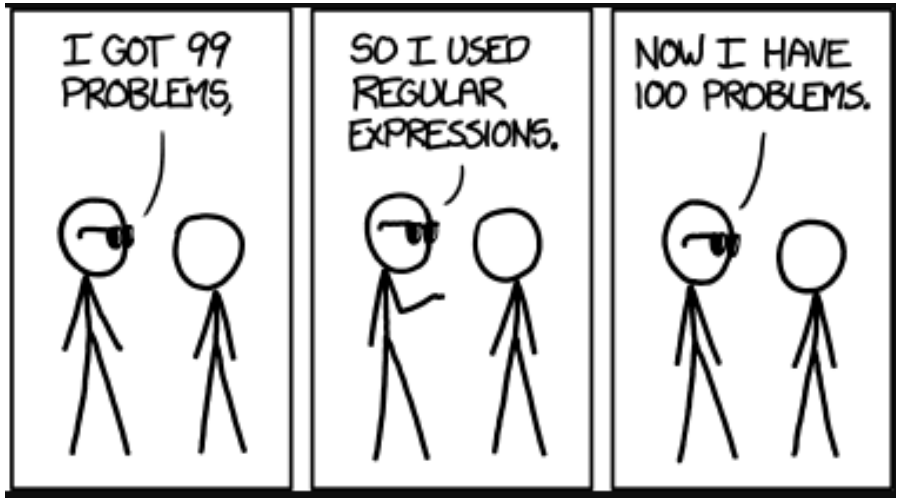
CMPSC 311 - Introduction to Systems Programming

PennState

Regular Expressions

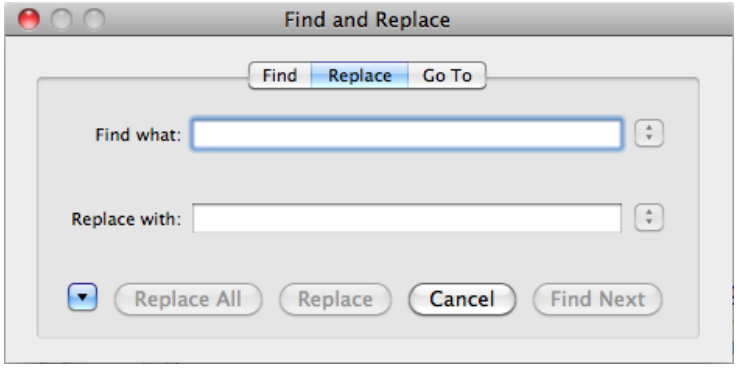
Professors:
Suman Saha
(Slides are mostly by *Professor Patrick McDaniel* and *Professor Abutalib Aghayev*)



Regular expressions

PennState

- Often shortened to “regex” or (more rarely) “regexp”
- Regular expressions are a language for matching patterns
  - Super-powerful find and replace tool
  - Can be used on the command line tools, in shell scripts, as a text editor feature, or as part of a program language (e.g., Python, Perl)



What are they good for?

