Regular Expressions

Devin J. Pohly <djpohly@cse.psu.edu>
Regular expressions

- Often shortened to “regex” or “regexp”
- Regular expressions are a language for matching patterns
  - Super-powerful find and replace tool
  - Can be used on the CLI, in shell scripts, as a text editor feature, or as part of a program
What are they good for?

• Searching for specifically formatted text
  ‣ Email address
  ‣ Phone number
  ‣ Anything that follows a pattern

• Validating input
  ‣ Same idea

• Powerful find-and-replace
  ‣ E.g. change “X and Y” to “Y and X” for any X, Y
Regex “flavors”

• Many languages support regular expressions
  ‣ Perl
  ‣ JavaScript
  ‣ Python
  ‣ PHP
  ‣ Java, Ruby, .NET, etc.

• Today we will be learn standard Unix “extended regular expressions”
On the command line

• The `grep` command is a regex filter
  ‣ That’s what the “re” in the middle stands for
  ‣ We have seen `fgrep`, which looks for literal (“fixed”) strings

• Today we will use `egrep`
  ‣ E for “extended” regular expressions
  ‣ Very close to other languages’ flavors
grep command syntax

• To find matches in files:
  `egrep regex file(s)`

• To filter standard input:
  `egrep regex`
  ‣ where regex is a regular expression, and file(s) are the files to search

• Options:
  ‣ `-i`: ignore case
  ‣ `-v`: find non-matching lines
  ‣ `-r`: search entire directories
  ‣ `man grep` for more
Okay, let’s begin!

```
$ cd /usr/share/dict
$ egrep hello words
...
$ cat words | egrep hello
...`

First lesson

• Letters, numbers, and a few other things match literally
  ‣ Find all the words that match “fgh”
  ‣ Find all the words that match “lmn”

• Note: a regex can match anywhere in the string
  ‣ Doesn’t have to match the whole string
Anchors

• Caret ^ matches at the beginning of a line
• Dollar sign $ matches at the end of a line
  ‣ Use '...' to protect special characters from the shell!
• Try it
  ‣ Find words ending in “gry”
  ‣ Find words starting with “ah”
• What happens if we use both ^ and $? 
Single-character wildcard

- Dot . matches any single character (exactly one)
  - Find a 6-letter word where the second, fourth, and sixth letters are “o”
  - Find any words that have at least 22 characters
Multi-character wildcard

- Dot-star \.* will match 0 or more characters
  - We’ll see why on the next slide
  - Find all the words that contain a, e, i, o, u in that order (with anything in between)
  - How about u, o, i, e, a?
Quantifiers

• How many repetitions of the previous thing to match?
  ‣ Star *: 0 or more
  ‣ Plus +: at least 1
  ‣ Question mark ?: 0 or 1 (i.e., optional)

• Try it out
  ‣ Spell check: necc?ess?ary
  ‣ Global awareness: colou?r
  ‣ Find words with u, o, i, e, a in that order and at least one letter in between each
Careful!

• Guess before you try: what happens if you search for $z^*$?

• Now try searching for the empty string
  ‣ Use '' to give the shell an empty argument

• Conclusion: make sure your regex always tries to match something!
Character classes

• Square brackets \([ \text{abc} ]\) will match any one of the enclosed characters
  ‣ What will \([ \text{chs} ]\)andy match?
  ‣ You can use quantifiers on character classes
    ▪ Find words starting with b where all the rest of the letters are s, a, or n
    ▪ Find all the words you can type with only ASDFJKL
    ▪ Find all the words you can type with AOEUHTNS!
Ranges

• Part of character classes

• You can specify a range of characters with \[ \text{a-j} \]
  ‣ One hex digit: \[ \text{0-9a-f} \]
  ‣ Consonants: \[ \text{b-df-hj-np-tv-z} \]
  ‣ Find all the words you can make with A through E
  ▪ … that are at least 5 letters long (hint: pipe the output to another egrep!)
Negative character classes

• If the first character is a caret, matches anything except these characters
  ‣ Consonants: [^aeiou]
    ▪ Not quite – why?
  ‣ Find words that contain a q, followed by something other than u
  ‣ Can be combined with ranges
    ▪ Any character that isn’t a digit: [^\d]
Groups

- Parentheses (... ) create groups within a regex
  - Quantifiers operate on the entire group
  - Find words with an m, followed by “ach” one or more times, followed by e
  - Find words where every other character, starting with the first, is an e
Branches

• The pipe | denotes that either the left or right side matches
  ‣ It’s the “or” operator
  ‣ Useful inside parentheses

• Guess before you try:
  ‣ book(worm|end)
  ‣ ^(out|lay)+$
Special characters

• We’ve seen a lot already
  ‣ ^$.*+?[ ]()\`

• Backslash \ will escape a special character to search for it literally
  ‣ For example, you could search your code for the expression ‘int \*' to find integer pointers
  ‣ What is the difference between the two *s?
Backreferences

• Groups in () can be referred to later
  ‣ Must match exactly the same characters again
  ‣ Numbered \1, \2, \3 from the start of the regex
  ‣ Try it: (....)to\1
  ‣ Find words that have a four-character sequence repeated immediately
Substituting – a demo

• The **sed** program has a lot of functions for modifying text

• Most useful is the *s///g* ("*substitute"*) command: regular expression find-and-replace
  
  ‣ Also available in Vim by typing :

  ```
  %s/regex/replacement/g
  ```

• Try it: run this command and type things

```
$ sed -r 's/([a-z]+) and ([a-z]+)/\2 and \1/g'
```
Enjoy puzzles?

- regexcrossword.com
  - Great way to practice your regex-fu
  - Starts with simpler tutorial puzzles and works up