Linux: Introductie tot de Command Line Interface

D. Leeuw

13 mei 2025

1 Over dit Document

1.1 Leerdoelen

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

- de Linux shell, de syntax, variabelen en quoting
- de commando's: ls, pwd, whoami, history, mkdir, cd, rmdir,
- de speciale betekenis van de ~
- de Shell-variabelen: PATH, HOME, USER
- quotes en escaping
- FHS - Filesystem Hierarchy Standard
- Bestands typen

Dit document sluit aan op de volgende onderdelen van de LPI:

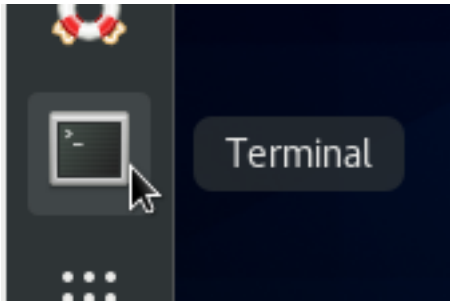
- LPI Linux Essentials 010-160 - 6.2.1 Command Line Basics (weight: 3)
- LPI Linux Essentials 010-160 - 6.2.3 Using Directories and Listing Files (weight: 2)

1.2 Voorkennis

Voor een goed begrip van dit document is er geen voorkennis vereist, wel is het van belang dat de lezer een Linux-distributie geïnstalleerd heeft.

2 Waarom de commandline interface?

De eerste vraag is meestal waarom we de command line zouden gebruiken als we al een grafische interface hebben. Het meest simpele antwoord is dat Unix van oorsprong alleen maar een command line interface of CLI had. Maar een beter antwoord is dat een grafische interface veel resources (geheugen en processor) gebruikt en die resources kunnen we beter inzetten voor de taken die we het systeem geven. Veel Linux machines draaien als servers in een serverruimte en staan op de achtergrond hun ding te doen, bijvoorbeeld als webserver. Voor die functie is geen grafische interface noodzakelijk terwijl op een drukbezochte website elk stukje processor of geheugen nodig kan zijn



Figuur 1: Terminal op de Dash

om de website soepel te laten lopen. Wat we niet nodig hebben installeren we dan ook niet op de machine, dus geen grafische interface.

Daarnaast zal je hopelijk ervaren dat, omdat Linux op de schouders staat van de vele jaren ervaring uit de Unix-wereld, dat de command line een enorm krachtige interface is om mee te werken. Vaak kan je op de command line dingen sneller en makkelijker doen dan je in een grafische interface zou kunnen. Wees niet bevreest als dat in eerste instantie niet zo lijkt. De leercurve kan, zeker als je niet veel ervaring hebt met bijvoorbeeld de command prompt of powershell van Windows, soms erg stijl zijn.

De voorbeelden in dit document zijn gemaakt op een Debian machine, maar daarmee niet Debian specifiek. Ze zullen op een Red Hat, Centos, Mint of Ubuntu machine geen wezenlijk andere resultaten opleveren. De grote verschillen tussen de verschillende distributies zitten vooral in de grafische interface, de verschillen op de command line zijn minimaal voor de onderwerpen die in dit document worden behandeld. Het doel van dit document is je vetrouwt maken met de CLI, niet het configureren van van het systeem.

3 Toegang tot de CLI

Een Linux server die geïnstalleerd is zonder grafische interface heeft op zijn monitor de prompt Login: daar kan je inloggen als gebruiker. Via de toets combinatie ALT en F1-F6 kan je schakelen naar verschillende consoles zodat je meerdere keren ingelogd kan zijn.

Als je werkt vanuit een grafische interface kan je gebruik maken van de Terminal of Terminal Emulator applicatie. Deze brengt je bij de command line interface, zie figuur 3

4 De shell

Alles wat je op de prompt intypt wordt opgevangen door de shell. De shell is een schil rond de kernel die commando's van gebruikers aanneemt en deze doorgeeft naar de kernel. Shell is het Engelse woord voor schelp en een schelp zorgt ervoor dat het weekdier dat in de schelp leeft beschermt wordt tegen de buitenwereld. Dat is bij de shell net zo, de shell beschermt de kernel tegen de gebruiker. De shell wordt ook wel een command interpreter genoemt omdat hij commando's van de gebruiker interpreteert.