PennState

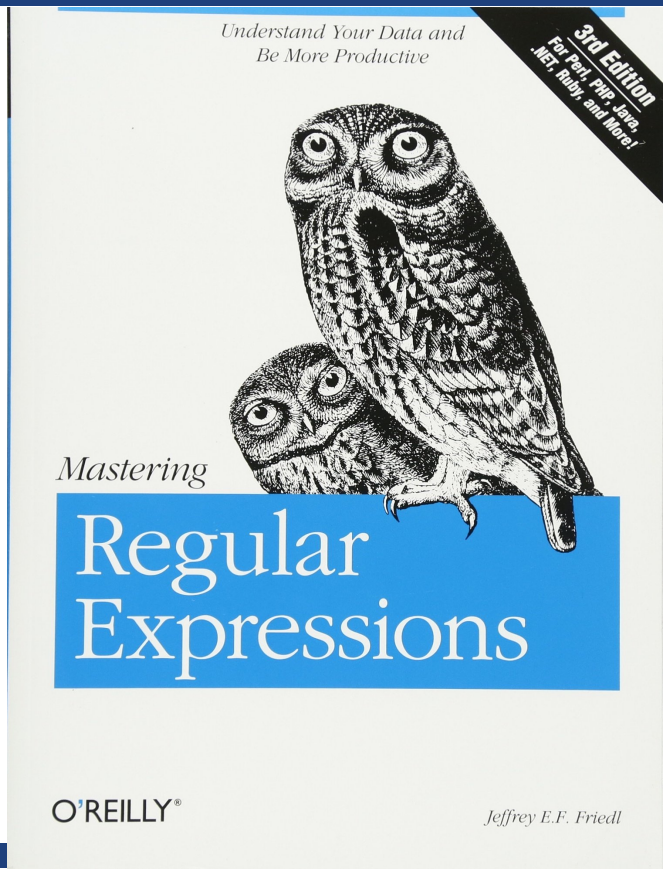
- Searching for specifically formatted text
  - Email address
  - Phone number
  - Anything that follows a pattern
- Validating input
  - Same idea
- Powerful find-and-replace
  - E.g. change “X and Y” to “Y and X” for any X, Y

Regex “flavors”

PennState

Regular Expression Flavors (Engines) (source: https://gist.github.com/CMCDragonkai/)

- JGsoft: This flavor is used by the Just Great Software products, including PowerGREP and EditPad Pro.
- .NET: This flavor is used by programming languages based on the Microsoft .NET framework versions 1.x, 2.0 or 3.x. It is generally also the regex flavor used by applications developed in these programming languages.
- Java: The regex flavor of the java.util.regex package, available in the Java 4 (JDK 1.4.x) and later. A few features were added in Java 5 (JDK 1.5.x) and Java 6 (JDK 1.6.x). It is generally also the regex flavor used by applications developed in Java.
- Perl: The regex flavor used in the Perl programming language, versions 5.6 and 5.8. Versions prior to 5.6 do not support Unicode.
- PCRE: The open source PCRE library. The feature set described here is available in PCRE 5.x and 6.x. PCRE is the regex engine used by the TPerlRegEx Delphi component and the RegularExrpessions and RegularExpressionsCore units in Delphi XE and C++Builder XE.
- ECMA (JavaScript): The regular expression syntax defined in the 3rd edition of the ECMA-262 standard, which defines the scripting language commonly known as JavaScript.
- Python: The regex flavor supported by Python's built-in re module.
- Ruby: The regex flavor built into the Ruby programming language.
- Tcl ARE: The regex flavor developed by Henry Spencer for the regexp command in Tcl 8.2 and 8.4, dubbed Advanced Regular Expressions.
- POSIX BRE: Basic Regular Expressions as defined in the IEEE POSIX standard 1003.2.
- POSIX ERE: Extended Regular Expressions as defined in the IEEE POSIX standard 1003.2.
- GNU BRE: GNU Basic Regular Expressions, which are POSIX BRE with GNU extensions, used in the GNU implementations of classic UNIX tools.
- GNU ERE: GNU Extended Regular Expressions, which are POSIX ERE with GNU extensions, used in the GNU implementations of classic UNIX tools.
- XML: The regular expression flavor defined in the XML Schema standard.
- XPath: The regular expression flavor defined in the XQuery 1.0 and XPath 2.0 Functions and Operators standard.

