Het Linux filesysteem (2)

In dit hoofdstuk maken we nader kennis met het Linux filesysteem en enkele commando's

Doelstelling

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

- het verschil tussen een absoluut en een relatief pad
- het wandelen door het filesysteem
- directories maken, weggooien en verplaatsen
- bestanden maken, weggooien en verplaatsen
- het disk gebruik van een filesyteem onderzoeken
- wildcards
- het vinden van bestanden en directories
- de bij al bovenstaande acties behorende commando's

---

absolute en relatieve paden

- **absoluut** pad: begint **altijd** met de rootdirectory (/). Dit betekent dat de plaats van een bestand ten opzichte van deze rootdirectory wordt aangegeven. Voorbeeld:
  - `/home/user2/vb1`
- **relatief** pad: wordt **niet** uitgegaan van de root directory maar van de directory waar de gebruiker zich op dat moment bevindt. Voorbeeld:
  - `user2/vb1`
- Een relatief pad begint **nooit** met een /

---

het commando cd

Wandelen doe je met het commando: `cd` (`c`hange `d`irectory)

- Voorbeelden:

`$ cd`	ga naar de home-directory van de gebruiker
`$ cd /`	ga naar de system's root directory (absolute padnaam)
`$ cd ..`	ga een directory omhoog (relatieve padnaam)
`$ cd ../..`	ga twee directories omhoog (relatieve padnaam)
`$ cd /home/user1`	ga naar /home/user1 (absolute padnaam)
`$ cd ../user1`	ga 1 dir omhoog en dan naar user1 (relatieve padnaam)
`$ cd oefeningen`	ga naar de directory oefeningen (relatieve padnaam)
`$ cd ./oefeningen`	ga naar de directory oefeningen (relatieve padnaam)

Speciale directories:

`..`	1 directory hoger
`.`	de huidige directory

pwd, ls

- `pwd [optie]...`

  print working directory

  - `~$ pwd`

    `/home/oscar`

- `ls [optie]... file...`

  toon (*list*) de inhoud van een directory

  - `ls -l` geeft een lange (*long*) lijst van alle bestanden met o.a. de naam, eigenaar en grootte (later meer)
  - `-a` geeft ook de verborgen (***hidden***) bestanden (beginnen met een `.`) weer. Dus ook `.` en `..`

Meer in de opgaven (en `man ls` natuurlijk).

Wildcards

Karakters met een speciale betekenis welke worden gebruikt in aanduidingen voor files en directories:

`*` willekeurige reeks van 0 of meer karakters

- `ls v*` geeft v, vb1, voorbeeld2 maar niet evaluatie.txt
- `ls *` geeft alle files en directories in de huidige directory.

`?` precies 1 willekeurig karakter op die plaats

- `ls user?` geeft user1, user2 maar niet user11

`[]` een karakterklasse: 1 van de tekens in de reeks op die plek

- `ls vb[123]` geeft vb1, vb2 of vb3 en niets anders
- `[!abc]` de ontkenning: juist **niet** 1 van die tekens
- een (ascii) range kun je aangeven met de hyphen (-):
  - `ls user[1-9]`

De shell en bv. het commando `find` kennen wildcards.

Maar `ls` bijvoorbeeld **niet** (!)

Bestanden vinden (1/5)

`locate [optie]... pattern...`

- veel simpeler dan find
- vind files op naam welke minimaal *pattern* bevatten.
- kijkt niet op systeem maar in database (gemaakt door `updatedb`)
- sneller maar soms niet op de hoogte (als de database achter loopt)

Opties:

- `-b` match niet op de padnaam maar op alleen de basename
- `--regex` zie *pattern* als een reguliere expressie (later meer over reguliere expressies)

Bestanden vinden (2/5)

`whereis en which`

- `whereis` laat zien waar een binary, de man-page van de binary en evt. source-code van de binary te vinden is

```
$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

- `which` laat alleen het pad naar een binary zien (indien in $PATH)

