Korte uitleg: Device files (`/dev/sda` en zo)

Het bovenstaande onderwerp zegt misschien niet veel Linux gebruikers iets, maar als ik zeg `/dev/sda`, dan is dat toch iets dat je wel eens gezien hebt. Iets meer kennis hiervan is noodzakelijk bij het begrijpen van bijvoorbeeld mounten. Overigens, device files worden ook wel special files genoemd.

In Unix was één van de uitgangspunten: alles is een bestand (everything is a file). Het gevolg is dat allerlei invoer/uitvoer (I/O) apparaten kunnen worden gelezen en beschreven op dezelfde manier als ieder ander bestand. Maar die bestanden zijn toch een beetje bijzonder. Dat blijkt uit het feit dat zij twee eigen types hebben. Je ziet dat met

```
henk@boven:~> l /dev/sda /dev/tty0
brw-rw---- 1 root disk 8, 0 13 mei 09:43 /dev/sda
crw--w---- 1 root tty  4, 0 13 mei 09:43 /dev/tty0
henk@boven:~>
```

We kennen als eerste letter van deze uitvoer al: `-` voor gewone bestanden, `d` voor directories en `l` voor symbolic links. Nu zien we `c` voor character device file en `b` voor block device file. Het verschil is dat block device files zijn voor apparaten waarvan de I/O naar via de buffer cache gaat (zoals schijven) en character device file voor apparaten die dat niet doen, maar regelrecht de bytes krijgen of leveren (zoals terminals).

Al deze device files staan in `/dev`. Dat is een afspraak en kan dus ook anders. Maar als je er ooit een vindt buiten `/dev` is dat zeer verdacht.

Verder zien we de permissie bitjes en de eigenaar user en group. Die bepalen dus wat iedereen regelrecht op het apparaat mag doen. Dat is zeer belangrijk. Stel je voor als iedereen zo maar willekeurig op een schijf zou kunnen schrijven, of nog geniepiger alles zou kunnen lezen. Dat zou alle beveiliging van de bestanden waardeloos maken.

Je ziet hier veelal bijzondere groepen als `disk` en `video` genoemd. Soms kunnen deze worden gebruikt om persmissie te geven aan één of meer gebruikers. Maar in het algemeen is het geen goed idee om dit te doen, zelfs als het "ergens op het internet" wordt aangegeven als oplossing voor een probleem.

Dan zien we een afwijking ten opzichte van de gegevens van "normale" bestanden. Twee door een komma gescheiden getallen.Dit zijn het "major number" and het "minor number". Het major number geeft de "driver" aan die in de Kernel gebruikt wordt voor dat type apparaat. Het minor number is een nadere aanduiding voor die driver. Dat kan bijvoorbeeld zijn om verschillende apparaten die door dezelfde driver worden behandeld te onderscheiden:

```
henk@boven:~> ls -l /dev/sd? /dev/sr0
brw-rw----  1 root disk   8,  0 12 mei 09:53 /dev/sda
```

```
brw-rw----  1 root disk   8, 16 12 mei 09:53 /dev/sdb
brw-rw----  1 root disk   8, 32 12 mei 09:53 /dev/sdc
brw-rw----  1 root disk   8, 48 12 mei 09:53 /dev/sdd
brw-rw----  1 root disk   8, 64 12 mei 09:53 /dev/sde
brw-rw----+ 1 root cdrom 11,  0 12 mei 09:53 /dev/sr0
henk@boven:~>
```

We zien dat voor `sr0` (een CD apparaat) een andere driver wordt gebruikt dan voor de schijven. En dat iedere schijf een eigen nummer heeft.

Die device files zijn dus heel belangrijk. Hoe komen we er aan? Op vroegere Unix systemen werden er een aantal in `/dev` klaargezet, nodig of niet. Toch moesten ze soms met de hand aangemaakt worden (bijvoorbeeld bij het monteren van extra schijven). Tegenwoordig kan het systeem via de moderne bussen en apparaten uitzoeken welke hardware aanwezig is en kan het systeem daarop aktie nemen. Dat besteedt de Kernel uit aan `udev`. In het kort:

- `/dev` is een tijdelijk filesysteem en verdwijnt dus als het systeem stopt;
- bij boot wordt het `/dev` filesysteem nieuw aangemaakt en gemount;
- uit de gegevens die van de apparaten worden verkregen worden de device files gemaakt met de correcte naam, en major en minor numbers en de eigenaar, groep, permissies. De zogenaamde `udev` regels definiëren wat er moet gebeuren;
- tijdens het draaien van het systeem kunnen ook dynamisch nieuwe device files worden aangemaakt. Bijvoorbeeld als er een stick in een USB aansluiting wordt gestopt.
- dit laatste is belangrijk om te begrijpen: de volgorde van aanmaken is dus de volgorde van detecteren, dezelfde schijf is dus vandaag `/dev/sdd` en morgen misschien `/dev/sdc`!