Linux gebruikt standaard de bash shell. Bijna elke linux distributie levert deze mee en heeft deze als standaard shell. Van oudsher werd op Unix systemen sh meegeleverd. Het was een van de eerste programma's die werd geschreven voor Unix door Ken Thompson. Tussen 1976 en 1979 schreef Stephen Bourne een vervanging voor sh wat de standaard werd in Version 7 Unix. De naam bleef echter sh op het Unix systeem. Toen het GNU project een vrij en open source systeem wilde maken was ook daar een van de eerste programma's die er moest komen een shell. Dat werd bash, de naam bash staat voor Bourne Again SHell. Het is een open source versie van de shell geschreven door Stephen Bourne.

Unix en Linux commando's en bestandsnamen zijn case sensitive. Dat betekent dat 'hello.txt' niet hetzelfde is als 'Hello.txt'. Dit zijn twee verschillende bestanden.

Commando's of opdrachten aan de shell hebben een vaste vorm (syntax). Ze zien er zo uit:

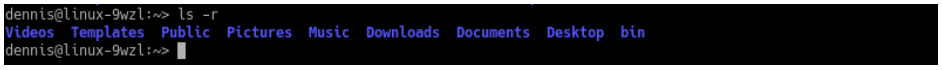
```
commando<spatie>optie(s)<spatie>argument(en)
```

De spaties zorgen ervoor dat de shell weet wanneer een volgend deel begint, voor de eerste spatie staat het commando, daarna volgen er geen of enkele opties en tot slot zijn er geen of enkele argumenten. Bijna alle commando's houden deze syntax aan, hoewel er ook uitzonderingen zijn.

Het commando ls laat een lijst met bestanden zien. Als we het gebruiken zonder argumenten dan ziet dat er ongeveer uit als in figuur 2

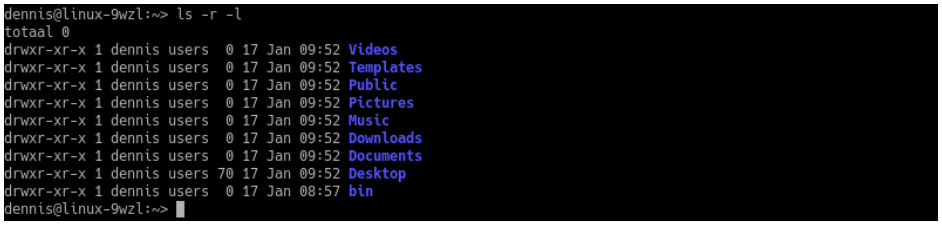
Figuur 2: Directory listing

Opties worden van argumenten onderscheiden doordat opties beginnen met een min-teken (-). Geven we aan ls een optie me, bijvoorbeeld -r voor reverse, of wel sorteer omgekeerd dan zien we wat we in figuur 3 zien. We zien nu dat Videos vooraan staat en bin achteraan.



Figuur 3: Reverse order listing

We kunnen ook meerdere opties meegeven. De optie -l geeft een long list, ofwel een lijst die veel meer informatie per bestand of directory laat zien. Je



Figuur 4: Reverse and long listing

mag de opties apart meegeven zoals we in figuur 4 gedaan hebben maar je mag ze ook samenvoegen zoals:

```
$ ls -lr
```

of

```
$ ls -rl
```

We kunnen ook een argument meegeven, bijvoorbeeld:

```
$ ls Documents/
```

Het argument is de naam van een directory, in dit geval Documents, en we krijgen geen output omdat de directory leeg is.

