LINUX SYSADMIN BASICS

INHOUD

EEN BEETJE GESCHIEDENIS KAN NOOIT KWAAD...

Unix varianten zijn er al sinds het begin der (automatiserings)tijden. Begonnen als een experiment is het al snel uitgegroeid tot een volwaardig OS dat met name op universiteiten en natuurlijk in defensie toepassingen werd gebruikt. De voorloper van het internet (ARPANET, de "Advanced Research Projects Agency" van het ministerie van Defensie van de VS), werd grotendeels gebouwd op de eerste versies van Unix.

De eerste implementatie waren zeer low-level, dwz dat de operator direct toegang had tot de onderliggende systeemfuncties.

Uiteraard zorgen de fabrikanten van "gebruikers software", denk aan bijvoorbeeld Microsoft en Apple, ervoor dat kritieke systeemfuncties niet zo 1-2-3 vanuit de zogenaamde GUI (Graphical User Interface) te benaderen en te modificeren zijn. Je zult hiervoor echt op de laag onder de GUI moeten zijn, bijna op het niveau van het operating system zelf. En dan bij voorkeur ook nog op dàt niveau waar eventuele restricties geen rol meer spelen.

Nou is dat bij Windows-systemen meestal bijna niet te doen. Windows is van oudsher (en dan denk ik aan MS-DOS, damals, im Krieg…) een operating system dat alle kritieke systeemfuncties ontoegankelijk gemaakt heeft.
Dat past ook wel in de filosofie van hoe Microsoft de computermarkt benadert: power to the people, en die systemen, laat dat maar aan ons over.
Pas toen Microsoft met Windows NT ook de servermarkt op ging – ergens halverwege de jaren '90 van de vorige eeuw – ontstond langzaam de behoefte en de noodzaak om veel onderliggende systeemfuncties te ontsluiten: aan servers worden toch echt wel andere eisen gesteld dan aan het gemiddelde huis-tuin-en-keuken-pctje.
De grote concurrent Apple, had dat al iets eerder in de smiezen. Met de gigantische stap die Apple begin deze eeuw gezet heeft van OS-9 naar het volledig nieuwe, en vanaf 0 opgebouwde, OSX, hebben ze bepaalde low level functies – weliswaar verborgen, maar ze zijn er – ontsloten voor de eindgebruiker. OSX is gebouwd op een apart gelicenseerde Unix-versie van Free-BSD, dat, in tegenstelling tot wat de naam doet vermoeden, geen Open Source versie is. Door deze stap sluit OSX eigenlijk naadloos aan bij het "Operating System of Choice" aan de serverkant van het internet: Unix (daar waar ik de term Unix gebruik, kun je net zo goed Linux of een van z'n varianten lezen).
Apple heeft hiermee eigenlijk de kracht van een enorm krachtige en gebruikersvriendelijke GUI gekoppeld aan het enorm krachtige Unix operating system.

HET AFZIEN VAN DE JAREN '90

In tegenstelling tot de Microsoft-approach, is de gebruiker wat Unix betreft een noodzakelijk kwaad. Als een Unix computer bezig is met opstarten, schreeuwt het hele operating system eigenlijk al: "Als je niet weet wat je aan het doen bent, dan lazer je maar op" - er flitsen allerlei onbegrijpelijke boodschappen over het scherm en uiteindelijk staat er òf de tekst "Kernel Panic" – dan weet je dat je als systeembeheerder hebt zitten prutsen en wordt het overwerken – òf het verlossende woord "username:" met een knipperende cursor erachter.

De meeste Linux-varianten booten tegenwoordig over het algemeen direct door naar een grafische desktopomgeving (de zgn. Window-manager). Maar op servers is deze feature omwille van memory over het algemeen niet geïnstalleerd. Alle systeemhandelingen (netwerk instellingen, het gebruik van USB of andere disks of apparaten) kun je ook via de command line uitvoeren. Sterker nog: de meeste configuratie software is niet meer dan een grafisch omhulsel van een command line tool. Het gebruik van eindgebruikers-toepassingen zoals tekstverwerkers of tekenprogramma's is natuurlijk een ander verhaal, maar verder kun je de hele computer via de command line besturen.

Unix is gebouwd voor ontwikkelaars en systeem-jongens en meisjes, pas met het verschijnen van Linux (en dan met name de Ubuntu varianten) ging de community zich eigenlijk pas richten op het "normale" gebruik van zo'n systeem.

