CSE 391

Regular Expressions

grep

Slides created by Josh Ervin. Based off slides made by Marty Stepp, Jessica Miller, Ruth Anderson, Brett Wortzman, and Zorah Fung

pollev.com/cse391

- How was the last homework assignment? How much time did it take?

ROADMAP

- Introduction to the command line
- Input/output redirection, pipes
- More input/output redirection, tee, xargs
- Git: Fundamentals
- Git: Branches, merging, and remote repositories
- Regular expressions
- More regular expressions, sed
- Users and permissions
- Bash scripting
- Industry applications

AGENDA

- Some notes on git
- Regular expressions
  - Regular expression syntax
  - Get started early-ish on the next homework assignment!

REGULAR EXPRESSIONS

- A *regular expression*, often referred to as a *regex,* is a description of a pattern of text.
    - Have a formal mathematical definition (DFA's in CSE 311)
- There are countless applications of regular expressions
    - Search for text in a given file (Purpose of this lecture)
    - Search/Replace text in a file
- Virtually all programming languages implement regular expressions
    - You've all see this before in java using `String split` (Grammar Solver, split on whitespace)
    - Regular expressions differ between programming languages
    - We will be using regular expressions in the context of `grep`.

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “Chocolate” candies.txt
```

- Notes:
  - grep is case-sensitive
  - Notice that running grep from a terminal prints out the entire line

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -Ei “Chocolate” candies.txt
```

- Notes:
  - We can make grep case-insensitive with -i
  - Notice that running grep from a terminal prints out the entire line

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “a” candies.txt
```

- Notes:
  - grep is case-sensitive
  - grep will highlight multiple matches per line

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -Ei “a” candies.txt
```

- Notes:
  - We can make grep case-insensitive with -i
  - grep will highlight multiple matches per line

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “ar” candies.txt
```

- Notes:
    - When we type a literal string (i.e. “ar”), grep will only print exact matches to this string. Subsets are not matches.
    -

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “.a” candies.txt
```

- Notes:
  - The . character matches **anything**

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “^K” candies.txt
```

- Notes:
  - ^ matches the beginning of the line

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “\<T” candies.txt
```

- Notes:
    - \< matches the start of a word
    - A word is considered a string of characters consisting of only letters, numbers, and underscores

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “t\>” candies.txt
```

- Notes:
    - \> matches the end of a word
    - A word is considered a string of characters consisting of only letters, numbers, and underscores

SIMPLE REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat.
Dark chocolate
```

```
grep -E “\.” candies.txt
```

- Notes:
  - \ is an escape character.
  - For the example above, this means “search for a literal period”

GLOSSARY

Syntax	Functionality
`.`	Any character
`^`	Start of line
`$`	End of line
`\<`	Start of word
`\>`	End of word
`\`	Escape the following character
`-i`	(Flag to grep) match case insensitively

THINK: pollev.com/cse391

candies.txt

```
Twix
Sweet Tart
Chocolate
Almond Joy
Jolly Ranchers
Kit Kats
Dark chocolate
```

Suppose we have the file candies.txt on the left. What is the full grep command to print out all lines that contain four-letter words that start with the letter T (Uppercase)?

Syntax	Functionality
.	Any character
^	Start of line
$	End of line
\<	Start of word
\>	End of word

PAIR: pollev.com/cse391

candies.txt

```
Twix
Sweet Tart
Chocolate
Almond Joy
Jolly Ranchers
Kit Kats
Dark chocolate
```

Suppose we have the file candies.txt on the left. What is the full grep command to print out all lines that contain four-letter words that start with the letter T (Uppercase)?

Syntax	Functionality
.	Any character
^	Start of line
$	End of line
\\<	Start of word
\\>	End of word

REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat
Dark chocolate
```

```
grep -E “Twix|Tarts” candies.txt
```

- Notes:
  - | is the logical or, it matches the thing on the left or the thing on the right.
  - **WARNING**: be really careful of spaces. Leading/trailing spaces after the | are interpreted literally. For example: “Twix | Tarts” searches for “Twix “ and “ Tarts”

REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat
Dark chocolate
Reeses
```

```
grep -E “(e|a)t” candies.txt
```

- Notes:
  - You can use parentheses to capture groups and encapsulate logical operators.
  - The above regex looks for all strings of two characters: either et or at

REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat
Dark chocolate
Reeses
```

```
grep -E “e*t” candies.txt
```

- Notes:
  - The * operator means “**zero** or more of the previous pattern”
  - This translates to “zero or more e’s followed by a t”

REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat
Dark chocolate
Reeses
```

```
grep -E “e+t” candies.txt
```