Overigens worden soms extra symbolic links aangemaakt, die naar de device file verwijzen:

```
henk@boven:~> ls -l /dev/disk/by*/* | grep sda5
lrwxrwxrwx 1 root root 10 12 mei 09:53 /dev/disk/by-id/ata-Hitachi_HDT725032VLA380_VFJ2
lrwxrwxrwx 1 root root 10 12 mei 09:53 /dev/disk/by-id/scsi-1ATA_Hitachi_HDT725032VLA38
lrwxrwxrwx 1 root root 10 12 mei 09:53 /dev/disk/by-id/scsi-SATA_Hitachi_HDT72503_VFJ20
lrwxrwxrwx 1 root root 10 12 mei 09:53 /dev/disk/by-id/wwn-0x5000cca311f7606f-part5 ->
lrwxrwxrwx 1 root root 10 12 mei 09:53 /dev/disk/by-label/System_B -> ../../sda5
lrwxrwxrwx 1 root root 10 12 mei 09:53 /dev/disk/by-path/pci-0000:00:1f.2-scsi-0:0:0:0-
lrwxrwxrwx 1 root root 10 12 mei 09:53 /dev/disk/by-uuid/67aabbdc-41c8-4c4c-a965-4ff467
henk@boven:~>
```

Merk op dat veel van die links onafhankelijk zijn van de plaats die de schijf in het systeem inneemt. Gebruik van bijvoorbeeld de `/dev/disk/by-id` link (bij openSUSE bij verstek) gebruikt altijd dezelfde schijf, ook al wordt hij bij booten ineens `sdb5`. Gebruik van `/dev/disk/by-label` is heel handig als je een Volume Label aan een file systeem hangt. Altijd goed gemount, waar hij ook zit.

Probeer het volgende goed te begrijpen:

- Het schrijven/lezen van bestanden gaat via de Kernel en het file systeem.

Op welk apparaat dat terecht komt zoekt de Kernel uit en is van geen belang voor de gewone gebruiker.
- Het kan nodig zijn om regelrecht op een schijf te schrijven. Bijvoorbeeld bij het partitioneren en bij het aanmaken van een file systeem. Daarvoor gebruik je dus de device files en dat mag alleen `root`.
- Als je mag schrijven naar een device file (en dus `root` bent) kan je dus heel makkelijk alles verknoeien. **DOE HET VOLGENDE NIET** (en als je het doet kom dan niet bij mij klagen):

```
echo "Hoera, systeem verpest" >/dev/sda
```

Dat betekent dat die tekst op de eerste plekken van het eerste blok van de schijf komen te staan. Daarmee overschrijf je de partitietabel en de boot informatie. Foetsie systeem.

Niet alleen echte apparaten (die je kunt aanraken) maar ook pseudo apparaten hebben device files. Een voorbeeld:

```
henk@boven:~> l /dev/tty*1
crw--w---- 1 root tty     4,  1 14 mei 09:23 /dev/tty1
crw--w---- 1 root tty     4, 11 14 mei 09:23 /dev/tty11
crw--w---- 1 root tty     4, 21 14 mei 09:23 /dev/tty21
crw--w---- 1 root tty     4, 31 14 mei 09:23 /dev/tty31
crw--w---- 1 root tty     4, 41 14 mei 09:23 /dev/tty41
crw--w---- 1 root tty     4, 51 14 mei 09:23 /dev/tty51
crw--w---- 1 root tty     4, 61 14 mei 09:23 /dev/tty61
crw-rw---- 1 root dialout 4, 65 14 mei 09:23 /dev/ttyS1
crw-rw---- 1 root dialout 4, 75 14 mei 09:23 /dev/ttyS11
crw-rw---- 1 root dialout 4, 85 14 mei 09:23 /dev/ttyS21
crw-rw---- 1 root dialout 4, 95 14 mei 09:23 /dev/ttyS31
henk@boven:~>
```

Ik heb de lijst ingekort door alleen de device files eindigend op `1` te tonen. Er zijn er veel meer.

Je ziet hier device files `ttyS`. Dat zijn de terminals (TTY betekent Teletype, een terminal uit de begintijd van Unix), die je kunt aansluiten aan een asynchrone poort (ook bekend als COM poort, zie ook dat je op zo'n verbinding eventueel kunt "uitbellen", allemaal nogal ouderwets). De `tty` (zonder `S`) device files worden gebruikt door terminal emulaties op de console (die je bereikt met Ctrl-Alt-Fn) en zijn dus pseudo devices.

Maar ook device files voor het volgende zijn eigenlijk voor pseudo apparaten omdat ze naar containers op mass-storage wijzen, die niet echt met de vinger aan te raken zijn:

- partities van schijven, device files zoals `/dev/sda1`, `/dev/sda2`;
- Logical Volumes van LVM, namen zoals `/dev/mapper/<vgnaam>/<lvnaam>`;
- MD devices van Linux Software RAID, namen zoals `/dev/md*` (en symbolic links daar naar in the `/dev/disk/by-*` directories).

Nog een aantal bijzondere device files van pseudo apparaten.

`/dev/null`

Dat is "het zwarte gat", alles wat je daarheen stuurt is weg, wordt veel gebruikt om uitvoer die je niet wilt zien te laten verdwijnen via een re-direction van stdout en/of stderr:

```
henk@boven:~> ping -c 1 beneden && echo "hij is er"
PING beneden.henm.xs4all.nl (10.0.0.155) 56(84) bytes of data.
64 bytes from beneden.henm.xs4all.nl (10.0.0.155): icmp_seq=1 ttl=64 time=0.174 ms

--- beneden.henm.xs4all.nl ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.174/0.174/0.174/0.000 ms
hij is er
henk@boven:~> ping -c 1 beneden >/dev/null && echo "hij is er"
hij is er
henk@boven:~>
```

`dev/zero`

Levert bij lezen binaire 0 bytes op. Daarmee (en met het programma `dd`) kun je dus op een bestand of een hele disk alle bytes op 0 zetten.

`/dev/random`

`/dev/urandom`

Levert bij lezen random bytes op. Daar kun je dus ook een hele disk mee vullen. Kost meer tijd dan met `/dev/null`, berekenen van random getallen kost CPU cycles.

`/dev/mem`

`/dev/kmem`

`/dev/port`

Hier kun je o.a. regelrecht uit het geheugen (RAM) lezen en er naar schrijven. Nooit doen!