Maar het onderliggende systeem is nog altijd heel sterk op ontwikkelaars en infrastructuur mensen gericht. Je ziet Unix-omgevingen vanaf eind jaren '70 tot aan deze eeuw eigenlijk alleen maar in specialistische omgevingen gebruikt worden. Met name voor grote database toepassingen als Oracle of Progress in administratieve omgevingen en in productie of industriële omgevingen waarbij enorme hoeveelheden data verwerkt moeten worden. Dat was ook een markt waar met name HP (met oa. de HP9000 servers die op HP-UX draaiden) en IBM (met AIX machines) peperdure computers verkochten, uitgerust met hun eigen CPU's (de PA-RISC van HP bijvoorbeeld) en hun eigen smaak Unix (HP-UX, AIX en Ultrix etc).

Deze machines – vaak ook fysiek imponerend, ik heb servers beheerd van 2m hoog waarin een betonplaat was gemonteerd tegen het trillen – waren bedoeld voor het betere stampwerk. Al in de jaren '90 volledig voorzien van SAN (storage area network) voorzieningen via glasvezel, full SCSI en RAID5 en nog meer van dat soort termen waarmee je op (nerd)feestjes de blits kon maken. Tegenwoordig geen cloud zonder shared storage, maar indertijd gewoon niet aanwezig in de Windows of Novell wereld.

Maar... het was natuurlijk helemaal geweldig dat dat allemaal zomaar kon. Maar je werd steevast begroet door die "username:" prompt. En nu dan?

Inloggen met de username/password combinatie die op verpakking staat, en vervolgens staat er:

$>

op een verder volkomen leeg scherm. Goed, een ton aan ijzer onder je vingers en dat kreng vertelt je niet meer dan $>…

Het was afzien in de jaren '90…

REVOLUTIE

Toen een Finse student in 1991 besloot om zijn eigen operating system te gaan bouwen, kwam de zaak in een stroomversnelling. Het was namelijk zijn idee om de kracht van Unix op een huis-tuin-en-keuken-pc te kunnen gebruiken. Al gauw had deze student (Linus Torvalds) een hoop programmeurs om zich heen verzameld, en Linux was geboren.

Omdat Unix nu ook op "gewone" Intel x86 CPU's kon draaien, en mèt een gratis, en open sourced besturingssysteem, was het niet meer noodzakelijk voor ondernemers om in dure hardware te investeren om toch een graantje mee te kunnen pikken van dat rare, hippe, nieuwe (hoewel de oorsprong in de jaren '60 ligt) iets dat ze "internet" noemden. Het duurde dan ook niet lang voordat allerlei Linux varianten (distributies genaamd) op de markt verschenen. Niet alleen werd de populaire webserver software "Apache" gebundeld in deze distributies, maar ook mail-servers, FTP-servers, database toepassingen (bij MySQL). In de moderne versies van bijna alle distributies kun je met een druk op de knop een volledig functionele webserver installeren.

Hierdoor kreeg de hosting-markt natuurlijk een ontzettende boost. En was de server-kant van het internet niet alleen toegankelijk voor grote bedrijven, maar ineens konden ook kleine hosting providers een rol op die markt gaan spelen.

De meeste hosting providers bieden hosting aan die gebaseerd is op een Linux variant, van de instap-pakketten tot aan de gespecialiseerde VPS (Virtual Private Server) managed en unmanaged. Dit heeft eind jaren '90/begin deze eeuw mede bijgedragen aan de enorme bloei die het internet doorgemaakt heeft.

Inmiddels is Linux een volwassen operating system geworden, alle grote leveranciers van server software (zoals Oracle, SAP, Progress) ondersteunen - of hebben zelfs hun eigen distributie - Linux. HP en IBM, van oudsher de grootste spelers op de Unix markt, leveren tegenwoordig standaard een Linux versie uit met hun servers en je ziet dat HP-UX en AIX langzaam maar zeker uitgefaseerd worden. En ook in de "embedded" en PLC-markt is het Linux dat de klok slaat.

MINI COMPUTERS

Hiernaast een HP9000 K-klasse server met twee diskcabinets. Een typisch voorbeeld van de zgn. "mini computers" die in de jaren '90 van de vorige eeuw het echte "stampwerk" deden.
Wie de term "mini" verzonnen heeft, heeft deze machines duidelijk nog nooit gezien.

