REGULAR EXPRESSIONS AND GREP

Filters

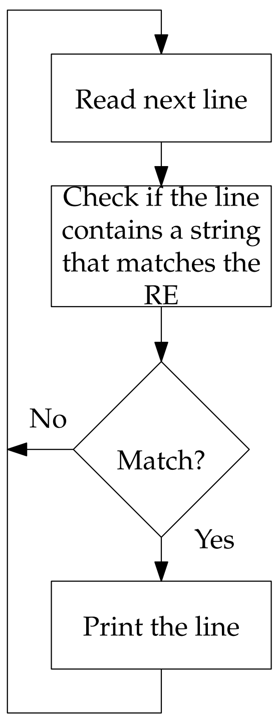
S Milton Rajendram

6 August 2019

1 Regular Expressions

- Text file, sequence of lines
- Search for a string: In which lines does a string occur?
- In which lines does a *pattern* occur?
- Regular Expression (RE): pattern of strings
- Language (alphabet, operators), RE defines a set of strings
- Command line tools and programming lnaguages
- POSIX standard – command line tools
- Some similarity to wildcards (`*,?,[]`). Shell matches wildcards with filenames

```
for each line in the inputs:
   if the line contains a string that matches the regular expression:
       print the line
```



2 `grep` Command

- Global Regular Expression Print
- Search text files for lines which contain strings matching a given regular expression, and output the lines

```
ls /bin > dirlist-bin.txt
ls /usr/bin > dirlist-usr-bin.txt
ls /sbin > dirlist-sbin.txt
ls /usr/sbin > dirlist-usr-sbin.txt

grep zip dirlist*.txt
ls /usr/bin | grep zip
```

- Command line `grep [options] regex [file...]`
- Symbols
  - Literal characters

  - Metacharacters `^ $ . [ ] { } - ? * + ( ) | \`
  - Escape `\`
    - Escaped metacharacters become literal characters
    - A few escaped characters (metasequences) become control characters

- A few metacharacters are common to shell and regular expressions
  - If such metacharacters are in a regular expression, quote the regular expression
  - Otherwise, shell will interpret them

```
ls | grep marks[123].txt
ls | grep 'marks[123].txt'
ls | grep marks[123].*
ls | grep 'marks[123].*'
```

3 Symbols (atoms)

A symbol matches a single character

- Literal character matches itself.
- Dot matches any single character, except newline.

```
grep '.zip' dirlist*.txt
grep -h '.zip' dirlist*.txt
ls | grep '.txt'
```

  `-h` hide filenames

- Anchors
  - `^r` matches `r` at the beginning of lines
  - `r$` matches `r` at the end of lines
  - `\<r` matches `r` at beginning of words
  - `r\>` matches `r` at end of words

```
grep -h '^zip' dirlist*.txt
grep -h 'zip$' dirlist*.txt
grep -h '^zip$' dirlist*.txt
```

- Match empty lines

  ```
  grep '^$' rudyard-kipling.txt
  grep -n '^$' rudyard-kipling.txt
  grep -c '^$' rudyard-kipling.txt
  grep -v '^$' rudyard-kipling.txt
  grep -cv '^$' rudyard-kipling.txt
  ```

- Match directories

  ```
  ls -l | grep '^d'
  ls -l | grep '^.......r-x'
  ```

- Words

  ```
  echo "A part of an apartment." | grep '\<part'
  ```

- Character class matches any *one* of the characters in a set.

  ```
  grep -h '[bg]zip' dirlist*.txt
  ```

  - In a character class, metacharacters are considered literal characters
  - Complement: `^` as the first character in a class

    ```
    grep -h '[^bg]zip' dirlist*.txt
    ```

  - Range

    ```
    grep -h '^[ABCDEFGHIJKLMNOPQRSTUVWXZY]' dirlist*.txt
    grep -h '^[A-Z]' dirlist*.txt
    grep -h '^[A-Za-z0-9]' dirlist*.txt
    ```

  - Escape: turn the metacharacters `-` and `^` to literals.

    ```
    grep -h '[-A-Z]' dirlist*.txt
    ```

- Back references
  - `\1, \2, \ldots, \9`
  - `\n` refers to the string matched by `n` th regular expression

4 Extended `grep`: `egrep` or `grep -E`

- Extended grep

5 Operators

- Sequence (concatenate) `r1r2` matches concatenation of two strings, the first one defined by `re1` followed by the second one defined by `re2`
- Alternative (union) `r1|r2` matches any string matched by `r1` or any string matched by `r2`
- Repetition (closure)
  - `r*` matches `r` zero or more times
  - `r+` matches `r` one or more times
  - `r?` matches `r` zero or one time

6 Concatenate

```
echo "This works." | grep -E '^[A-Z][A-Za-z]*'

echo "this does not work." | grep -E '^[A-Z][A-Za-z]*'
```

7 Alternatives

```
echo "AAA" | grep AAA
echo "BBB" | grep AAA
echo "AAA" | grep -E 'AAA|BBB'
echo "BBB" | grep -E 'AAA|BBB'
echo "CCC" | grep -E 'AAA|BBB'
# more than two alternatives
echo "AAA" | grep -E 'AAA|BBB|CCC'
# alternative as a part of re
grep -Eh '^(bz|gz|zip)' dirlist*.txt
grep -Eh '^bz|gz|zip' dirlist*.txt
```

