Operating System fundamentals

Regular expressions

Karel de Grote Hogeschool

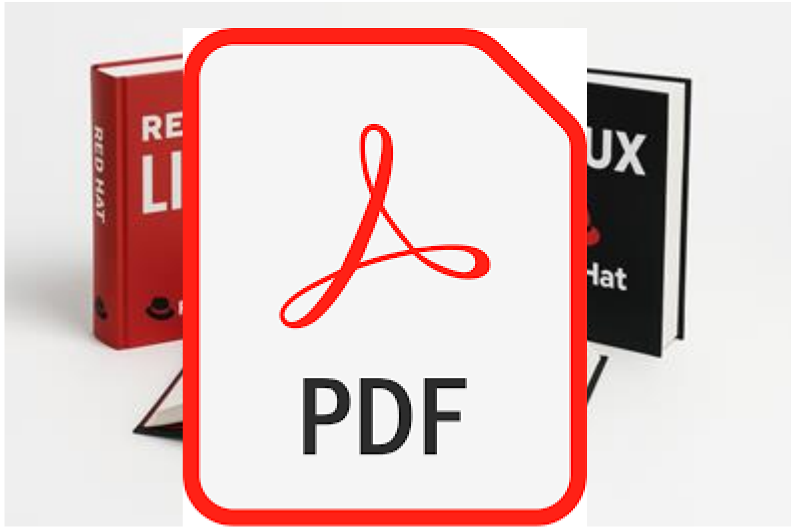
Content

Karel de Grote Hogeschool

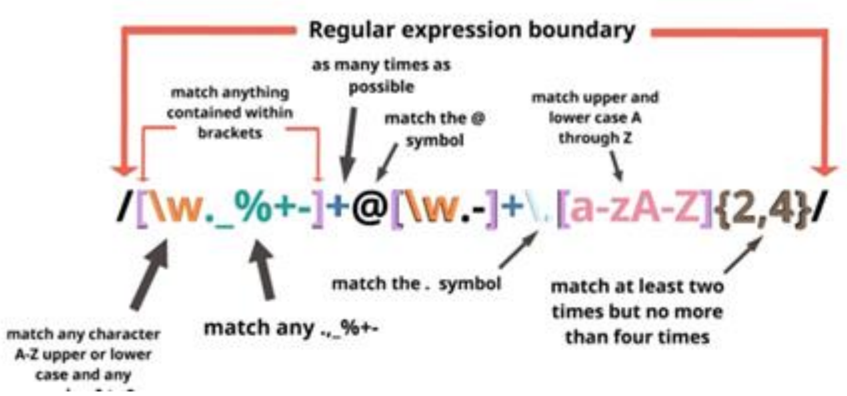
Course text

- chapter 6: Text Files - Regular Expressions

  → no RedHat material



Regular expressions with grep


Regular expression boundary
as many times as possible
match anything contained within brackets
match the @ symbol
match upper and lower case A through Z
/[\w._%+-]+@[\w.-]+\.[a-zA-Z]{2,4}/
match the . symbol
match at least two times but no more than four times
match any character A-Z upper or lower case and any
match any ._%+-

Regular expressions

Searching/Matching: to find occurrences of a specific pattern within a larger body of text.

  - e.g.: a word, a telephone number, an account number, …

- Validation eg. Email address
- Extraction/parsing eg. extracting domain names
- Substitution and Replacement: to find occurrences of a pattern and replace them with other text.
- Text Processing and Transformation: to manipulate large amounts of text programmatically, often involving a combination of searching, extracting, and replacing.

