CSE 15L, Hacker edition

or, The Joy of Regex

Matt Chan

April 29, 2016

Abstract

This material was developed for a workshop on regex and command line tricks, hosted by Eve Security on April 29, 2016. It aims to demystify the command line and teach common poweruser shortcuts. Each section explains a tool or concept, illustrated with exercises. (Note to presenters: work through exercises, referring to notes to explain concepts)

1 Regular Expressions

or, how to use find-replace properly.

(xkcd 208)

1.1 Math

A regular expression R is defined by the grammar

$$
\begin{array}{rr}
R \coloneqq \{a \mid a \in \Sigma\} & \text{chars in alphabet} \\
\mid \varepsilon & \{\texttt{""}\} \\
\mid \emptyset & \{\} \\
\mid R \cup R & \text{or} \\
\mid R \circ R & \text{concat} \\
\mid R^* & \text{Kleene's star}
\end{array}
$$

Some mathematical syntactic sugar:

$$
\begin{aligned}
\{R_1, R_2, \ldots\} &\rightsquigarrow R_1 \cup R_2 \cup \ldots \\
R^+ &\rightsquigarrow R \circ R^* \\
R^? &\rightsquigarrow R \cup \varepsilon \\
\{R_1, R_2, \ldots\}^c &\rightsquigarrow \Sigma \setminus \{R_1, R_2, \ldots\} \\
R^n &\rightsquigarrow \underbrace{R \circ R \circ \ldots \circ R}_{n \text{ times}} \\
R^{\{n,m\}} &\rightsquigarrow R^n \cup R^{n+1} \cup \ldots \cup R^m
\end{aligned}
$$

What are the equivalents in code?

1.2 POSIX regex

- `abc` is equivalent to $a \cup b \cup c$
- `* + ?` are the same as you'd expect
- `|` is 'OR'
- `.` is any char (except newline)
- `\{n,m\}` is n reps to m reps, leave m blank to leave the upper bound unspecified
- `^` marks the beginning and `$` marks the end of a line
- `\<char>` escapes a character.
  - `\t \n \r` are tab, linefeed, and carriage return respectively
- `\(...\)` captures a group (have to escape the parens in Emacs). Then you can refer to the group in the replace expr by `\<N>` (1-indexed)
- Boundary: `\b<char*>`. Does not consume the marker. TODO TODO TODO
- Classes:

- `[...]` matches anything in a *class* where `...` is a string of chars e.g. `abcd`, and the class is equiv to `a|b|c|d`. Classes CANNOT contain expressions that are strings (concatenations).
- `[^...]` matches anything NOT in the class, i.e.
- `[<begin>-<end>]` denotes a range, e.g. `[0-9]` or `[a-z]`
- `\w \d \s` are the classes: word `[0-9a-zA-Z]`, digit `[0-9]`, whitespace
- `\W \D \S` are the complements of those classes
- More classes:
  * https://en.wikipedia.org/wiki/Regular_expression#Character_classes
  * https://www.emacswiki.org/emacs/RegularExpression#toc1

1.2.1 POSIX Basic vs POSIX Extended

- The above is written in terms of POSIX Basic.
- NOTE: have to escape `{} () + |` in POSIX Basic regex but not POSIX Extended regex

https://www.emacswiki.org/emacs/RegularExpression

2 Tools that harness the power of Regular Expressions

2.1 sed

2.2 awk

2.3 find

2.4 grep

2.5 Emacs / vim

3 Shell commands, shell scripting

The shell is full of surprises and shortcuts designed to save you time. For a full listing, lookup `man builtin`.