De linux shell heeft ook een handigheidje om het leven makkelijker te maken, of beter om minder te hoeven typen, dat handigheidje heet automatisch aanvullen. Als je op de prompt een p typt zonder een enter te geven en daarna de tab-toetst indrukt dan gebeurd er niets, druk je nog een keer op de tab-toets dan krijg je de melding:

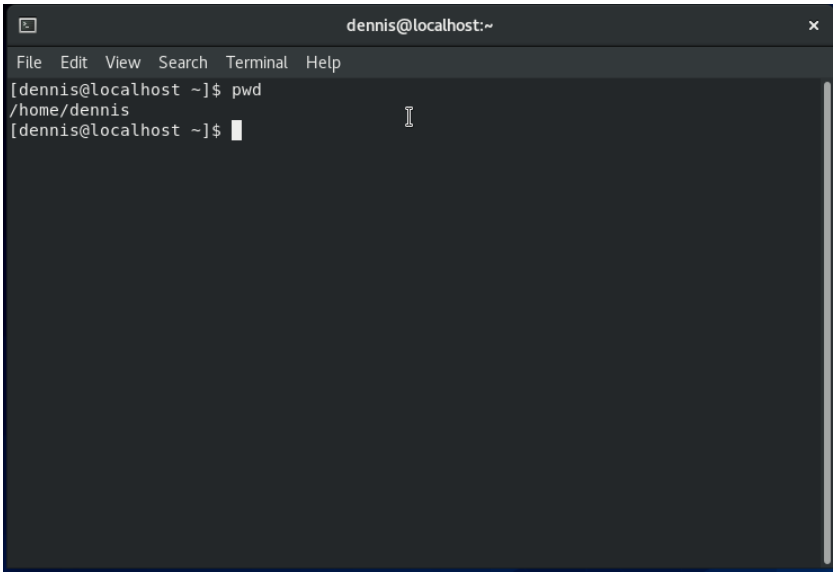
```
Display all 177 possibilities (y or n)
```

Type n want we willen niet 177 mogelijkheden zien. Wat het systeem ons verteld heeft is dat het 177 commando's kent die met een p beginnen, voegen we nu aan onze p een w toe en typen we 1x tab. Dan vult de shell dit aan met de d en staat er pwd.

Dit automatische aanvullen kun je doen met commando's maar ook met bestanden en directories. Er wordt weleens gezegd dat je het toetsenbord van een Linux-beheerder kan herkennen aan de versleten tab-toets.

4.1 De prompt

Als je op de command line bent heb je een commandprompt. De commandprompt is het deel voor de cursor, zie 5



Figuur 5: Console

Helemaal links begint het met de login naam, dennis, daarnaast op welke host je bent in gelogd, localhost, daarna volgt de directory waarin je je nu bevindt, ~, en tot slot de prompt voor een normale gebruiker: $. Als je ingelogt bent als root dan staat er inplaats van een $ een #. In deze cursus zullen we alle commando's die je in moet of kan typen op de prompt vooraf laten gaan door een $ als je het commando als gewone gebruiker moet intypen en door een # als je het als root moet doen. De $ en # moet je dus niet intypen.

Typen we

```
$ pwd
```

dan zien we het pad naar de directory waarin we ons bevinden. In dit geval zal dat onze gebruikers directory zijn met als volledig pad /home/<gebruikersnaam>. Voor het voorbeeld waarbij de gebruiker dennis ingelogd is is dat dus /home/dennis. Merk op dat op Linux de directories gescheiden worden door de forward-slash, /, dit in tegenstelling tot Windows waar de backward-slash, \, gebruikt wordt als scheidingsteken.

Nu weten we waar we zijn. Als we willen weten onder welke naam we zijn ingelogd dan gebruiken we

```
$ whoami
```

Als je dit commando nu intypt dan zal je je eigen loginnaam zien.

```
$ hostname
```

laat zien op welke machine we zitten. Als ik dit commando intyp krijg ik terug 'localhost.localdomain', wat betekent dat mijn machine nog geen echte naam gekregen heeft. Dit kan op je eigen netwerk anders zijn, omdat dit afhankelijk is van server die de IP-adressen uitdeelt.

```
$ hostname -s
```

Geeft alleen de naam van de hostname terug. De -s staat voor short, ofwel kort.

