# Linux: Bestandsrechten

D. Leeuw

5 april 2025
v.1.0.0

# Over dit Document

## 0.1 Leerdoelen

Na het bestuderen van dit document heeft de lezer kennis van:

- de Linux shell, de syntax, variables en quoting
- hoe er gebruik gemaakt kan worden van de aanwezige documentatie op het systeem (man, info)
- het aanmaken van gebruikers en groepen en waar deze informatie wordt opgeslagen
- hoe er gewerkt kan worden met bestand en directories en de rechten daarop
- welke speciale files en direcories er zijn
- de manipulatie van bestanden en directories (grep, less, cat, head, tail, sort, cut wc)
- shell scripting
- hoe er informatie over de hardware verzameld kan worden via de commandline
- hoe er netwerk informatie van het systeem opgevraagd kan worden
- logging en systeem monitoring

## 0.2 Voorkennis

Voor een goed begrip van dit document is er geen voor kennis vereist, wel is het van belang dat de gebruik een Linux-distributie geïnstalleerd heeft staan.

# Inhoudsopgave

# Hoofdstuk 1

# Toegangsrechten op bestanden en directories

Elk besturingssysteem dat meer dan één gebruiker kent heeft een manier nodig om ervoor te zorgen dat de twee gebruikers niet bij elkaars bestanden kunnen als ze dat niet willen. Ook moeten gebruikers niet bij de bestanden van de beheerder (root) kunnen komen. Er moet dus door het systeem bijgehouden worden wie welke rechten heeft op een bestand.

Bij het ontwerp van Unix was de centrale gedachte dat alles binnen het systeem gerepresenteerd werd door het idee van een bestand: 'Everything is a file'. Dus niet alleen documenten zijn bestanden, maar ook aangesloten printers, harddisks, etc. Omdat alles een bestand is kan je met rechten op bestanden ook bepalen wie toegang heeft tot bepaalde stukken hardware.

Dit hoofdstuk gaat over de rechten op Unix-achtige systemen zoals Linux. Met `ls -l` kan je een lijst van bestanden opvragen die meer weergeeft dan alleen de bestandsnaam. De output van -l -l zou er zo uit kunnen zien:

```
total 31657256
drwxr-xr-x  3 dennis dennis        4096 Jan  9 08:38  Apps
-rw-r--r--  1 dennis dennis 30752636928 Mar  5  2020
    bootdisk_32G_20200305.img
drwxr-xr-x  2 dennis dennis        4096 Jan  8  2020  Desktop
drwxr-xr-x 10 dennis dennis        4096 Jul 13  2022  Documents
drwxr-xr-x  4 dennis dennis       16384 Jan 19 13:21  Downloads
-rw-r--r--  1 dennis dennis          13 Jan 22  2020  hello.txt
-rw-r--r--  1 dennis dennis         131 Jan 22  2020  HELLO.txt
drwx------  2 root   root         16384 Sep 16  2019  lost+found
drwxr-xr-x  5 dennis dennis        4096 Mar 15  2022  Nextcloud
-rw-r--r--  1 dennis dennis      137665 Mar  3  2022  output.pdf
drwxr-xr-x  4 dennis dennis        4096 May 26  2020  PGP
drwxr-xr-x  3 dennis dennis        4096 Apr  5  2022  Pictures
drwxr-xr-x  4 dennis dennis        4096 Sep 30  2019  Projects
```

```
drwxr-xr-x  3 dennis dennis        4096 Feb  1  2022  src
-rw-r--r--  1 dennis dennis         420 Mar 31  2020  Teams.txt
drwxr-xr-x  2 dennis dennis        4096 Sep 25  2019  Templates
-rw-r--r--  1 dennis dennis          65 Apr  6  2022  test.py
-rwxr--r--  1 dennis dennis         345 Apr 16  2020  test.sh
drwxr-xr-x  2 dennis dennis        4096 Sep 23 15:32  tmp
drwx------ 24 dennis dennis        4096 Jan 18 13:23 'VirtualBox VMs'
```

De kolommen van links naar rechts geven weer:

- De rechten op een bestand,
- Het aantal links naar dit bestand
- De eigenaar van het bestand
- De groepseigenaar van het bestand
- De grootte van het bestand
- De maand van de laatste wijziging
- De dag van de laatste wijziging
- Het jaar van de laatste wijziging (of het tijdstip van de laatste wijziging als het minder dan een jaar geleden is)
- De naam van het bestand

# Hoofdstuk 2

# Bestandstypen

Waar Windows de extensie van een bestand gebruikt om te bepalen wat voor type bestand het is, gebruik Linux hoofdzakelijk de eerste bytes van een bestand om aan te geven wat voor bestand het is. Met het `file` commando kunnen we achterhalen met wat voor bestand we te maken hebben zonder het bestand te openen.

```
$ touch onbekend_bestand.txt
$ file onbekend_bestand.txt
onbekend_bestand.txt: empty
```

We hebben een leeg bestand, dus weet het `file` commando niet wat voor bestand we hebben, zelfs niet met de .txt uitgang.

```
$ echo '%PDF-1.5' > onbekend_bestand.txt
$ file onbekend_bestand.txt
onbekend_bestand.txt: PDF document, version 1.5
```

Een bestand met een .txt extensie en een kop in het bestand dat aangeeft dat het een PDF zal door het systeem gezien worden als een PDF.

```
$ echo '#!/bin/bash' > onbekend_bestand.txt
$ file onbekend_bestand.txt
onbekend_bestand.txt: Bourne-Again shell script, ASCII text executable
```

De #! geeft aan dat het een shell-script is en /bin/bash zegt dat het om de Bourne-Again shell gaat.

