Korte uitleg: Mounten van filesystemen

Mounten (van filesystemen) wordt in beter Nederlands 'aankoppelen' genoemd. Hoe je het ook noemt, waar gaat het om? Voor velen is dit een raadselachtig gebeuren. Dat is het uiteraard niet, maar enige achtergrondkennis is nodig. Lees daarom eerst:

- *Korte uitleg: De enkelvoudige hiërarchische bestandsorganisatie (directory tree, absolute and relative pathes)*
- *Korte uitleg: Device files (/dev/sda en zo)*
- *Korte uitleg: File systems*
- *Korte uitleg: Disk partitionering*

Een Unix/Linux systeem kent dus één boomstructuur van directories. Uiteraard wordt alles uiteindelijk op mass-storage gezet. Maar je kunt verschillende stukken mass-storage hebben (door mij containers genoemd). Hoe koppel je die zo dat die boomstructuur ter beschikking komt?

Eerst een voorbeeld van zo'n boom. Hoewel het volgende is ingekort (anders wordt het veel te lang) komt wat er staat overeen met een werkelijk systeem.

```
/
├── boot
├── dev
├── etc
├── home
│   ├── henk
│   └── marian
├── lib
├── lib64
├── lost+found
├── media
├── mnt
│   └── B
│       ├── boot
│       ├── dev
│       ├── etc
│       ├── home
│       │   ├── henk
│       │   └── marian
│       ├── lib
│       └── var
├── opt
├── proc
├── root
├── run
├── sbin
├── selinux
├── srv
├── sys
├── tmp
├── usr
└── var
```

Ik heb met een kleurtje aangegeven wat de verschillende filesystemen zijn die

hier zijn gebruikt om de boom op te bouwen.

We zien de boom vanaf de root (`/`) en daarboven de standaard directories die daar thuishoren. Echter bij `/home` zien we een andere kleur. `/home` en alles wat daarin zit staat dus op een ander filesysteem en dus op een andere mass-storage container.

Bij `/mnt/B` zien we ook een andere kleur. Daar zit dus ook een ander filesysteem. In dit geval is dat file systeem eerder lange tijd gebruikt als het root filesysteem van openSUSE 12.3. Ik vind dat handig, ik kan dan bijvoorbeeld kijken wat er in `/etc/apache2` stond zoals dat op openSUSE 12.3 werd gebruikt. Maar op 12.3 had ik ook een aparte `/home`. Een kopie daarvan heb ik nu staan in `/mnt/B/home`.

Dat die verschillende filesystemen daar aangekoppeld zijn kunnen we zien met:

```
henk@boven:~> mount | grep /dev/sd
/dev/sda2 on / type ext4 (rw,relatime,data=ordered)
/dev/sda3 on /home type ext4 (rw,relatime,data=ordered)
/dev/sda5 on /mnt/B type ext4 (ro,relatime,data=ordered)
/dev/sda6 on /mnt/B/home type ext4 (ro,relatime,data=ordered)
henk@boven:~>
```

Elk van die filesystemen heeft al de structuur van een stuk van de boom. Het enige wat we nu moeten doen is de boom in elkaar zetten. We beginnen met het onderste deel van de boom: het root filesysteem. En dan prikken we op plekken in de boom die wij daarvoor geschikt vinden verdere filesystemen.

Het `mount` commando

Om iets te kunnen mounten heb je een aantal dingen nodig:

- de container waar het te mounten filesysteem in zit; dat is dus de block device file van die container;
- het mountpoint, dat is de plek in de al bestaande directory boom waar we deze nieuwe takken willen ophangen, een directory dus;
- het file systeem type; tenslotte moet de Kernel weten met welke software dit stuk behandeld moet worden.

Dat is het belangrijkste. Uiteraard kunnen nog allerlei parameters worden toegevoegd.

Mounten is niets anders dan aan de Kernel vertellen dat we op bepaalde directory (die daarom een mountpoint wordt genoemd) in de al bestaande boom een nieuw stuk boom (een tak met zijtakken) willen aankoppelen. Als alles goed gaat doet de Kernel dat. En de Kernel weet vanaf dat moment dat bijvoorbeeld `/home/henk/test` op een ext4 filesysteem staat dat te bereiken is via device file `/dev/sda3`.