Nu weet je een beetje wie je bent en waar je je bevindt, zowel op welke machine als in welke directory. Welkom thuis.

4.2 De home-directory

Wat waarschijnlijk wat uitleg behoeft is dat je prompt laat zien dat je je in de directory ~ bevindt. Dat is geen werkelijk bestaande directory maar een handige afkorting die we binnen de CLI kunnen gebruiken om de home-directory van een gebruiker aan te duiden. Je eigen plek waar je je bestanden kan opslaan heet je home-directory en die kan dus ook aangeduid worden met de ~ (tilde). Je bevindt je dus in je eigen home-directory.

De werkelijke plek waar je je nu bevindt in het bestandssysteem kan je laten zien door

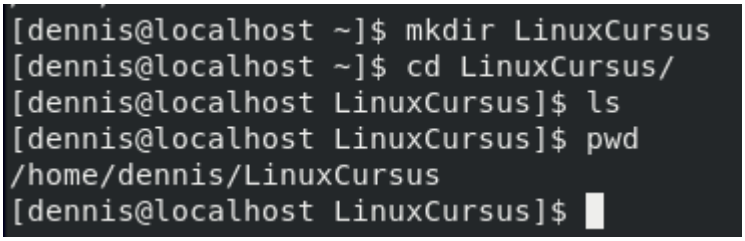
```
$ pwd
```

te typen. Het pwd commando toont de werk-directory en pwd staat dan ook voor print working directory. Je zult zien dat pwd iets terug geeft als

```
/home/dennis
```

waarbij 'dennis' vervangen is door je eigen gebruikersnaam. De / (forward slash) is het scheidingsteken dat in Linux gebruikt wordt om directory namen van elkaar te scheiden. Je bevindt je dus drie directories diep vanaf de root van het bestandssysteem. / is de root-directory, home/ is de tweede directory en dennis/ is de derde directory.

Laten we eens kijken wat er in onze directory te vinden is. Om een overzicht te krijgen van de directories en bestanden in onze directory typen we 'ls'. 'ls' is een afkorting voor list. Ofwel geef een lijst van aanwezige files



Figuur 6: Het maken van de LinuxCursus directory

en directories. Afhankelijk van de distributie of de systeembeheerder zullen er al wat zaken aanwezig zijn.

We gaan onze eerste eigen directory aanmaken. Type hiervoor:

```
$ mkdir LinuxCursus
```

met behulp van 'ls' kan je zien dat er een nieuwe directory bijgekomen is.

Met 'cd', change directory, kunnen we ons verplaatsen in de directory:

```
$ cd LinuxCursus
```

na de enter zal je zien dat de prompt gewijzigd is en een 'ls' geeft een lege output omdat er niets in de directory staat. Met 'pwd' kan je controleren dat je daadwerkelijk in de nieuwe directory staat.

Nu gaan we een aantal directories tegelijk aanmaken hiervoor typen we:

```
$ mkdir Data Documenten Muziek
```

na een 'ls' zie je nu drie nieuwe directories. Zo zie je dat je door meerdere argumenten mee te geven aan mkdir meer dan 1 directory kan aanmaken.

Met rmdir kan je directories weggooien.

```
$ rmdir Data
```

Dit gooit de Data directory weg, na 'ls' zie je nu nog maar twee directories.

Als we het systeem verder willen verkennen dan kunnen we met

```
$ cd ~
```

ervoor zorgen dat we weer in onze home-directory terecht komen, dat scheelt toch een hoop typen die ~. Als we nu

```
$ cd ..
```

typen dan gaan we terug in de directory boom. We komen in de /home/ directory terecht. De dubbele punten naast elkaar (..) staan voor de directory die 1 stap lager ligt in de boom. We staan nu in de home/ directory die standaard alle directories van alle gebruikers bevat, dus als er meer gebruikers op

het systeem zouden zijn aangemaakt dan zouden ook van deze gebruikers de home-directories hier te vinden zijn. Een uitzondering is de home-directory van de systemadministrator van een Linux systeem. Deze beheerder heet 'root' en zijn of haar directory is /root/. Daar komen we later nog een keer op terug.