MARKTAANDEEL

De Gartner-group schat dat het aandeel van Linux servers op het internet inmiddels ongeveer 32% is. Microsoft heeft ongeveer 35% en de andere 33% is "onbekend". Geheid dat een groot deel daarvan ook een of andere Linux variant is.

UNIX HISTORIE IN VOGELVLUCHT

1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

1969 UNIX WORDT BIJ BELL/AT&T ONTWIKKELD VOOR MET NAME MAINFRAME SYSTEMEN

1973 EERSTE BSD VERSIE, BESCHIKBAAR BUITEN BELL/AT&T

1982 AT&T VERWEFT LICENTIE

1983 EERSTE RELEASE SYSTEM V

1987 X11 ZIET HET LICHT

1991 LINUS TORVALDS KNUTSELT LINUX

1998 ALLE GROTE SPELERS ONDERSTEUNEN LINUX

2001 APPLE OSX OP BSD UNIX

2010 GOOGLE ANDROID

OPBOUW VAN HET SYSTEEM

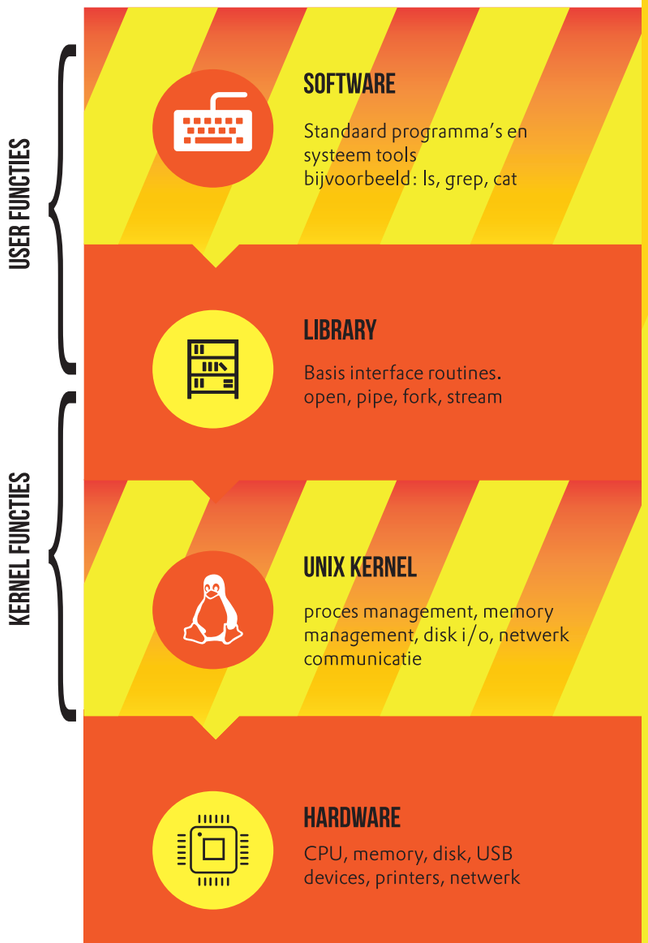
Om een beetje de structuur van het systeem te begrijpen, is het handig om even een overzichtje te schetsen hoe zo'n systeem nou daadwerkelijk opgebouwd is. In tegenstelling tot Windows-machines communiceer je in Unix altijd met behulp van bestanden.