Zo'n verzoek aan de Kernel kan alleen een root proces doen en gaat met het

commando `mount`. De `man` pagina van `mount` is zeer uitgebreid. Vooral omdat er niet alleen de algemene opties van `mount`, maar ook de speciale opties van de diverse filesystemen in staan. Een optie die je kent van ext4 kan dus totaal onbekend zijn (of iets anders betekenen) bij een VFAT file systeem.

Een simpel `mount` commando, dat alleen de drie bovengenoemde basisgegevens bevat:

```
mount -t ext4 /dev/sda3 /home
```

Dit probeert om een ext4 filesysteem dat op de disk partitie `/dev/sda3` staat aan te koppelen op de directory `/home`.

Kan het met nog minder? Ja, dat kan. Als je

```
mount /dev/sda3 /home
```

doet gaat `mount` met intelligent raden proberen uit te vinden welk type filesysteem er op `/dev/sda3` staat. En dat lukt heel vaak goed. Maar je kunt het natuurlijk beter opgegeven, gaat ook sneller.

Het `mount` commando kan nog korter, maar daarvoor moet je de gegevens ergens anders neerzetten zodat `mount` ze kan vinden. Dat is het bestand `/etc/fstab`. Om weer af te koppelen is het commando `umount`. Als parameter is alleen de device file of het mountpoint nodig.

Het configuratiebestand `/etc/fstab`

De gegevens te gebruiken in een mount commando kunnen voorgedefiniëerd worden in /etc/fstab:

```
henk@boven:~> cat /etc/fstab
/dev/disk/by-id/ata-Hitachi_HDT725032VLA380_VFJ201R23XUEXW-part2 /
                        ext4  acl,user_xattr  1 1
/dev/disk/by-id/ata-Hitachi_HDT725032VLA380_VFJ201R23XUEXW-part1 swap
                        swap  defaults        0 0
/dev/disk/by-id/ata-Hitachi_HDT725032VLA380_VFJ201R23XUEXW-part3 /home
                        ext4  defaults        1 2
/dev/disk/by-id/ata-Hitachi_HDT725032VLA380_VFJ201R23XUEXW-part5 /mnt/B
                        ext4  ro,acl          1 2
/dev/disk/by-id/ata-Hitachi_HDT725032VLA380_VFJ201R23XUEXW-part6 /mnt/B/home
                        ext4  ro,acl          1 2
henk@boven:~
```

(de regels zijn hier in tweeën gebroken om ze netjes in dit document te krijgen, in het echt mag dat niet).
Hier zien we:

- de device file, we zien hier gebruik van de symbolic links uit `/dev/disk/by-id`, gebruikelijk bij openSUSE;
- het mountpoint, `swap` is geen mountpoint, maar een tekst om dit veld niet

leeg te laten zijn;
- het type filesysteem, `swap` is geen fileysteem type en deze container wordt niet gemount, maar gebruikt als swap/wisselgeheugen;
- de parameters, `default` betekent volgens de `man` pagina `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, and `async`, let ook op de parameter `ro`, deze file systemen worden read-only gemount;
- en nog twee velden, gebruik `1 2` voor alle andere dan het root file systeem dat `1 1` krijgt.

Als we nu doen

```
mount /home
```

dan zoekt het `mount` commando in `/etc/fstab` naar het veld `/home` en vindt dan daarbij alle andere parameters. Hetzelfde gebeurt bij

```
mount /dev/sda3
```

Dit is helemaal handig bij het volgende commando:

```
mount -a
```

want dat loopt alle definities van `/etc/fstab` na en mount ze indien zich niets daartegen verzet. Dat gebeurt overigens ook tijdens het booten.

Vanuit de desktop

Het bovenstaande werkte allemaal jaren lang naar tevredenheid. De systeembeheerder zorgde voor een correcte `/etc/fstab` en als er iets bijzonders gedaan moest worden handelde hij/zij dat met `mount` statements af. Tenslotte was het aansluiten van nieuwe schijven iets waarvoor het systeem spanningsloos moerst worden gemaakt en dat ging mooi allemaal samen in geplande onderhoudstijd. Tegenwoordig hebben we aparaten die bij draaiend systeem kunnen worden aangesloten en verwijderd (CDs, DVDs, USB stick, geheugenkaarten) en omdat bij Linux de gebruiker eigenlijk ook operator is, wil die gebruiker dat zelf doen zonder een systeembeheerder te moeten inschakelen. Dat lijkt simpel, maar dat is het niet:

- mounten kan alleen door een root process;
- Linux is een multi-user systeem en er kunnen dus meer gebruikers tegelijk zijn ingelogd. Wie mag nu die USB stick gebruiken? (Daar zijn leuke anekdotes over);
- er moet een goed mountpoint worden aangemaakt op een plek en met een naam die uniek is;
- er moet bepaald worden welke parameters nodig zijn, dit punt is vooral belangrijk omdat veel van de gebruikte apparaten niet-Linux filesystemen bevatten.

