Filters, redirection, piping

In dit hoofdstuk maken we nader kennis met filter commando's, redirection, piping en de rol van de shell hierbij.

Doelstelling

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

- enkele filter commando's
- input, output en error *streams*
- redirection van deze streams
- command piping
- meer doen met filter commando's
- reguliere expressies
- de commando's grep en sed
- de shell en de commandoregel
- het belang van filtering (commando's als bouwstenen)

---

enkele filter commando's (1/6)

Veel commando's in Linux werken als een filter: aan de hand van bepaalde kenmerken en condities kunnen **delen** van tekst getoond worden

Filtering werkt met tekst-data (niet met binary data) en werkt **per regel**

Hier volgen enkele commando's met een filterwerking:

`cat [OPTION]... [FILE]...`

toont de inhoud van een bestand

- `-T` toon tabs als `^I`
- `-E` toon regeleinden als `$`
- `-vet` toon niet zichtbare karakters, tabs als `^I` en regeleinden als `$`

enkele filter commando's (2/6)

`head [optie]... [file]...`

- toon zonder opties de eerste 10 regels (”**head**”) van een bestand
- `-n` *n* toon de eerste *n* regels van een bestand. Als *n* negatief is toon dan alle regels behalve de laatste *n*
- `-c` *n* toon de eerste *n* bytes van een bestand. Als *n* negatief is toon dan alles behalve de laatste *n* bytes

`tail [optie]... [file]...`

- toont zonder opties de laatste 10 regels (”**tail**”) van een bestand
- `-n` *n* toon de laatste *n* regels
- `-n +`*n* toon alle regels beginnend bij de *n*-de regel
- `-c` *n* toon de laatste *n* bytes
- `-c +`*n* toon alle bytes beginnend bij de *n*-de byte

enkele filter commando's (3/6)

`cut optie... [file]...`

knip (**cut**) secties van elke regel

Je kunt per karakter of per veld knippen. De standaard veldscheider (*delimiter*) is de TAB

- `-c`*`lijst`* De lijst, volgend op `c` (zonder spatie), geeft **karakterposities** aan. De verschillende posities worden gescheiden door een komma of door een hyphen (-). Met een hyphen wordt een **range** aangegeven.
  - `cut -c1,5-10` toont van elke regel het eerste karakter en de karakters 5 t/m 10
- `-f`*`lijst`* de lijst volgend op `f` (zonder spatie), geeft **velden** aan. De verschillende veldaanduidingen worden gescheiden door een komma of een hyphen (-). Met een hyphen wordt een **range** aangegeven.
- `-d`*`FS`* gebruik een andere veldscheider dan de default TAB
  - bv. `cut -d: -f1,3,6,7 /etc/passwd`

---

enkele filter commando's (4/6)

`sort [optie]... [file]...`

sorteert alle regels

- zonder optie sorteert op ASCII volgorde (o.a. 0-9 < A-Z < a-z)
- `-n` numerieke sort
- `-r` draai resultaat om (*reverse*)
- `-t`*`FS`* wijzig de veldscheider (standaard de overgang van zichtbaar naar niet-zichtbare karakters)
- `-k pos1[,pos2]` sorteer alleen vanaf `pos1` t/m `pos2`. Positie wordt aangegeven met `F`*`n`*`[.C`*`n`*`]` (bv. `4.3` = veld `4`, `3`-de karakter).
  - `-k 2,5.3` sorteer vanaf 1e karakter van het 2e veld t/m het 3e karakter van het 5-de veld

---

enkele filter commando's (5/6)

`nl [optie]... [file]...`

geeft regelnummers

- `-b a` (alle regels), `t` (**niet** de lege regels)
- `-n {ln|rn|rz)`
  - `ln`: links georienteerd
  - `rn`: rechts georienteerd
  - `rz`: rechts georienteerd met voorafgaande nullen

`fmt [-width] [optie]... [file]...`

formatteer tekst

- `-w` *n* maak de output *n* wijd (kan ook met `fmt` *-n*, wel als eerste optie)
- `-s` breek lange regels af maar begin daarna weer op een nieuwe regel (in de opgaven wordt dit duidelijker...)

enkele filter commando's (6/6)

`expand [optie]... [file]...`

converteer tabs naar spaties

- `-i` converteer alleen ”leading” tabs
- `-t` *n* vervang tab door *n* spaties

`unexpand [optie]... [file]...nl [optie]... [file]...`

converteer spaties naar tabs, standaard alleen voor ”leading” spaties

- `-a` converteer alle spaties
- `-t` *n* converteer *n* spaties naar een tab

Oefeningen

Tijd voor wat oefening!

---

redirection

normaal:

- input van keyboard (stdin - *stream* 0)
- output naar scherm (stdout - *stream* 1)
- error output naar (stderr - *stream* 2)

Deze streams zijn te redirecten:

- input kan bv. komen uit een file i.p.v. het keyboard
  - redirecten met <

```
mail student2 < message.txt
```

- uitvoer kan bv. naar een file i.p.v. het scherm
  - stdout redirecten met >

```
ls -l > mijn_bestanden
```

  - stderr redirecten met 2>

```
find / -size +1G -exec ls -lh {} \;  2>/dev/null
```

---

append

Met `> file` en `2> file` wordt `file` **overschreven** als het bestaat.

