Bash: regular expressions

Glenn Bruns
CSUMB

What is a regular expression?

- Examples:
  - gr[ae]y matches 'grey' and 'gray'
  - ^File matches 'File' at beginning of line
- In Unix-like systems, they are a pattern-matching notation
- Lots of Unix tools use regular expressions for pattern-matching (including editors)
- Regular expressions are closely related to finite-state automata
- Regex flavors vary from system to system!

Lecture Objectives

When this lecture is finally over, you should be able to:

- write regular expressions that can be used with grep, ls, sed, awk, and more

Remember: regular expressions are different from bash filename expansion (aka globbing), as in:

```
$ ls *.c
```

Plain text

```
$ echo 'that the address is' | grep 'add'
that the address is
$
$ echo 'that the address is' | grep 't t'
that the address is
$
$ echo 'That the address is' | grep 'that'
$
```

The 'grep' command searches its input for lines containing text that matches the pattern.

For every such line, the entire line is returned.

```
$ grep [OPTIONS] PATTERN FILE
```

Special characters

Bash has special characters, like $ and # and *

Sometimes you want them treated ‘literally’ -- not as special characters.

Put characters in quotes so they’re not treated specially by bash.

```
$ echo '> $?'      # single quotes: nothing treated specially
> $?
$ echo "> $?"      # double quotes: only $ is treated specially
> 0
```

In regular expressions, we also have special characters and ways to treat them literally.

stackoverflow.com/questions/6697753/difference-between-single-and-double-quotes-in-bash

Match any character

.   matches any character

```
$ cat temp.txt
ARG 16k sz
arg size 16k
arg soze 16k
$
$ grep siz temp.txt
arg size 16k
$ grep 's.z' temp.txt
arg size 16k
arg soze 16k
$ grep ' ..z' temp.txt
arg size 16k
arg soze 16k
```

Remember: grep lists the entire line if the pattern matches anywhere on a line.

Special characters in regular expressions

regular expressions use lots of special characters:

- basic: ^$[]*.\ extended: ?+{}()|
- “escape” them with ‘\’ to not use special meaning

```
$ echo 'how the address was formed' | grep '.'
how the address was formed
$
$ echo 'how the address was formed' | grep '\.'
$
```

Match beginning, end of line

^ beginning of line

$ end of line

"anchor characters"

```
$ cat temp.txt
ARG 16k size
arg size 16k
$
$ grep size temp.txt
ARG 16k size
arg size 16k
$ grep 'size$' temp.txt
ARG 16k size
$ grep '^arg' temp.txt
arg size 16k
```

What would you do if you wanted to match on an actual $ character?

Note: single quotes so that bash won't interpret $ itself

Question

```
$ cat temp.txt
abc abd abe
abcde
```

What is the output?

```
$ grep abd temp.txt
```

What is the output?

```
$ grep '^abd' temp.txt
```

What is the output?

```
$ grep 'b.d' temp.txt
```

Zero or more repetitions

a* zero or more repetitions of ‘a’

```
$ cat temp.txt
fl
fle
fleeeeee

$ grep 'fle*$' temp.txt
fl
fle
fleeeeee
```

Question: what lines will be matched here?

```
$ grep 'fle*.$' temp.txt
fle
fleeeeee
```

Match any of a group of characters

[ab]   matches ‘a’ or ‘b’

“character class”

[a-f]   matches any character from ‘a’ to ‘f’

```
$ cat temp.txt
flam
blam
glam
glum
$ grep '[af]lam' temp.txt
flam
$ grep '[a-f]lam' temp.txt
flam
blam
$ grep 'gl[au]m' temp.txt
glam
glum
```

Match any of a group of characters

characters in [ ] don’t have to be letters

[0-9_] matches a digit or underscore

```
$ cat temp.txt
1a
2b
35
$ grep [12]. temp.txt
1a
2b
$ grep [0-9][a-z] temp.txt
1a
2b
$ grep [125] temp.txt
1a
2b
35
```

Any but certain characters

[^abc] maches any character but ‘a’ or ‘b’ or ‘c

```
$ cat temp.txt
flam
blam
glam

$ grep '[^fb]lam' temp.txt
glam

$ grep '^[^g]' temp.txt
flam
blam

$ grep '^[^g]*$' temp.txt
flam
blam
```

Note that ^ changes its meaning when in brackets!