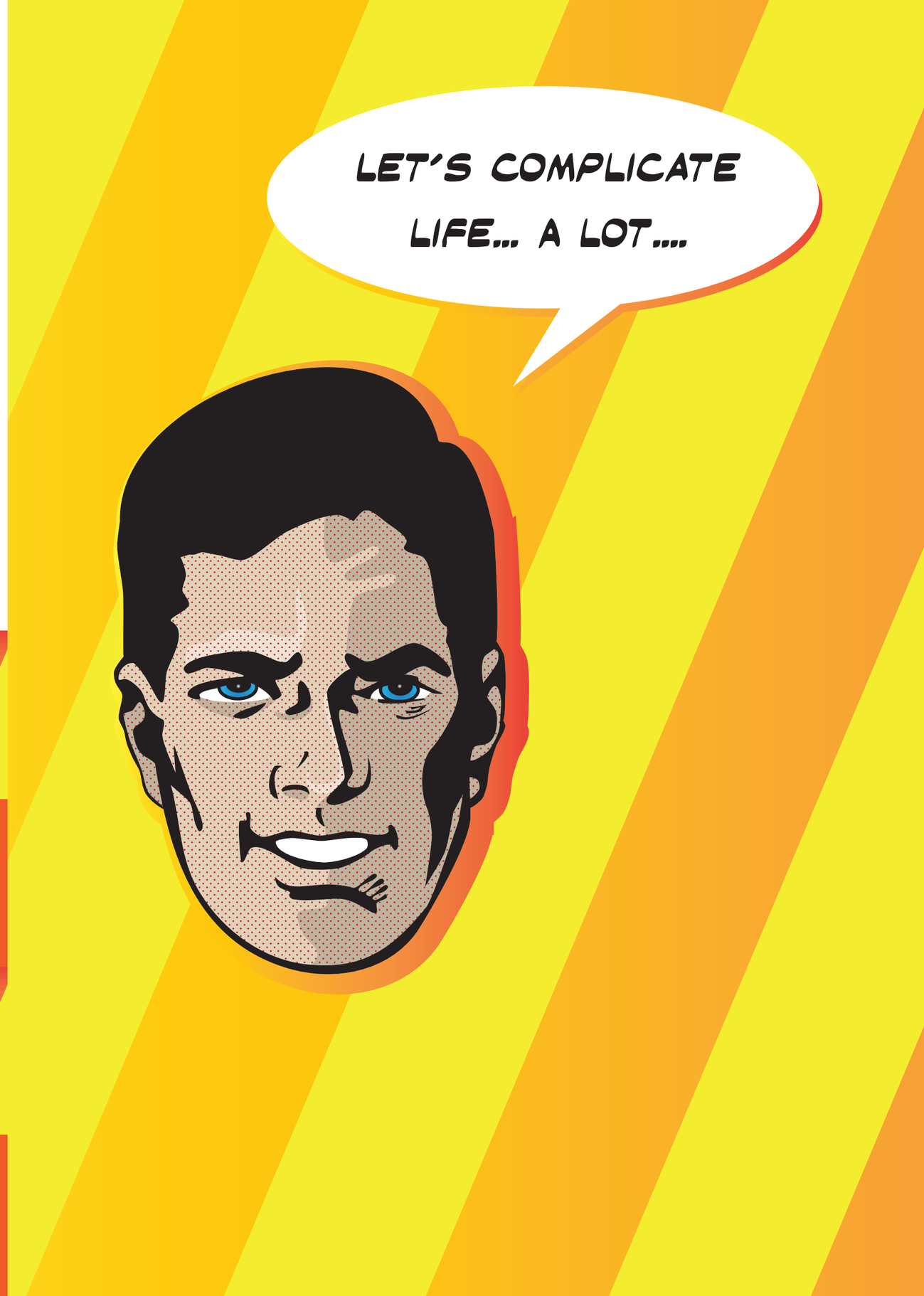
Devices (hardware) zijn op systeem niveau niet meer dan bestanden met een speciale functie. Zo is ook je beeldscherm "gewoon" een bestand en kun je met een commando uitvoer (output) naar dat bestand schrijven. Standaard wordt alle output naar het bestand dat in de software STDOUT (over het algemeen het beeldscherm van de ingelogde gebruiker) genoemd is geschreven, maar je kunt output ook "redirecten" naar een ander bestand/uitvoer medium. Dat is handig omdat je vaak de uitvoer van een commando als invoer voor een ander commando wilt gaan gebruiken. Ook kun je conditioneel werken, zodra de uitvoer van een bepaald programma "X" is, dan voer "Y" uit en anders "Z".

Dat lijkt een beetje op programmeren, en dat is het ook. Je kunt de computer bepaalde taken laten uitvoeren (dat noemen ze over het algemeen "batches") waarin je een bepaalde foutmarge opneemt. Bijvoorbeeld als de backup niet gelukt is, dan stuur een mailtje naar de systeembeheerder. We zullen straks een aantal voorbeelden behandelen en dan kun je je eigen batch schrijven.

SECURITY

De beveiliging is op een Unix systeem "ingebakken" in de kernel. Alle bestanden zijn voorzien van een zgn. ACL (Access Control List) en deze wordt op filesystem niveau door de kernel beheerd.
Hierdoor is het zo goed als onmogelijk voor virussen een Unix systeem te besmetten: heb je niet de juiste rechten, dan mag je het bestand niet wijzigen!
In werkelijkheid functioneert het iets anders, maar in grote lijnen komt het wel hierop neer. Het voert wat te ver om precies te gaan uitleggen hoe de zogenaamde inodes werken en hoe de kerneltransacties verlopen.





