Korte uitleg: Directories, meta-data van bestanden, inodes

Wat is een directory eigenlijk en wat staat er in? De belangrijkste antwoorden daarop zullen we in deze Korte uitleg behandelen.

Zoals in *Korte uitleg: De enkelvoudige hiërarchische bestandsorganisatie (directory tree, absolute and relative pathes)* is uitgelegd, is een directory een vertakkingsknoop in de boomstructuur op een file system. Een directory is zelf ook een bestand en bevat de lijst van bestanden die "in die directory zitten". Dat is een lijst van bestandsnamen met aan iedere bestandsnaam gekoppeld het adres van een inode. Een wat? Een inode, of i-node, of index-node.

Inodes zijn gegevens in een file system (op mass-storage). Zij behoren tot de manier waarop een file system in elkaar zit. Zij bevatten gegevens over bestanden (dat noemt men 'meta-data'). Welke gegevens zijn dat? Dat verschilt per type file system (EXT4 doet dat anders dan Reiserfs), maar er zijn in ieder geval:

- de grootte van het bestand in bytes;
- de `uid` van de eigenaar van het bestand;
- de `gid` van het bestand;
- de mode, dat is een rijtje bits dat het type en de bescherming/gebruiksmogelijkheden van het bestand aangeven (zie *Korte uitleg: Wie mag wat met welk bestand?*);
- tijdstempels/timestamps (zie hieronder);
- een teller die aangeeft hoeveel "harde links" dit bestand heeft (zie *Korte uitleg: Harde en zachte links*);
- verwijzingen naar de plek(ken) op de schijf waar de inhoud van het bestand staat.

We hebben dit hierboven in een top-down benadering uitgelegd. Als we het nu bottom-up samenvatten hebben we:

1. de inhoud van een bestand, dat is dus de inhoud waarin je meestal bent geïnteresseerd (en LibreOffice document met een brief aan je grootmoeder, een plaatje in PNG formaat, ...);
2. de gegevens over het bestand (de meta-data): waar staat het, wat is het, wie mag er iets mee, hoe groot is het, wanneer gewijzigd (maar niet de naam van het bestand) in een inode;
3. een bestand van het type directory met daarin een tabel van de bestandsnamen in die directory met een verwijzing naar de inode van die bestanden.

Kun je zien welk bestand welke inode heeft?

```
henk@boven:~/test> ls -1 -i
1441878 bestanden
```

```
1441888 fff
6029554 file
6029495 hhh
6029668 script
6030011 spsp
henk@boven:~/test>
```

Het programma `ls` hoeft hiervoor dus niet in de inodes te kijken, de inhoud van de directory is genoeg om dit te maken.

Maar voor het volgende moet `ls` wel in de inodes wezen:

```
henk@boven:~/test> ls -lc
totaal 20
drwxr-xr-x 2 henk wij 4096 17 mei 15:48 bestanden
drwxr-xr-x 2 henk wij 4096 18 mei 12:37 fff
-rw-r--r-- 1 henk wij    0 26 mrt 10:31 file
-rw-r--r-- 1 henk wij  566 13 mrt 11:08 hhh
-rwxr--r-- 1 henk wij   49 13 mrt 10:39 script
-rw-r--r-- 1 henk wij    3 31 mrt 19:29 spsp
henk@boven:~/test>
```

Behalve de naam komen alle andere gegevens uit de diverse inodes. Merk op dat in dit geval de `ctime` wordt getoond in plaats van de `mtime`. we kennen de volgende timestamps:

- `mtime`: de laatste keer dat het bestand is gewijzigd;
- `atime`: de laatste keer dat het bestand is geraadpleegd;
- `ctime`: de laatste keer dat de inode is gewijzigd.

Enkele gevolgen van de scheiding tussen bestandsnaam en inode:

- Als je een bestand van naam verandert verandert de inode niet.
- Als je een bestand naar een andere plek in een filesysteem verhuist verandert de inode niet. Alleen de vermelding in de ene directory (bestandsnaam/inode) wordt verplaats naar de andere directory.

Beide bovenstaande handelingen gaan dus heel efficiënt, hoe groot het bestand ook is. Maar verplaatsen naar een ander filesystem houdt het kopieëren van het hele bestand in. Vandaar dat men soms niet begrijpt waarom de ene "move" zoveel vlugger klaar is dan de andere.

- Als een bestand wordt verwijderd verdwijnt de verwijzing in de directory, maar de inode, en ook de ruimte van het bestand, wordt pas vrijgegeven nadat alle processen die dat bestand nog gebruiken het bestand hebben gesloten (of het proces is gestopt). Het bestand blijft dus bruikbaar voor processen die het al in gebruik hadden. Dit maakt het vervangen van een programma (update) makkelijker. Bijvoorbeeld:
  - het bestand `/usr/bin/amarok` wordt weggegooid, maar de muziek draait gewoon door en eventuele benodigde geheugen segmenten worden gewoon ingelezen want alle gegevens staan nog op de schijf;
  - een nieuw bestand `/usr/bin/amarok` wordt aangemaakt. Als Amarok nu

weer wordt gestart is dat de nieuwe versie en die draait samen met de oude versie.

- Meer bestandsnamen kunnen naar dezelfde inode verwijzen. Zie daarvoor *Korte uitleg: Harde en zachte links*.

Als laatste een gereedschap dat de zaken uit een dirctory en een inode uitgebreider laat zien dan `ls`:

```
henk@boven:~/test> stat lnt
  Bestand: ‘lnt’
  Grootte: 4096         Blokken: 8          IO-blok: 4096   map
Apparaat: 806h/2054d   Inode: 6029480      Koppelingen: 2
Toegang: (0755/drwxr-xr-x)   UID: (  500/    henk)   GID: (  500/     wij)
Toegang:   2013-07-05 17:42:10.646708743 +0200
Gewijzigd: 2013-07-05 17:42:08.669694987 +0200
Veranderd: 2013-07-05 17:42:08.669694987 +0200
Ontstaan:  -
henk@boven:~/test>
```

Misschien is e.e.a. duidelijker in het Engels:

```
henk@boven:~/test> LANG=C stat lnt
  File: 'lnt'
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 806h/2054d      Inode: 6029480     Links: 2
Access: (0755/drwxr-xr-x)  Uid: (  500/    henk)   Gid: (  500/     wij)
Access: 2013-07-07 21:09:20.926417472 +0200
Modify: 2013-07-05 17:42:08.669694987 +0200
Change: 2013-07-05 17:42:08.669694987 +0200
 Birth: -
henk@boven:~/test>
```