Karel de Grote Hogeschool

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

https://xkcd.com/208/

Karel de Grote Hogeschool

Searching text: grep

- Look for a pattern in a text
  - cat /etc/passwd | **grep** 'nobody'
  - cat /etc/passwd | **grep -i** 'nobody' → case **i**nsensitive
  - cat /etc/passwd | **grep -v** 'nobody' → show the lines that do NOT contain the pattern (in**v**ert match)
  - cat /etc/passwd | **grep -E 'w{3}'** → the pattern is an 'extended regular expression'
  - (sudo) grep **-l 'status'** /var/log/*log → show a **l**ist of filenames that contain the pattern
  - grep **-s 'bla' /root/*** → suppress error messages (**s**ilent)

Karel de Grote Hogeschool

Examples

Find this file on the class server:
/opt/share/example/text.txt

```
cat text.txt | grep -E 'word'
     grep -E 'word' text.txt
     egrep 'word' text.txt
cat text.txt | grep -vE 'word'
cat text.txt | grep -ivE 'word'
cat text.txt | grep -E 'w..d'
cat text.txt | grep -E 'w.rd'
cat text.txt | grep -E 'word.'
cat text.txt | grep -E 'word\.'
cat text.txt | grep -E 'w[aeiou]rd'
cat text.txt | grep -E '[A-Z]ord'
cat text.txt | grep -E '(the|a)'
cat text.txt | grep -E 'th(at|is)'
```

```
Here is a word.
And here is another word, but with more after it.
The plural of word is words, but you already knew that.
Many words have already been written.
Words can also be written with a capital character.
If I write w0rd, that is a typo.
The pattern to search for is not here...
Blablabla too many words
I wonder where king Arthur left his sword
And here's a word just before the newline character: word
```

best practice:
always put the expression between double or single quotes

Karel de Grote Hogeschool

Regular expressions

grep -E '**<**blah': a word beginning with “blah”

grep -E '**b**blah' a word beginning with “blah” (recommended)

grep -E 'blah**>**': a word ending in “blah” (even if followed by punctuation)

grep -E 'blah**b**' a word ending in “blah” (recommended)

grep -E '**^**blah': 'blah' at the beginning of a line

grep -E 'blah**$**': 'blah' at the end of a line

Exercise

1. Find lines where 'word' appears at the beginning of a line.
2. Do this again, but 'word' can also start with a capital character.
3. Now do it again, but 'word' must be at the end of a line and be a whole word (not part of a larger word)

Karel de Grote Hogeschool

Regular expressions

- Square brackets choose 1 character out of a list:
  - '**[aeiou]**': a vowel
  - '**[a-z]**' or **'[[:lower:]]'** : 1 lower case character
  - '**[0-9]**' or **'[[:digit:]]'**: 1 digit
  - '**[[:alnum:]]**': 1 character or digit
  - '**[[:alpha:]]**': 1 character
  - '**[[:blank:]]**': 1 space , tab
  - '**[[:space:]]**': 1 space, tab, newline, cr, vertical tab, or form feed

  - **'[^abc]'** : not the character a, nor b, nor c

  - … (see PDF course text for more posix character classes)

Regular expressions

- Repeating patterns
  - [[:lower:]]**+** → 1 or more lower case characters
  - [[:lower:]]**?** → 0 or 1 lower case character
  - [[:lower:]]***** → 0 or more lower case characters
  - [[:lower:]]**{5}** → exactly 5 times a lower case character

Regular expressions

Meta characters:

- . (dot): ANY ONE character except newline.
  Same as [^\n]
- \w : ANY ONE word character
  Same as [a-zA-Z0-9_] or [[:alnum::]]
- \b : ANY ONE word boundary character
- \< : start of word
- \> : end of word

Karel de Grote Hogeschool

Regular expressions

Escape character:

If you want to use a special character in your regular expression, you need to use an escape character: \

eg. grep -E '**\\$**20**\\.**00' :

Search for the string: **$**20**.**00

File globbing vs regex

don’t confuse these!

File “globbing”