ZO, NU AAN HET WERK!

DE SHELL

Zodra je bent ingelogd, start een specifiek programma – een systeembeheerder kan dat voor je instellen – op waarmee je toegang krijgt tot het systeem. Een bepaald set programma's stellen je in staat om commando's op het systeem uit te voeren. Dit noemen ze de "shell"-programma's. Er bestaan diverse varianten die elk met hun eigen specifieke syntax (een soort programmeer taal) werken.

De bash wordt tegenwoordig het meeste gebruikt, daarnaast heb je nog de ksh, de sh en de csh. De sh (van shell, of Bourne-shell – ontwikkeld door Stephen Bourne) is een uitgeklede variant die erg beperkt is. De csh heeft een syntax die nogal op de programmeertaal C lijkt, nogal omslachtig in het gebruik dus. De ksh (Korn-shell) was zeer populair vanwege de uitgebreide mogelijkheden en deze is later omgeschreven naar de bash (en dat staat voor "Bourne Again Shell" – een uitbreiding op de Bourne-shell en het klinkt als "born again", welkom in Nerdistan…) met nog meer mogelijkheden. De shell stelt je dus in staat om commando's (programma's) uit te voeren. Een soort van Windows-start knop, maar dan anders.

Over het algemeen begroet zo'n shell je met een zgn. "prompt" waarin meestal je gebruikersnaam staat en de directory waarin je je bevindt. De systeembeheerder kan die prompt helemaal voor je configureren. De knipperende cursor geeft aan dat je iets kunt invoeren.

BASIS COMMANDO'S

Het eerste commando dat je over het algemeen leert is het commando 'ls' (list) en dit geeft je een overzicht van de bestanden in de actieve directory.

Als je dit nu uitvoert zullen er verdacht weinig bestanden in je (home) directory aanwezig zijn, maar daar gaan we snel verandering in brengen.

Aantekeningen

#> commando -[flag,flag] [parameter]

Toelichting: de #> representeert de aanwezige prompt, de ***<tekst tussen de haken>*** *geeft een toetsenbord-actie aan. In sommige gevallen kun je kiezen tussen een aantal "****flags****" dit wordt aangegeven door* ***-[a,b,c,d]*** *- dit betekent dat er (optioneel) een of meerdere flags gebruikt kunnen worden.*

Het is de bedoeling dat je de opdracht zo op de command prompt invoert en het system dit uit laat voeren.

Opdracht

```
#> ls <enter>

#> ls -a <enter>
```

Type nu maar eens ls –a, je zult zien dat je nu ineens meer ziet dan zonet. De bestanden die met een . beginnen (bijvoorbeeld .bash-profile) zijn "hidden files", verborgen bestanden. Dat zijn vooral configuratie files en bestanden waarvan je niet wil dat ze gemakkelijk bewerkt kunnen worden.

De "-a" in deze opdracht noemen ze een "flag". Veel commando's hebben een serie aan flags die gebruikt kunnen worden om het commando een specifieke opdracht te geven. De "-a" betekent hier dat ook verborgen bestanden getoond moeten worden. Veel flags kunnen gecombineerd worden, andere weer niet.

Aantekeningen

Opdracht

```
#> ls -ltra /usr/bin <enter>
```

De inhoud van een specifieke directory kun je opvragen door de naam van de directory op te geven achter de flag(s). Dit noemt men een argument of parameter. Als het goed is, flitst nu de inhoud van `/usr/bin` directory over het scherm.

De `-ltra` flag vertelt het `ls` commando dat het een volledige lijst moet tonen, aflopend gesorteerd op bewerkingsdatum en tevens alle verborgen bestanden en directory's moet tonen.

De standaard notatie van een Unix commando/programma is als volgt - en dit geldt voor zo'n beetje alle standaard commando's:

Opdracht

```
#> commando -[flag,flag,flag] [parameters] <enter>
```

"Da's leuk" zul je zeggen, "Moet ik die flags allemaal onthouden?". Nou... ja dus... Maar niet getreurd, zoals het een goed systeem betaamt, is er een "help functie" aanwezig. En zoals bij zoveel systemen, is ook hier de help functie geschreven door techneuten. Oftewel, veel plezier bij het uitvogelen ervan! Je kunt de help functie aanspreken door het "man" (van manual) commando uit te voeren (op voorwaarde dat de plaatselijke systeembeheerder de 'man pages' geïnstalleerd heeft).