# Hoofdstuk 3

# De eigenaar

Met `chown` kan de eigenaarschap van een bestand wijzigen. Om dit te kunnen testen moeten we een aantal zaken regelen:

- Maak een groep aan met de naam **samen**
- Voeg jezelf en de gebruiker eengebruiker toe aan deze groep
- Maak een directory aan `/home/samen`

Nu gaan we ervoor zorgen dat de groep samen toegang heeft tot de directory samen en dat jezelf de hoofd eigenaar wordt. We beginnen met het laatste:

```
$ sudo chown $(id -un) /home/samen
```

We hebben in dit commando een extraatje toegevoegd. We kunnen een variabele gebruiken om de gebruikers naam in te zetten en die gebruiken bij `chown`. Er bestaat echter ook een mogelijkheid om in de shell direct een commando aan te roepen en deze als variabele te gebruiken en dat is wat we hier gedaan hebben. We hebben `id -un` aangeroepen (wat onze gebruikersnaam terug geeft) en deze hebben we gebruikt als optie aan `chown`. Dus eigenlijk staat er `sudo chown username /home/samen`. Bekijk met `ls -l` het resultaat.

Nu gaan we zorgen dat de beide gebruikers gebruik kunnen maken van deze directory. Eerst moeten we zorgen dat de groep **samen** de groepseigenaar wordt van de directory:

```
$ sudo chown .samen /home/samen
```

Door een punt voor **samen** te zetten gevevn we aan dat we de groepseigenaarschap willen wijzigen. Bekijk met `ls -l` het resultaat.

Beide commando's hadden we ook in één keer kunnen doen:

```
$ sudo chown dennis.samen /home/samen
```

# Hoofdstuk 4

# read, write and execute

De rechten op een bestand zijn opgedeeld in drie blokken. Elk blok kan de waarden r, w, en x bevatten. Er zijn dus in totaal 9 posities (rwxrwxrwx). De mogelijke rechten zijn:

**r** Read

**w** Write

**x** Execute

Elk blokje heeft zijn eigen betekenis:

| Type | Owner | Group | Other |
|---|---|---|---|
| d | rwx | rwx | rwx |

Er zijn dus rechten te vergeven voor de eigenaar, voor de groep en voor de rest van de wereld.

Voor normale bestanden geldt dat je met read rechten een bestand mag openen en dus het kunt lezen, met write rechten mag je bestand ook schrijven en dus wijzigen en met execute rechten mag je een bestand opstarten, dat is natuurlijk alleen handig als je een bestand ook daadwerkelijk op mag starten zoals een script of programma.

Voor directories zijn de regels even anders. Met leesrechten mag je zien welke bestanden er in een directory staan en mag je deze lezen, je kan dus `ls` gebruiken en een bestand openen in de directory, met schrijfrechten mag je nieuwe bestanden aanmaken en met execute het je daadwerkelijk toegang tot de directory kortom je kunt `cd` gebruiken om in de directory te komen.

Computers zijn slecht in namen, maar goed in nummers, dus de rechten r, w en x moeten omgezet worden naar een getal waarmee de computer kan werken. Omdat computers heel goed zijn in binair zijn de rechten omgezet naar binaire getallen. Een 1 betekent dat het recht gegeven is, een 0 zegt dat het recht niet gegeven is. 101 betekent dus leesrechten en execute-rechten. Deze binaire manier van tellen kan ook decimaal geschreven worden 101 is

dan 5. Een blok van rechten voor eigenaar, groep en de wereld zou er zo uit kunnen zien: rwxr-x-r–. Omgerekend naar binair is dat 111 101 100 en dat per stukje omgezet naar decimaal is 754.

Om de rechten op een bestand te wijzigen is er het commando `chmod`. Je kan `chmod` gebruiken om de rechten op bestanden te wijzigen door gebruik te maken van read, write en execute of door gebruik te maken van de decimale waarden van de rechten. Een voorbeeld van het gebruik van de decimale waarden zou voor de directory **samen** er zo uit kunnen zien:

```
$ sudo chmod 777 /home/samen
```

We hebben nu de eigenaar, de groep en de wereld alle rechten gegeven, dus de beide gebruikers in de groep **samen** kunnen nu bij alle documenten die ze in deze directory aanmaken.

