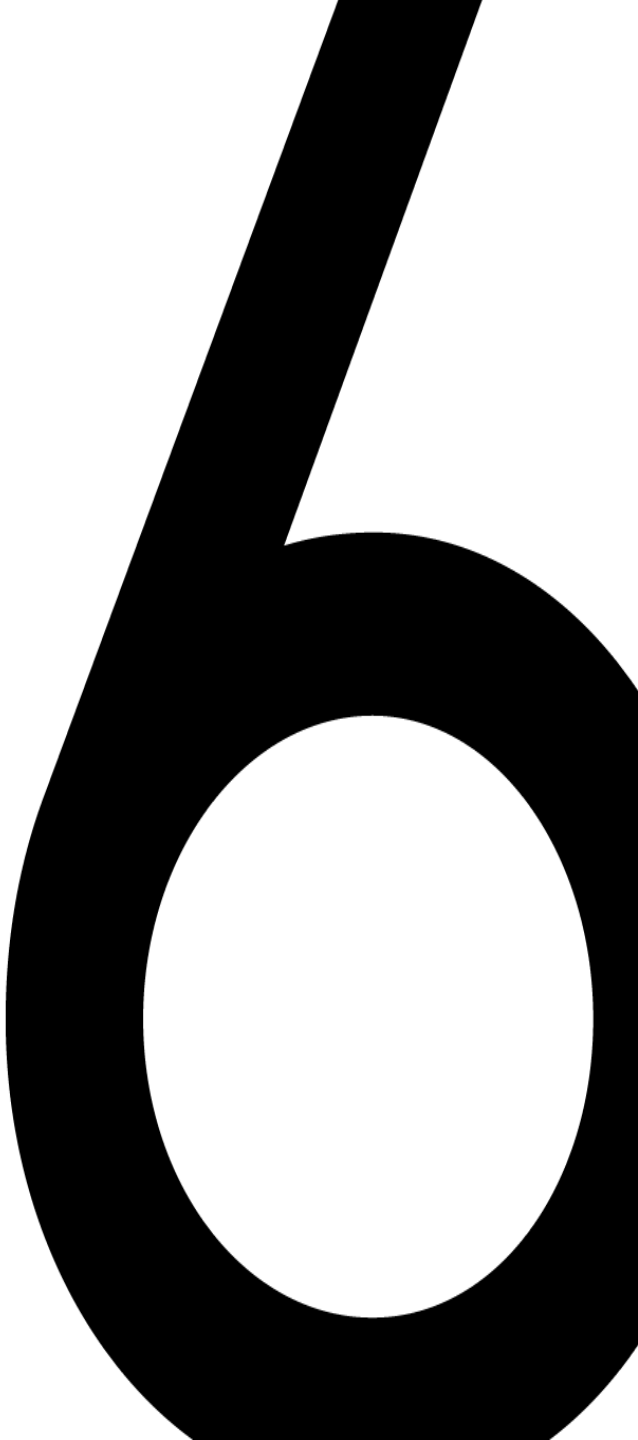
Bash Regex



REGEX

- soorten
- expressies
- =~
- groups
- extglob

Soorten regex

- POSIX2.0 Regex
  - Extended Regex (ERE)
  - egrep/bash
  - Oud: Basic Regex(BRE)
  - escapen van ? + { | (en)
  - standaard grep (zie CS1...)
- Perl Compatible Regex (PCRE)
  - UTF-8 en UNICODE support
  - perl/python/java/javascript/c#
  - grep -P

Extended en basic regex voorbeeld

- Extended regex met grep (egrep):
  - grep -E "groen|rood" bestand.txt
- Basic regex met grep
  - grep "groen\|rood" bestand.txt

Enkele regex expressies

- .
- [a-z]
- [0-9]
- [a|b]
- a{4}
- a{1,4}
- +
- *

Enkele regex expressies

- . 1 karakter
- [a-z] karakter a tot en met z in kleine letters
- [0-9] getal 0 tot en met 9
- [a|b] karakter a of karakter b
- a{4} vier keer a
- a{1,4} van één tot 4 keer a
- + 1 tot n keer
- * 0 tot n keer

POSIX: Soorten karakters extended regex

- [[:digit:]] een cijfer
- [[:space:]] spatie,tab, newl, return
- [[:alnum:]] letters en cijfers
- [[:alpha:]] letters
- [[:blank:]] spatie en tab
- [[:lower:]] lowercase letters
- [[:print:]] afdrukbare karakters

=~ operator

```
#!/bin/bash
content="Karel de Grote-Hogeschool, Nationalestraat 5,B-2000 Antwerpen"
regex="B-[0-9]{4}"
 if [[ $content =~ $regex ]]
  then
    echo "postnummer gevonden" ;  exit 0
  else
    echo "postnummer niet gevonden" >&2 ; exit 1
fi
```

OPGELET: $regex is NOOIT met dubbele quotes!

