

# Linux samenvatting

## Historie

- Alle moderne computer systemen stammen af van UNIX
- UNIX werd in 1969 ontworpen in de labs van AT&T Bell
- De source code word voor iedereen ter beschikking gesteld
- De source code werd voortdurende aangepast en verbeterd door anderen.
- Jaren later werd UNIX commercieel verkocht, dit terwijl de helft van de source code door anderen geschreven was ondertussen.
- Hier kwam een rechtszaak van, wat uiteindelijk in 2 versies van UNIX resulteerde:
  - Official AT&T Unix
  - Free BSD Unix
- Veel bedrijven waaronder HP, Sun en IBM maakte hun eigen Unix versie
- Het resultaat was een soep van gelijkaardige commandos en dialecten
- Richard Stallman maakte hier een einde aan dmv het GNU project
- GNU = GNU IS NOT UNIX
- De bedoeling was een gratis besturingssysteem waar iedereen aan kon meewerken
- 90s: Linux Torvalds ontworpen een nieuwe kernel en samen met de GNU Tools vormde dit **LINUX**
- Meer dan 90% van alle supercomputer werkt op Linux
- De helft van alle smartphones draait op Linux
- Miljoenen home PCs draaien op Linux
- 70% van alle webserver draaien op Linux
- Veel tablets, dvd-players, wasmachines, modems, routers, etc draaien Linux
- = **Meest gebruikte OS ter wereld**

## Distros

- Meestal open source
- Collectie van software bovenop een Linux kernel
- Deze distros bevatten verschillende dingen:
  - Server software
  - System Management
  - Documentatie
  - Desktop apps in een centrale repository
- Distros zien er vaak gelijkaardig uit
- Zijn veilig
- Zijn makkelijk te gebruiken
- Hebben vaak een specifiek doel qua werking

De meest populaire distros zijn:

- **Red Hat**
- **Ubuntu**
- **Debian**
- CentOS |
- Oracle Enterprise Linux | -> Gebaseerd op Red Hat Enterprise Linux
- Oracle Scientific Linux |
- Linux Mint |
- Edubuntu | -> Gebaseerd op Ubuntu
- En anderen |

## Rechten

- FOSS = Free Open Source Software (tegenhanger van proprietary software = merk)
- Public Domain Software = Rechten opgegeven, mogen geen rechten op verleend worden
- Freeware = geen van beiden, maar proprietary zonder kosten.
- Voorbeelden van free software:
  - GIMP
  - MYSQL
  - GCC

## GNU GPL

- Copyleft principe

Iedereen die de software gebruikt moet verbetering aan de software terug delen met de community. Deze verbetering mogen niet onder eigen naam gelicenseerd of verkocht worden.

- GPLv3 software

Deze software runnen mag op vrij basis, bij het aanpassen of distribueren van de software ga je automatisch akkoord met de voorwaarden.

Als je de software lokaal aanpast en enkel daar gebruikt moet je je software niet distribueren. Bij eener welk ander gebruik zal als distributie bekeken worden moet je je aan Copy Left voorwaarden houden.

## Manual

<b>man</b> whois	manual opzoeken
<b>man</b> syslog.conf	config file manual zoeken
<b>man</b> syslog.d	daemon manual opzoeken
<b>man -k</b>	= apropos. Zoeken in man pages via string
<b>whatis</b> route	korte beschrijving van een manual
<b>whereis</b> route	locatie van de manpage
<b>man 5</b> passwd	locatie binnen een manpage meteen openen
<b>man</b> man	manual van de manual
<b>mandb</b>	update van de manuals uitvoeren

## Directories

<b>pwd</b>	huidige map tonen
<b>cd</b>	van folder veranderen
<b>cd ~ (cd)</b>	naar je homefolder gaan
<b>cd ..</b>	naar de parentmap gaan
<b>cd -</b>	naar de vorige map gaan
<b>cd /etc</b>	absoluut pad
<b>cd etc</b>	relatief pad
<b>tab</b>	tab completion
<b>ls</b>	bestanden weergeven
<b>ls -a</b>	bestanden weergeven, inclusief de verborgen bestanden
<b>ls -l</b>	long listing (=extra info zoals rechten, owner, grootte, datum)
<b>ls -lh</b>	human readable format (=grootte niet meer in bytes, maar GB, MB, KB, etc)
<b>mkdir</b>	map aanmaken
<b>mkdir -p</b>	parent mappen aanmaken
<b>rmdir</b>	map verwijderen
<b>rmdir -p</b>	onderliggende mappen ook verwijderen
<b>pushd</b>	directory op de stack zetten
<b>popd</b>	directory van de stack halen