8 Repetition

- `r?` matches `r` zero or one time
  - that is, `r` is optional

  ```
  echo "2229-4254" | grep -E '^[1-9][0-9][0-9][0-9]-?[0-9][0-9][0-9][0-9]$'
  echo "22294254" | grep -E '^[1-9][0-9][0-9][0-9]-?[0-9][0-9][0-9][0-9]$'
  ```

```
echo "2229 4254" | grep -E '^[1-9][0-9][0-9][0-9]-?[0-9][0-9][0-9][0-9]$'
```

- `r*` matches `r` zero or more times

```
echo "This works." | grep -E '^[A-Z][A-Za-z ]*\.'
echo "This Works." | grep -E '^[A-Z][A-Za-z ]*\.'

echo "this does not work." | grep -E '^[A-Z][A-Za-z ]*\.'

echo "varname" | grep -E '^[A-Za-z][A-Za-z0-9_]*$'
echo "var_name" | grep -E '^[A-Za-z][A-Za-z0-9]*$'
echo "var_name_2" | grep -E '^[A-Za-z][A-Za-z0-9]*$'
echo "_var_name_3" | grep -E '^[A-Za-z][A-Za-z0-9]*$'
echo "VarName_4" | grep -E '^[A-Za-z][A-Za-z0-9]*$'

echo "5varname" | grep -E '^[A-Za-z][A-Za-z0-9]*$'
```

- `r+` matches `r` one or more times

```
echo "This that" | grep -E '^([A-Za-z]+ ?)+$'
echo "This that and nine" | grep -E '^([A-Za-z]+ ?)+$'

echo "This that and 9" | grep -E '^([A-Za-z]+ ?)+$'
echo "This that  and nine" | grep -E '^([A-Za-z]+ ?)+$'
```

- `r{}` matches `r` a specific number of times
  - `r{n}` matches `r` exactly `n` times
  - `r{n,m}` matches `r` at least `n` times and at most `m` times ($n \leq \ldots \leq m$)
  - `r{n,}` matches `r` at least `n` times ($n \leq \ldots$)
  - `r{,m}` matches `r` at most `m` times ($\ldots \leq m$)

```
echo "2229-4254" | grep -E '^[1-9][0-9]{3}-?[0-9]{4}$'
echo "22294254" | grep -E  '^[1-9][0-9]{3}-?[0-9]{4}$'
echo "2229 4254" | grep -E  '^[1-9][0-9]{3}-?[0-9]{4}$'
```

9 Summary

(decreasing order of precedence)

`c`	any non-special character `c` matches itself
`\c`	turn off any special meaning of character `c`
`^`	beginning of line
`$`	end of line
`[...]`	any one of characters in ...; ranges like `a-z` are legal
`[^...]`	any single character not in ...; ranges are legal
`\n`	what the `n`'th `\( ... \)` matched (grep only)
`r*`	zero or more occurrences of `r`
`r+`	one or more occurrences of `r` (egrep only)
`r?`	zero or one occurrences of `r` (egrep only)
`r1r2`	`r1` followed by `r2`
`r1 | r2`	`r1` or `r2` (egrep only)
`\(r\)`	tagged regular expression `r` (grep only); can be nested
`(r)`	regular expression `r` (egrep only); can be nested
	No regular expression matches a newline.

10 Options

- `-i --ignore-case`
  Ignore case. Do not distinguish between uppercase and lowerc

- `-v --invert-match`
  Invert match. Normally, grep prints lines that contain a match. This option causes grep to print every line that does not contain a match.

- `-c --count`
  Print the number of matches (or non-matches if the -v option is also specified) instead of the lines themselves.

- `-l --files-with-matches`
  Print the name of each file that contains a match instead of the lines themselves.

- `-L --files-without-match`
  Like the -l option, but print only the names of files that do not contain matches.

- `-n --line-number`
  Prefix each matching line with the number of the line within the file.
- `-h --no-filename`
  For multi-file searches, suppress the output of filenames.

11 Examples

11.1 Example 1

```
cat files/phonelist.txt
```

```
(782) 109-1816
(180) 383-1301
(304) 176-9993
(263) 205-2981
(251) 24-2931
(264) 185-1088
(526) 102-2988
(300) 193-2433
(971) 165-221
(275) 205-3699
(674) 190-4401
```

```
grep -vE  '^\([0-9]{3}\) [0-9]{3}-?[0-9]{4}$' files/phonelist.txt
```

```
(251) 24-2931
(971) 165-221
```

11.2 Example 2

```
cat files/isaiah.txt
```

```
For it is precept upon precept, precept upon precept,
line upon line, line upon line,
here a little, there a little.
```

```
grep -E '(precept).*(upon).*\1.*\2' files/isaiah.txt
grep -E '(precept).*(upon).*\2.*\1' files/isaiah.txt
```

```
For it is precept upon precept, precept upon precept,
For it is precept upon precept, precept upon precept,
```