Anthony Ventresque
anthony.ventresque@ucd.ie

Regex, Grep and Sed

Quotes

- There is four type of quotes in bash
  - Command substitution (`…`)
  - Weak quote (“…“)
  - Strong quote (‘…‘)
  - Character quote (\x , with x a character)

E.g.,

```
filename=‘myfile.txt‘
echo ‘Text in a file‘ > $filename
```

```
echo `cat $filename`
>> Text in a file
echo “cat $filename“
>> cat myfile.txt
echo ‘cat $filename‘
>> cat $filename

echo \‘cat $filename\‘
>> ‘cat myfile.txt‘
echo \‘cat \$filename\‘
>> ‘cat $filename‘
```

Regular Expressions: regex

A regular expression is a sequence of characters that form a **pattern**.

E.g., ^[hH]ello[1-5]*

Regex is useful for searching words matching that pattern.

It is also **case sensitive.**

A regex is structured in **three parts**:

- **Anchor**: used to specify the position of the pattern a line
- **Character Set**: the characters that need to be matched
- **Modifiers**: how many times the previous characters are repeated.

Regex: Anchor

“**^**”: at the beginning of the line:

E.g., ^Takfarinas

Takfarinas is the TA

Is Takfarinas the TA?

“**$**”: at the end of the line:

E.g., lecturer$

Anthony is the lecturer

The lecturer is Anthony

Without any anchor: at any position in the line:

E.g., lecturer

Anthony is the lecturer of COMP30640

Regex: Character Set

- A character set is formed by a sequence of characters (e.g., a, d, 4, 7, #, \$, etc).

  E.g., hardPasswd12

You can also use:

“.”: any character at the position it is mentioned including empty space:

E.g., .ard.asswd12.

“[]”: to select any character from the defined subset:

E.g., hard[A-Za-z]as[zs]wd[1-9]2

“[^]”: to select any character **except** those specified:

E.g., hardPasswd[^2-9]2

Regex: Modifier

“*”: it indicates that the character or the sequence before it can appear zero of multiple times

E.g., Hel*o

>> Matches with Heo, Helo, Hello, Hellllo, etc

You can also apply “*” to:

“.”: to indicate a sequence of any characters.

E.g., H.*o

>> matches with Ho, Hydro, Hallo, Hereinto , etc

“[]”: to indicate a sequence made of characters defined by the square brackets.

E.g., passwd[0-9]*

>> matches passwd, passwd9156, passwd1234561089, etc

Grep

Grep prints out the lines containing a string matching the given pattern.

E.g.,

- grep root /etc/passwd
- grep ‘^ro*t‘ /etc/passwd
- cat /etc/passwd | grep ‘root:*:0:[0-9]‘

Grep

You can also use grep with more than one pattern:

- pattern1 **or** pattern2:
  - grep ‘pattern1\|pattern2‘ filename
  - grep −e pattern1 −e pattern2 filename
- pattern1 **and** pattern2: there is no “**and**” in grep, but we can simulate it with an “**or**”:
  - grep ‘pattern1.*pattern2\|pattern2.*pattern1‘ filename

Grep

You can use grep with different options:

- **-c**: only gives the number of matching lines
- **-v**: only shows the lines that do not match the pattern. Inverted search.
- **-i**: ignore case
- **-n**: gives the line number as well as the matching lines.
- Etc.

Stream Editor: Sed

If you would like to write a program that modifies the content of a file, sed is a good tool to use.

The essential command in sed is the **substitution** indicated by "**s**".

E.g., sed 's/day/night/' filename

- cat filename | sed 's/day/night/'
- sed 's_day_/night_' filename

Note: sed goes through filename line by line and substitutes only the 1st occurrence of day in each line.

If a line contains : **I do it everyday except Monday.**

Sed will turn it into: **I do it everynight except Monday.**

Sed

Usage:

sed ‘**s/Search Pattern/Replacement String/**‘ filename

A sed command is composed of four parts:

- **s** : the substitute command
- **/../../** : a delimiter. Note that you can use other characters instead of “/” e.g., “_” this way **_.._.._**
- **Search Pattern** : a regular expression
- **Replacement String** : a string

Matching string as a replacement string

You might want to add a prefix or a suffix to the strings that match your pattern.

Solid pattern:

- sed ‘s/user23/user23/‘ filename

Regex pattern:

- sed ‘s/user[0-9]*/&/‘ filename

Splitting the matching string

If you are searching for a string with a particular token, but would like to take he token out:

E.g., if you would like to rename the files:

Rename module IDs from COMP**ID** to CS**ID**

- sed ‘s/COMP\([1-5][0-9]*\)/CS\1/‘ filename

If you would like to switch the order of two words:

Reverse the order between first name and family name separated by a space

- sed ‘s/\([^]*\)[ ]\([^]*\)/\2 \1/ ‘ filename

Specifying which occurrence using flags

If you have a line with multiple strings matching the pattern:

By default, only the 1st occurrence gets substituted

You can specify which occurrence you want to substitute using: /1, /2

E.g., deleting the last name, or hiding the password:

- sed ‘s/[^]/DELETED/1‘ filename
- sed ‘s/[^]/******/2‘ filename

Note: don’t get confused between **\1** and **/1**. The 1st is used as a replacement string. The 2nd is used as a flag.

You can apply the substitution on all the matching strings using: /g

E.g. turning every space into a comma:

- sed ‘s/[ ]/;/g‘ filename