CSc 352
Text Processing and Regex

Benjamin Dicken

Announcements

- Exam 2
- Study guide
- Review session hosted my Michael
    - Wednesday April 5, 5pm

Processing Text

- By "processing text" (in UNIX) we mean any commands that can search, arrange, and modify text files or text streams.

- **Many** commands on a UNIX system can be used for this

```
sort  sed  cut  grep  head  tail  sed  tr  awk . . .
```

Processing Text

Why should we be comfortable with text-processing?
   (either in UNIX, or just with programming in-general)

- Sifting through large log files from a program
- Manipulating data files, such as XML, CSV, JSON
- Searching for patterns, functions, keywords in large codebases (without an IDE)
- Web scraping
- Data cleansing

Three tools in UNIX

`grep` - A "simple" tool for searching for patterns within a file / stream and printing out the matches

`sed` - A tool that can *manipulate* the content of files by searching for patterns, replacing text. (more "advanced" than grep)

`awk` - A programming language that can be used for text searching / processing / manipulation

More simplistic

More advanced
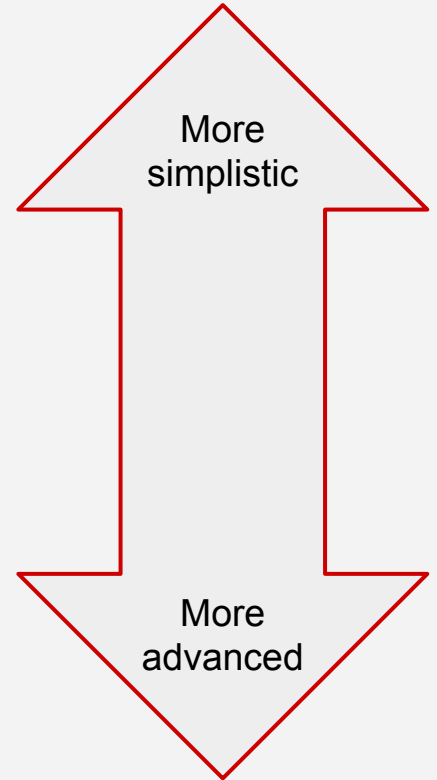
Regular Expressions

- These three tools use ***regular expressions***
- A regular expression is a *description of a pattern of text, specified with text*
- Regular expressions are not only useful for these unix tools, but in other contexts too (web dev, functionality in other languages)

Go through this tutorial! → https://regexone.com/

Reference → https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html

Regular Expressions

- Not every tool supports the exact same set of regex features
- BRE and ERE

    https://www.gnu.org/software/sed/manual/html_node/BRE-vs-ERE.html

- Use the -e option for ERE
- Also PCRE, use -P (with grep)

sample.txt (use for learning regex)

```
the quick brown fox
The Quick Brown Fox
THE QUICK BROWN FOX
fox brown quick the
Th3 qu1ck br0wn f0x
thequickbrownfox
100 quick brown foxes
nearly 200 speedy animals
one huge intelligent brain
the briin on point
the speeeeeeeeeeeedy animal
the speeeeed of the animal
the car spewed oil
```

Find every line containing "the"

```
$ grep the sample.txt
```

```
$ cat sample.txt | grep the
```

Many ways to accomplish the exact same thing

```
$ grep the < sample.txt
```

```
$ sed -n /the/p sample.txt
```

```
$ awk '/the/{print $0}' sample.txt
```

Find every line containing "the"

```
$ grep the sample.txt

$ cat sample.txt | grep the

$ grep the < sample.txt

$ sed -n /the/p sample.txt

$ awk '/the/{print $0}' sample.txt
```

These are basic regular expressions

Can make these more complex

Regex Special Keywords

Regex has many special keywords that represent something other than the literal character. Some of these are:

^ $ [   ]   .   +   *   -   \   {   }

Beginning and end of line

The **^** and **$** match the beginning and end of a line of input

Useful when you want search for something at the beginning of a line, end of a line, or match an entire line

For example:

```
$ grep ox$ sample.txt
```

Will search for lines that ***end*** with the word "**ox**"

```
the quick brown fox
The Quick Brown Fox
THE QUICK BROWN FOX
fox brown quick the
Th3 qu1ck br0wn f0x
thequickbrownfox
100 quick brown foxes
nearly 200 speedy animals
one huge intelligent brain
the briin on point
the speeeeeeeeeeedy animal
the speeeeed of the animal
the car spewed oil
```

Match exactly one character

The dot (**.**) matches any character at that position

Does this seem familiar?