## Files

Alles is een bestand in Linux, ook mappen (mappen kunnen dus ook een extensie hebben)

<b>file</b> file33.jpg	Geeft weer wat voor een bestand het is (jpg image file)
<b>file -s</b> /dev/sda	Wordt gebruikt bij speciale files in bv /dev en /proc
<b>touch</b> file33.jpg	Maakt een bestand aan
<b>touch -t</b> 200505052359 file33.jpg	Maakt een bestand aan en geeft meteen een aanmaakdatum mee
<b>rm</b>	Bestand wissen
<b>rm -i</b>	Bestand wissen, maar eerst om bevestiging vragen
<b>rm -rf</b>	Dient om mappen met inhoud te wissen (r=recursive, f=force)
<b>cp</b>	Bestand kopiëren
<b>cp -r</b>	Rekursief kopiëren, inclusief subdirs en inhoud dus.
<b>cp</b> file1 file2 dir1/file3 dir2/file5	Meerdere bestanden tegelijk kopiëren naar verschillende mappen
<b>cp -p</b>	Behoudt timestamps en permissies van het bronbestand
<b>mv</b>	Bestand verplaatsen (of renamen)
<b>rename</b>	Complexer -> 's/txt/png/' *.txt

## File content

<b>head</b>	eerste lijnen van een bestand opvragen
<b>tail</b>	laatste lijnen van een bestand opvragen
<b>cat</b>	inhoud van een bestand opvragen ( <b>concatenate</b> )
<b>cat "Hello" &gt; test.txt</b>	Steekt "Hello" in test.txt
<b>cat &gt; winter.txt</b>	Maakt een lege file, dan kan je pas typen (CTRL-D om te stoppen)
<b>cat &gt; winter.txt &lt;&lt; stop</b>	Maakt een lege file, dan kan je pas typen (stop typen om te stoppen)
<b>cat winter.txt &gt; cold.txt</b>	Bestand kopiëren
<b>tac</b>	Hetzelfde als cat, maar output is omgekeerd
<b>more</b>	Laat meer zien indien de inhoud te groot is voor het scherm
<b>less</b>	Laat minder zien indien de inhoud te groot is voor het scherm
<b>strings</b>	Zoekt ASCII tekst in een bestand en geeft deze weer

## File system

man hier

/

/bin

/home/serena/bin

/sbin

/lib

/lib/modules

/lib32

/lib64

/opt

/boot

/etc/init.d

/etc/X11

/etc/skel

/etc/sysconfig

/home

/root

/srv

/media

/mnt

/tmp

/dev

/dev/tty

/dev/null

/proc

/proc/interrupts

/proc/kcore

/sys

/usr

/usr/bin

/usr/include

/usr/lib

/usr/local

/usr/src

/var

/var/log/messages

/var/cache

/var/spool

/var/lib

/var/...

Manual van de hierarchy

Root folder

User binaries

Binaries voor enkel die user

System binaries, enkel voor root users

Libraries die gebruikt worden door /bin en /sbin

Kernel modules

libraries voor 32-bit architecturen (ELF files, Executable Linkable Format)

libraries voor 64-bit architecturen

Optionele software

Boot files, /boot/grub/grub.cfg definieert het boot menu

Scripts om daemons (background processen) te starten

X Windows System (of X)

Skeleton map. deze bevat de default files voor nieuwe users (bv. .bashrc)

Configuratie bestanden (booten, harddisks, firstboot, HW config, Keymapping)

Alle user mappen staan hier (\$USER)

Map van de root gebruiker

Data die gedeeld kan worden (FTP, Rsync, www)

Camera, USB-drive en CD-ROM-drive mount points

Tijdelijke mount points (bv geshared mappen tussen host en guest)

Map voor tijdelijke bestanden (~RAM), is leeg na reboot.

