Bash

Introductie tot de Linux command line en bash scripts

Wat is Bash?

> Bash is a command processor that typically runs in a text window, where the user types commands that cause actions. Bash can also read and execute commands from a file, called a script. (Wikipedia)

Waarom Bash en waarom command line?

- Snelheid van werken: één command is vaak sneller dan zoeken naar een bepaalde optie in het menu van bijvoorbeeld je bestandsbeheerder
- Verschillende programma's combineren om één krachtig commando te verkrijgen dat exact voldoet aan je eisen
- Scripts kunnen veel zaken automatiseren. Zoals commando's die vaak achter elkaar gebruikt worden. Bash is daar ideaal voor.

Bash als command line

Programma's uitvoeren

```
bash$ date
wo nov 8 18:30:00 CET 2017
```

- `bash$` duidt het begin van elke lijn aan. Dit voorvoegsel kan steeds anders zijn op andere computers. Soms wordt er in de plaats van `bash` ook de naam van de computer gebruik gevolgd door de huidige maap. Bv. `archlinux:~$`. Dit instelbaar.
- `date` is het programma dat we willen uitvoeren
- `wo nov 8 18:30:00 CET 2017` is de uitvoer van dat programma

Argumenten voor programma's

Programma's kunnen ook gestart worden met bepaalde argumenten.
Om bijvoorbeeld Firefox te openen (op Linux) met de site "vdb.space" kan je dit doen:

```
bash$ firefox vdb.space
```

Note: Op MacOS gaat dit niet omdat het programma dat we willen aanspreken zich bevindt in een applicatiebundel (waarin ook bijvoorbeeld icoontjes zitten voor het programma) en niet waar bash zoekt naar programma's.

Argumenten in de vorm van flags

```
bash$ ls
draft.md  presentation.md  template
```

Toont de inhoud van de huidige map. `ls -l` geeft hetzelfde in lijstvorm.
Je kan ook sorteren op tijd met `ls -lt`, als je de lijst wilt omdraaien voeg je daar nog een `r` aan toe.

```
bash$ ls -rtl
totaal 12
drwxr-xr-x 2 bram bram 4096  3 nov 16:29 template
-rw-r--r-- 1 bram bram  267  3 nov 16:30 draft.md
-rw-r--r-- 1 bram bram 2073  3 nov 16:50 presentation.md
```

Man pages

Je kan steeds voor de meeste programma's en commando's een "manual" vinden. Deze kan je via je terminal bekijken door middel van:

```
bash$ man ls
LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory ...
```

Andere nuttige commando's (1)

```
bash$ pwd
/home/bram/Documenten/infogroep/2017-2018/lesje-bash/
```

Print de naam van de huidige "working directory", de map die dient als basis voor alle relatieve paden. Wanneer we dan:

```
bash$ ls ../
lesje-bash
```

Kijken we naar de inhoud van de map onder de huidige "working directory" (`..` is steeds gelinkt aan de map waar de huidige map in staat). `../` is dus een relatief pad voor `/home/bram/Documenten/infogroep/2017-2018/`

Andere nuttige commando's (2)

```
bash$ cd ../
bash$ pwd
/home/bram/Documenten/infogroep/2017-2018/
```

Verander de huidige working directory (`cd` = **c**hange **d**irectory).

```
bash$ ls
lesje-bash
bash$ mkdir lesje-bash-advanced
bash$ ls
lesje-bash lesje-bash-advanced
```

`mkdir` maakt een nieuwe map aan op het meegeven relatief of absoluut pad.

Andere nuttige commando's (3)

```
bash$ cat lesje-bash/presentation.md
---
# Wat is Bash?

> Bash is a command processor ...
```

`cat` voegt de inhoud van de bestanden die zijn meegeven als argumenten achter elkaar aan één en print het resultaat uit. Meestal wordt `cat` gebruikt om snel de inhoud van een bepaald bestand te zien.