Helaas hebben we ook alle rechten gegeven aan Other. Dat betekent dat de hele wereld bij alle documenten kan. We kunnen met chmod ook rechten afnemen. Gebruik eens:

```
$ sudo chmod o-x /home/samen
```

Na een `ls -l` zal je zien dat van other (o) de execute-rechten (x) verdwenen zijn. Dat kunnen we ook met meerdere rechten doen:

```
$ sudo chmod g-w,o-rw /home/samen
```

We ontnemen hier van other de read en write rechten en van de group rechten verwijderen we de schrijfrechten. Het plus-teken kunnen we gebruiken om rechten toe te kennen:

```
$ sudo chmod g+w /home/samen
```

zorgt ervoor dat de groep weer schrijfrechten heeft.

# Hoofdstuk 5

# Het 4de-bit

De rechten r, w en x zijn elke keer blokjes van 3-bits. Het totaal is dus 3x3 is 9 bits lang. Dat is een raar getal in de computerwereld en dan ook niet helemaal correct, eigenlijk is het blok 12-bits lang en bestaat het uit 4x3 bits. De triplet 777 is dus eigenlijk een quadlet 7777. De eerste 7 wordt gebruikt om extra zaken in te coderen. De rechten op een bestand zonder extra functionaliteit is dus eigenlijke 0777, maar daar laten we de 0 meestal weg.

Met de extra rechten kunnen we de volgend functionaliteit weergeven:

- SUID bit
- SGID bit
- Sticky bit

## 5.1 SUID-bit

Het SUID-bit is het meest linkse bit uit de reeks en heeft dus een decimale waarde van 4. Het is de bit die hoort bij de user dus kan je het SUID-bit ook setten door `u+s` te gebruiken. Voorbeelden:

```
$ mkdir SUID.d
$ touch SUID.txt
$ chmod u+s SUID.d
$ chmod 4777 SUID.txt
$ ls -ld SUID*
drwsr-xr-x 1 dennis dennis  4096 Feb  7  2022 SUID.d
-rwsrwxrwx 1 dennis dennis 63960 Feb  7  2022 SUID.txt
```

Het eerste rwx blokje is nu veranderd in rws om aan te geven dat het SUID-bit gezet is.

Het bit op een bestand zorgt ervoor dat, als je het bestand kunt opstarten, de applicatie draait onder de username van de eigenaar. Dus als een bestand als eigenaar heeft root.admin en het SUID-bit is gezet dan zal bij opstarten het programma draaien met root-rechten. Het voordeel is dat gewone gebruikers zo programma's kunnen opstarten met rechten die ze normaal niet hebben. Het nadeel is dat er een security-lek zou kunnen ontstaan, dus je moet heel voorzichtig zijn met deze rechten.

Een voorbeeld op een Linux systeem waar het SUID-bit gebruikt wordt is op het `passwd` programma. Een gebruiker moet instaat zijn om zijn wachtwoord te wijzigen, terwijl het `/etc/shadow` bestand alleen lees en schrijfbaar is door root.

Het bit op een directory heeft geen betekenis in GNU/Linux.

## 5.2 SGID-bit

Het SGID-bit is het tweede bit uit de reeks en is dus decimaal: 2. Het behoort bij de groepsrechten en kan dus gezet worden met `g+s`. Voorbeelden

```
$ mkdir SGID.d
$ touch SGID.txt
$ chmod g+s SGID.d
$ chmod 2777 SGID.txt
$ ls -ld SGID*
drwxr-sr-x 2 dennis dennis 4096 Jan 24 09:28 SGID.d
-rwxrwsrwx 1 dennis dennis    0 Jan 24 09:28 SGID.txt
```

Het SGID-bit op een bestand zet de effectieve groep waaronder een applicatie draait. Dus net als wat de SUID-bit doet voor de gebruiker doet het SGID-bit voor de groep,

Het SGID-bit op directories betekent dat de groep eigenaarschap wordt doorgegeven aan nieuwe bestanden of directories die aangemaakt worden in de directory. Dus als we een directory hebben met het SGID-bit en de groepseigenaar van de directory is **samen** en we maken in die directory een nieuwe bestand aan dan wordt de groepseigenaar weer **samen** ongeacht de groep waarin de gebruiker zit die het bestand aanmaakt. Natuurlijk moet een van de groepen van die gebruiker wel **samen** zijn.

## 5.3 Sticky-bit

Het Sticky-bit kan gezet worden met de decimale waarde 1 of met `o+t`. Voorbeelden:

```
$ mkdir sticky.d
$ touch sticky.txt
$ chmod o+t sticky.d
$ chmod 1777 sticky.txt
$ ls -dl sticky*
drwxr-xr-t 2 dennis dennis 4096 Jan 24 10:04 sticky.d
-rwxrwxrwt 1 dennis dennis    0 Jan 24 10:04 sticky.txt
```

Bij het rechtenblok van other is de x vervangen door een t.

Voor bestanden heeft het Sticky-bit in Linux geen betekenis.

Als het Sticky-bit gezet is op een directory dan betekent dat alleen de eigenaar, de eigenaar van de directory en root het bestand kunnen weggooien en hernoemen. Zelfs als iemand in de juiste groep zit en de groep heeft schrijfrechten dan nog heeft die persoon niet de rechten. Een voorbeeld van het gebruik van het Sticky-bit is het `/tmp` directory.

# Index