Device bestanden (SATA, IDE, ATAPI, USB, SCSI)

/dev/tty1 stelt uw terminal (CLI) voor. /dev/pts/1 (terminal in GUI)

Zwart gat, hier kan je bv de **stderr** naar toe sturen

Schijnbaar lege bestanden die met de kernel communiceren

BIOS interrupts

Enkel leesbaar met debugger, stelt uw RAM geheugen voor

Hot plug devices (USB, IEEE 1394 Firewire)

Unix System Resources (read only via NFS protocol)

Unix commandos

Algemene bestanden die dienen voor C

Libraries die niet voor users of scripts dienen

Map waar een admin software lokaal in kan installeren

Kernel Source Files

Centrale map voor log files

Bestand dat info bevat over de laatste gebeurtenissen

Cache Data voor apps

Printer, mail en cron spool

App toestand info

PID bestanden (Process ID), file locks

## Argumenten

- Voegt extra functionaliteit aan commandos toe
- Dit gebeurt door de Command Line Scan
- Na het scannen van de lijn knipt dit het commando in stukjes op
- Dit hele process heet **Shell Expansion**
- White spaces (spaties, tabs, etc) worden automatisch verwijderen
- Om dit te vermijden moet je enkele of dubbele quotes gebruiken
  - bv **echo Hallo      Hallo      Hallo** wordt **Hallo Hallo Hallo**
  - bv **echo "Hallo      Hallo      Hallo"** wordt **Hallo      Hallo      Hallo**
  - Alles tussen de quotes wordt als 1 argument bekeken
- Escape karakters kunnen gebruiken bij echo met de optie **-e**
  - bv **echo -e "Hallo\nHallo"** wordt **Hallo**  
**Hallo**
  - **\t -> tab**

## Externe/Interne commandos

- Externe commandos staan in /sbin of /bin
- Interne commandos maken deel van de shell zelf
- Met **type** kan je bepalen of het commando intern of extern is
  - Bv. **type cd** geeft als output -> cd is a shell builtin >>> INTERN
  - Bv. **type cat** geeft als output -> cat is /bin/cat >>> EXTERN
- Sommige commandos hebben zowel een interne als externe versie
- Met **-a** kan je kan zien of een commando zowel een interne als externe versie heeft
  - Bv. **-a echo** geeft als output -> echo is a shell builtin  
echo is /bin/echo
- Om expliciet het externe commando te gebruiken moet je het volledige pad intypen
- Met **which** kan je naar binaries zoeken in de \$PATH variabele
  - Bv. **which cp ls cd mkdir pwd** geeft als output ->
  - /bin/cp
  - /bin/ls
  - /usr/bin/which: no cd in (/usr/kerberos/sbin:/usr/kerberos/bin:...)
  - /bin/mkdir
  - /bin/pwd

## Aliassen

- Een alias aanmaken gebeurt met het **alias** commando
  - Bv. **alias ll='ls -lh --color=auto'**
  - Bv. **alias c='clear'**
- Met aliassen kan je bepaalde opties van een commando als standaard instellen
  - Bv. **alias rm='rm -i'**
  - Vanaf nu zal het **rm** command altijd vragen om bevestiging.
- Aliassen bekijken doe je ook met het **alias** commando
  - Bv. **alias c ll**
  - **alias c='clear'**
  - **alias ll='ls -lh --color=auto'**
- Een alias wissen doe je met **unalias**
  - Bv. **unalias rm**
- Als je wil zien of een commando als alias werkt of niet kan je de Shell Expansions bekijken
- Dit doe je met het commando **set -x** om het aan te zetten, **set +x** is om het uit te schakelen.

## Control Operators

- Deze dienen om meer dan 1 command op 1 regel te typen
- ; commandos scheiden
- & commandos naar de achtergrond sturen (de shell wacht dus niet)
- \$? Laatste exit code van het vorige commando
  - Bv. **rm fileblablabla**
  - **echo \$?**
  - 0 -> Gelukt
  - 1 -> Niet gelukt
- && Logische EN. Het tweede commando wordt enkel uitgevoerd als het eerste lukt
- || Logische OF. Het tweede commando wordt enkel uitgevoerd als het eerste mislukt
- && en || **If-Then-Else**
- # Commentaar, alles na dit teken wordt genegeerd door de shell
- \ Karakters **escapen** -> Het eerst volgende teken is dus een speciaal teken
- \ Kan ook een **EOL** (End-Of-Line) betekenen op het einde van een echo
  - Hierdoor kan je 1 commando over verschillende regels typen