* = 0..n random characters

? = 1 random character

. = the “.” symbol

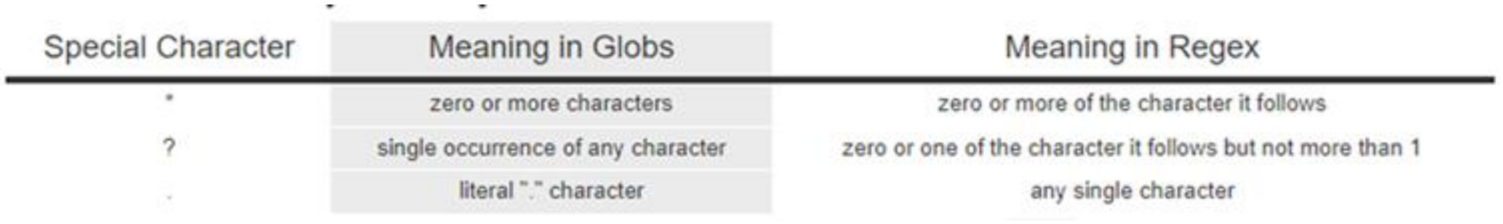
```
ls *.txt
```

Regex

x* = 0..n times the previous character

x? = 0 of 1 times the previous character

. = one random character



Special Character	Meaning in Globs	Meaning in Regex
*	zero or more characters	zero or more of the character it follows
?	single occurrence of any character	zero or one of the character it follows but not more than 1
.	literal "." character	any single character

Regular expressions in Java

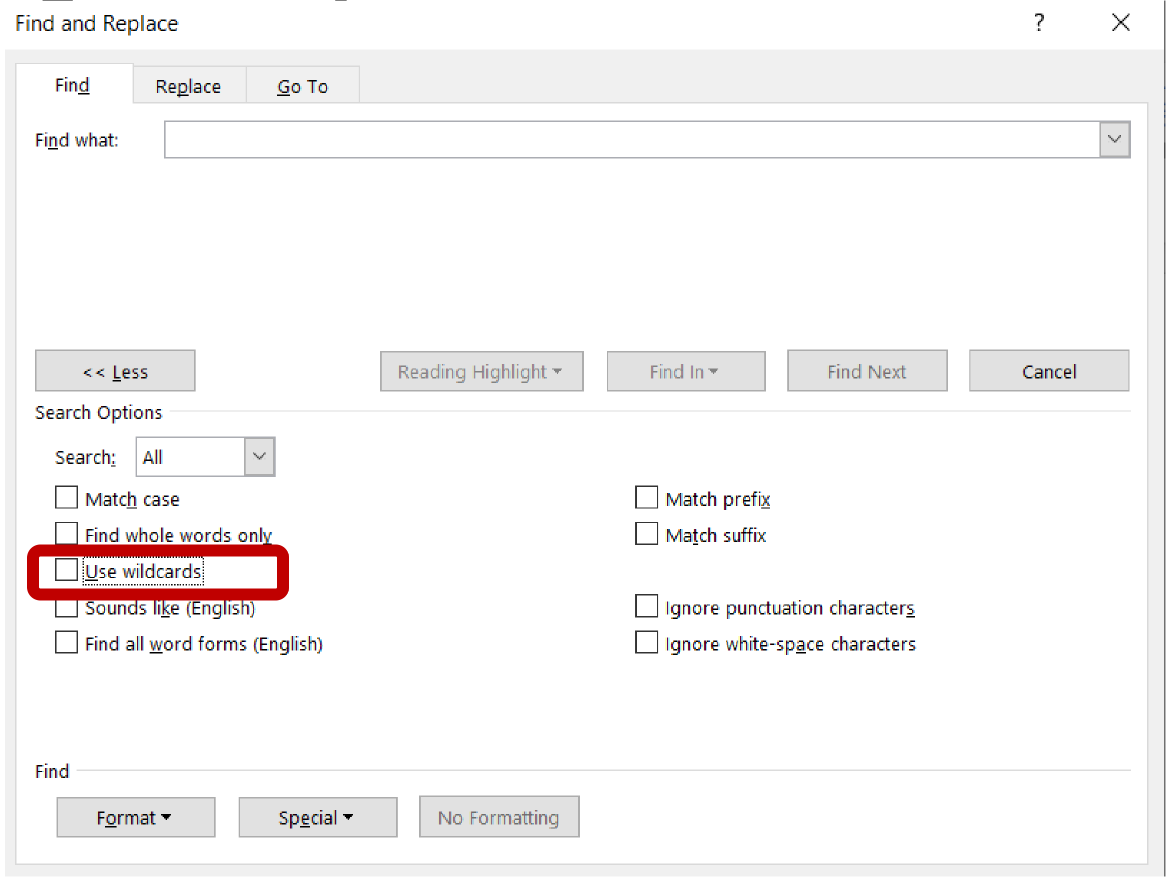
```
Scanner keyboard = new Scanner(System.in);
boolean inputIsCorrect;
do {
  System.out.print("Enter a word: ");
  String word = keyboard.nextLine();
  inputIsCorrect = word.matches("[a-zA-Z]+");
  if (!inputIsCorrect) {
    System.err.println("You may only enter alphabet
                            characters!!!");
  }
} while (!inputIsCorrect)
```