- Notes:
  - The + operator means “**one** or more of the previous pattern”
  - This translates to “one or more e’s followed by a t”

REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat
Dark chocolate
Reeses
```

```
grep -E “r?t” candies.txt
```

- Notes:
    - The ? operator means “**zero or one** of the previous pattern”
    - This translates to “rt or t”

REGEX

candies.txt

```
Twix
Sweet Tarts
Chocolate
Almond Joy
Jolly Ranchers
Kit Kat
Dark chocolate
Reeses
```

```
grep -E “(es)+” candies.txt
```

- Notes:
  - Wildcard operators, such as *, +, and ? can be applied to groups of characters using parentheses

GLOSSARY

Syntax	Functionality
|	Logical or
*	Zero or more of
+	One or more of
?	Zero or one of
( )	Group characters together

THINK: pollev.com/cse391

kitkats.txt

```
KitKats
Kit Kats
Kit kat
kitkats
Kit Kats
kitkat
kitkats
KITKATS
```

Suppose we have the file candies.txt on the left. We want to print out all lines containing kitkats. All of the following criteria are valid:

1) Starts with an upper or lowercase K
2) May or may not contain a space after the first t
3) The second K may be upper or lowercase
4) Optionally ends with an s

Syntax	Functionality
|	Logical or
*	Zero or more
+	One or more
?	Zero or one
( )	Group characters

PAIR: pollev.com/cse391

kitkats.txt

```
KitKats
Kit Kats
Kit kat
kitkats
Kit Kats
kitkat
kitkats
KITKATS
```

Suppose we have the file candies.txt on the left. We want to print out all lines containing kitkats. All of the following criteria are valid:

1) Starts with an upper or lowercase K
2) May or may not contain a space after the first t
3) The second K may be upper or lowercase
4) Optionally ends with an s

Syntax	Functionality
|	Logical or
*	Zero or more
+	One or more
?	Zero or one
( )	Group characters

REGEX

passwords.txt

```
password123
supersecret
NoHackingPlz
122password
Not a valid password.
won’t remember
```

```
grep -E “[abc]” passwords.txt
```

- Notes:
    - [ ] denotes a **character set**. It is equivalent to an or statement, but is easier to read
        - [abc] = (a|b|c)
    -

REGEX

passwords.txt

```
password123
supersecret
NoHackingPlz
122password
Not a valid password.
won't remember
```

```
grep -E "[a-z]" passwords.txt
```

- Notes:
  - a-z is a range of characters. It's some shorthand included to make things like letters and numbers easier to write
    - a-z is all lowercase letters
    - A-Z is all uppercase letters
    - 0-9 is all digits
    - a-zA-Z is all upper and lowercase letters

REGEX

passwords.txt

```
password123
supersecret
NoHackingPlz
122password
Not a valid password.
won't remember
```

```
grep -E "[.]" passwords.txt
```

- Notes:
  - Special characters inside of character sets do not usually need to be escaped.

REGEX

passwords.txt

```
password123
supersecret
NoHackingPlz
122password
Not a valid password.
won't remember
```

```
grep -E "[^ao]" passwords.txt
```

- Notes:
  - When inside of a character set, ^ negates all of the contents of it.
  - **WARNING:** This is different from the ^ outside of the character set, which denotes the start of the line.

REGEX

passwords.txt

```
password123
supersecret
NoHackingPlz
122password
Not a valid password.
won't remember
```

```
grep -E "[0-9]{2}" passwords.txt
```

- Notes:
  - Curly braces are another type of modifier that allow us to specify how many of a certain match we want
  - {x} - match exactly x characters of the pattern
  - {x,} - match x or more characters of the pattern
  - {,x} - match x or fewer characters of the pattern
  - {x,y} - match between x and y characters of the pattern
  - All of these ranges are inclusive

REGEX

passwords.txt

```
password123
supersecret
NoHackingPlz
122password
Not a valid password.
won't remember
```

```
grep -E “(..)\1” passwords.txt
```

- Notes:
  - The \1 is a back reference - it is used to reference an earlier match.
  - The above string is search for “any two characters, followed immediately by those same two characters.”
  - If you have more than one grouping (i.e. things captured by parentheses) you can reference them with \1, \2, \3, etc

GLOSSARY

Syntax	Functionality
`[ ]`	Character set
`[^ ]`	Negate character set
`[a-z]`	All lowercase characters
`[A-Z]`	All uppercase characters
`[0-9]`	All digits
`\1`	Back reference earlier character