Interlude: special character craziness

```
$ cat movies.txt
My ratings:
Exit through the gift shop ****
Star Wars **
Headhunters ***
$ *
$ grep $ movies.txt
My ratings:
Exit through the gift shop ****
Star Wars **
Headhunters ***
$ *
$ grep '$' movies.txt
My ratings:
Exit through the gift shop ****
Star Wars **
Headhunters ***
$ *
```

```
$ grep '\$' movies.txt
$ *
$
$ grep * movies.txt
grep: bash-hw: Is a directory
grep: bin: Is a directory
$
$ grep '****' movies.txt
My ratings:
Exit through the gift shop ****
Star Wars **
Headhunters ***
$ *
$
$ grep '\*\*\*\*' movies.txt
Exit through the gift shop ****
```

Some extended regex

a{2,4} ‘a’ appears 2 to 4 times

```
$ cat temp.txt
bet
beet
beat
beeeet
$ egrep 'b[ae]{2,3}t' temp.txt
beet
beat
$
```

Some tools support more regex features than others

$ egrep
is same as
$ grep -E

One or more repetitions

a+ ‘a’ appears at least once

```
$ cat temp.txt
fl
fle
flee

$ egrep 'fle+' temp.txt
fle
flee
$
```

Zero or one repetitions

a?   ‘a’ appears optionally

```
$ cat temp.txt
color
colour
$
$ egrep 'colou?r' temp.txt
color
colour
```

Grouping

(pattern)+

(pattern)*

(pattern)?

Repetition of any pattern, not just a character

```
$ cat temp.txt
AabA
AacabA
AadA
adAacA
$ egrep 'A(a[bc])+A' temp.txt
AabA
AacabA
adAacA
$
```

Grouping, cont'd.

```
$ cat temp.txt
555-1212
(630)555-1212
$
$ egrep '^[0-9]{3}-[0-9]{4}' temp.txt
555-1212
$
$ egrep '^\([0-9]{3}\)[0-9]{3}-[0-9]{4}' temp.txt
(630)555-1212
$
$ egrep '^(\([0-9]{3}\))?[0-9]{3}-[0-9]{4}' temp.txt
555-1212
(630)555-1212
```

Sometimes called a 'match group' . The stuff within parentheses is said to be 'captured'.

Predefined character classes

[:alpha:]	upper or lower case alphabetic
[:alnum:]	[[:alnum:]] is same as [0-9a-zA-Z]
[:blank:]	space or tab
…	plus many more

```
$ grep '[[:alpha:]][[:blank:]]' temp.txt
a b
$
$ grep '[[:alpha:][:blank:]]' temp.txt
9 9
aa9
a b
```

From manual at gnu.org: ‘Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket expression.’

More extended regex

Shorthand classes: \d digit, \w word

Anchors: \b word boundary

Support for extended regex varies from system to system

Look at online cheat sheets, and online training such as:

gnu.org/software/grep/manual/
html_node/Regular-Expressions.html#Regular-Expressions

http://regexone.com/lesson/introduction_abcs

Summary

- We learned the basics of regular expressions
- Some important features:
  - `$ ^` (anchors)
  - `.` (match any character)
  - `a*` (zero or more matches of character a)
  - `[abc]` (any of the characters)
  - `[^abc]` (none of the characters)
- We will use them with grep, sed, awk, and other Linux tools
- Commands introduced in this lecture:
  - `grep, egrep`