## Shell Variabelen

- **\$** Bij een \$ teken gevolgd door een woord zal de shell naar de variabele van dat woord zoeken en het vervangen door de waarde van de variabele.
  - Bv. \$USER, \$HOSTNAME, \$UID, \$SHELL, \$HOME
  - Shell variabelen zijn hoofdlettergevoelig!!!
- Variabelen aanmaken kan je met dit commando:
  - **MyVar=555**
  - **echo \$MyVar** geeft als output -> 555
- Quotes kunnen enkel of dubbel gebruikt worden bij variabelen, maar er is een verschil!
  - Bv. **echo "\$MyVar"** geeft als output -> 555
  - Bv. **echo '\$MyVar'** geeft als output -> \$MyVar
- **set** Geeft de variabelen weer die momenteel bestaan
- **set | more** Geeft de shell functies ook weer
- **unset** Verwijderen van een variabele
- **\$PS1** Deze variabele bepaalt je shell prompt, je kan hier speciale tekens gebruiken.
  - Bv. **PS1='\u\h:\W\$'**
  - Dit geeft als prompt:
  - **user@host\$**
- **\$PATH** Deze variabele bepaalt waar de shell naar commandos gaat zoeken
- **env** Doet hetzelfde als **set**, maar exclusief de variabelen die geexporteerd zijn naar child shells
- **\$SHELL** Geeft weer welke shell gebruikt wordt
- **\$LANG** Geeft weer welke locale gebruikt wordt
- **export** Shell variabelen exporteren naar child shells, omgekeerd gaat dit niet naar de parent shell.
- Variabelen concateneren met strings kan problemen opleveren, dit los je op deze manier op:
  - prefix=Super
  - **echo Hello \${prefix}man and \${prefix}girl**
  - Geeft als output -> Hello Superman and Supergirl
- Unbound variabelen zijn variabelen die (nog) niet bestaan
  - Met het **set -o nounset** commando kan je een error laten zien bij het oproepen van een onbestaande variabele.
  - **set -o nounset of set +o nounset (aan/uit)**
  - **set -u of set+u (aan/uit)**

## Shell Embedding

Shells kunnen embedded zijn op de command line. Dit betekent simpelweg dat je een nieuwe shell kan gebruiken in een bestand commando. Dit is handig als je resultaten wil combineren 1 commando.

Bv. **echo \$var!**  
**echo \$(var1=5;echo \$var1)**  
geeft als output: 5

**De variabele var1 bestaat enkel in de subshell, de waarde wordt via een echo gedrukt, de buitenste echo drukt deze dan nog eens opnieuw.**

- Backticks kunnen ook gebruikt worden om shells te embedden, op die manier kan je bv voorkomen dat je bij het **cd** commando van directory wijzigt.

Bv. **echo `cd /etc ; ls -d \* | grep pass`**

Je kan ook meerdere shells embedden in mekaar, dit noemt dan **nested embedding**.

**Dit kan echter alleen met \$(), niet met backticks!**

**Verwar backticks trouwens niet met enkel quotes:**

**BACKTICK:** ` `

**ENKELE QUOTE:** ' '

## Shell Options

Met **echo \$-** kan je alle opties zien die momenteel actief zijn voor de shell.

Bv. **echo \$-**  
himbh  
**set -C ; set -u**  
**echo \$-**  
himuBCH

## Shell History

<b>!!</b>	Herhaal laatste commando
<b>!string</b>	Herhaal laatste commando startend met de string die je getypt hebt.
<b>history</b>	Bekijk de command history
<b>history -5</b>	Bekijk de laatste 5 commandos
<b>!5</b>	Voer commando nummer <b>5</b> uit
<b>CTRL-R</b>	Zoeken in de history
<b>\$HISTSIZE</b>	Hoeveel commandos er in de history bijgehouden worden
<b>\$HISTFILE</b>	Het bestand waar de history in opgeslaan is
<b>\$HISTFILESIZE</b>	Bepalen hoeveel commandos er bijgehouden moeten worden

- Regex in history
  - cat file1
  - Bv. !c:s/1/2
  - cat file2

## File Globbing

*	0, 1 of meer karakters	Bv.	ls File*
?	Gelijkaardig, maar enkel 1 karakter	Bv.	ls Fil?24
[]	Eender welk karakter binnen de []	Bv.	ls file[a5]
!	Logische NIET	Bv.	ls file[a5][!Z]
[a-z]	Enkel letters	Bv.	ls fil[a-z]
[0-9]	Enkel cijfers	Bv.	ls file[0-9]

File globbing kan je ook voorkomen dmv quotes of een backslash

Bv.	<b>echo *</b>	->	globbing
	<b>echo \*</b>		
	<b>echo '*'</b>		
	<b>echo "*"</b>	->	geen globbing, zoekt letterlijk op *

## Pipes and Commands



The keyboard often serves as **stdin**, whereas **stdout** and **stderr** both go to the display. This can be confusing to new Linux users because there is no obvious way to recognize **stdout** from **stderr**. Experienced users know that separating output from errors can be very useful.