Als we weer

```
$ cd ..
```

typen dan komen we in de / directory terecht. Lager kunnen we niet gaat op een Linux systeem, we zijn nu bij de root-directory aangekomen. Let op de verwarring die kan ontstaan tussen de /root/ directory en de / directory beiden heten de root-directory maar de ene is van de gebruiker root en de ander is van het bestandssysteem.

Met alleen het 'cd' commando

```
$ cd
```

kom je ook weer terug in je eigen home-directory.

4.3 History

Een ander handigheidje van de shell is dat hij een geschiedenis (history) opslaat van gebruikte commando's. Met de pijltjes toetsen ↑ (pijltje-omhoog) en ↓ (pijltje-omlaag) kan je door de geschiedenis van je commando's scrollen. Omhoog is terug in de tijd en omlaag is het omgekeerde. Met CTRL-c breek je af waar je gebleven bent en met enter voer je het commando uit.

Met CTRL-r kan je zoeken in je history. Type maar eens CTRL-r en dan de p. Er zal vrijwel direct pwd verschijnen. Op enter drukken is dan het enige dat nog nodig is om het commando uit te voeren. Als je toch besluit dat je het commando niet wil uitvoeren dan druk je op CTRL-c om de zoekfunctie te verlaten.

Met !! herhaal je het laatste commando dat je gedaan hebt en met !<commando> herhaal je het laatste <commando> dat je gedaan hebt. Type nu eens

```
$ !!
```

dan zal het pwd commando opnieuw uitgevoerd worden.

```
$ !hostname
```

zal de hostname -s uitvoeren want dat is het laatste hostname commando dat we gedaan hebben.

4.4 Shell variabelen

De shell houdt voor de gebruiker informatie bij over de omgeving waarin hij werkt. Met welke gebruikersnaam zijn we ingelogd, wat is onze home-directory en nog veel meer van dit soort informatie. Deze informtie is opgeslagen in variabelen. Met het `echo` commando kunnen we die informatie uit die variabelen opvragen:

```
$ echo $USER
$ echo $SHELL
$ echo $HOME
$ echo $PWD
```

om een complete lijst te krijgen van alle variabelen die je in je huidige sessie tot je beschikking staan is er het commando `env`.

4.5 Eigen variabelen

Het is soms handig om eigen variabelen te gebruiken:

```
$ aap=1
$ echo $aap
```

Zoals je ziet gebruiken we bij het toewijzen van een waarde aan een variable (stop 1 in de variabele aap) geen $-teken voor de variabele. Alleen bij het gebruik van de variabele zetten we er een $-teken voor.

Variabelen in de shell hebben ook geen type, er bestaat geen integer of een string, dus we kunnen net zo makkelijk doen:

```
$ aap="Dag aap!"
$ echo $aap
```

het is een goede gewoonte om voor eigen variabelen kleine letters of kleine letter gemixed met hoofdletters te gebruiken en dat variabelen met alleen hoofdletters gereserveerd zijn voor de interne variabelen van de shell.

4.6 Shell Quotes en Escapes

In het hoofdstuk de Shell is de commando syntax uitgelegd. Je hebt daar geleerd dat het commando, de opties en de argumenten geschieden worden door spaties. Stel dat we een directory willen aanmaken met de naam `Mijn Documenten`. De naam bevat dus een spatie en zou gezien worden als twee argumenten. Om dit op te lossen gaan we gebruik maken van quotes of escape characters en dat is waar deze sectie over gaat.