Andere nuttige commando's (4)

```
bash$ ls
lesje-bash lesje-bash-advanced
bash$ touch test.txt
bash$ ls
lesje-bash lesje-bash-advanced test.txt
```

`touch` maakt een bestand aan op het meegeven relatief of absoluut pad.

```
bash$ echo "Hello World"
Hello World
```

`echo` print het stuk tekst uit dat is meegeven als argument.

Andere nuttige commando's (5)

Een goede tekst editor (grafisch) op Linux is `gedit`. Een bestand openen met `gedit` doe je als volgt:

```
bash$ gedit bestandsnaam
```

Andere niet grafische tekst editors zijn `vim`, `nano` en `emacs`.

Commando's combineren

Voer commando B uit na commando A:

```
bash$ A && B
```

Stuur de uitvoer van commando A naar commando B:

```
bash$ A | B
```

Commando's combineren (2)

Bijvoorbeeld: zoek in de presentatie naar een bepaald woord:

```
bash$ cat presentation.md | grep inhoud
Toont de inhoud van de huidige map. `ls -l` geeft hetzelfde in lijstvorm
...
```

Het programma `cat` is verantwoordelijk voor de inhoud van `presentation.md` naar zijn uitvoer te sturen. Deze uitvoer `pipen` we dan (sturen we door) naar de invoer van `grep`. `grep` gaat dan zoeken naar het woord "inhoud" en geeft elke lijn van de invoer waar dat woord voorkomt.

Input/Output redirection

Elk programma heeft drie I/O poorten:

- STDIN: de invoer van het programma (0)
- STDOUT: de uitvoer van het programma (1)
- STDERR: de uitvoer van het programma waar fouten naartoe worden gestuurd (2)

Je kan de uitvoer van elk programma sturen naar een bepaald bestand:

```
bash$ echo "Hello World" > test.txt
bash$ cat test.txt
Hello World
```

Dit stuurt de `STDOUT` poort naar het bestand test.txt.

Input/Output redirection (2)

Op dezelfde manier kan je de fouten uitvoer van een programma sturen naar een bestand.

```
bash$ program-with-errors 2> err.log
```

Als je geen errors wilt zien kan je de errors sturen naar de special file `/dev/null`.

```
bash$ program-with-errors 2> /dev/null
```

Bash scripts

Een reeks van bash commando's kunnen ook in één bestand geplaatst worden. Dit bestand noemen we een bash script.

Open eerst een teksteditor (vim, gedit, nano):

```
bash$ gedit myscript.sh
```

Je eerste script

```
#!/bin/bash

echo "Hello World"
```

`#!/bin/bash` vertelt aan het besturingssysteem dat het tekstbestand als een bash script geinterpreteerd moet worden.

Uitvoeren van je script

```
bash$ bash myscript.sh
Hello World
```

of

```
bash$ chmod +x myscript.sh
bash$ ./myscript.sh
Hello World
```

`chmod` gaat de rechten veranderen op een bepaald bestand. De `+x` wilt zeggen dat elke gebruiker het bestand mag uitvoeren. Verder heb je `+r` om elke gebruiker leesrecht te geven en `+w` om elke gebruiker een schrijfrecht te geven. (zie `man chmod`)

Rechten op Linux/Unix uitgebreid

Elk bestand en map heeft een eigenaar gebruiker en een eigenaar groep. De groep kan een groep zijn waar de eigenaar gebruiker zelfs niet inzit, maar dat is meestal wel het geval. Rechten op bestanden en mappen kunnen met `chmod` veranderd worden.

Er zijn vier rechten:

- lezen: het recht hebben een bepaald bestand te lezen (**r**)
- schrijven: het recht om aanpassingen in een bestand of map aan te brengen (**w**)
- uitvoeren: het recht om de inhoud van een bestand als iets uitvoerbaar te zien en dat uit te voeren (**x**)
- setuid: het recht dat een bestand wordt uitgevoerd in de naam van de eigenaar ervan (**s**)

Rechten op Linux/Unix uitgebreid (2)

De rechten kunnen aangepast worden voor de volgende drie categorieën:

- eigenaar gebruiker (**u**)
- eigenaar groep (**g**)
- andere gebruikers die noch de eigenaar zijn en noch behoren tot de groep (**o**)

Bijvoorbeeld: leesrechten geven aan de eigenaar gebruiker voor bestand test.txt:

```
bash$ chmod u+r test.txt
```

Let op: te vrije rechten kunnen onveilig zijn op een bestandsysteem

Argumenten in je bash script

Command line argumenten kunnen opgevraagd worden in een bash script met `$n`. `$0` geeft de scriptnaam `$1` geeft de waarde van het eerste argument, `$2` de waarde van het tweede argumen,etc.

```
#!/bin/bash
echo "Hello $1"
```

Uitvoer:

```
bash$ ./myscript.sh Bram
Hello Bram
```

Controlestructuren

We kunnen volgorde waarin expressies in het bash script worden uitgevoeren beinvloeden via controlestructuren.

- if: op basis van een bepaalde conditie, code al dan niet uitvoeren
- loops: meerdere keren bepaalde expressies uitvoeren

If statements

```
#!/bin/bash
if cond-expr; then
   expressies wanneer cond-expr waar
fi
```

Voorbeeld:

```
#!/bin/bash
if [[ $1 == "secret password" ]]; then
   echo "welcome to the secret bash script"
fi
```

Conditionele expressies

- [[TEKST1 == TEKST2]] slaagt wanneer TEKST1 gelijk is aan TEKST2
- [[TEKST2 != TEKST2]] slaagt wanneer TEKST2 niet gelijk is aan TEKST2
- [[-a PAD]] slaagt wanneer het bestand op PAD bestaat
- [[-d PAD]] slaagt wanneer de map op PAD bestaat
- [[-z TEKST]] slaagt wanneer TEKST leeg is (kan gebruikt worden om te controleren of bepaalde command line argumenten gegeven zijn)

Meer expressies zijn hier te vinden.

Let op: de spaties die volgen achter `[[` en voor `]]` zijn belangrijk. Anders is de syntax incorrect

Conditionele expressies (2)

Voorbeeld:

```
#!/bin/bash
if [[ -z $1 ]]; then
   echo "Gebruik: ./myscript.sh naam"
   exit
fi

echo "Hello $1"
```

Meerdere condities kunnen ook worden toegevoegd tussen de `[[` en `]]` door gebruik te maken van `&&` wanneer de conditie achter deze operand en ervoor beide waar moeten zijn (and) en `||` wanneer één van beide waar moet zijn (or).

For loops

Voer voor elk element uit een lijst bepaalde expressies uit.

```
#!/bin/bash
for var in list-expr; do
   reeks van expressies
done
```

Het variabele `$var` zal voor de reeks van expressies binnen de for loop telkens gebonden worden aan één element van de lijst.

For loops (2)

```
#!/bin/bash
for var in 1 2 3 4 5; do
    echo $var
done
```

Uitvoer:

```
bash$ ./myscript.sh
1
2
3
4
5
```

For loops (3)

De volgende soorten list expressies zijn toegestaan in Bash zijn:

- Letterlijke lijsten: Bv. `1 2 3 4`
- Reeksen (ranges): `{1..2}`
- Lijsten van commando's: `$(ls)`

Functies

Code kan ook herbruikt worden op meerdere plaatsen in een script door functie voor deze te implementeren. Op die manier kunnen complexe commando's ook verborgen worden.

```
#!/bin/bash

functieNaam () {
   expressies
}

# oproep van de functie (deze lijn is commentaar)
functieNaam argument1 argument2
```

Functies (2)

De argumenten van een functie zijn positioneel.
Je kan ze op dezelfde manier binnen een functie opvragen zoals dat het geval is bij de command line argumenten.

Bijvoorbeeld:

```
#!/bin/bash

zegHallo () {
   echo "Hallo $1"
}

zegHallo "Bram"
```

Nuttige bronnen

- Manual pages via het `man` commando
- http://tldp.org/LDP/Bash-Beginners-Guide/html/
- https://www.gnu.org/software/bash/manual/html_node/