>	output naar <b>stdout</b> sturen (Identiek aan <b>1&gt;</b> )
<b>set -o</b>	noclobber, hiermee voorkom je het per ongeluk wissen van de inhoud van een bestand dmv het > teken
<b>set +o</b>	Noclobber uitschakelen
>	Noclobber overrulen wanneer deze aanstaat
>>	Append -> output toevoegen aan file
2>	<b>stderr</b> bv naar /dev/null sturen
2>&1	Zowel <b>stdout</b> als <b>stderr</b> samen naar een file sturen
&>	Zelfde effect als <b>2&gt;&amp;1</b>
<	<b>stdin</b> Bv. <b>cat &lt; test.txt</b> Bv. <b>tr 'onetw' 'ONEZZ' &lt; test.txt</b>
<<	"Here document" -> Invoer bij cat stopt bij dit woord. Bv. <b>cat &gt; test.txt &lt;&lt; stop</b>
>filename	Snel leegmaken van een bestand
> filename	Snel leegmaken van een bestand wanneer noclobber actief is

## Filters

	Output doorsturen naar het volgende commando	
tee	Tussenresultaten wegschrijven naar een bestand	
grep	Lijnen filteren uit tekst	
grep -i	case insensitive	
grep -v	not-matching (NIET-functie)	
grep -A1	Neemt het resultaat en de eerst volgende lijn	
grep -B1	Neemt het resultaat en de lijn voordien	
grep -C1	Neemt het resultaat, de lijn voordien en de lijn nadien	
cut	Kolommen uit tekst nemen op basis van een delimiter	
	Bv. <b>cut -d: -f1,3</b>	
	Bv. <b>cut -d" " -f1</b>	
	Je kan ook ook een aantal karakters uit een tekst halen:	
	Bv. <b>cut -c2-7 /etc/passwd</b>	
tr	<b>translate</b> karakters	
	Bv. <b>cat tennis.txt   tr 'e' 'E'</b>	
	Bv. <b>cat tennis.txt   tr 'a-z' 'A-Z'</b>	
	Bv. <b>cat count.txt   tr '\n' ' '</b>	
	Bv. <b>cat spaces.txt   tr -s ' '</b>	-s = squeeze spaties
	Bv. <b>cat tennis.txt   tr -d e</b>	-d = delete karakter
wc	Word count.	
	Bv. <b>wc -l tennis.txt</b>	lijnen tellen
	Bv. <b>wc -w tennis.txt</b>	woorden tellen
	Bv. <b>wc -c tennis.txt</b>	karakters tellen
sort	sorteren	
	Bv. <b>sort -k1 country.txt</b>	sorteer op kolom 1
	Bv. <b>sort -n -k3 country.txt</b>	sorteer numeriek op kolom 3
uniq	duplicaten verwijderen	
	Bv. <b>sort music.txt   uniq</b>	sorteren
	Bv. <b>sort music.txt   uniq -c</b>	sorteren en aantal keren ervoor printen
comm	Inhoud van bestanden vergelijken	
	Bv. <b>comm list1.txt list2.txt</b>	
	Abba	
	Bowie	
	Cure	
	Queen	
	Sweet	
	Turner	
	Kolom1 -> Enkel list1.txt	
	Kolom2 -> Enkel list2.txt	
	Kolom3 -> Allebei	
od	text omvormen naar hexadecimale bytes	
sed	tekst aanpassen dmv regex	
	Bv. <b>echo level5   sed 's/5/42'</b>	-> level42
	Bv. <b>echo level5 level7   sed 's/level/jump/g'</b>	-> jump5 jump 7
	Bv. <b>cat tennis.txt   sed '/BE/d'</b>	-> verwijder lijnen

## Pipe Voorbeelden

```
who | wc -l  
who | cut -d' ' -f1 | sort  
grep bash /etc/passwd | cut -d: -f1
```

Geeft weer hoeveel users er ingelogd zijn  
Een gesorteerde lijst van ingelogde users  
Geeft alle bash user accounts weer