Je ziet nu de manual page van ls op het scherm. Met de pijltjes-toetsen kun je vooruit en achteruit scrollen, met <q> beëindig je het programma (een en ander kan afhankelijk zijn van de Linux versie waar je mee werkt, `man man` zou kunnen helpen).

Opdracht

```
#> man ls <enter>
```

Opdracht

```
#> pwd <enter>

# het systeem laat je iets zien in de geest van:
/home/[username]
```

BASIS COMMANDO'S: JE WEG VINDEN

Aangezien Unix volledig file georiënteerd is, is het natuurlijk van belang dat je de juiste files weet te vinden. Er zijn een aantal commando's van belang om door je bestanden en directory's te navigeren. We hebben net al stil gestaan bij het 'ls' commando. Om te achterhalen in welke directory je momenteel bent (en soms kun je dat niet aan de prompt zien), kun je het commando '**pwd**' (print working directory) gebruiken.

[username] staat voor de gebruikersnaam waaronder je bent ingelogd. Je ziet dat Unix gebruik maakt van forward-slashes om directory's en subdirectory's aan te geven (wellicht herken je dit van de URL's op een website, dat heeft nl. dezelfde oorsprong).

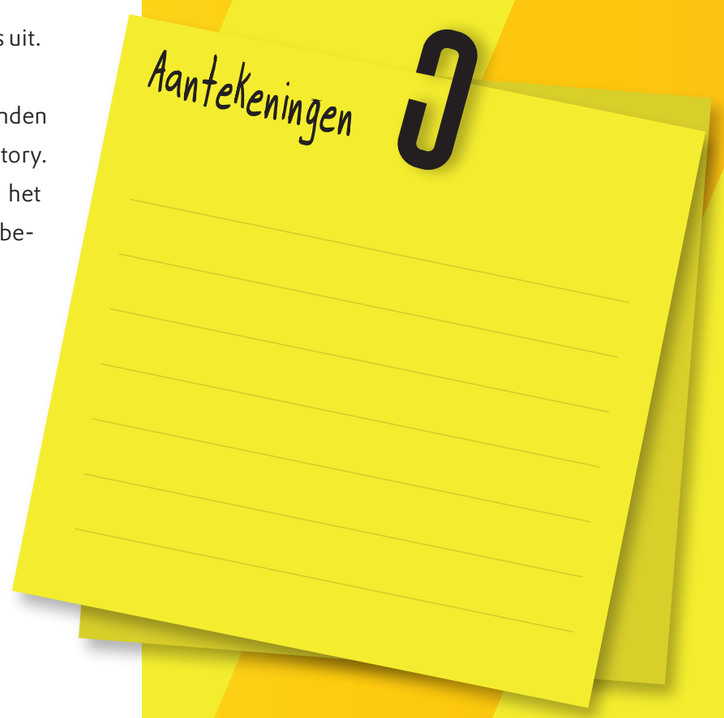
Opdracht

```
#> pwd <enter>

/home/[username]

#> cd /tmp <enter>

#> pwd <enter>

/tmp

#> ls -ltra <enter>
```

Om van directory te wisselen, kun je het commando 'cd' (change directory) gebruiken. Als je 'cd' zonder extra argument gebruikt, dan spring je vanzelf terug naar je home directory. Gebruik je 'cd' met een slash ervoor, dan staat dit voor de directory direct onder de root-directory (de basis van het systeem).

Voer de bovenstaande reeks commando's uit.

Je krijgt nu een overzicht van alle bestanden (mochten die er zijn) in de /tmp directory. Deze directory wordt overigens door het systeem gebruikt om allerlei tijdelijke bestanden weg te schrijven.



HET FILESYSTEM

Alle Unix systemen zijn volgens dezelfde structuur opgebouwd. Dit is vastgelegd in een standaard (POSIX) en als je als OS-ontwikkelaar daaraan voldoet dan mag je je OS "*nix" noemen. Dat begint met de directory (mappen) structuur. Net als alle andere (moderne) systemen werkt een Unix systeem met directory's en subdirectory's waarin de diverse bestanden zijn ondergebracht. Er zijn vanaf de root-directory tal van subdirectory's waarin diverse software is onder gebracht. Binnen deze context zijn ze niet allemaal even relevant, onderstaand overzicht toont de belangrijkste met een korte omschrijving van het soort bestanden dat je hierin kunt vinden.