Karel de Grote Hogeschool

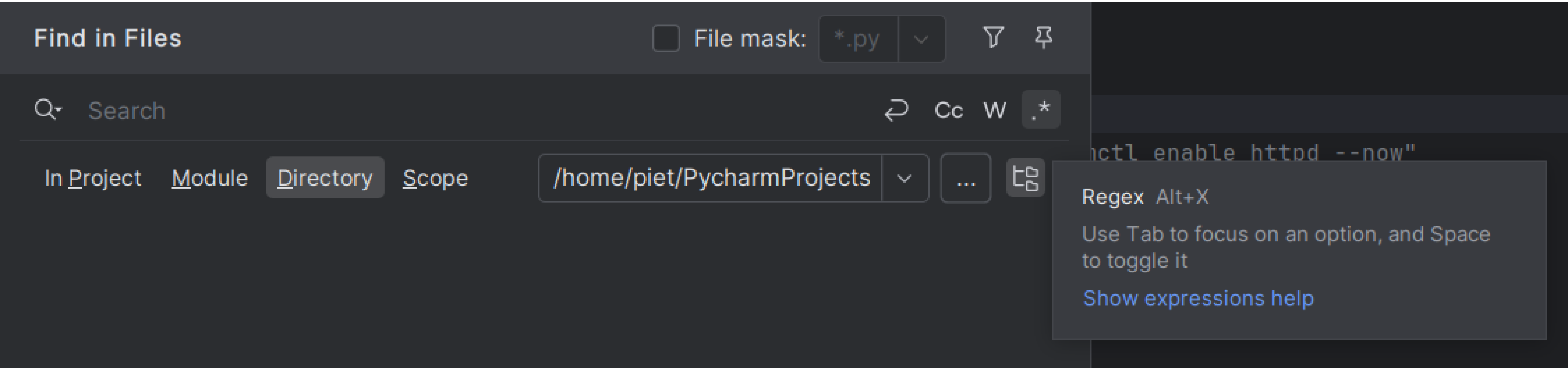
Regular expressions in SQL

```sql
-- SQL code to select valid French licence plates
-- 2 chars, followed by 3 digits, followed by 2 chars.
-- chars cannot be IOU
SELECT *
FROM cars
WHERE licence_plate ~ '^[A-HJ-NP-TV-Z]{2}-[[:digit:]]{3}-[A-HJ-NP-TV-Z]{2}$';
```

Regular expressions in MS Office - Word



Regular expressions in IntelliJ - PyCharm



```
ctl enable httpd --now"
```

Regex Alt+X

Use Tab to focus on an option, and Space to toggle it

Show expressions help

Karel de Grote Hogeschool

Regular expressions in JavaScript

- Eg. Validate an email address in a form

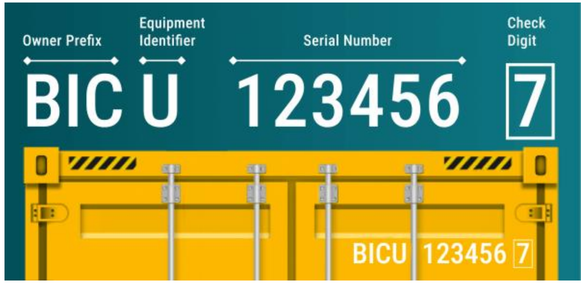
```
const email = "user@example.com";
const regex = /^[\w.-]+@[\w.-]+\.[A-Za-z]{2,}$/;

console.log(regex.test(email)); // true
```

Exercise

- How can you recognise Belgian licence plates? 1-HDY-463, 9-IVG-537, …
  Use sample file from /opt/share/example/licence_plate

- How to validate a shipping container number?
  - The first three characters of the container number represent the owner of the container. For example, “MAE” is the code for Maersk.
  - The fourth character indicates the type of equipment, such as a freight container (U), trailer (Z), or detachable container-related equipment (J).
  - The following six digits are a unique serial number assigned by the container owner.
  - The final (7th) digit is a check digit, used to verify the accuracy of the container number.