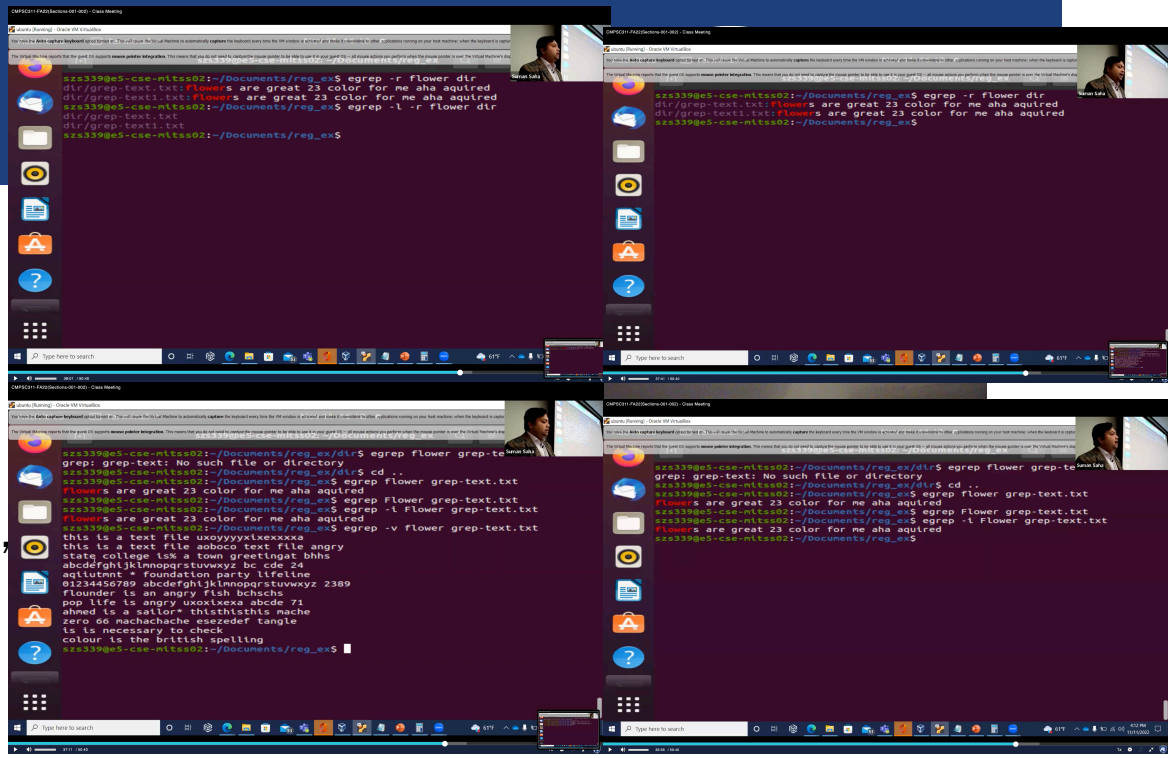


On the command line

- The grep command is a regex filter
  - That's what the "re" (regular expression) in the middle stands for
  - grep –F looks for literal strings
- Today we will use grep –E
  - E for "extended" regular expressions
  - Very close to other languages' flavors
  - Or egrep

grep command syntax

- To find matches in files:
  - grep -E regex file(s)
- To filter standard input:
  - grep -E regex
  - where regex is a regular expression, and file(s) are the files to search
- Options (aka “flags”):
  - -i: ignore case
  - -v: find non-matching lines (inverted search)
  - -r: search entire directories to find matches
  - -l: list the files that match, not the lines



Playing along …

PennState

- First, go to the course website and grab the `grep-text.txt` file and put it in your VM (it is under the files tab of canvas)
  - E1*: Now test it out:

```
% grep -E this grep-text.txt
this is a text file
this is a text file text file
%
```

*These numbered exercises are used throughout this lecture to help illustrate regular expression use. Answer key given later.

First lesson

PennState

- Letters, numbers, and a few other things match literally
  - E2: Find all the lines containing "`fgh`" $ egrep fgh grep-text.txt
  - E3: Find all the lines containing "`lmn`" $ egrep lmn grep-text.txt

- Note: a regex can match anywhere in the string
  - Doesn't have to match the whole string

Anchors

PennState

- Caret `^` matches at the beginning of a line
- Dollar sign `$` matches at the end of a line
  - Use `"..."` to protect characters from the shell, e.g.,

```
grep -E "blah" grep-text.txt
```

- Try it
  - E4: Find words ending in "`gry`"
  - E5: Find words starting with "`ah`"

```
egrep "gry$" grep-text.txt
egrep "^ah" grep-text.txt
egrep "^ah$" /user/share/dict/words
```

directory

- What happens if we use both anchors?

only `ah`

Single-character wildcard

PennState

- Dot . matches any single character (exactly one)
  - E6: Find the lines line with 6-letter word where the second, fourth, and sixth letters are “o”
  - E7: Find lines with words that start with ”gr” and end with ”at” with exactly one character in the middle

egrep ".0.0.0" grep-text.txt

egrep "gr.at" ...