```
$ which ls
/bin/ls
```

Bestanden vinden (3/5)

`find [pad]... [expressie]`

- Zoek naar files vanaf `pad`
- Veel zoekmogelijkheden

Expressies:

- `-name` *`name`* zoek naar *`name`* (find kent wildcards)

  `find ~ -name 'v*'` zoek naar alle files onder je home directory die beginnen met een `v` (**Quoten!** (waarom?))
- `-type {d|f}` zoek naar files van type d (directory) of f (regulier file), of ...
- `-size {+|-}`*`size`* zoek naar bestanden kleiner (-) of groter (+) dan `size` (`size`: blokken van 512 bytes)

Bestanden vinden (4/5)

`find [pad]... [expressie|actie]`

Meer expressies:

- `-mtime {+|-}n` alle files meer dan (+) n dagen of minder dan (-) n dagen geleden gewijzigd (*modified*)
- `-atime {+|-}n` alle files meer dan (+) n dagen of jonger dan (-) n dagen gebruikt (*accessed*)
- `-newer file` alle files later veranderd dan `file`

Logische AND/OR combinaties van expressies mogelijk:

- `-name '*.jpg' -a -atime -7` alle files eindigend op .jpg **en** minder dan 7 dagen nog gebruikt (`-a` = AND, kan weg: is default)
- `-name '*.old' -o -atime +100` alle files eindigend op `.old` of meer dan 100 dagen niet meer gebruikt zijn

---

Bestanden vinden (5/5)

`find [pad]... [expressie|actie]`

Acties:

- `-print` laat de gevonden files zien (default)
- `-ls` geef een lange list van de gevonden files (gelijk aan `ls -dils`)
- `-exec` *cmd* `{} \;` laat een commando los op de gevonden files, bv.: `-exec chmod o-rwx {} \;`

Directories maken, weggooien en verplaatsen

`mkdir [opties] directory ...`

- maken van 1 of meer directories
- `-p` maak meerdere levels van directories tegelijk

`rmdir [opties] directory ...`

- verwijdert 1 of meer directories. Directory moet leeg zijn.

`mv source-dir target-dir`

- verplaatst een hele boom. Als target-dir bestaat komt source-dir er onder
- Als target-dir niet bestaat wordt source-dir de target-dir

Bestanden kopieren

`cp [opties] source [...] dest`

- copy `source` naar `dest`.
- meerdere source files: dan moet `dest` een directory zijn.
- `-R, -r` copy directories recursively (beter: `rsync`)
- `-i` (interactive): vraag om `dest` te overschrijven (indien deze bestaat)

Voorbeelden:

```
$ cp -i vb1 vb2
$ cp vb1 vb2 /home/user1
$ cp * /home/user2
```

Bestanden hernoemen

`mv [opties] source [...] dest`

- hernoem (*rename*) source naar `dest` (zowel voor files als voor directories)
- meerdere source files: dan moet `dest` een directory zijn.
- `-i` (interactive): vraag om te overschrijven

Voorbeelden:

```
$ mv vb1 voorbeeld1
$ mv * /home/user2
$ mv /home/user2 /home/student2
```

Bestanden verwijderen

`rm [optie ...] file ...`

- verwijder 1 of meerdere files
- `-i` (interactive): vraag om te verwijderen
- `-R, -r` verwijder directory en alle inhoud *recursive* (pas op: erg krachtig!)

Voorbeelden:

```
$ rm vb1 voorbeeld1
$ rm -r /var/tmp/zut
```

Disk gebruik

Twee handige commando's.

`df` (disk free) Reeds gezien: geef overzicht op partitie niveau snel te zien welke partitie vol is

```
$ df -h -t ext3
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             958M  399M  511M  44% /
/dev/sda7             958M  230M  680M  26% /home
/dev/sda6             958M   51M  858M   6% /tmp
/dev/sda9             865M  546M  276M  67% /var
/dev/sda5             3.8G  1.1G  2.5G  31% /usr
```

- `-h` human readable format
- `-t type` geef allen df voor filesystem `type`

`du` (disk usage) geef overzicht op directory/file niveau. Beter ”in te zoomen” wat de oorzaak is. Handig met de optie `-s` en de wildcard *

```
$ du -sh /*
```

`-s` laat totalen per argument zien

Samenvatting

Na een disk indelen (**partitioneren**), deze geschikt maken voor gebruik door er een filesysteem op te maken (**`mk2efs`**) en deze aan elkaar te koppelen (**`mount`**) hebben we in dit hoofdstuk gekeken naar het daadwerkelijk gebruik van het filesysteem:

- het wandelen door het filesysteem (`cd`)
- kijken naar de inhoud van het file systeem (`ls`)
- het gebruik van wildcards om meerdere bestanden aan te duiden
- zoeken naar bestanden (met name `find`)
- maken, kopieren, verwijderen, verplaatsen en hernoemen van files en directories (`mkdir`, `rmdir`, `rm`, `cp`, `mv`)
- disk gebruik bekijken (`df` en `du`)

Oefeningen

Tijd voor oefening!