Exercise: auth.log

Find auth.log on classerver in folder /opt/share/example/

- Use the grep command to show only the log lines where an invalid user tried to log in:

```
2025-03-02T02:18:28.606615+01:00 rpi2 sshd[14681]: Invalid user admin from 196.251.87.45 port 
39134
```

Exercise: auth.log

- Use grep to show only log lines from **March 1 till March 3**, 2025.
- How many times does the string authentication failure occur?
- Create an overview of successful logins.
- Create a list of all lines containing the string **session opened** but <u>not</u> the string **CRON**.

Karel de Grote Hogeschool

2

Sed

Editing text: sed

- sed is a 'line-editor'
- sed is <u>not interactive</u> like vi or nano: it reads from standard input, performs an action on that and writes the result to standard output
- sometimes called a 'stream-editor' (used in pipelines)
- output goes to standard output, not to a file
  (unless it is redirected, or using an option)

Editing text: sed

- possible actions
  - Replace a pattern (regex) with something else.
  - Delete lines.
  - Print lines.
  - Insert/append lines.

sed: substitute (find&replace)

- cat /etc/passwd | **sed -E 's/home/Users/g'**
  - Replaces 'home' with 'Users' everywhere on the line.
- cat /etc/passwd | **sed -E 's_home_Users_g'**
  - same as above
- echo 'hahahaahaa' > file
- **sed -E -i 's/haa/hoo/'** file
  - Replaces the first 'haa' with 'hoo'.
  - The result is written back into the file.
  - Don’t use this in a pipeline
- echo '123 abc 456' | **sed -E 's/[0-9]+/& &/'**
  - Repeats the first number on the line
  - ‘&’ stands for the recognized pattern.
- echo '123 abc 456' | **sed -E 's/[0-9]+/& &/g'**
  - Repeats all numbers on the line
  - g stands for global

Exercise

1. How can you put all numbers in the file /etc/passwd between parentheses?
2. What if you also want to put decimal numbers (e.g., 3.14) between parentheses?

sed: editing

[address]	[command]	[arguments]
1 = first line	s(ubstitute)	/<pattern>/<subst>/g
$ = last line	d(elete)	
1,5 = lines 1-5	a(ppend)	<text>
/pattern/	c(change)	<text>
	i(insert)	<text>
…	…	

sed '1i line1' : insert line before the first

sed '3a line5': insert line after the third

sed '$d' : delete last line

sed '1,5s/X/Y/g': replace all occurances of X by Y on the first 5 lines

Karel de Grote Hogeschool

Exercise

1. Make a copy in your homedir of the file /opt/share/example/word.txt
2. Use a pipeline and redirection to add linenumbers to the word.txt file. (call the new file word_nl.txt)
3. Use sed to:
   - Delete the third line
   - Delete the last line
   - Insert "Hello Word" at the beginning of the file
   - Insert "This is the End" at the end of the file
   - Find and replace w0rd with word on line 6
   - Replace word or Word by WORD on the first 5 lines

sed: removing lines

- echo -e 'a\nb\nc\nd\ne\nf\ng\n' >text.txt
- nl text.txt | **sed '1d'**
  - removes line 1

- nl text.txt | **sed '2,4d'**
  - removes lines 2 till 4

- can you also do this with head and tail?

Karel de Grote Hogeschool

sed: printing lines

- nl text.txt | **sed -n '2,4p'**
- can you also do this with head and tail?

sed: adding lines

- nl text.txt | sed '**4a** test'
- nl text.txt | sed '**4i** test'
- Why is the new line not numbered? How can you number it?

Exercises

Karel de Grote Hogeschool

Exercises

- KdG
  - 6.1 Exercises: grep-sed-tr
  - 6.2 Exercises: Regex
  - 6.3 Fun Exercises
- RedHat
  - none