3.1 `coreutils`: the standard suite of little programs

Composing little programs (all in GNU coreutils, `info coreutils`)

- mv
- cp
- cat
- rev
- more/less
- echo

- cut
- wc
- uniq
- seq
- sort

3.2 Shell scripting

- Redirection: standard in/out

4 Other tools worth a mention

4.1 processes: ps, killall, kill

4.2 make

4.3 wget

1. Download all lectures from http://people.orie.cornell.edu/shmoys/or630/#handouts

   Command: =wget -4 -nH –cut-dirs=3 -r –no-parent -l1 "${l}"=
   Problem: plain 'ol wget doesn't work (redirect error)
   look @ html, extract all links with emacs / sed

```
cat lecs | while read l do
  wget -4 -nH --cut-dirs=3 -r --no-parent -l1 "${l}"
done
```

4.4 scp and rsync

4.5 "Advanced" plain text editing

- Org-mode
- Markdown
- Pandoc
- Jekyll, static site generators

5 Exercises

See `exercises/README.md` — most problems here are duplicated there.

- Do a few rounds of Regex crossword
- Find all files with phone numbers
- Find words with 20 letters or more

- Find all words in English that have x as their second letter, and n as their second-to-last.
- Replace all spaces in a filename with underscore for a given directory.
- Reformat dates from MM-DD-YYYY to DD-MM-YYYY
- Update all copyrights from `Copyright (c) 2015` to `Copyright (c) 2016` in a directory of files.
- Verify that a password contains at least one upper case char, one lower case char, a digit, a symbol, and is > 8 chars long.
- Find consecutive identical words, delete all but one, i.e. `foo foo foo bar` $\rightsquigarrow$ `foo bar`
- Delete trailing whitespace
  - PSA: please set your editor to do this automatically.
- Uglify a source file by compressing it into a single line, removing as much whitespace as you can.
- Pad numbers in a seq of files, i.e. `chap1.pdf chap2.pdf ... chap14.pdf chap15.pdf` $\rightsquigarrow$ `chap01.pdf chap02.pdf ... chap14.pdf chap15.pdf`
- Replace all C99 comments with C89 comments

```
M-x replace-regexp RET //\(.*\)$ RET /* \1 */
```

- Extract contents enclosed in an html tag. Why can you not match/extract from *multiple nested* tags?
- You just torrented an entire season of a TV show, and the files have fucked up names. Luckily, they're fucked up in a regular way (heh). Bulk rename and cleanup the files.
- Find all gmail usernames from a bunch of email addresses.
  - Validate RFC822 email addresses (please no holy shit wtf)
- Implement a fast (linear time) regexp matcher.
  - As a DSL, with a normaliser.
- What is PCRE? Figure out how to install the damn PCRE headers on a Mac.
- Find all files with phone numbers.
- Editor scripting: Automatically fix indentation for a whole **web** project, where html+css+javascript is all mushed into the same file. (use emacs web-mode, write lisp)
- Standardize filenames in a directory
- dd a disk image onto a thumbdrive

From `http://matt.might.net/articles/what-cs-majors-should-know/`

- Find the five folders in a given directory consuming the most space.

- Report duplicate MP3s (by file contents, not file name) on a computer.
- Take a list of names whose first and last names have been lower-cased, and properly recapitalize them.
- Find all words in English that have x as their second letter, and n as their second-to-last.
- Directly route your microphone input over the network to another computer's speaker.
- Replace all spaces in a filename with underscore for a given directory.
- Report the last ten errant accesses to the web server coming from a specific IP address.

6 References and links

This workshop is mostly based on a series of articles by Matt Might[1]:

- http://matt.might.net/articles/sculpting-text/
- http://matt.might.net/articles/best-tools-for-using-and-learning-linux-and-unix/
- http://matt.might.net/articles/bash-by-example/
- http://matt.might.net/articles/sql-in-the-shell/
- http://matt.might.net/articles/basic-unix/
- http://matt.might.net/articles/settling-into-unix/
- http://matt.might.net/articles/what-cs-majors-should-know/

You may also find these other resources useful:

- http://www.commandlinefu.com/commands/browse
- http://sites.harvard.edu/~lib113/reference/unix/writingtools.html

[1]: with examples drawn from personal experience.