4.6.1 Escapes

Speciale characters die in de shell al een betekenis hebben kunnen we gebruiken door ze te escapen, van een speciale teken te voorzien. Het escape character in de shell is de \, backslash. Voor het maken van een directory of bestand met een spatie erin kunnen we dit doen:

```
mkdir Directory\ met\ spaties
```

4.6.2 Quotes

In het vorige hoofdstuk hebben we gezien dat we aan de variable aap een waarde 1 gaven, maar ook de waarde "Dag aap!". Bij de ene gebruikten we geen quotes en bij de andere wel. Ook dit had te maken met het gebruik van spaties. Een slimmerik onder jullie zou kunnen zeggen, dan kan je ook een escape gebruiken en dat is helemaal waar, maar stel dat ik het escape-character letterlijk wil nemen en zodat die niet gebruikt wordt als escape, wat dan?

Voer het volgende maar eens uit:

```
$ aap=hello\ world
$ echo $aap
```

en nu met quotes:

```
$ aap="hello\ world"
$ echo $aap
```

Nu we toch met variabelen aan het spelen zijn, dan kan je ook een variable in een variabele gebruiken zoals hier:

```
$ mies="kees"
$ aap="Hello\ $mies"
$ echo $aap
```

Als we mies niet als variabele willen gebruiken dan kunnen we het dollarteken escapen:

```
$ aap="Hello\ \$mies"
$ echo $aap
```

we hebben nu twee escapes in één variabele.

We kunnen enkele quotes gebruiken zodat we helemaal geen escapes meer nodig hebben en alles letterlijk wordt weergegeven:

```
$ aap='Hello $mies'
$ echo $aap
```

Als we aan aap een complete zin als waarde willen geven kunnen we elke spatie escapen, maar is het korter om enkele quotes te gebruiken.

5 Het bestandssysteem

Net als met de standaardisatie van Unix in een POSIX standaard werden er in het begin op Linux Distributies soms bestanden in verschillende directories neergezet. Dat is voor programma's die op die systemen moeten draaien niet handig. Als de ene distributie /var/db heeft voor het plaatsen van databases en de ander /var/databases dan schept dat verwarring. De oplossing die hiervoor gekomen is is de Filesystem Hierarchy Standard. Deze is beschikbaar op `https://refspecs.linuxfoundation.org/fhs.shtml`. Hier gaan we heel globaal in op een aantal belangrijke directories, mocht je alle ins en outs willen weten dan raden we je aan om het document een keer te lezen.

/ De basis van het bestandssysteem wordt bepaald door de root-directory, zo genoemd omdat de vertakkende directories op een boom structuur lijkt en het Engelse root betekend wortel.

Een ls van / laat ons een aantal verschillende directories zien. Waarvan we er een aantal zullen behandelen.

/**home**/ bevat de directories waarin gebruikers hun bestanden kunnen zetten. Een uitzondering hierop is de directory waarin de root gebruiker (de baas of administrator van het systeem), zijn bestanden kan opslaan. Die directory is `/root/`.

/**etc**/ Deze directory bevat de configuratiebestanden van het systeem. Als je een instelling systemwide wilt wijzigen is dit de plek om te gaan zoeken. De configuratie bestanden voor een gebruiker staan in zijn of haar home-directory.

/**boot**/ Deze directory bevat bestanden die cruciaal zijn voor het opstarten maar die geen commando zijn. Hier vinden we de kernel en bestanden die behoren bij de bootloader.

/**dev**/ Omdat op een Linus systeem alles een bestand is vind je in deze directory de bestanden die verwijzen aan devices. Devices worden verder besproken in het hoofdstuk over devices. Dus daar gaan we later nog op in.

/**var**/ is de directory voor de systeem opslag van variabele data zoals bijvoorbeeld de logbestanden, databases, etc. De log bestanden kan je vinden in `/var/log/`.

/srv/ bevat de data van de diensten die door het systeem worden aangeboden. Data van web- of ftp-servers kan hier gevonden worden.

