Unix Talk #2

Regular Expressions

vi / sed / grep / fgrep / egrep / awk

What is a Regular Expression?

- Pattern to match all or part of a line of text
  - Expressed in a formal, albeit weird, language
  - For example:

```
^.*:Mike
```

    matches lines that start (^) with any string (.*) and contain a colon followed by Mike
- ^, . and * are called *meta characters*
  - They do not represent themselves, but have other special meaning.
- :, M, i, k and e are normal characters
  - The do represent themselves

Where are Regular Expressions Used?

- Special commands that “know” about them
  - vi
    - Uses them in searching for a string: /regexp/
    - Uses them in substitite command: s/regexp/replace/sed
  - Edit a file (like vi) as a filter in a pipeline

    … | sed *editing-commands* | …
  - grep , fgrep (fixed grep) , egrep (extended grep)
    - Print lines of a file that match a regexp
  - awk
    - Process input files looking for lines that match a regexp and processing those lines

Regular expression

- A regular expression is a pattern of characters used for describing sets of strings
- A pattern or sequence of characters
  - Upper and lower case
  - Digits
  - Space, underscore, etc
- Metacharacters

Basic Regular Expressions

- Patterns that match a single character
  - All regular letters match themselves

    ```
    a  b  z  T  Q  0  1  9
    ```

  - . (a single dot) matches any single character except newline.
    - In awk, dot can match newline also
    - (like ? in filename generation)
  - A set of characters that matches any single character from the set (just like filename generation)

    ```
    [aeiou]
    [a-z0-9]
    [A-Za-z-]
     [a-m]
    ```

Metacharacter

* Matches 0 or more occurrences of the **preceding** char

[…] Matches any one of characters enclosed between the brackets.

- dash indicates a range when inside sq bkts.

[^ - negates what's inside brackets]

Metacharacter (con't)

\ - backslash - escape character - just like before.

- \. means match a dot
- This means \ is a meta character
- \\ means match \

Positional indicators:

- ^ anchor to beginning of line
- $ anchor to end of line

A Simple Example Using sed

- Try this example and see what happens …

```
STRING="Four score and seven years ago"
echo "Start: $STRING"

STRING=`echo $STRING | sed -e 's/and/or/'`
echo "Step1: $STRING"

STRING=`echo $STRING | sed -e 's/s..../xxxxx/'`
echo "Step2: $STRING"

STRING=`echo $STRING | sed -e 's/s..../xxxxx/'`
echo "Step3: $STRING"
```

- "Some people, when confronted with a Unix problem, think ‘I know, I’ll use sed.’ Now they have two problems.”

  - unknown (page 206 Unix Haters Group)

Anchoring the Match

- Two rules:
  - Normally, matches are unanchored … i.e., the match can occur any place in the string.
  - Normally, substitutions apply to only the first match in the string.
    - To apply the substitution to all matches in the string, append a 'g' following the last '/'.

Another Example

- Try this

```
STRING="Four score and seven years ago”

echo $STRING | sed 's/^and/or/'

echo $STRING | sed 's/s..../xxxxx/g'

echo $STRING | sed 's/ago$/<END>/'
```

Regular expression examples

- Peach
- a*c
  - cxxx, acxxx, aaacxxxx
- a.c
  - a+c, abc, match, a3c
- [tT]he
  - The, the
- Ch[^0-9]
  - Chapter, Chocolate
- ^the
  - Start with the
- Friends$
  - End with Friends

Regular expression examples

- L..e
- \$[0-9]*\.[0-9]
- ^[0-9]file.dat
- [^0-9]file.dat

- MM-DD-YY or MM/DD/YY
  - [0-1][0-9][-/][0-3][0-9][-/][0-9][0-9]

Named classes of characters

- [0-9] ---OR --- [[:digit:]]
- [a-z] --- OR --- [[:lower:]]
- [A-Z] --- OR --- [[:upper:]]
- [a-zA-Z] --- OR --- [[:alpha:]]
- [a-zA-Z0-9] ---OR --- [[:alnum:]]

- egrep ‘^[[:lower:]]*$’

Extended Metacharacter (egrep and awk)

Available in egrep and awk NOT in vi, sed, grep or fgrep

? matches zero or one occurrence of the preceding char

+ Matches 1 or more occurrences of the preceding char

| Specifies that either the preceding or following regular expression can be matched

( ) Groups regular expressions

Examples

- “.*” matches all characters between the quotations
- ^$ matches blank lines
- ^.*$ matches the entire line
- Big(Computer)?
- Compan(y|ies) # note: the | is a pipe symbol
- SSN: [0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9]
- grep ‘\.H[123]’ ch0[12]
  - ch01: .H1 “Contents of Distribution Tape”
  - ch02: .H2 “A Quick Tour”

Repetition Examples

- Consider a file named text containing:

```
We're off to see the wizard.
The wonderful wizard of oz.
What a wonderful wizard he was.
The end.
```

- Try the following commands and explain the output:

```
egrep 'f+' text
grep 'ff*' text
grep 'f\{2\}' text
egrep 'z?' text
```

Some More Examples

- A price: `\$[0-9]*\.[0-9][0-9]`
- A filename, at the start of a line, that starts with a digit
  `^[0-9]file\.dat`
- A filename, anyplace in the line, that starts with a non-digit
  `[^0-9]file\.dat`
- A social security number
  `[0-9]{3}-[0-9]{2}-[0-9]{4}`
- From 4 to 6 digits: `[0-9]\{4, 6\}`
- A date - MM-DD-YY or MM/DD/YY:
  `[0-1][0-9][-/][0-3][0-9][-/][0-9][0-9]`
- A line containing only upper case letters:
  `^[A-Z]*$`

Back References (tag)

- vi, sed and grep family only

  `/.*\(love\).*\1.*$/`

  – Finds lines that contain ‘love’ at least twice

  – The \(… \) is a way to parenthesize a part of the regexp

  – The \1 is a reference to what was matched by the 1st regexp

  – Can use up to nine tags (\1 … \9)

Replacement in sed and vi

- Applies to substitute command:
  - s/regexp/replacement/
  - Replacement string can contain the metacharacter ‘&’ which means the string that was matched
- Try this

```
STRING="Start Again"
echo $STRING | sed 's/Again/& &/'
```

Replacement in sed and vi

- Try:

```
echo $STRING | sed 's/Again/&ANOTHER&/'
```

- Then try:

```
STRING="Beastie Boys getting live on the spot"
echo $STRING | sed 's/Beastie/& & & &/'
```

Alternation

- Only in egrep and awk
  - regex1 | regex2 | regex3 …
    - Matches regex1 if it can
    - If not goes on to regex2
    - Etc. until one matches or they all fail
- Example:
  - egrep "Brown|Smith" file
    - Prints lines containing Brown or Smith or both