Er zijn verschillende software pakketten voor gemaakt en weer vervangen.

Het lijkt er op dat de huidige stand van zaken redelijk stabiel en correct is en meestal is het voor de eindgebruiker een intuitieve gang van zaken. Toch gaat hier nog wel eens iets mis of wordt verkeerd begrepen. Daarom een aparte *Korte uitleg: Mounten van filesystemen door de desktopgebruiker*.

Aandachtspunten en tips

- Een directory die als mountpoint gebruikt gaat worden moet natuurlijk wel bestaan. In het voorbeeld boven betekent dat, dat de directory `/mnt/B` er moet zijn en ook dat `/dev/sda5` er eerst op gemount moet worden anders kan `/dev/sda6` niet op `/mnt/B/home` gemount worden. Volgorde, ook in `/etc/fstab`, is dus belangrijk.

- Een mountpoint is een gewone directory. Daar kan dus al van alles in staan. Als je iets mount op zo'n directory kun je niet meer bij alles wat in die directory staat. Het is niet weg. Als je `umount` is alles weer bereikbaar.

```
boven:~ # l /mnt/B
total 8
drwxr-xr-x 2 root root 4096 May 29 13:50 ./
drwxr-xr-x 3 root root 4096 Sep 27  2013 ../
-rw-r--r-- 1 root root    0 May 29 13:50 hier-is-niets-gemount
boven:~ # mount /dev/sda6 /mnt/B
boven:~ # l /mnt/B
total 40
drwxr-xr-x 10 root   root  4096 May 20 16:42 ./
drwxr-xr-x  3 root   root  4096 Sep 27  2013 ../
drwxr-xr-x  6 mysql    108 4096 Nov 12  2013 databases/
drwxr-xr-x 84 henk   wij   4096 Jan 18 14:02 henk/
drwx------  2 root   root  4096 Oct 25  2009 lost+found/
drwxr-xr-x 34 marian wij   4096 Aug  3  2013 marian/
drwxr-xr-x 12 mgi    users 4096 Jan 14 10:49 mgi/
drwxr-xr-x 17 smweb  www   4096 Jan 14 10:14 smweb/
drwxr-xr-x  8 wappl  www   4096 Nov 12  2013 wappl/
drwxrwxrwx 16 henk   wij   4096 Jan 16 14:12 wij/
boven:~ # umount /mnt/B
boven:~ # l /mnt/B
total 8
drwxr-xr-x 2 root root 4096 May 29 13:50 ./
drwxr-xr-x 3 root root 4096 Sep 27  2013 ../
-rw-r--r-- 1 root root    0 May 29 13:50 hier-is-niets-gemount
boven:~ #
```

- Nogmaals, mountpoints zijn gewone directories. Dat betekent dat eigenaarschap (gebruiker en groep) en permissies precies hetzelfde doen als bij alle andere directories. Denk daar eens aan als er iemand roept "Ik kan niet schrijven op mijn USB stick!" of woorden van gelijke wanhopige strekking.

- Als je als systeembeheerder regelmatig een mass-storage apparaat aansluit voor bijvoorbeeld backup, maak dan een regel voor de mount in `/etc/fstab`. Je kunt dan met een zeer kort mount commando werken en toch alles altijd goed doen. Dan moet je wel de `noauto` optie in die regel zetten, anders probeert het systeem bij het booten dat apparaat aan te koppelen.

- Je hoeft dit echt niet allemaal "met de hand" te doen in de hoop dat je niets vergeet. YaST > System > Partitionering is een grote hulp. Als je daar alles goed invult, wordt een mountpoint gemaakt als het niet bestaat, een benodigde regel in `/etc/fstab` wordt aangemaakt, kortom in de meeste gevallen is YaST je vriend.

- Bij afkoppelen (met `umount`) moet je bedenken dat dit alleen kan als er geen enkel proces meer met het filesysteem bezig is. Alle bestanden moeten dus gesloten zijn en ook mag er geen enkele shell een directory in dat filesysteem als "working directory" gebruiken.