5.1 Everything is a file

In Unix en Unix-achtige operating systems zoals Linux is het basis principe dat alles een bestand (file) is. Dit betekent dat alles binnen het systeem; documenten, directories, harddisks, printers, toetsenborden, maar ook processen weergegeven worden als bestanden. Het voordeel hiervan is dat dezelfde commando's en API's gebruikt kunnen worden voor verschillende onderdelen van het besturingssysteem.

Het filesysteem is een enkele boom van bestanden en directories, zonder onderscheid tussen disks en partities. Zelfs verplaatsbare media zoals USB-sticks en DVDs zijn onderdeel van deze boom als ze 'gemount' zijn. Ook processen (/proc) en de kernel (/sys) is voor een groot deel benaderbaar via het bestandssysteem.

Met `ls` kan je in bijvoorbeeld `/proc` kijken en zien welke processen er zijn. Je ziet er nummers staan en die nummers komen overeen met de process nummers zoals ook `ps` die heeft. Doe maar eens:

```
$ ps aux
```

en je ziet in de tweede kolom dezelfde nummers staan als in `/proc/`. `ps` is ook het commando om te zien welke processen er op je systeem actief zijn. We zullen `ps` in een volgend hoofdstuk uitgebreid behandelen.

5.1.1 Bestandstypen

In de eerste kolom van `ls -l` vinden we de bestandsrechten en het geeft tevens aan met welk type bestand we te maken hebben. Naast normale bestanden (zoals tekstbestanden) hebben we op POSIX-compliant systemen ook speciale bestanden zoals directories. De onderstaande tabel geeft de mogelijke bestandtypen weer.

Mode veld	Bestandstype	Beschrijving
-	normaal bestand	Documenten, etc.
d	directory	Directories bevatten geen bestand, maar een overzicht van de bestandsnamen waaraan gekoppeld referenties naar iets wat inodes worden genoemd. De inodes bevatten de daadwerkelijke bestanden en meta-data (eigenaar, groep, permissies, time stamps, etc.). Door deze manier van werken kan een bestand (met meta-data) verschillende namen hebben (hard-link), mits binnen één bestandssysteem (partitie of disk). Bij verschillende bestandsnamen kunnen dezelfde inodes vermeldt staan.
l	Symbolic link	een link naar een bestand die over bestandssystemen heen kan gaan
b	Block device	Een apparaat waar van of waar naar toe data in een random manier gestuurd kan worden. Het hoeft dus niet in de juiste volgorde te zijn. Denk aan harddisks waar eerst sector 2014 en dan sector 5678 geschreven kan worden.
c	Character device	Een apparaat waar data in een stroom van characters naar of naar toe gestuurd kan worden.
p	FIFO	Ook bekend als named pipes. Een pipe verbindt het ene proces met het andere proces zodat data van proces 1 naar proces 2 gestuurd kan worden. Dit kan maar één kant op.
s	Socket	Verbindt net als FIFO's processen, maar dan op een manier dat er twee weg communicatie mogelijk is.

6 Vragen en opdrachten

6.1 Over de shell

1. Als ik ingelogd ben als gewone gebruiker met als gebruikersnaam `karel` en ik type in `cd ~` en ik type daarna `pwd` wat krijg ik dan als antwoord van de shell?

2. Wat geeft de shell als antwoord na het commando (test het op je eigen systeem):

```
$ echo $PATH
```

3. Welk commando gebruiken we om een lege directory weg te gooien?

4. Welke shell gebruiken we op een Linux systeem?

5. In mijn home-directory wil ik een directory met de naam `Mijn Documenten` maken wat moet ik dan intypen op de prompt om zeker te weten dat deze directory in mijn home-directory aangemaakt wordt?

6. Ga naar de LinuxCursus directory, laat de inhoud van de directory zien en maak een screenshot

6.2 Over het bestandssysteem

1. Waar kan ik op een Linux systeem de configuratie bestanden vinden?

2. Als ik `ls -l` gebruik hoe kan ik dan zien of een bestand een directory is?

3. Als er iets niet goed gaat op een Linux systeem in welke directory zou je dan kunnen gaan zoeken naar de fout?

Index