CS2043 - Unix Tools & Scripting

Cornell University, Spring 2014[1]

Instructor: Bruno Abrahao

February 7, 2014

[1] Slides evolved from previous versions by Hussam Abu-Libdeh and David Slater

Regular Expression

- A new level of mastery over your data.
- Pattern matching with regular expressions is more sophisticated and more powerful than shell expansion.
- A regular expression is a set of strings that **match** the expression.

Regular Expression

- Standard features in a wide range of languages (Perl, Python, Ruby, Java, VB.NET and C#, PHP, and MySQL)
- Many Unix tool takes in a **regular expression** as its input, e.g., `grep`

Regular Expressions are used all over the place. Where else can we use Regular Expressions?

Example: less and vim

When we are reading something using less/vim, if we hit / and type a regular expression, less will highlight everywhere it occurs

- press `n` to move to the next match
- press `N` to move to the previous match

Regular Expression

- Regular Expressions use different syntax than shell expansion
- We enclose them in single quotes to distinguish them from shell expansion.

Regular Expression Rules

Some RegExp patterns perform the same tasks as our earlier wildcards

Single Characters

Wild card: ? RegExp: .

- Matches any single character

Wild card: [a-z] RegExp: [a-z]

- Matches one of the indicated characters
- Don't separate multiple characters with commas in RegExp form (e.x. [a,b,q-v] becomes [abq-v]).

Example:

`grep 't.a'` - prints lines with things like tea, taa, and steap

Warning!

Like shell wildcards, RegExps are case-sensitive. What if you want to match any letter, regardless of case?

- What will [a–Z] match?

Character Sorting

Different types of programs sort characters differently. In the C language, characters A-Z are assigned numbers from 65-90, while a-z are 97-122. Thus, the range [a-Z] would equate to [65-122].

- There are non-alphabet characters within [a-Z].
- To specify all letters safely we would use [a-zA-Z].

- Note: not everything treats sorting like C. For example, a dictionary program might sort its characters aAbBcC...

Fortunately We Can Get Around This Easily

Fortunately there are shortcuts for many ranges of characters we typically run into:

POSIX character classes

- `[:alnum:]` - alphanumeric characters
- `[:alpha:]` - alphabetic characters
- `[:digit:]` - digits
- `[:punct:]` - punctuation characters
- `[:lower:]` - lowercase letters
- `[:upper:]` - uppercase letters
- `[:space:]` - whitespace characters

Example:

```
ls | grep [[:digit:]]
```

- list all files with numbers in the filename

Class Shorthands

Support for the following shorthands is common:

Class Shorthands

- `\d` - digit
- `\D` - non-digit
- `\w` - part of word character, i.e., `[a-zA-Z0-9]`
- `\W` - non-word character
- `\s` - whitespace character
- `\S` - non-whitespace character

The Not Operator

We can also negate ranges of characters:

Not

- `[^abc]` - matches any character that is not a b or c
- `[^a-z]` - matches any non lowercase letter

Matching Repetitions

A RegExp followed by one of these repetition operators defines how many times that pattern should be matched:

- `*` - matches 0 or more occurences of the expression
- `\?` - matches 0 or 1 occurrences of the expression
- `\+` - matches 1 or more occurrences of the expression

Examples:

- `grep 't*a'` - matches things like `aste`, `taste`, `ttaste`, `tttaste`
- `grep '[[:alpha:]]\+a'` - matches the letter a only when it is preceeded by at least one letter.
- `grep '"\?Hello World"\?'` - matches Hellow World with or without quotes.

Beginning and End

Another thing RegExp can do is match the beginning and end of a line

Positional Operators

- `^` matches the beginning of a line
- `$` matches the end of a line

Examples:

`grep 'o$'`

- matches lines ending with "o"

`grep '^[A-Z]'`

- matches lines beginning with a capital letter

`ls -l | grep '^l'`

- prints all files that are links

Example

Let's play with the file `/var/log/system.log`.

Matching A Range of Repetitions

- `\{n\}` - preceeding item is repeated exactly n times
- `\{n,\}` - preceeding item is repeated at least n times
- `\{i,j\}` matches between `i` and `j` occurrences of strings that match `e`.

Question: How to print all social security numbers in a file (both `111-11-1111` and `111111111`)?

Grouping Expressions

Grouping Expressions

`\(expr\)` : matches expr

- useful for grouping expressions together

Examples:

`a\(boat\)*` finds a, aboat, aboatboat, etc.

Regular Expression Rules

And a few more:

- `c1\|c2` matches the expression c1 or the expression c2.
- `\<` matches the beginning of a word
- `\>` matches the end of a word

•This• •illustrates• •word•'•s• •boundaries•.

- **Question**: What's the difference between `[ab]` and `a\|b`?

Regular Expression Rules

And a few more examples:

`grep '\(left\)\|\(right\)'` matches left or right.

`grep 'top\{3\}'` searches for toppp.

