  - JavaScript (client-side webpages)
  - Lisp (Emacs)