Het Filesystem

```
/---
   - bin
   - dev
   - etc
   - home
       -- [user1]
       -- [user2]
       -- [etc...]
   - lib
   - opt
   - usr
       -- bin
       -- local
            -- bin
       [etc...]
   - tmp
   - var
       -- log
       -- www
       -- [etc...]
```

DIRECTORIES

/bin

In de /bin directory staan zgn "systeem commando's", zoals bijvoorbeeld het 'ls' commando, maar ook de diverse shells,

/dev

In /dev wonen de devices. Als gezegd, is Unix volledig file georienteerd, elk stuk hardware is als file in deze directory vertegenwoordigd. Zo is er bijvoorbeeld een bestand /dev/console dat de communicatie met het aangesloten beeldscherm verzorgd. Maar je vindt hier ook de ingangen voor harddisks en zelfs je terminal waar je nu in werkt staat erbij.

/etc

De /etc directory is bedoeld voor allerlei configuratie bestanden. Er zijn tal van subdirectory's voor bijvoorbeeld apache, mail en minder tot de verbeelding sprekende software (ntp, ldap, ftp).
Veruit het bekendste bestand zal /etc/hosts zijn (ook op windows voorhanden), dit is een wezenlijk onderdeel van het TCP/IP protocol en bevat bekende machine's in het netwerk met hun ip-adres.

/home

De /home directory bevat alle, de naam zegt het al, home directory's van de op het systeem geconfigureerde gebruikers - die gebruikers zitten overigens in het bestand /etc/passwd.

/lib

Deze directory bevat de zgn. shared libraries. Systeemfuncties die door elk programma gebruikt worden (bijv. voor het openen van bestanden en het schrijven naar scherm of disk). Enigszins vergelijkbaar met de DLL bestanden van Windows.

/opt, /var en /tmp

Deze directory's zijn bedoeld voor software en gegevens die in principe niks met het systeem van doen hebben. Dus hier installeer je bijvoorbeeld Oracle oid. De /var is de vergaarbak van oa. log-bestanden maar ook Apache zet hier z'n www-directory bijvoorbeeld.

/usr

/usr tenslotte bevat "bijzaak" commando's. Zoals de commandline tools die we later gaan gebruiken. Vanuit de history is er een /usr/bin directory (voor tools) en een /usr/local/bin directory (voor je eigen programma's en tools). Maar dat wordt nogal door elkaar gebruikt

DIRECTORIES & PERMISSIES

Veel van de tools/programma's die we gaan gebruiken staan in de /bin, /usr/bin en /usr/local/bin directory. Deze directory's zijn in de shell standaard opgenomen in de **PATH**-variabele.
Deze zogenaamde "omgevingsvariabele" vertelt aan het shell programma in welke directory's het moet zoeken om bepaalde programma's te vinden. Als de directory van een programma niet in de PATH-variabele staat, zul je het hele pad naar het programma moeten intypen.

Bijvoorbeeld: als /bin niet in het PATH zou staan, dan moet je de volledige directory en de naam van het programma - gescheiden door '/' intoetsen om het programma uit te voeren (dit noemen ze een pad).

Opdracht

```
#> /bin/ls <enter>
```

Als we straks zelf een script gaan schrijven en dat willen gaan uitvoeren, moet of de directory in het **PATH** worden opgenomen, of we moeten het volledige pad naar het script intoetsen.

Ik neem aan dat jullie nu al terug verlangen naar de Verkenner of de Finder en smachten naar een "dubbelklik"?

Hoe weet je nou of een directory wel of niet in het pad staat? Nou, dat kan door de naam van het programma in te toetsen op de commandline:

Opdracht

```
#> /bestaat-niet <enter>
-bash: bestaat-niet: command not found
```

De shell komt nu met de melding "command not found". Maar ja, stel het is er wel en het doet iets vreselijks met je systeem - je bent tenslotte een hacker - dan zijn de gevolgen niet te overzien natuurlijk. We kunnen beter even checken wat de inhoud van de omgevingsvariabele **PATH** is. Dat doen we als volgt:

Opdracht

```
#> echo $PATH <enter>
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

Hier zie je welke directory's er in het **PATH** zijn opgenomen. De shell doorzoekt de directory's in het **PATH** in de volgorde waarin ze zijn opgenomen, en zodra het programma gevonden is wordt dit uitgevoerd.