Chapter 8: Regular Expressions

Introduction to Regular Expressions

- Regular expressions are *patterns* that only certain commands are able to interpret
- Patterns should be protected by strong quotes (single quotes)
- Regular expressions are expanded to match certain sequences of characters in text
- The `grep` command is very useful in demonstrating regular expressions

The grep command

- The `grep` command displays all lines that match a regular expression (sys* as below):

  $grep sys* /etc/passwd
  sys:x:3:3:sys:/dev:/usr/sbin/nologin
  syslog:x:101:104::/home/syslog:/bin/false
  sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin:/bin/bash

- Use the `--color` option to highlight what the regular expression matched
- Examples in this presentation use the `--color` option by default via an alias

Types of regular expressions

- Two primary types:
  - Basic Regular Expressions (regex)
  - Extended Regular Expressions (extended regex)
- Some commands only support Basic
- Use `grep -E` or `egrep` to use Extended

Basic Regular Expressions

Basic Regex Character(s)	Meaning
.	Any one single character
[ ]	Any one specified character
[^]	Not the one specified character
*	Zero or more of the previous character
^	If first character in the pattern, then pattern must be at beginning of the line to match, otherwise just a literal "^"
$	If last character in the pattern, then pattern must be at the end of the line to match, otherswise just a literal "$"
\	Treat the special character after “\” literally

Example of grep

- Display all lines of the `/etc/passwd` file that contain the expression "root":

  $grep root /etc/passwd
  root:x:0:0:root:/root:/bin/bash
  operator:x:1000:37::/root:

Anchor example

- Use the "^" anchor to make sure that this pattern "root" appears at the beginning of the line:

```
$grep ^root /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Anchor example

- Use the "$" anchor to make sure that this pattern "root" appears at the end of the line:

  $grep $'bash' /etc/passwd
  root:x:0:0:root:/root:/bin/bash
  sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin:/bin/bash

  Note: in the example above, bash must be placed in single quotes due to the special nature of the $ character to the shell

. example

- Use the . character to match any character (except for the new line character):

```
$grep ro.. /etc/passwd
root:x:0:0:root:/root:/bin/bash
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
operator:x:1000:37::/root:
```

[ ] example

- Use the [ ] characters to match any character of a specific set of characters:

```
$grep chapter[1-5] bookfile
chapter1
chapter2
chapter3
chapter4
chapter5
```

[ ] example

- If the first character inside the square brackets is the "^", then it negates the characters

  $grep chapter[^1-5] bookfile

  chapter6

  chapter7

  chapter8

* example

- The * character match zero occurrences of the character (or pattern) that precedes it:

```
$grep sys* /etc/passwd
sys:x:3:3:sys:/dev:/usr/sbin/nologin
syslog:x:101:104::/home/syslog:/bin/false
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin:/bin/bash
```

Extended Regular Expressions Summary

Extended Regex Character(s)	Meaning
+	One or more of the previous pattern
?	The preceding pattern is optional
{ }	Specify minimum, maximum or exact matches of the previous pattern
|	Alternation - a logical "or"
( )	Used to create groups

+ example

- The + character match one or more occurrences of the character (or pattern) that precedes it

Example	Result
egrep '[a-z]+ing' *file*	Displays all lines having one or more letters followed by "ing." (i.e. ring, learning, computing, etc.)

? example

- The ? character matches zero or one occurrence of the character (or pattern) that precedes it

Example	Result
egrep ‘bel?ville’ *file*	Displays all lines containing “belville” or “bellville.”

{ } example

- The { } character are used to specify how many times to repeat the previous character or pattern

Curly Brace Example	Other Regex Equivalent	Meaning
'a{0,}'	'a*'	Zero or more 'a' characters
'a{1,}'	'a+'	One or more 'a' characters
'a{0,1}'	'a?'	Zero or one 'a' characters
'a{5}'	N/A	Five 'a' characters
'a{,5}'	N/A	Five or fewer 'a' characters
'a{3,5}	N/A	From three to five 'a' characters

| example

- The | character matches either one expression or another

Example	Result
egrep ’64|128’ *file*	Displays all lines containing either “64” or “128.”

( ) example

- The ( ) characters group patterns together
- They are good when you want to use *, +, ? or { } on more than one character
- They are also good for limiting scope of | character

Example	Result
egrep '(computer|ling)' *file*	Displays all lines containing either "computer" or "computing"

( ) example

- Some commands capture what is in ( )
- \1 returns first ( ) match
- \2 returns second ( ) match

Special sequences

Backslash Sequence	Matches
\b	A word boundary
\B	Not a word boundary
\w	A word character
\W	Not a word character
\s	A whitespace character
\S	Not a whitespace character
\\	A backslash character

Special sequences example

- Bad pattern match:

- Using word boundaries:

\ example

- The \ character makes the following character literal (not special):

The fgrep command

- The `fgrep` (fast grep) command always treats patterns as literal characters, so the backslash is not needed to “escape” the characters:

Example	Result
fgrep ‘*’ *file*	Displays all lines containing the * string.

Common grep, egrep, & fgrep options

Option	Meaning	Example
-i	Case insensitive	grep –i Root
-v	Invert search results (logically negates criteria) - returns all lines that don't contain specified pattern	grep –v chapter[0-5] bookfile
-l	Display a list of files containing a string	grep –l Chapters
-r	Perform a recursive search including subdirectories	
-w	Match whole word only	grep – w sysadmin /etc/passwd
-q	Quietly operate without producing output. Used in scripts to check if a line is present	grep –q 'linux_enable="YES"'