Multi-character wildcard

PennState

- Dot-star `.*` will match 0 or more characters
  - We'll see why on the next slide
  - E8: Find any words that start with "`gr`" and end with "`at`" with any number of characters in the middle

egrep "gr.*at" grep-text.txt

Quantifiers

PennState

- How many repetitions of the previous thing to match?
  - Star `*`: 0 or more
  - Plus `+`: at least 1
  - Question mark `?`: 0 or 1 (i.e., optional)
  - Brackets `{#}` : exactly # of times

`{#}` e.g. `egrep "o{2}" /user/share/dict/words`

- Try it out
  - E9: Spell check: `necc?ess?ary`
  - E10: Outside the US: colou?r
  - E11: Find lines with words with `u`, `o`, `i`, `e`, `a` in that order and at least one letter in between each

`egrep "u.+o.+i.+e.+a" grep_text.txt`

Character classes and spaces

PennState

- Spaces can be specified with "`\s`" egrep "\sgr.at" grep-text.txt
- Square brackets `[abc]` will match any one of the enclosed characters
  - You can use quantifiers on character classes
    - E12: Find words starting with `b` where all the rest of the letters are `c`, `h`, or `s`

egrep "b[chs]+" grep-text.txt

Ranges

PennState

- Part of character classes
- You can specify a range of characters with `[a-j]`
  - One hex digit: `[0-9a-f]`
  - Consonants: `[b-df-hj-np-tv-z]`
  - E13: Find lines that have all the words you can make exclusively with `a` through `e`

```
egrep "[a-e]+" grep_test.txt
```

Negative character classes

- If the first character is a caret, matches anything except these characters
  - Consonants: `[^aeiou]`
  - Can be combined with ranges
    - Any character that isn't a digit: `[^0-9]`
  - E14: Find words that contain "`aq`", followed by something other than "`u`"

egrep "aq[^u]" grep-text.txt

( ) group

[ ] set

[a-b] range
[^□] match every thing except □

Groups

PennState

- Parentheses `(...)` create groups within a regex
  - Quantifiers operate on the entire group
  - E15: Find words with an "`m`", followed by "`ach`" one or more times, followed by "`e`"
  - E16: Find words where every other character, starting with the first, is an "`e`"

egrep "m(ach)+e" grep_text.txt

egrep "(e.)+" grep_text.txt

Branches if/else

PennState

- The vertical bar | denotes that either the left or right side matches
  - It's the "or" operator
  - Useful inside parentheses
  - E17: Find lines with the word "this" or "fish"

egrep "this|fish" grep_test.txt

Special characters

PennState

- We've seen a lot already
  - `^$.*+?[]()|\`
- Backslash `\` will escape a special character to search for it literally
  - E18: Search for all lines with an asterisk (`*`)

egrep "*" grep_text.txt

Backreferences

PennState

- Groups in `( )` can be referred to later
  - Must match the exact same characters again
  - Numbered `\1`, `\2`, `\3` from the start of the regex
  - E19: Find lines that have words that have a four-character sequence repeated immediately

e.g. = "x(abc).+(abc).+(abc)"

↳ = "x(abc).+(abc).+\1"

\1 \2 = "\1" means the first group from the start of the regex

↳ egrep "(....)\1" grep_text.txt

Want to learn more?

PennState

- regexcrossword.com
  - Great way to practice your regex-fu
  - Starts with simpler tutorial puzzles and works up
- Write your own regex engine