${BASH_REMATCH[0]}

- ${BASH_REMATCH[0]} is een array met het resultaat van de match. Element 0 is de hele match

```
#!/bin/bash
# Functie: Regex voor een postnummer
content="Karel de Grote-Hogeschool, Nationalestraat 5,B-2000 Antwerpen"
regex="B-[0-9]{4}"

 [[ $content =~ $regex ]]
echo "${BASH_REMATCH[0]}"
```

Output: B-2000

regex voorbeeld: email adres

jan . *celis* @ *kdg* . *be*

[[:alnum:]]+ . [[:alnum:]]+ @ [[:alnum:]]+ . [[:alpha:]]+

Het eerste stuk kan ook zonder punt zijn, dus dat plaatsen we optioneel:

jancelis @ *kdg* . *be*

([[:alnum:]]+.){0,2} [[:alnum:]]+ @ [[:alnum:]]+. [[:alpha:]]+

Het tweede deel kan ook uit subdomeinen bestaan:

jan . *celis* @ *student* . *kdg* . *be*

([[:alnum:]]+.){0,2} [[:alnum:]]+ @ ([[:alnum:]]+.){1,3} [[:alpha:]]+

Het laatste domein deel is van 2 tot 3 karakters (zonder cijfers):

jan . *celis* @ *student* . *kdg* . *be*

([[:alnum:]]+.){0,2} [[:alnum:]]+ @ ([[:alnum:]]+.){1,3} [[:alpha:]]{2,3}

regex groups

- Een group selecteert met haakjes een onderdeel van de regex
- Het resultaat zit in array n ${BASH_REMATCH[n]}

```
content="Karel de Grote-Hogeschool, Nationalestraat 5,B-2000 Antwerpen"


regex="([- a-zA-Z]+), ([[:alpha:]]+) ([[:digit:]]+),(.*) (.*)"
[[ $content =~ $regex ]]
echo "Naam: ${BASH_REMATCH[1]}"
echo "Straat: ${BASH_REMATCH[2]} Nr: ${BASH_REMATCH[3]}"
echo "Postcode: ${BASH_REMATCH[4]} Stad: ${BASH_REMATCH[5]}"
```

Output:

Naam: Karel de Grote-Hogeschool

Straat: Nationalestraat Nr: 5

Postcode: 2000 Stad: Antwerpen

“Regex” met parameter substitution

```
#!/bin/bash

content="  Karel de Grote-Hogeschool, Nationalestraat 5,B-2000
Antwerpen"

shopt -q -s extglob

content2=${content#*+([[:digit:]])}
content=${content##*+([[:digit:]])}

shopt -q -u extglob

echo "ungreedy: $content2"

echo "greedy: $content"
```

```
Output:
 ungreedy: ,B-2000 Antwerpen
 greedy:  Antwerpen
```

http://wiki.bash-hackers.org/syntax/pattern

Linux

- Vraag:
  - Ik begrijp de opbouw van de regex niet helemaal die gebruikt wordt in de parametersubstitutie in volgende voorbeelden:
  - 1) zie cursus p 42: content=${content##*+([[:digit:]])} Dit verwijdert alles tot en met het eerste cijfer. Wat betekent de combinatie van * en + hier?
  - 2) content=${content##*([[:space:]])} Dit verwijdert alle leading spaties. Ook hier begrijp ik niet goed wat het sterretje wil zeggen.

- Verklaring:
  - Extglob = extended globbing
  - Zoals regular expressions, maar met een paar eigenaardigheden
  - quantifiers (+ * ? ...) hebben dezelfde betekenis als in regex, maar moeten voor het patroon (dat tussen ronde haakjes staat) staan ipv erachter

Linux

- Verklaring:
  - http://wiki.bash-hackers.org/syntax/pattern
  - Let op: regular expressions belangrijker! (Slides als randinfo.)

Linux

- Verklaring:

Normal pattern language

Sequence	Description
`*`	Matches **any string**, including the null string (empty string)
`?`	Matches any **single character**
`X`	Matches the character `X` which can be any character that has no special meaning
`\X`	Matches the character `X`, where the character's special meaning is stripped by the backslash
`\\`	Matches a backslash
`[…]`	Defines a pattern **bracket expression** (see below). Matches any of the enclosed characters at this position.