`grep '[0-5]\{2\}\|[6-9]\{2\}'` searches for things like 12, 15, 68, 97, but not 19, 61.

A word about extended regular expressions

With extended regular expressions you do not need to escape special characters such as ?, +, () and {}.

- Some tools enable its use, but not all, e.g., `grep -E` or egrep (limited availability).

Extended regular expressions tend to be cleaner and easier to read:

`grep '\(woo\+t\)\{2,3\}'` becomes `egrep '(woo+t){2,3}'`.

Why we quote regular expressions

Suppose we have a directory with the following files in it:

```
num, num2, test
```

Now suppose we want to search the file test for the regular expression `nu*`. If we don't quote,

```
grep nu* test
```

gets expanded to

```
grep num num2 test
```

, which searches num2 and test for the string num.

Regular Expression Examples

Let's play with file /usr/share/dict/words

- **Question**: How would you match any word that begins with c and ends with d?
- **Question**: Fing 5 letter words beginning with c and ending with d?

  `caged`
  `caked`
  `caned`
  `caped`
  `cared`

⋮

  Great for crosswords!

Regular Expression Rules

Single Characters

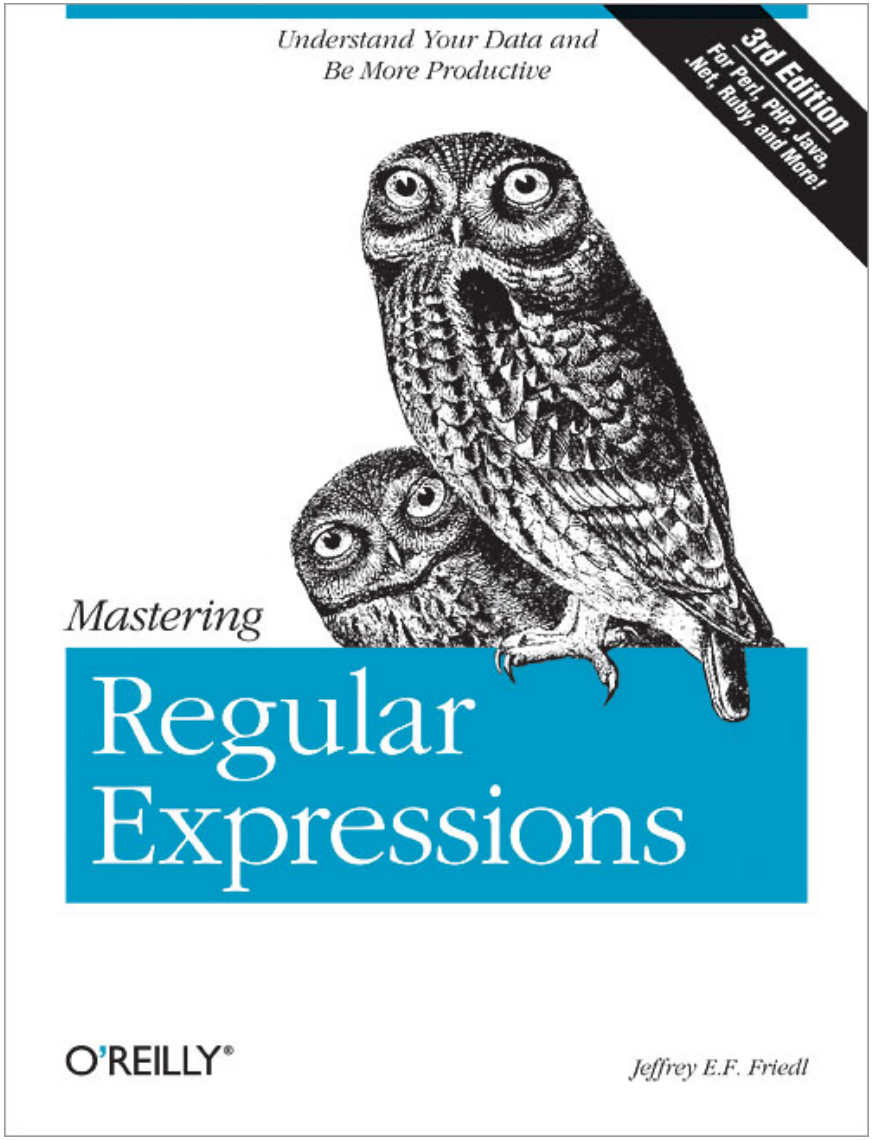
RegExp: .

- Matches any single character

How can we search for an url? e.g., `www.cs.cornell.edu`?

Using . would result in anything like `www2csicornell9edu`

Example: `[[A-Z].]` matches an upper case letter or a dot character.
Example: `\.` is the escaped version that matches a dot character.


Understand Your Data and
Be More Productive
3rd Edition
For Perl, PHP, Java, .Net, Ruby, and More!
Mastering
Regular
Expressions
O'REILLY®
Jeffrey E.F. Friedl