

CSc 352

Text Processing and Regex

Benjamin Dicken

Announcements

- I'm back!
- Video from Russ
- Exam 2 is on Wednesday
- PA 8 is due next Tuesday (April 12th)
- PA 6 grades
- Pop quizzes will return :) :(
- TA applications
 - <https://www.cs.arizona.edu/ugta>

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)

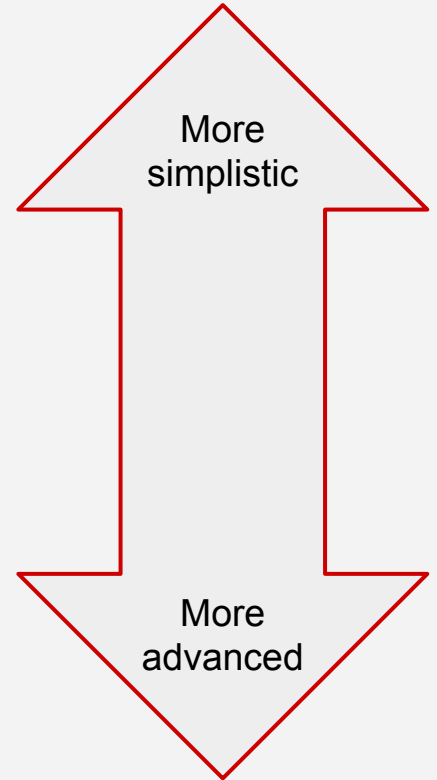
- Sifting through large log files from a program
- Manipulating data files, such as XML, CSV, JSON
- Searching for patterns, functions, keywords in large codebases
(without an IDE)
- Web scraping
- Data cleansing

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



Regular Expressions

- These three tools use ***regular expressions***
- A regular expressions 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

sample.txt (use for learning regex)

The quick brown fox
Jumped over the lazy dog
Sitting under the tree
By the road next to the river
the super quick red fox
leapt over the sleeeeeeeeping dog
sitting beneath the treeeeeeeeeee
By the path close to the river
treeeeeeeeeee houses

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
```

***Many ways to
accomplish the
exact same thing***

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 mountains$ sample.txt
```

Will search for lines that *end* with the word “mountains”

Search The File

Write a bash command to print lines from **sample.txt** that *begin* with “the”

Write a bash command to print lines from **sample.txt** that *end* with “tree”

Match exactly one character

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

Does this seem familiar?

For example:

```
$ grep p..h sample.txt
```

Will search for lines that contain a “p” followed by any two characters, and end in an “h”

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 scre+ch sample.txt
```

Will search for lines that contain “screch”, “screech”, “screeech”, etc

Search The File

Write a bash command to print lines from **sample.txt** that begin with “the” and end with “fox” and can have any amount of text in-between

Groups and Ranges

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

Use `-` to specify a range within a group

For example:

```
$ grep r[aeiou][aeiou]ch sample.txt
```

```
$ grep r[a-z]d sample.txt
```

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]
```

```
$ cat input.txt | grep z[eio]+[aeiou]*
```

```
$ cat input.txt | grep ..[0971]..
```

Special Categories of Characters

\d match any digit

\D match any non-digit

\w match any alphanumeric

\W match any non-alphanumeric

\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

sed (Stream EDitor)

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


sed command



options



**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