## Basic Unix Tools

```
find /etc > etcfiles.txt
```

```
find . -type f -name "*.conf"
```

```
find . -name "*.conf"
```

```
find /date -type d -name "*.bak"
```

```
find . -newer file42.txt
```

```
find /date -name "*.odf" -exec cp {} /backup/ \;
```

```
find /date -name "*.odf" -ok rm {} \;
```

```
find /data -name "*.txt"
```

```
find /data -name *.txt
```

```
locate
```

```
updatedb
```

```
date
```

```
date +%A %d-%m-%Y'
```

```
date +%s
```

```
cal
```

```
cal 2 1970
```

```
sleep
```

```
time
```

```
gzip - gunzip
```

```
zcat - zmore
```

```
bzip2 - bunzip2
```

```
bzcat - bzmor
```

```
cp -r /data/*.odf /backup/
```

Alle bestanden in /etc en de lijst in etcfiles.txt plaatsen

Alle \*.conf bestanden (geen mappen)

Alle \*.conf bestanden (incl mappen)

Alle \*.bak mappen

Alle bestanden nieuwer dan file42.txt

Alle \*.odf bestanden en kopiëren naar /backup

Alle \*.odf bestanden verwijderen na bevestiging

**Alle txt files, ook in submappen**

**Alle txt files in huidige map én alle files in .txt mappen**

Zoekt via index = sneller

Index updaten

Datum oproepen

**Saturday 17-04-2010**

**Seconden sinds 1969**

Kalender oproepen

Kalender van Februari 1970

Aantal seconden wachten

Duur van een commando testen

Zippen/unzippen met gzip

Gzip bestanden bekijken

Zippen/unzippen met bzip2

Bzip2 bestanden bekijken n

**Bij miljoenen bestanden zal dit commando niet meer uitgevoerd worden. Alle filenames worden namelijk op 1 regel geplaatst. Als dit commando hierdoor te lang wordt, zal er een error volgen. De volgende oplossing werkt wel bij miljoenen files.**

```
find /data -name "*.odf" -exec cp {} /backup/ \;
```

## Regular Expressions

BRE	Basic Regular Expressions	
ERE	Extended Regular Expressions	-E
PRCE	Perl Regular Expression	-P

<b>grep u names</b>	Alle lijnen die u bevatten
<b>grep in names</b>	Alle lijnen die 'ia' bevatten
<b>grep -E 'i a' list</b>	Alle lijnen die een i of een a bevatten in BRE moet dit als: <b>grep 'i a' list</b>
<b>grep -E 'o*' list</b>	Alle lijnen die 0, 1 of meer o's bevatten
<b>grep -E 'o+' list</b>	Alle lijnen die 1 of meer o's bevatten
<b>grep -E 'a\$ names</b>	Alle lijnen die op a eindigen
<b>grep -E ^F names</b>	Alle lijnen die met een f beginnen

Het \$ en ^ teken worden anchors genoemd in een regex

<b>grep '\bover\b' test</b>	Alle lijnen waar het woord 'over' in staat (geen spaties, letters, cijfers, punten, komma's, etc er voor of erna)
<b>grep -w over text</b>	Identiek
<b>grep -i</b>	Case insensitive
<b>grep -v</b>	Exclude
<b>grep -w</b>	Woord
<b>grep -A5</b>	5 regels erna
<b>grep -B5</b>	5 regels ervoor
<b>grep -C5</b>	5 regels ervoor en erna

**Een regex altijd quoten, op die manier voorkom je shell expansion bij gebruik van het \$ teken**

rename is een Perl script, dit commando gebruikt dus ook Perl regex

<b>rename 's/TXT/text/' *</b>	Alle TXT vervangen door text
<b>rename -n 's/TXT/txt/g' aTXT.TXT</b>	-n = laat zien wat het gaat doen ipv het effectief te doen s = switch g = global = alle occurrences
<b>rename 's/.text/.txt/i' *</b>	i = case insensitive

**Om extensies te renamen gebruik je best het \$ teken om het einde van de string aan te geven**

<b>sed</b>	stream editor met regex
<b>echo Sunday   sed 's/Sun/Mon/'</b>	output -> Monday
<b>echo Sunday   sed 's/Sun/##/'</b>	output -> SunSunday
<b>echo Sunday   sed 's_\(Sun\)_\1ny_'</b>	output -> Sunnyday
<b>echo 2014-01-01   sed 's/....-...-../YYYY-MM-DD/'</b>	output -> YYYY-MM-DD
<b>echo 2014-04-01   sed 's/\(...\)-\(.\)\/\1+\2+\3/'</b>	output -> 2014+04+01

