Regular Expressions

Benjamin Brewster

Adapted from slides by Jon Herlocker, OSU

Regular Expressions

- Regular expressions are a way to specify a pattern of strings that you'd like returned as part of a search

- From Windows, searching for *.exe finds all executibles

Regular Expressions

- REs are used by many UNIX programs
  - grep, sed, vi, emacs, regexp, etc.
- Used extensively by many scripting lanugages
  - Perl, Tcl/Tk
- There is an entire course (CS321) that goes over REs, and other grammars

Regular Expression Libraries

- Several libraries exist for regular expressions
  - grep *(basic)*
  - /usr/xpg4/bin/grep –E *(extended)*
  - Perl *(Out of control crazy super Extended)*

Simple RE Example

- Lets use grep to find all lines that have the word “FINDME” anywhere in them:

```
% grep "FINDME" fileToSearch
```

Simple RE Example

- Lets use grep to find all lines that have the word “FINDME” in them

```
% cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
% grep "FINDME" fileToSearch
FINDME first line
second FINDME line
third line FINDME
sixth lFINDMEine
```

`grep`?

- What is `grep` doing?
  - Filtering: `grep` is a filter

- The correct meaning of `grep`'s name:
  - "search **g**lobally for lines matching the **r**egular **e**xpression, and **p**rint them"
  - So `grep` is synonymous with "search"

Another `grep` example

```
% ps -ef | grep brewsteb
root     29541 3760  0 11:26 ?     00:00:00 sshd: brewsteb [priv]
brewsteb 29543 29541 0 11:26 ?     00:00:00 sshd: brewsteb@pts/1
brewsteb 29544 29543 0 11:26 pts/1 00:00:00 -csh
brewsteb 30737 29544 0 11:44 pts/1 00:00:00 ps -ef
brewsteb 30738 29544 0 11:44 pts/1 00:00:00 grep brewsteb
```

Basic REs - Operators

- More operators
  - * (asterisk) – Matches 0 or more of the *previous character*
    - *Warning – this is different than Windows, and UNIX command line usage!*
  - ^ (circumflex) – When placed at the beginning of a RE, indicates the RE must start at the beginning of the string
  - $ (dollar sign) – When placed at the end of an RE, matches the end of the string

The asterisk – 0 or more

Pattern	**Matches**
A*	A or AA or AAA or …
Ab*	Ab or Abb or Abbb or …
FINDME*	FINDME or FINDMEE or FINDMEEE…

Note: ***Not*** FINDME or FINDMEFINDME or …

Binding to Beginning and End

- Unless you use the ^ and $ operators, a RE will match substrings

Pattern	**Matches**
Jon	Will match any string that contains Jon
^abc	Any string that starts with abc
XYZ$	Any string that ends with XYZ
^Ben Brewster$	Any string that matches “Ben Brewster” exactly

Single Char Matching

- The following operators are available
  - . (Period) – Matches any single character
  - \. (Backslash)– quotes a special character
    - Like the period character itself
  - [abc] – Matches any one character inside the brackets
  - [^abc] – Matches any character except any of the ones inside
  - Any other non-special character matches itself

Period Example 1

```
% cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
% grep "FINDM." fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
sixth lFINDMEine
```

Period Example 2

- .* means match any single char any number of times
  - This is the “anything, any length” wildcard

```
% grep ".*" ./* > newFile
```

  - What does this do?
  - Find all lines in all files in the current dir, and store this list of lines in the new file called newFile

Brackets Example 1

```
% cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
% grep "FINDM[E3]" fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
sixth lFINDMEine
```

Brackets Example 2

```
% cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
% grep "FINDM[^3]" fileToSearch
FINDME first line
second FINDME line
third line FINDME
sixth lFINDMEine
```

Brackets Example 3

```
% cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
% grep "[^3]" fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
```

Ranges

- When using the square brackets [ ], you can specify ranges of characters to match
- The proper ordering is defined by the ASCII character set – see http://www.neurophys.wisc.edu/www/comp/docs/ascii.html

Pattern	**Matches**
[a-z]	a b c d e f … y z
[^a-z]	Anything but the characters a-z

Quoting with the backslash

- The backslash causes the REs to literally interpret special characters

Pattern	**Matches**
\.	.
\$	$
*	*

Or: |

```
% cat fileToSearch
I dislike cats
I dislike dogs
% grep "cat\|dog" fileToSearch
I dislike cats
I dislike dogs
% grep "I dislike \(cat\|dog\)s" fi…
I dislike cats
I dislike dogs
```

Note that we parenthesize the 'or', here, and that each syntax symbol is escaped

Matching a Repeated Pattern

- We can search for a pattern that is repeated at least once

```
% cat fileToSearch
catdogcatdog
catdog

% grep "\(catdog\)\1" fileToSearch
catdogcatdog
```

Matching a Repeated Pattern

- Curly braces specify the number of repeats (at least) that we're looking for to register a match

```
% cat fileToSearch
dig
digdig
digdigdig
digdigdigdig

% grep "\(dig\)\{2\}" fileToSearch
digdig
digdigdig
digdigdigdig
```

Backreferences

```
% cat fileToSearch
You dislike You
I dislike You

% grep "\(You\) dislike \1" fileToSearch
You dislike You
```

Backreferences

- \(\) – (parentheses) These operators will capture a matched string for later use

- \1, \2, etc. – (escaped integer) This allows you to specify that the string should match the *n*th pattern that you have previously captured, where *n* is the number following the backslash