Reguliere expressies

```
^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$
```

In dit hoofdstuk maken we nader kennis met zoekpatronen (reguliere expressies) en bijbehorende commando's

Doelstelling

Aan het eind van dit hoofdstuk is de cursist bekend met:

- wat reguliere expressies zijn
- wat reguliere expressie operatoren zijn
- basis versus uitgebreide reguliere expressies
- het nut van reguliere expressies
- de grep familie en de stream editor sed
- nogmaals het belang van de shell en quoten

---

reguliere expressies

- veel filter commando's maken gebruik van zoekpatronen om bepaalde stukken tekst te selecteren.
- een moeilijk woord voor deze zoekpatronen: reguliere expressies
- reguliere expressie:
  - simpele string
  - gevormd met karakters die een speciale betekenis hebben. Die karakters worden dan reguliere expressie operatoren genoemd

---

reguliere expressie operatoren

`.`	op die plek mag een willekeurig karakter staan
`[abc]`	op die plek kan 1 van de karakters a, b of c staan
`[^abc]`	op die plek alle karakters **behalve** a, b of c
`[a-z]`	duidt een range van mogelijke karakters aan
	Speciale karakter ranges: [:upper:], [:lower:], [:alnum:], [:digit:], ...
`^`	dit duidt het begin van de regel aan
`$`	dit duidt het eind van de regel aan
`?`	het voorafgaande karakter komt 0 of 1 keer voor
`*`	het voorafgaande karakter komt 0 of meer keer voor
`+`	het voorafgaande karakter komt 1 of meer keer voor
`\`	ontdoe het volgende karakter van de speciale betekenis

reguliere expressie operatoren

Voorbeeld regel:

```
Er was eens een beginnend Linux gebruiker die vi leerde kennen.
```

Matchen de volgende zoekpatronen?

```
Unix
^E
^E.$
^E[[:alpha:][:space:]]+$
^E[[:alpha:][:space:]]+\.$
.*
^Er ww*
```

---

de grep familie (1/2)

Onderscheid basis en uitgebreide reguliere expressie operatoren:

De karakters `? + { } | ( )` behoren tot de uitgebreide reguliere expressie operatoren. Verschillende tools gaan verschillend om met de implementatie van basis en/of uitgebreide operatoren, het commando GNU **grep** kent ze allebei, maar:

- uitgebreide operatoren vooraf laten gaan door een `\` (!)
- optie -E (Extended) meegeven of **egrep** gebruiken

Gebruik grep: `grep [optie]... 'zoekpatroon' file...`

Enkele opties voor grep:

`-i` ignore case (geen onderscheid hoofd- en kleine letters)

`-v` laat juist de regels zien die **niet** voldoen aan de regexp

`-r` recursive - zoek in alle files onder een directory

`-l` laat alleen de filenamen zien die matchen en niet de regels

de grep familie (2/2)

```
egrep <=> grep -E
rgrep <=> grep -r
fgrep <=> grep -F
```

```
$ fgrep "Nijmegen
> Amsterdam" klanten
H.      Koster          Mr.     Kroosstraat 33          Amsterdam
Mc.     Smits           Miss    Koningsplein 3          Nijmegen
Ch.     Fuchte          Mr.     Staddijk 23             Nijmegen
```

`fgrep` hoeft geen rekening te houden met operatoren -> sneller

---

de stream editor sed (1/3)

- deze editor kan iedere bestandsgrootte aan
- wijzigt standaard niets, leg wijzigingen vast met redirection
- sed doet per regel:
  1) leest de regel van stdin en plaatst deze in een buffer
  2) kijk of de regel matcht met een selectie en zo ja voer het sed commando uit. Selectie van een regel kan op 2 manieren:
     - numeriek: op regelnummer
     - met een reguliere expressie: alleen regels die matchen komen in aanmerking
  3) Na het commando wordt de evt. veranderde regel naar output geschreven

Een 'sed programma' bestaat dus uit *adressen* (regelnummers of reguliere expressies) en commando's

de stream editor sed (2/3)

sed commando's:

- `a` (append) voegt toe na de regel(s)
- `i` (insert) voegt in voor de regel(s)
- `c` (change) vervangt de opgegeven regel(s)
- `d` (delete) verwijdert opgegeven regel(s)
- `s` (substitute) vervangt het meegegeven patroon
- `p` (print) toont regels, sed schijft standaard alle (ook de niet geslecteerde regels) naar output. Met de optie `-n` onderdruk je dit en dan kun je met `p` die regels printen die je wilt

---

de stream editor sed (3/3)

Voorbeelden:

- `sed -n '/Linux/p' file`

  laat alleen de regels met de string Linux zien (= `grep Linux file`)

- `sed 's/100/honderd/g' file`

  vervang op elke regel 100 door honderd voor **elke** (**g**lobal) instantie van 100 op de regel

- `sed '1,5 d' file`

  delete de eerste 5 regels

- ```
  sed '5,6 c\
  > regels 5 en 6 worden vervangen door\
  > deze tekst
  > ' file
  ```

  ```

de commandoregel en quoten

Belangrijk hoe de shell met de commandoregel om gaat:

- eerste wordt de hele commando regel gescanned (van links naar rechts)
- substituties en acties (bij bv. variabelen, wildcards, redirecten en pipes) worden gedaan als de shell deze tegenkomt
- **pas dan** komt het commando (of komen de commando's) in actie
- speciale karakters voor de shell o.a.: `$`, `*`, `?`, `[]`, `<`, `>`, `|`
- quote wat je verborgen **wilt** houden voor de shell!
  - enkele quotes verbergen **alle** speciale karakters voor de shell
  - dubbele quotes ook behalve de $, \ en `
  - de backslash voor een karakter is hetzelfde als enkele quotes om dat karakter

Oefeningen

Tijd voor oefening