Linux

- Verklaring:

Bracket expressions

The bracket expression `[…]` mentioned above has some useful applications:

Bracket expression	Description
`[XYZ]`	The "normal" bracket expression, matching either `X`, `Y` or `Z`
`[X-Z]`	A range expression: Matching all the characters from `X` to `Y` (your current **locale**, defines how the characters are **sorted**!)
`[[:class:]]`	Matches all the characters defined by a POSIX® character class: `alnum`, `alpha`, `ascii`, `blank`, `cntrl`, `digit`, `graph`, `lower`, `print`, `punct`, `space`, `upper`, `word` and `xdigit`
`[^…]`	A negating expression: It matches all the characters that are **not** in the bracket expression
`[!…]`	Equivalent to `[^…]`
`[]...]` or `[-…]`	Used to include the characters `]` and `-` into the set, they need to be the first characters after the opening bracket
`[=C=]`	Matches any character that is eqivalent to the collation weight of `C` (current locale!)
`[[.SYMBOL.]]`	Matches the collating symbol `SYMBOL`

Linux

- Verklaring:

Examples

Some simple examples using normal pattern matching:

- Pattern `"Hello world"` matches
  - `Hello world`
- Pattern `[Hh]"ello world"` matches
  - ⇒ `Hello world`
  - ⇒ `hello world`
- Pattern `Hello*` matches (for example)
  - ⇒ `Hello world`
  - ⇒ `Helloworld`
  - ⇒ `HelloWoRlD`
  - ⇒ `Hello`
- Pattern `Hello world[[:punct:]]` matches (for example)
  - ⇒ `Hello world!`
  - ⇒ `Hello world.`
  - ⇒ `Hello world+`
  - ⇒ `Hello world?`
- Pattern `[[.backslash.]]Hello[[.vertical-line.]]world[[.exclamation-mark.]]` matches (using collation sybols)
  - ⇒ `\Hello|world!`

Linux

- Verklaring:

Extended pattern language

If you set the shell option `extglob`, Bash understands some powerful patterns. A `<PATTERN-LIST>` is one or more patterns, separated by the pipe-symbol (`PATTERN|PATTERN`).

`?(<PATTERN-LIST>)`	Matches **zero or one** occurrence of the given patterns
`*(<PATTERN-LIST>)`	Matches **zero or more** occurrences of the given patterns
`+(<PATTERN-LIST>)`	Matches **one or more** occurrences of the given patterns
`@(<PATTERN-LIST>)`	Matches **one** of the given patterns
`!(<PATTERN-LIST>)`	Matches anything **except** one of the given patterns

- Verklaring:
  - Extglob = extended globbing
  - Zoals regular expressions, maar met een paar eigenaardigheden
  - quantifiers (+ * ? ...) hebben dezelfde betekenis als in regex, maar moeten voor het patroon (dat tussen ronde haakjes staat) staan ipv erachter

Linux

- Vraag:
  - 1) content=${content##*+([[:digit:]])}
  - 2) content=${content##*([[:space:]])}

Linux

- Toepassing:
  - content=${content##*+([[:digit:]])}
    - String functie om te knippen
    - # een zo klein mogelijk string vinden
    - ## een zo groot mogelijk string vinden
    - 1 of meer cijfers (equivalent met [[:digit:]]+ in regex)
    - * = pattern = any string (including NULL)
  - Besluit:
    - Zoeken de grootst mogelijke string die start met eender wat en eindigt op 1 of meer getallen en knippen die weg.

Linux

- Toepassing:
  - content=${content##*+([[:digit:]])}
    - String functie om vooraan te knippen
    - # een zo klein mogelijk string vinden
    - ## een zo groot mogelijk string vinden
    - 1 of meer cijfers (equivalent met [[:digit:]]+ in regex)
    - * = pattern = any string (including NULL)
  - Besluit:
    - Zoeken de grootst mogelijke string die start met eender wat en eindigt op 1 of meer getallen en knippen die weg.

Linux

- Toepassing:
  - content=${content##*([[:space:]])}
    - String functie om vooraan te knippen
    - # een zo klein mogelijk string vinden
    - ## een zo groot mogelijk string vinden
    - 0 of meer cijfers (zie extended patterns)
  - Besluit:
    - Zoeken naar 0 of meer spaties en deze wegknippen.

Hulpmiddelen

- http://regexraptor.net/
- http://www.regexr.com/
- CheatSheet in cursus...