Dit laatste (haken) noemt men **grouping**

<b>echo -e 'today\tis\twarm'</b>	output -> today is warm
<b>echo -e 'today\tis\twarm'   sed 's_\s_g'</b>	output -> today is warm
<b>cat list2   sed 's/ooo?/A/'</b>	? betekent optioneel
<b>cat list2   sed 's/o{3}/A/'</b>	Exact 3 o's
<b>cat list2   sed 's/o{2,3}/A/'</b>	Minimaal 2, maximaal 3 o's

## VI

ESC	Command Mode
a	Typen na huidig karakter
A	Typen na huidige lijn
i	Typen voor huidig karakter
I	Typen voor huidige lijn
o	Typen op een nieuwe lijn na de huidige lijn
O	Typen op een nieuwe lijn voor de huidige lijn
x	Verwijder karakter onder de cursor
X	Verwijder karakter voor de cursor
r	Vervang karakter onder de cursor
p	Plak karakter na de cursor
xp	Verwissel twee karakters
u	Undo
.	Repeat
dd	Knip de huidige lijn
yy	Kopieer de huidige lijn
p	Plak na de huidige lijn
P	Plak voor de huidige lijn
3dd	Knip 3 lijnen
4yy	Kopieer 4 lijnen
0	Spring naar het begin van de huidige lijn
^	Spring naar het begin van de huidige lijn
\$	Spring naar het einde van de huidige lijn
d0	Verwijder tot het begin van de lijn
d\$	Verwijder tot het einde van de lijn
j	Join twee lijnen
yyp	Duplicate twee lijnen
ddp	Verwissel twee lijnen
w	1 woord vooruit
b	1 woord terug
3w	3 woorden vooruit
dw	Verwijder woord
yw	Kopieer woord
5yb	Kopieer 5 woorden terug
7dw	Verwijder 7 woorden
:w	Opslaan
:w naam	Opslaan met naam
:q	Quit
:wq	Opslaan en quit
ZZ	Opslaan en quit
:q!	Geforceerde quit
:w!	Geforceerd opslaan (read-only bv)
/string	Zoeken naar string
?string	Achterwaarts zoeken naar string
n	Ga naar de volgende string die hij gevonden heeft
/^string	Voorwaarts zoeken aan het begin van de lijn
/string\$	Voorwaarts zoeken aan het einde van de lijn
/br[aeio]l	Zoeken naar bral, brel, bril en brol
/\<he\>	Zoek naar <b>he</b> ( <b>niet here of the</b> )

:4,8\$ s/foo/bar/g	Vervang foo met bar op lijn 4 tot en met 8
:1,\$ s/foo/bar/g	Vervang foo met bar op alle lijnen
:r fname	Lees bestand en plak de inhoud
:r !cmd	Voer commando uit en plak de output
"add	Verwijder de huidige lijn en stop deze in buffer a
"g7yy	Kopieer 7 lijnen in buffer g
"ap	Plak van buffer a
vi file1 file2 file3	Drie bestanden editen
:args	Laat bestanden zien en markeert het actieve bestand
:n	Volgend bestand editen
:e	Togglen tussen het huidige bestand en het vorige
:rew	Terug naar eerste bestand
:ab str long string	Wanneer je str intypt zal er long string komen te staan
:una str	Verwijder deze afkorting
:set number	Lijnnummering aanzetten
:set nonumber	Lijnnummering uitzetten
:syntax on	Syntax inkleuren aanzetten
:syntax off	Syntax inkleuren uitzetten
:set all	Alle opties bekijken
:set tabstop=8	Hiermee stel je het aantal kolommen van 1 tab in
:set tx	CR/LF einde
:set notx	Uitzetten

**Settings opslaan voor de volgende keer doe je in ~/.vimrc voor vim en in ~/.exrc voor vi**

## Scripting

- Altijd beginnen met declaratie van de shell -> **#!/bin/bash --**
- **De -- dient je script tegen root-misbruik te beveiligen**
- Commentaar schrijven doe je met het # teken -> **# Voornaam Achternaam**
- Variabelen in een script bestaan enkel in het script!
- Je kan een script ook sourcen, hiermee laadt je een script in een ander script in
  - Bv. **source ./vars**
  - Ipv een script kan dit ook een .conf file zijn, hier staan dan bv al je variabelen al in gedeclareerd en geïnitieerd.
- Een script debuggen kan door je script te runnen met het **bash** commando.
- **bash -x** gebruik je shell expansion toe te passen (= zien welk commando er uitgevoerd wordt)