Om data **toe te voegen (append)**: gebruik double output redirection (>>):

- ```
  cat >> klanten
  K.   Appel Mr.     Vrolikstraat 453   Amsterdam
  <Ctrl>-d
  ```

- `./myscript 2>> mijn_fouten`

Aparter: double **input** redirection (<<)

- ```
  cat >> klanten <<'Here'
  K.   Appel Mr.     Vrolikstraat 453   Amsterdam
  Here
  ```

Ook wel een ”here-document” genoemd. String `Here` is vrij te kiezen

Handig voor (non interactief) gebruik in scripts

Combineer stdout en stderr

Redirect stdout en stderr naar hetzelfde file:

Fout:

- `commando >output.txt 2>output.txt`

Goed:

- `commando >output.txt 2>&1` (”*koppel 2 aan waar stream 1 op dat moment gekoppeld is*”).

Anders dan (!):

- `commando 2>&1 >output.txt`

Command piping

Niet alleen redirection van/naar **files**: ook van/naar **commando's:** de output van het ene commando kan als input dienen voor het andere commando

- `sort tekst.zootje | lpr` (**output** van `sort` komt op de **input** van `lpr` waarmee het bestand `tekst.zootje` gesorteeerd op de printer komt.

Een van de krachten van Linux: kleine utilities vormen samen een krachtiger geheel!

Vandaar het belang voor een utility: **doe maar 1 ding en doe het op een standaard manier**: dus lezen van stdin en schrijven naar stdout. Dan zijn deze utilities goed te combineren.

Bijvoorbeeld:

- `ls` toont het aantal files in de huidige directory, **1 file per regel**
- het commando `wc -l` telt regels
    - `ls | wc -l` geeft dus het aantal files in de huidige directory

---

combineren

Command piping en redirection ook goed te combineren:

- `du -s /var/log/* | sort -rn > usage.txt`

Speciaal:

het commando `tee`

- Data zowel door pipe als naar file ('T' splitsing):
    - `sort tekst.zootje | tee tekst.geordend | lpr`

het commando `script`

- Neemt uw ”werk” (input en output) op
    - handig om te laten zien wat u hebt gedaan
    - output komt standaard in het file `typescript` (maar u kunt een andere filenaam als argument meegeven
    - er wordt een aparte shell gestart, `script` afsluiten met `<Ctrl>-d`

---

creativiteit wordt beloond!

```
~:# cat access_log | cut -d' ' -f1 | sort | uniq -c | sort -rn | head -5
2243 85.223.50.224
 182 6*.2**.66.1
 167 *2.1*.94.1
 143 2a01:3a8:100:16:1:cafe:0:80
 122 *2.1**.9*.9
```

```
~:# du -sh /var/log/* | sort -rh | head -5
5.7M    /var/log/dist-upgrade
3.3M    /var/log/apache2
1.5M    /var/log/samba
1.1M    /var/log/kern.log.1
852K    /var/log/installer
```

---

het xargs commando

- xargs kan commando's uitvoeren
- input van stdin
- xargs buffert argumenten. Veelal gebruikt als voor een commando de argumentlijst te lang is. Bijvoorbeeld als er heel veel file in een directory staan dan krijg je bv.:

```
~$ rm *.old
/bin/rm: cannot execute [Argument list too long]
```

Oplossing:

```
find ~ -name '*.old' | xargs -d '\n'  rm
```

---

het wc commando

`wc [optie]... [file]...`

Telt karakters, woorden en regels

`-c` toon alleen aantal karakters

`-w` toon alleen aantal woorden

`-l` toon alleen aantal regels

```
$ who | wc -l
$ echo ”Het werkboek bevat `wc -c <werkboek.txt` karakters”
```

---

het tr commando

`tr [optie]... set1 [set2]`

kopieert van invoer naar uitvoer waarbij set1 vervangen wordt door set2.

```
tr ”[a-z]” ”[A-Z]” < klanten > KLANTEN
tr -d ”aeiou” < klanten > klanten_zonder_klinkers
tr -s ”a” < klanten
```

(`-s` = *squeeze*, reduceer tekens die meerdere keren achter elkaar voorkomen tot 1 teken)

---

het od commando

`od [optie]... [file]...`

laat de inhoud als octal dump (of een ander formaat zien)

```
$ echo "A a        1" | od -b
0000000 101 040 141 011 061 012
0000006
```

Voor als je niet printbare karakters wilt zien.

Ook handig om verschil in spaties en tabs te zien (maar daar is `cat -vet` handiger voor).

---

join en paste

`paste [optie]... [file]...`

Makkelijk 2 kolommen te maken

- **Geen key nodig (zoals bij join)**
- Verenigd files regel voor regel, plaatst <tab> tussen de verenigde regels.q

`join [optie]... file1 file2`

- Verenigd (*joint*) files regel voor regel
- Eerste veld wordt standaard gebruikt als key
- **Alleen geschikt voor files met dezelfde key-velden en het liefst ook evenveel regels**

split

```
split [optie]... [file1 [prefix]]
```

- splitst een file in even grote stukken, genummerd met een prefix
- splitst standaard op 1000 regels
- standaard *prefix* is x, files: xaa, xab, xac, ...

Meer in de opgaven

Oefeningen

Tijd voor oefening!
