CSE 374: Programming Concepts and Tools

Eric Mullen
Spring 2017
Lecture 7: sed

Administrivia

- Homework 0: graded and returned
- Homework 1: being graded as we speak
- Homework 2: due Thursday at midnight
- Homework 3: out on Friday
- ssh keys: working?
- No office hours at 11am tomorrow

Where we're at

- Regular expressions are powerful for finding strings
- Output of `egrep` is subset of its input
    - Use -v to invert pattern: print everything that doesn't match
- What if you want to manipulate strings instead?

sed

- **S**tream **ED**itor
- Terrible little language for editing streams of strings
- single most common use:

```
sed -E -e 's/pattern/replacement/g' file
```

- For each line in the file, replace every occurrence of the pattern with the replacement, and print result to stdout
- Many options, you should look them up AND try them out
- `-E` allows us to use extended syntax, just like `egrep`

sed

- There is so much more you can do with `sed`
- If you find yourself trying, ask yourself if you should
- Generally if it's hard to write, it's harder to debug, and this is very true with `sed`

Newlines

- sed doesn't match newlines: they're removed before processing and added back before printing
- It's hard but possible to do multiple line things with sed. Again, not *can I?* but *should I?*
- Newlines are a pain across operating systems

Typewriters

- You can trace newline woes back to typewriters
- Carriage Return (CR): \r
    - would return carriage to beginning of line
- Line Feed (LF): \n
    - would advance paper by 1 line

Newlines

- How should modern Operating Systems represent newlines?
  - CR 
  - LF 
  - CR+LF 
  - LF+CR Thankfully nothing since the 80s

We’re done with Bash

- Any questions?