## Scripting Loops

test

**test 10 -gt 55 ; echo \$?** output -> 1  
**[ 56 -gt 55 ] && echo true || echo false** output -> true

**test geeft altijd een true of false terug**



if then else

```
if [ -f isit.txt ]  
then echo isit.txt exists!  
else echo isit.txt not found!  
fi
```

for loop

```
for i in 1 2 4  
do  
    echo $i  
done
```

```
for counter in `seq 1 20`  
do  
    echo counting from a to 20, now at $counter  
    sleep 1  
done
```

```
for counter in {1...20}  
do  
    echo counting from 1 to 20, now at $counter  
    sleep 1  
done
```

```
for (( counter = 1; counter <= 20; counter++ ))  
do  
    echo counting from 1 to 20, now at $counter  
    sleep 1  
done
```

while loop

```
i=100;  
while [ $i -ge 0 ]  
do  
    echo Counting down from 100 to 0, now at $i;  
    let i--;  
done
```

until loop

```
let i=100;  
until [ $i -le 0 ]  
do  
    echo Counting down from 100 to 1, now at $i;  
    let i++;  
done
```

## Scripting Parameters

Een bash shell script kan parameters hebben. Deze worden in het script opgeroepen dmv het \$ teken.

\$0 is de naam van het script

\$1 is het eerste argument

\$2 is het tweede argument

...

Het **shift** commando kan in een loop door deze parameters lopen.

```
if [ "$#" == "0" ]
then
    echo You have to give at least one parameter.
    exit 1
fi
while (( $# ))
do
    echo You gave me $1
    shift
done
```

Met het **read** commando kan je een gebruiker iets laten intypen (zoals de Invoer klasse bij Java).

```
echo -n Enter a number:
read number
```

Een Script kan ook script options hebben, deze verschillende tov parameters.

```
while getopts ":afz" option;
do
    case $option in
        a)
            echo received -a
        ;; f)
            echo received -f
        ;; z)
            echo received -z
        ;; *)
            echo "invalid option -$OPTARG"
        ;; esac
done
```

Je geeft vervolgens bij het runnen van het script 0, 1 of meerdere opties mee.

Bv. `./test.sh -afz`

Een option kan ook een eigen parameter hebben.

```
while getopts ":af:z" option;
do
  case $option in
    a)
      echo received -a
  ;; f)
      echo received -f with $OPTARG
  ;; z)
      echo received -z
  ;; :)
      echo "option -$OPTARG needs an argument"
  ;; *)
      echo "invalid option -$OPTARG"
  ;; esac
done
```

Je geeft vervolgens bij het runnen van het script 0, 1 of meerdere opties mee, inclusief een argument indien een optie deze nodig heeft.

Bv. `./test.sh -zaf 42`

De shell zelf heeft ook opties, met het **shopt** commando kan je aan de shell vragen of deze opties geset zijn of niet.

```
shopt -q cdspell ; echo $?
```

## More Scripting

Met het **eval** commando kan je de waarde van een variabele, als variabele zelf gebruiken

```
answer=42
word=$answer
eval x=\$$word ; echo $x
```

**eval** gebruik je ook indien je een argumenten meegeeft aan een commando dat op zijn beurt in een variabele zit. Door **eval** gebruiken zal het commando lukken, anders niet.

```
lastweek='date --date="1 week ago"'
echo $lastweek          -> lukt niet
eval $lastweek          -> nu wel
Thu Mar 8 21:36:39 CET 2012
```

Door **(( ))** te gebruiken kan je numerieke expressies evalueren, oftewel berekeningen testen op true of false.

```
( 42 > 33 ) && echo true || echo false
```

Met het **let** commando kan je berekeningen maken.

```
let x="3 + 4" ; echo $x
```

Met een **case** kan je zoals in Java een switch/case (of selectieblok) maken in een script.

```
echo -n "What animal did you see ? "
read animal
case $animal in
    "lion" | "tiger")
        echo "You better start running fast!"
    ;; "cat")
    ;; "dog")
        echo "Let that mouse go..."
        echo "Don't worry, give it a cookie."
    ;;
    "chicken" | "goose" | "duck" )
        echo "Eggs for breakfast!"
    ;; "liger")
    ;;
    "bafelfish")
    ;; *)
    ;; esac
```

Je kan ook **functions** gebruiken in scripts.

```
#functie
function greetings {
    echo Hello World!
    echo and hello to $USER to!
}
#main
echo We will now call a function
greetings
echo The end
```