For example:

        `$ grep br..n sample.txt`

Will search for lines that contain a "br" followed by any two characters, and end in an "n"

```
the quick brown fox
The Quick Brown Fox
THE QUICK BROWN FOX
fox brown quick the
Th3 qu1ck br0wn f0x
thequickbrownfox
100 quick brown foxes
nearly 200 speedy animals
one huge intelligent brain
the briin on point
the speeeeeeeeeeedy animal
the speeeeed of the animal
the car spewed oil
```

Previous Character

+ matches ***one*** or more of the character that comes before it

* matches ***zero*** or more of the character that comes before it

? matches ***zero*** or ***one*** of the character that comes before it

For example:

        $ grep -P sle+p sample.txt

Will search for lines that contain "screch", "screech", "screeech", etc

```
the quick brown fox
The Quick Brown Fox
THE QUICK BROWN FOX
fox brown quick the
Th3 qu1ck br0wn f0x
thequickbrownfox
100 quick brown foxes
nearly 200 speedy animals
one huge intelligent brain
the briin on point
the speeeeeeeeeeeedy animal
the speeeeed of the animal
the car spewed oil
```

Groups and Ranges

Use **[ ]** to specify a group of characters to match

Use **-** to specify a range within a group

For example:

```
$ grep sp[aeiou]+ch sample.txt
$ grep br[O0o]wn sample.txt
```

```
the quick brown fox
The Quick Brown Fox
THE QUICK BROWN FOX
fox brown quick the
Th3 qu1ck br0wn f0x
thequickbrownfox
100 quick brown foxes
nearly 200 speedy animals
one huge intelligent brain
the briin on point
the speeeeeeeeeeeedy animal
the speeeeed of the animal
the car spewed oil
```

What will it match?

Determine at least one string that each grep command will match

```
$ cat input.txt | grep [T-X][a-z][a-p]
   Match:        Tap
   Not match:    a5:)
$ cat input.txt | grep -E z[eio]+[aeiou]*
   Match:        zoooooooooooo
   Not Match:    dug
$ cat input.txt | grep ..[A-Za-z]..
   Match:      00000
   No match:     123456789
```

What will it match?

Determine at least one string that each grep command will match

```
$ cat input.txt | grep [T-X][a-z][a-p]
    Tub
$ cat input.txt | grep z[eio]+[aeiou]*
    zeeaeiou
$ cat input.txt | grep ..[0971]..
    br9uh
    cs120
```

Write the regex that....

- Searches for all lines start with "the"
- And end with the letter "l" (el)
- Contain at least one vowel in-between

```
the quick brown fox
The Quick Brown Fox
THE QUICK BROWN FOX
fox brown quick the
Th3 qu1ck br0wn f0x
thequickbrownfox
100 quick brown foxes
nearly 200 speedy animals
one huge intelligent brain
the briin on point
the speeeeeeeeeeeedy animal
the speeeeed of the animal
the car spewed oil
```

Special Categories of Characters

\d   match any digit

\D   match any non-digit

\w   match any alphanumeric

\W   match any non-alphanumeric   (with -P)

\s   match any whitespace

\S   match any non-whitespace

.   match any character

Escaping

As with string literals in code, use backslash to escape special characters

For example, if you want to actually search for a period, brackets, etc.

Grep: A Few Flags

-E Use extended regex (or use `egrep` instead)

-o Print only the matching part of a line

-R Recursive-search through directories and subdirectories

-v Print non-matching lines

-P Use PCRE

sed (Stream EDitor)

```
$ sed -r -n -E '/the/p' file_name.txt
```

options

sed command

The sed command (address)

File name (don't include if reading from standard input)

sed Commands (Addresses)

`'/the/p'` ← **Print lines matching 'the'**

`'s/night/day/p'` ← **Substitute occurrences of 'night' with 'day'**

`'s/\s[a-z][a-z]\s/ ZZ /g'` ← **Substitute two-letter words surrounded by whitespace with ' ZZ '**

See: `$ man sed`

Practical Use for Regex, grep, sed

- In the context of being a software developer working on UNIX systems
- What are some practical examples of the usefulness of regex, grep, sed?

Process Display

Show only the username and PID for all processes running on the system, in sorted order

Process Display

Show only the username and PID for all processes running on the system, in sorted order

```
ps aux | sed -r -n 's/([a-z]+)[ \t]+([0-9]+) .*/\1 \2/p' | sort
```

Further Reading

https://www.digitalocean.com/community/tutorials/the-basics-of-using-the-sed-stream-editor-to-manipulate-text-in-linux

https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux