Basiscursus Linux

Table of Contents

Inleiding

Wat is linux?

Linux is een Unix-achtig besturingssysteem. Strikt genomen bestaat Linux alleen maar uit het hart van dit operating system (de kernel). Meestal bedoelen we met Linux de kernel en alle software die reeds geschreven werd voor die kernel.
Duizenden programmeurs werkten gedurende jaren vrijwillig aan het besturingssysteem. Tegenwoordig worden er veel van deze mensen aangeworven bij bedrijven die met Linux bezig zijn. Zowat alle sleutelfiguren die ooit vrijwillig gewerkt hebben aan Linux zijn nu in dienst van deze bedrijven. Dit was belangrijk voor de doorbraak van Linux.

Historiek (Unix)

Het ontstaan van Unix situeert zich in de jaren '70 bij AT&T Labs, waar een systeem werd geschreven voor een mainframe. AT&T voerde een vrijgevig beleid met betrekking tot de licentierechten van de UNIX broncode ten opzichte van universiteiten en onderzoeksinstellingen. Omwille van deze beschikbaarheid is UNIX opgegaan in een reeks van besturingssystemen die niet compatibel waren met elkaar. Men kent wel twee stromingen: BSD (Berkeley Software Distribution) en SysV (AT&T). Fabrikanten zoals Sun, IBM, DEC, SCO, and HP wijzigden hun Unix variant om hun product te onderscheiden van anderen. Dit leidde gedeeltelijk tot een versplintering van Unix, maar niet in die mate als meestal aangenomen wordt.
Enkele voorbeelden van Unices die gebaseerd zijn op de AT&T code: HP-UX (Hewlett Packard), SunOS / Solaris (Sun), IRIX (Silicon Graphics), Digital Unix / Tru64 (Compaq), SCO (SCO / Caldera), AIX(IBM),...
Ondertussen zijn er verschillende projecten die op de UNIX-filosofie gebaseerd zijn, maar geen AT&T broncode bevatten zoals Linux, *BSD (FreeBSD, OpenBSD, NetBSD) en GNU Hurd.

Historiek (Linux)

Linux is ontstaan toen in 1991 een Finse student, Linus Torvalds, een OS (operating system) ontwierp dat gebaseerd was op Minix. Minix was een soort baby-Unix, geschreven door Andy Tanenbaum, hoogleraar aan de Vrije Universiteit van Amsterdam. Het Minix-systeem was bedoeld om studenten met een OS vertrouwd te maken.
Linus zocht een oplossing voor een technisch probleem. De computer van zijn universiteit kon niet meer dan 16 gebruikers tegelijk aan. Wat Linus deed was van grond af aan (maar op basis van zijn kennis van Minix) een nieuw OS schrijven.
Linux bleek de uitkomst voor de computeraars die zelf wilden programmeren. Ze konden, omdat de source code vrij te verkrijgen was, zelf aanpassingen maken.
Ondertussen is Linux aanvaard bij systeembouwers zoals IBM, Dell en Compaq als alternatief OS.

Kernel

De kernel is het hart van een OS. Het is een stuk software dat de hardware beheert (oa. geheugenbeheer, processcommunicatie, processorbeheer,...).
De linuxkernel is met broncode vrij verkrijgbaar op www.kernel.org.

Distributies

Een distributie wordt gemaakt door een bedrijf of een groep personen die een installatieprogramma, de linuxkernel en bijhorende software op een informatiedrager (CD, DVD, internet) plaatst. Enkele

voorbeelden:

- **RedHat**: de meest gekende distributie. Het is Amerikaans van oorsprong, maar wordt ook vaak in Europa en Azië gebruikt
- **Mandrake**: Frans van oorsprong. Deze distributie is vooral gericht op gebruiksvriendelijkheid
- **SuSE**: een Duitse versie van linux die populair is in Europa
- **TurboLinux**: Aziatische distributie
- **Caldera**: Eigenaar van SCO, en maker van oa. DR-DOS
- **Debian**: gemaakt door vrijwilligers
- **Slackware**: gelijkaardig met Debian
- **Yellow Dog**: Linux voor de Mac

Licenties

Linux wordt beschermd door de GNU Public License (GPL). Hoofdzakelijk houdt de GPL in dat de broncode van Linux altijd vrij beschikbaar moet zijn. Iedereen kan aanpassingen maken, maar de broncode van deze aanpassingen moeten ook vrij beschikbaar blijven.
BSD (Berkeley) is een andere open source licentie. Deze stelt dat iedereen de broncode mag nemen en eventueel gebruiken in commerciële software zonder de aanpassingen terug te geven aan de "gemeenschap". Een voorbeeld hiervan is het gebruik van BSD code voor de TCP/IP implementatie in Windows.

Kenmerken van linux

Stabiliteit
Linux is veel stabieler dan Windows. Ook in Linux kan een programma vastlopen, maar het zal niet je hele besturingssysteem vellen. Er zijn veel mensen die Linux maanden en zelfs jaren draaiende houden op hun PC zonder één enkele reboot.

Prijs
Naast het feit dat je Linux gratis kan downloaden of copiëren kan je voor weinig geld een distributie aanschaffen. Je krijgt dan één of meerdere cd's boordevol software.

Open Source
Hier draait alles om bij Linux. Ben je een programmeur en wil je de software aan je noden aanpassen, geen probleem. Je hebt toegang tot de broncode. Nadien moet je natuurlijk op jouw beurt de code ter beschikking stellen voor anderen. Door het systeem van open source kunnen bugs heel snel opgespoord worden, tenslotte mag iedereen aan de software werken.

Multi-Tasking Multi-User
Linux is een multi-tasking en multi-user besturingssysteem. Terwijl je een groot document afprint, een cd schrijft en een DVD afspeelt zal Linux niet gaan vertragen. Er kunnen ook meerdere gebruikers terzelfdertijd van één computer gebruik maken. De configuratie- en persoonlijke bestanden staan per gebruiker in een eigen directory.

Platform onafhankelijk
Linux is portable. Dit houdt in dat Linux werkt op PC's, Apple Macintosh, Sun werkstations, Alpha computers,...

Scalable
Het toepassingsgebied is enorm uitgebreid. Linux draait van handhelds, mainframes tot op je PC.

Installatie

Stap 1 : Backup maken

Voor dat we ook maar iets gaan doen met de PC is het ten zeerste aangeraden om een backup te maken van alles wat je lief is. Het is niet ondenkbaar dat er iets verkeerd gaat. Denk aan de Wet van Murphy: als er bij een experiment iets fout kan gaan, dan gaat het ook fout.

Stap 2 : Windows opschonen

Veel PC's bevatten een harde schijf waarbij er maar 1 partitie op aangemaakt is die alles in beslag neemt en daar is dan nog eens windows op geïnstalleerd. Hier moeten we enkele aanpassingen doorvoeren:

- alle programma's, behalve windows zelf, uitschakelen.
- virtueel geheugen uitschakelen (deze computer / configuratiescherm / systeem / prestaties / virtueel geheugen / uitschakelen) + rebooten.
- taakplanner onderbreken.
- defragmenteren (deze computer / C aanduiden / eigenschappen / schijf opruimen (alles) / tab extra / nu defragmenteren).

Op dit moment is onze partitie goed opgekuist en staat alle data vooraan waardoor we achteraan nu plaats hebben om een andere partitie te maken. Hiervoor moeten we onze partitie waar windows op staat wel verkleinen.

Indien je een tool zoals Partition Magic hebt, kun je deze gebruiken om de huidige partitie te verkleinen. Als je dat niet hebt, kun je bijvoorbeeld qtparted gebruiken dat op onze linux CD staat.

Stap 3 : Installatie

Ik heb hier voor de knoppix cd gekozen omdat deze vanaf cd kan opgestart worden. Dit is altijd handig indien de computer in de prut loopt. Bovendien staat er een virusscanner op, want ook wel eens handig kan zijn. Verder bevat de cd ongeveer alles wat we zouden kunnen nodig hebben. Deze installatie verschilt uiteraard van computer tot computer. Het principe blijft het zelfde, de parameters kunnen veranderen. (Vb. andere grootte hard disk, qwerty i.p.v. azerty, ...)

- reboot vanaf de knoppix cd
- snuffel eens rond wat er allemaal op de cd staat, bekijk de meegeleverde software eens...
- zet het keyboard op BE
- Konsole openen
- su
- /sbin/qtparted
    - verklein /dev/hda1 (windows) naar vb. 1 GB
    - maak een extended partitie (= /dev/hda2) aan van 8 GB
    - maak in deze extended partitie een linux-swap partitie aan van 256 MB (= 2 x RAM geheugen)
    - maak in deze extended partitie ook een linux partitie aan die de rest van de 8 GB in beslag neemt
    - De rest van de schijf kan later nog voor iets anders gebruikt worden.
- /usr/local/bin/knx-hdinstall
    - install: ok
    - choose harddisk: hda
    - launch cfdisk: ok

        - afsluiten, want we hebben dit met qtparted gedaan
    - use swap partition: y
    - swap partition: hda5
    - set up hda5 as swap: y
    - root partition: hda6
    - filesystem type: reiserfs
    - create filesystem on hda6: y
    - copying files to hard disk: ok (15 minuten)
    - copying has finished: ok
    - start mail server: y
    - start ssh server: y
    - start samba server: y
    - start printing server: y
    - start kdm: y
    - name for this machine: ccmsbox
    - use DHCP broadcast: n
    - ip addresses: allemaal ok
    - password root: ccms#root
    - password knoppix: ccms#knoppix
    - install lilo in mbr: y
    - create floppy: y
    - insert floppy: ok
    - finished: ok

- rebooten vanaf harddisk + testen of windows nog werkt
- rebooten vanaf harddisk + testen of linux werkt
- keyboard + locales nog configureren
    - als knoppix inloggen + kde configureren
        - country=belgium, language=english
        - de rest = allemaal OK
    - als root inloggen + kde configureren
        - country=belgium, language=english
        - de rest = allemaal OK
    - KDE / Settings / Control Center / Regional & Accessibility / Keyboard layout --> Belgian
    - keyboard veranderen in text mode
        - vi /etc/init.d/keymap.sh
        - in loadkeys statement: .../azerty/azerty.kmap.gz zetten
        - file saven
        - /etc/init.d/keymap.sh restart
    - keymapping voor kdm (login scherm) aanpassen
        - vi /etc/X11/XF86Config-4
        - verander XkbLayout naar "be"
        - file saven
    - locales aanpassen
        - vi /etc/sysconfig/i18n
        - LANG="nl_BE@euro"
        - COUNTRY="be"
        - LANGUAGE="us"

Structuur

Unix filosofie

Unix moet je je voorstellen als een bijzonder rijk gesorteerde gereedschapskist waar alle werktuigen gespecialiseerd zijn in een bepaalde taak of handeling. Door een aantal van deze kleine tools te laten samenwerken, kunnen grote en ingewikkelde bewerkingen worden uitgevoerd. De gebruiker moet zelf voor elke taak een coherent geheel samenvoegen.

Partities

Linux kan op verschillende manieren worden geïnstalleerd. De meest gangbare manier is het gebruik van lege partities. Deze partities formateer je dan met de "installer". Een andere manier is het gebruik van een "loopback systeem". Hiermee kan je het volledige bestandssysteem wegschrijven naar een bestand (dat zich bevindt op bijvoorbeeld een windowspartitie). Het gebruik van een "loopback" systeem heeft nadelige gevolgen voor de systeemperformantie.

Swap

De swap wordt gebruikt indien het geheugen volloopt. De computer gaat dan data die niet frequent wordt gebruikt wegschrijven naar de vaste schijf. Dit systeem vertraagt de computer, maar kan voorkomen dat het RAM geheugen volloopt.
Linux vereist minstens één swap partitie. Om performantieredenen mag je een aantal swap partities spreiden over meerdere harde schrijven. Het is mogelijk om linux te gebruiken met een swapbestand (zoals in Windows), maar geen enkele linuxdistributie ondersteunt standaard deze "setup". Een swap partitie heeft ten opzichte van een swapbestand enkele voor- en nadelen. Een aparte swap partitie is een stuk sneller omdat deze gestructureerd is om als swap te dienen. Daarentegen is een swapbestand flexibel omdat de grootte en plaats kan worden aangepast zonder wijzigingen te maken aan de partities.

Bestandsstructuur

Net zoals Windows heeft Unix een hiërarchische bestandsstructuur. Het startpunt noemt men de root (wortel, voorgesteld door "/"). Hieronder bevinden zich de directories (of "mappen"). In directories kunnen bestanden of andere directories worden geplaatst.
In Linux bestaan er geen "drives" (vb. "c:"), maar er worden andere schijven bevestigd (mounting) binnen het bestandssysteem. Praktisch voorgesteld zou een subdirectory eventueel een andere schijf kunnen zijn.
Nog een verschil met DOS/Windows is dat Windows gebruik maakt van de "\" (backslash) terwijl het bij Linux/Unix een "/" (slash) voorstelt.
Er is een indeling die op zowat ieder Unix systeem terug te vinden is:

/dev: Speciale bestanden voor de benadering van hardware.
/bin: Deze directory bevat de meest essentiële hulpprogramma's.
/sbin: De meest essentiële hulpprogramma's voor de administrator.
/etc: Hier worden de configuratiegegevens bijgehouden.
/tmp: Programma's kunnen hier hun tijdelijke bestanden schrijven.
/lib: Bibliotheken voor programma's.
/boot: Plaats waar de kernel wordt opgestart.
/var: Various, meestal worden hier logbestanden in geplaatst.

/home: Directory waar gebruikersdata in terecht komt.
/usr: Plaats waar meestal programma's in terecht komen.
/proc: Verwerking van systeeminformatie. Enkel beschikbaar op Linux.
/mnt: Gangbare plaats waar bestandssystemen worden aangekoppeld.

Speciale directories

Binnen Linux bestaan er twee speciale directories: /dev en /proc.
De /dev directory dient om rechtstreeks randapparatuur aan te spreken.
De /proc directory kan worden gebruikt om configuratiegegevens uit te halen (hoeveelheid vrij geheugen, hardwareinfo, status van het systeem,...), maar ook om "at runtime" systeemparameters te wijzigen (routing aan/uitzetten, gebruik van de hardware aanpassen,...).

Virtuele consoles

De gebruiker heeft meerdere consoles ter beschikking. De toetsencombinatie CTRL-ALT-Fx (1 <= x <= 4). Je kan op deze manier meerdere keren inloggen en dus ook meerdere programma's tegelijk draaien.
In grafische mode heb je ook een gelijkaardige werking, maar daar moet je meerdere grafische omgevingen opstarten om er gebruik van te maken. Voor de grafische omgevingen is CTRL-ALT-F6 gereserveerd. Men kan met deze toetsencombinaties ook schakelen tussen tekst en grafische mode.

Gebruikers

Voordat je Linux kan gebruiken, moet je eerst inloggen als een user (gebruiker) met een paswoord. Aan de hand van de gebruikersnaam kan de computer weten welke dingen de gebruiker van het systeem mag doen.
Naast de gebruikersnaam heb je ook groups (groepen). Je kan bepaalde rechten toekennen aan een groep, die groep kan je dan aan iedereen toekennen die de rechten op het systeem nodig heeft.

Er is één gebruiker die rechten heeft op alles, deze wordt "root" genoemd.
Iedere gebruiker heeft zijn eigen home-directory. In deze directory komen alle persoonlijke gegevens terecht alsook het gebruikersprofiel van alle applicaties.

Unix bestandspermissies

Windows NT, moderne Unices en Linux ondersteunen ACL (Access Control Lists). ACL is een nieuwere vorm voor het toekennen van gebruikersrechten op bestanden. Het heeft als voordeel dat je permissies van meerdere gebruikers en groepen kan koppelen aan een bestand (het is flexibel). ACL onder Unix/Linux is een uitbreiding op de Unix bestandspermissies. Jammer genoeg worden de meeste linuxdistributies geleverd zonder ACL ondersteuning. Zelfs indien er ACL voorhanden is, kan het nog altijd nuttig zijn om kennis te hebben van de structuur van Unix toegangsrechten.
Aan een bestand zijn er drie soorten gebruikers: owner, group, other.

- **Owner**: Staat voor de gebruiker zelf. De gebruiker kan instellen welke rechten hij/zij op de bestanden heeft.
- **Group**: Je kan hiermee instellen welke rechten de gebruikers hebben die in dezelfde groep zitten als jij.
- **Other**: Hier kan men de toegangsrechten instellen van iedere andere gebruiker.

Aan ieder van deze drie soorten gebruikers kan je volgende rechten toekennen: read (lezen), write (schrijven) en execute (uitvoeren).

Daemons

Een daemon is een programma dat op de achtergrond bewerkingen uitvoert. Het zijn processen die worden gestart bij het booten en ze maken handelingen die belangrijk zijn voor de functionaliteit van het systeem. Normaal gezien hebben deze geen interactie met de gebruiker.
Enkele voorbeelden van daemons: inetd (netwerkserver), crond (het regelmatig uitvoeren van taken), syslogd (systeemlogging), httpd (webserver), ...

Systeeminitialisatie

Iedere keer een linuxsysteem start, worden er een aantal initialisatiescripts gestart. Men kan deze scripts vinden in "/etc/init.d". De scripts worden gebruikt om daemons op te starten. Er is ook een soort "batch-bestand" waar je zelf een aantal dingen kan in plaatsen die moeten gestart worden. In RedHat en aanverwante distributies is dit "/etc/rc.d/rc.local". Opstartscripts kunnen enkel gewijzigd worden door "root".
Naast de systeemscripts bestaan er ook bestanden die aangeven wat moet gedraaid worden indien een gebruiker aanlogt. In /etc/bashrc kan men bewerkingen zetten die voor ieder gebruiker geldig zijn, $HOME/.bashrc ($HOME = de homedirectory) dient om bewerkingen uit te voeren voor die bepaalde gebruiker die eigenaar is van die homedirectory. Iedere gebruiker kan voor zichzelf dit bestand aanpassen.

Bestandsnamen

Unix laat toe om bestandsnamen te gebruiken met zowat alle karakters, maar algemeen aangenomen vermijd je spaties, tabs en karakters met een speciale betekenis zoals: & ; ( ) | ? \ ' " ' [ ] { } < > $ - ! /
De lengte van een bestandsnaam hangt af van het gebruikte bestandssysteem. Zowat ieder systeem ondersteunt een bestandsnaam van 256 karakters of meer.
Het is ook van belang te weten dat bestandsnamen hoofdlettergevoelig zijn, bijvoorbeeld file en FILE wordt aanzien als een verschillend bestand.
Bestanden die beginnen met een "." worden als verborgen beschouwd.
Bij het opvragen van bestanden zijn er nog enkele "wildcards" die kunnen gebruikt worden.

- ***** Komt overeen met een willekeurig aantal opeenvolgende tekens.
- **?** Één willekeurig teken.
- **[..]** Specifieert een teken uit de door vierkante haken omsloten verzameling. Bijvoorbeeld [a-z] wil een teken uit de verzameling letter van a tot z zeggen.

Redirection & Pipes

Unix laat toe om de uitvoer van het ene commando als input te gebruiken voor het andere. Dit noemt men een "pipe".
Bijvoorbeeld: ls | more (opvragen inhoud directory, met gebruik van pagina's).
Pipes laten toe om meerdere programma's aan mekaar te koppelen (denk aan de gereedschapskist waarmee je complexere commando's kan opbouwen)
Met redirections kunnen we de uitvoer van een programma naar een bestand schrijven.
Bijvoorbeeld: ls > lijst (we schrijven de lijst van de directory in het bestand "lijst").
Er bestaat ook een ">>" teken dat bijna dezelfde functie heeft als ">". Bij ">" wordt eerst het bestand leeggemaakt, met ">>" wordt er achteraan het bestand verdergeschreven.

Environment variabelen

Je kan in de shell een aantal variabelen definiëren waar de shell of externe programma's rekening kunnen mee houden.
Bijvoorbeeld: export PATH=/bin
Je kan deze variabele gebruiken in zelf geschreven programma's of scripts. Je kan ze oproepen via $naam.
Bijvoorbeeld: echo $PATH

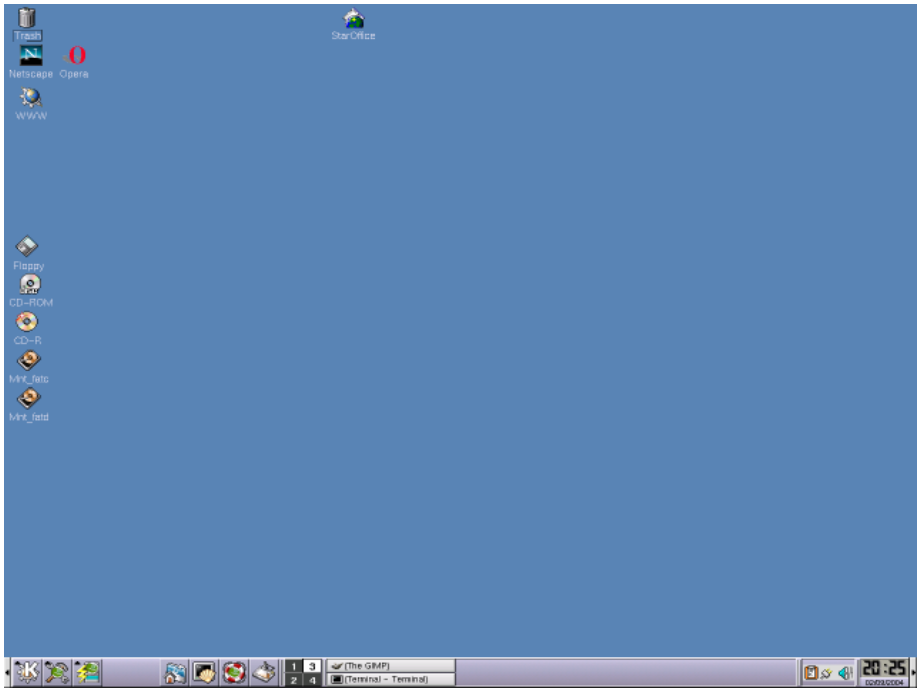
Achtergrondprocessen

Als je een commando start met een "&" teken achteraan, dan wordt dit uitgevoerd op de achtergrond. De console is dan vrij om andere commando's in te geven terwijl dat programma draait. Als men uitlogt, blijft het programma nog verder draaien.

Verkenning X

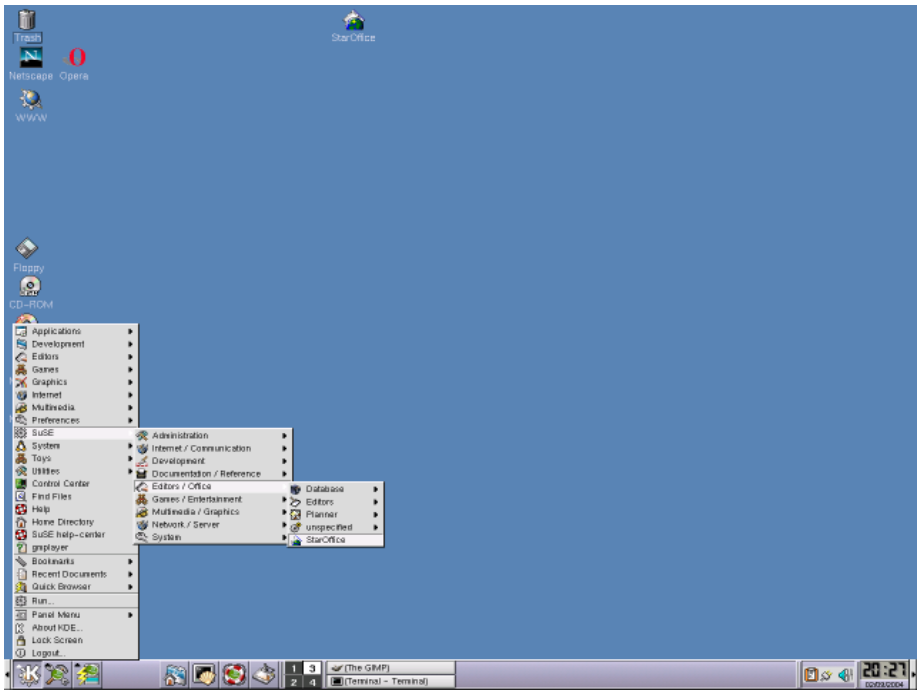
X

De grafische omgeving van Linux (X) biedt ongeveer de zelfde mogelijkheden als die van concurrent Windows. Je vindt er de taakbalk, de icoontjes om applicaties op te starten, de startknop, de verschillende toepassingen in hun window...
Er zijn in de Linux wereld 2 grote strekkingen, nl. KDE en Gnome



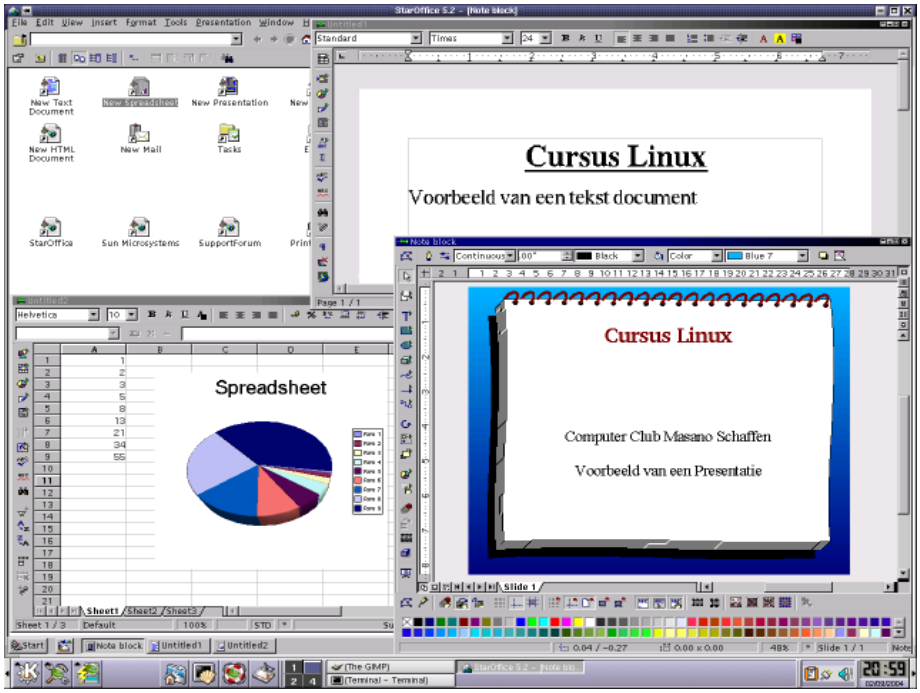
Een lege desktop

De taakbalk kan net zoals in Windows aan elke zijde van de desktop geparkeerd worden. De startknop, in KDE met de grote letter K, kan men gebruiken om applicaties te starten



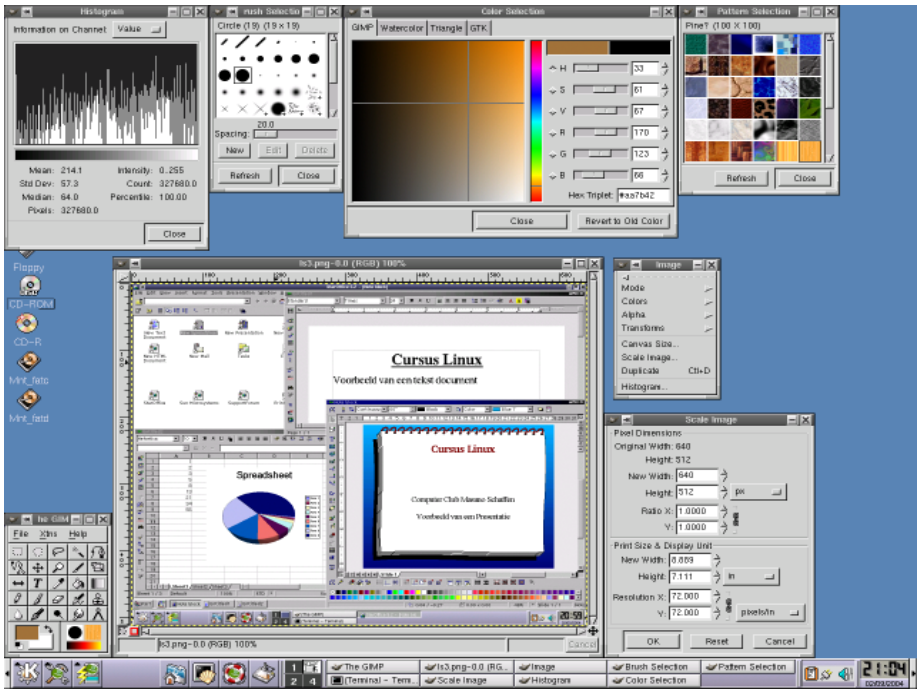
Het menu via de startknop

Onder Linux draaien office pakketten waarmee men documenten, spreadsheets, presentaties, enz kan maken. Er bestaan 2 bekende gratis offices: Star Office en Open Office.



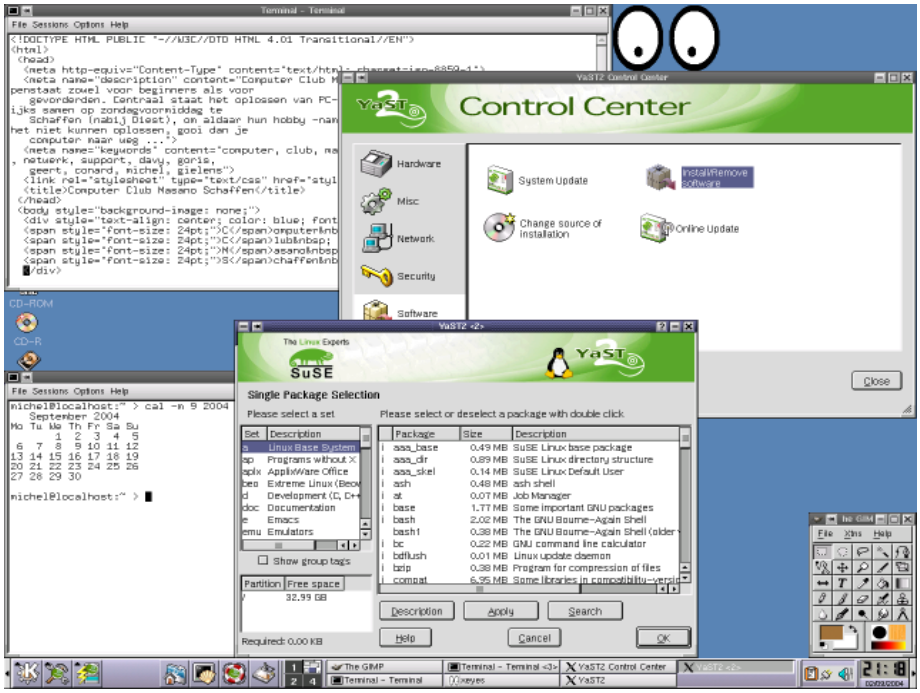
Star Office

Een professioneel grafisch pakket mag ook niet ontbreken. Met het heel uitgebreide programma 'The GIMP' kun je het onderste uit de kan halen wat betreft het verwerken van afbeeldingen.



The GIMP

Yast helpt je met de configuratie van je computer of bij het installeren van programma's. Hier zie je het samen met enkele andere toepassingen.



Yast

Heb je ook gemerkt dat je het aantal desktops zelf kunt kiezen? Met CTRL-F1 t/m CTRL-F4 kun je hier switchen naar een andere desktop. Het is ook mogelijk een applicatie naar een andere desktop te verhuizen of naar alle desktops tegelijk!

Bestandssystemen

Net zoals elk ander besturingssysteem moet voor Linux een bestandsstructuur aangebracht worden op de harde schijf.
Je hebt bij Linux minimum 2 partities nodig:
- één voor het bestandssysteem zelf (root).
- één voor het virtueel geheugen (swap).

Meestal worden er meerdere partities aangemaakt, om het systeem nog beter te beschermen tegen vastlopen of dataverlies.

Schijven

De bestandsstructuur van Unix/Linux is gemaakt dat randapparaten gekoppeld (gemount) worden aan de bestandsstructuur. Een randapparaat kan dan ook perfect aangesproken worden alsof het een bestand is. Op de achtergrond (eigenlijk in de kernel; ingebakken of als module) draait dan de driver voor dat randapparaat.

Al deze speciale bestanden (randapparaten) bevinden zich in een speciale directory /dev (devices). Harde schijven moeten aangesproken worden als /dev/hdx waarbij x een letter [a-z] is, die overeenkomt met de plaats waar de harde schijf wordt aangesloten.

Voor normale IDE-systemen, met twee IDE controllers, waaraan 2 toestellen kunnen aangesloten worden, kun je maximaal 4 harde schijven aansluiten:

IDE1 (primary)	Master	/dev/hda
	Slave	/dev/hdb
IDE2 (secundary)	Master	/dev/hdc
	Slave	/dev/hdd

Ook als er een CDROM of een ander medium is geïnstalleerd, krijgt deze hetzelfde speciale bestand toegewezen in /dev.

Beschik je over een SCSI-systeem, dan kan je deze harde schijven aankoppelen:
/dev/sda, /dev/sdb, ... of /dev/sr0, /dev/sr1, ... (tot 15)

SCSI CDROM's krijgen de *devices* /dev/scd**x** toegewezen (waarbij **x** een getal is tussen 0 en 15).
SCSI tapes worden aangeduid met /dev/st**x**.

Opgelet: Bij een CD-schrijver wordt de SCSI-emulatie gebruikt om CD's te kunnen schrijven (deze wordt automatisch aangezet bij recente distributies). In dit geval moet je het toestel gebruiken alsof het een SCSI-apparaat is (vb. /dev/scd0)

Partities

Een partitie is eigenlijk een stuk van een harde schijf waar je het bestandssysteem op plaatst.
Op een X86 systeem bestaan er 3 soorten partities:

- primaire partitie
- logische partitie (logic)
- uitgebreide partitie (extended)

Een aantal tools om partities te wijzigen: fdisk, cfdisk, sfdisk,...

Er kunnen maar **vier** *primaire partities* zijn. In sommige gevallen is dit weinig. Men heeft daarop een eenvoudige oplossing voor gevonden. De vierde partitie wordt een *uitgebreide partitie*. Daarin kunnen meerdere *logische partities* geplaatst worden.
In Linux kan je deze structuur voorstellen zoals in onderstaande tekening:

/dev/hda	**IDE1 master**
/dev/hda1	1e primaire partitie
/dev/hda2	2e primaire partitie
/dev/hda3	3e primaire partitie
/dev/hda4	1e uitgebreide partitie
/dev/hda5	1e logische partitie
/dev/hda6	2e logische partitie

Bestandssystemen

Er bestaan een hele hoop bestandssystemen die voor linux geschreven zijn. Vooral het laatste jaar zijn er enkele nieuwkomers die een geweldige "feature list" kunnen tonen.
Er zijn drie soorten bestandssystemen.

Klassieke bestandssystemen

Allereerst zijn er de **klassieke** bestandssystemen, die gebaseerd zijn op *inodes* of een *file allocation table*. Dit zijn delen op de schijf die aangeven waar op de schijf bestanden zijn weggeschreven. Het wegschrijven gebeurt zonder enige gegevensbescherming.

Voordelen:
- snel
- simpel

Nadelen:
- fragmentatie (verdeling van grote bestanden over de gehele schijf)
- geen gegevensbescherming (bij beschadiging van informatie kan er geen recuperatie van de gegevens meer gebeuren).

Voorbeelden zijn FAT (Windows), minix en ext2 (Linux).
Modernere versies hebben wel een verbetering ten opzichte van fragmentatie en gegevensbescherming.

Minix: Het allereerste bestandssysteem dat beschikbaar was voor Linux omdat het in de beginjaren afhankelijk was van een Minix installatie. Het minix-bestandssysteem komt uit Minix, geschreven door Andy Tanenbaum. De structuur is verouderd en het is afgeraden dit systeem nog te gebruiken.

FAT: Dit systeem wordt gebruikt in DOS en Windows9x/ME. Er zijn meerdere verbeteringen binnen dit systeem die compatibel zijn met elkaar (onder andere FAT-16 en FAT-32, deze laatste was een verbetering om grotere schijven te kunnen adresseren).
FAT heeft geen bescherming tegen fragmentatie of gegevensverlies. Het heeft ook geen implementatie in verband met security.
Linux heeft een zeer goede ondersteuning voor het lezen en schrijven op een FAT-partitie.

Ext2: Ext2 is al lange tijd hèt bestandssysteem van Linux. Het heeft een beveiliging tegen datacorruptie door meerdere keren het *superblock* weg te schrijven (superblock = data die verwijst naar de inhoud van de schijf). Het probeert ook fragmentatie tegen te gaan (er is geen defragmentatietool beschikbaar).

Softupdates

Het *softupdates* systeem wordt vooral gebruikt in het aangepaste UFS systeem in *BSD. Softupdates is een simpele manier om gegevensbeveiliging te bekomen.
Voor er data weggeschreven wordt, gaat het systeem dit melden in een logbestand. Als de data effectief weggeschreven is, wordt het logbestand opnieuw aangepast.
Dit komt in ruwe lijnen overeen met *journaling* (het verschil zit vooral in de technische details). Aanhangers van dit type bestandssysteem beweren dat het sneller werkt dan *journaling*. UFS wordt ondersteund in Linux, maar er is geen "native" implementatie in linux (er is wel een bestandssysteem in de maak dat TUX2 heet).

Journaled file systems

Journaled bestandssystemen zitten iets ingewikkelder in mekaar dan softupdates. De gegevensbeveiliging gaat ook veel verder, en in de meeste gevallen is de snelheid beter dan een klassiek/softupdates. Er zijn voor linux niet minder dan 4 journaled file systems. Indien er een crash is van een journaled file system, moet er bij de reconstructie van de gegevens geen *chkdsk/fsck* meer uitgevoerd worden.

NTFS: Het bestandssysteem van Windows NT/2000/XP. Dit bestandssysteem wordt niet goed ondersteund door Linux. De driver ondersteunt zonder problemen het lezen van de partities, maar het is nog niet veilig om op NTFS te schrijven.
Ext3: De gelijkenissen met ext2 zijn groot. Ext3 is eigenlijk een ext2 partitie met een bijkomende journal. Ext3 partities kunnen zonder problemen gelezen worden door systemen die enkel ext2 ondersteunen. Andersom is het ook mogelijk om ext2 partities om te zetten naar ext3 door het bijvoegen van een journal log.
Reiser:Reiser was het eerste journaled file system voor Linux. Dit systeem heeft zich sinds een lange tijd bewezen heel stabiel te zijn. Er zijn in Reiser enkele nieuwigheden geïmplementeerd die voorheen ondenkbaar waren. Reiser is enorm performant op bewerkingen met een groot aantal kleine bestanden. Er is ook een "tail" optie om meerdere bestanden in één inode te plaatsen wat diskruimte spaart.
JFS: Origineel was dit het bestandssysteem van AIX. IBM heeft besloten dit systeem te herschrijven voor Linux. JFS heeft zich al bewezen als één van de beste bestandssystemen in de Unix wereld.
XFS: Gelijkaardig met JFS heeft SGI (Silicon Graphics) zijn IRIX bestandssysteem herschreven om op Linux te gebruiken. XFS heeft een uitzonderlijke reeks van "features". Eén van de kenmerken is dat het geoptimaliseerd is om met grote bestanden te werken.

Mounting

Een bestandssysteem aankoppelen noemt men "mounten". Het vreemde bestandssysteem wordt dan opgenomen binnen de Unix-boomstructuur. Meestal wordt dit gedaan in de /mnt directory, maar als de gebruiker dit wenst, kan het ook een andere (lege) directory zijn.

Om een CD-ROM te mounten gebruiken we (in volledige notatie): `mount -t iso9660 /dev/cdrom /mnt/cdrom`
De optie "-t iso9660" specificeert dat we een bestandssysteem willen aankoppelen dat conform is met de cd-standaarden. De inhoud van de CD zal te vinden zijn in /mnt/cdrom. Zonder de optie "-t" zal mount het bestandssysteem zelf proberen te detecteren. Twee andere voorbeelden:

- Vaste schijf: `mount /dev/hda1 /home`
- Floppy: `mount -t vfat /dev/fd0 /mnt/floppy`

Indien men een wisselbare schijf aangekoppeld heeft, moet men deze eerst "unmounten" voordat ze kan verwijderd worden.

Vele fabrikanten van distributies hebben enkele gebruiksvriendelijke aanpassingen gemaakt. Zo kan men meestal een CD-ROM of floppy laden door respectievelijk `mount /mnt/cdrom` en `mount /mnt/floppy`.
In de grafische omgevingen van Linux zijn er ook gemakkelijkere manieren om te mounten, in de meeste gevallen gebeurt dit zelfs automatisch.

fstab

In dit bestand (/etc/fstab) staat een lijst welke bestandssystemen gekend zijn voor het systeem en welke automatisch worden aangekoppeld.
De bestandssystemen waarbij als *devicename* "none" staat, zijn speciale bestandssystemen. Ze worden door het systeem gegenereerd en worden niet opgeslagen op de schijf.

Bootloader

Na de installatie van Linux zal de computer bij het opstarten een scherm tonen met een keuzemenu waaruit je alle besturingssystemen die geïnstalleerd zijn op de PC kan laden. Als je bijvoorbeeld al Windows hebt geïnstalleerd, dan kan je kiezen tussen *Linux* en *Windows*.
Vanuit dit menu kan je ook verschillende versies van Linux starten.

De twee meest gekende bootloaders (op het X86 platform) die bij Linuxdistributies geleverd worden, zijn **LILO** en **GRUB**. Beide programma's kunnen zowel via configuratiebestanden als een GUI (vb. Linuxconf) worden geconfigureerd.
Deze programma's hebben tijdens het opstarten geen toegang tot de configuratiebestanden, dus is het belangrijk om na een wijziging een "update" uit te voeren van de configuratie. Bij LILO vind je het configuratiebestand in /etc/lilo.conf. Na een wijziging moet je het commando `/sbin/lilo` uitvoeren om de veranderingen effectief te maken.

Basiscommando's

Dit hoofdstuk heeft als doel de meest gebruikte Unix/Linux commando's te beschrijven. Na de theorie volgen oefeningen.

cat De inhoud van een bestand tonen.
Vb: `cat filename`

man Het opvragen van de handleiding.
Vb: `man cat` (Geeft een handleiding van cat)

cd Veranderen van de actuele directory.
Vb: `cd /usr/bin`

mkdir Het aanmaken van een nieuwe directory.
Vb: `mkdir directory`

ls Geeft een overzicht van bestanden. Bij gebruik van de optie "-l" krijg je gedetailleerde voorstelling.

cp Copiëert een bestand of directory.
Vb: `cp bronbestand doelbestand`
Je kan ook hele directories dupliceren door de optie "-r".
Vb: `cp -r brondirectory doeldirectory`

mv Verplaatst een bestand of directory. Dit commando kan ook gebruikt worden om bestanden te hernoemen.

chmod Wijzigen van bestandspermissies.

chown Wijzigt de eigenaar (gebruikersnaam en/of groep) van een bestand.

dd Converteert en copiëert bestanden. Meestal wordt dit commando gebruikt om "images" te maken van een schijf.
Vb: `dd if=/dev/cdrom of=bestandsnaam.iso`

file Bepaalt tot wat voor type een bestand behoort.
Vb: `file testbestand.jpg`

find Zoekt naar bestanden in directorytrees.
Vb: `find . -name testbestand`

locate Heeft ongeveer dezelfde functie als "find", maar in plaats van de directorytree te doorlopen werkt locate met een database die regelmatig wordt ge-update.

ln Maakt een bestand bekend onder een andere plaats/locatie.
Vb: `ln -s bestandsnaam symlinknaam`
De optie "-s" dient om een "soft symlink" te maken. Hierbij is het zichtbaar dat het bestand eigenlijk een link is.

nl Laat ingelezen regels voorafgaan door regelnummers.

pwd Laat de actuele directory zien.

rm Verwijdert bestanden en/of directorytrees.
Vb: `rm teverwijderenbestand`
Met de optie "-r" kan je ook de directories en hun inhoud verwijderen.

head Drukt de eerste regels van een bestand af.

tail Drukt de laatste regels van een bestand af. Met de optie "-100" (waarbij 100 ook een ander getal kan zijn), druk je de laatste 100 regels af.
De optie "-f" volgt continu het einde van het bestand. Zo kan je op een gemakkelijke wijze bijvoorbeeld logbestanden volgen.
Vb: `tail -f /var/log/messages`

tar Dit is een tool om backups van bestanden/directories te maken.
Om bijvoorbeeld een backup te maken van een logdirectory gebruik je: `tar -cf log.tar /var/log`
Om een .tar bestand uit te pakken gebruik je: `tar -xf log.tar`
Met de optie -v (verbose) kan je volgen welke bestanden worden gelezen of uitgepakt.

gzip & gunzip Deze tools dienen om bestanden te compresseren. Gzip gebruik je om gezipte bestanden te maken, gunzip om ze uit te pakken. Gzip kan maar één enkel bestand per keer compresseren, daarom wordt het vaak in combinatie gebruikt met tar. Dit verklaart waarom .tar.gz bestanden een populair formaat is (tar beschikt ook over de optie -z om automatisch gzip bestanden aan te maken).

touch Actualiseert de creatie-, benaderings-, en wijzigingsdatum. Het "touch" commando wordt ook vaak gebruikt om een leeg bestand aan te maken.

wc Word Count. Telt het aantal woorden, tekens en regels in bestanden.

mail Verzendt en ontvangt elektronische post. Mail wordt vooral gebruikt in scripts.

write Verzendt een boodschap naar een specifieke gebruiker.
Vb: `write root`
Je kan dan teksten typen die bij de andere gebruiker op het scherm komen. Telkens je op ENTER druk wordt die tekst verstuurd. Je sluit het programma af door op CTRL-C te drukken. Indien je geen berichten wenst te ontvangen, gebruik dan het commando mesg.

mesg Geeft/stopt de toestemming dat een andere gebruiker berichten kan sturen naar jouw terminal.
Vb: `mesg n` (geen toestemming) of `mesg y` (wel toestemming)

awk Patroonherkenningstaal.

cmp Vergelijkt twee bestanden op gelijkheden.

diff Zoekt de verschillen tussen twee bestanden.

grep Zoekt naar een bepaalde tekststring in bestanden.
Vb: `grep "string die ik zoek" mijnbestand`

uniq Verwijdert dubbele regels uit gesorteerde bestanden.

sort Sorteert de regels van samengevoegde bestanden.

vi Teksteditor.

passwd Wijzigen van het paswoord.

echo Toont een tekststring op het scherm.

lpr Print de inhoud van een bestand af. Het bestand kan drie formaten hebben: tekst, postscript of raw (taal van de te gebruiken printer).
Vb: `lpr test.ps`

kill Beëindigt een aangegeven proces.
Vb: `kill 1201` (beëindig het proces met nummer 1201)
Met de optie "-9" kan je aangeven dat het systeem het proces op hardhandige wijze stopt. Gebruik enkel de optie "-9" indien een proces op hol slaat en het niet op een andere manier kan gestopt worden.

killall Heeft dezelfde functie als kill, maar met killall wordt een procesnaam gebruikt in plaats van een procesnummer.

ps Geeft een lijst van de gestarte processen. Met het commando ps -ef krijg je een lijst van alle processen die op het systeem draaien.

top Top toont een lijst van de processen die het systeem het zwaarst belasten. Je kan er ook andere statistische gegevens in terug vinden.

df Het opvragen van een lijst van alle "gemounte" partities. Bij de uitvoer wordt ook getoond hoeveel vrije ruimte er beschikbaar is. Met de optie "-h" krijg je een duidelijkere uitvoer.

du Toont de hoeveelheid schijfruimte in gebruik voor een opgegeven directorytree.

nice Voert een proces uit met een verminderde prioriteit.

date Toont of wijzigt de tijd van het systeem.

su Dit staat voor "Switch User". Je hebt de mogelijkheid om van gebruiker te veranderen.
Vb: `su gebruiker2`
Indien geen gebruikersnaam wordt opgegeven, wordt er vanuit gegaan dat "root" wordt bedoeld. Voor het wijzigen van de gebruiker heb je wel het paswoord van die andere gebruiker nodig.

finger Opvragen van gebruikersgegevens.
Vb: `finger gebruikersnaam`

mount Het aankoppelen van externe bestandssystemen. Meestal gebruiken we de verkorte notaties.
Vb: `mount /mnt/floppy` of `mount /mnt/cdrom`

umount Het afkoppelen van externe bestandssystemen.
Vb: `umount /mnt/floppy` of `umount /mnt/cdrom`

who Toont wie op het systeem aangelogd is.

clear Maakt het scherm leeg.

Oefeningen (start)

Linux commando's leren gebruiken

Deze opgaven moeten je vertrouwd maken met de basisbegrippen van het geven van commando's.

Zorg ervoor dat een Linux venster is geactiveerd.

Probeer enkele eenvoudige commando's die geen argumenten of opties vereisen.

`date`	Toon datum en tijd.
`pwd`	Toon de actuele directory.
`ls`	Toon de lijst van bestanden in de huidige directory.
`clear`	Maak het scherm leeg.
`who`	Geef een lijst wie aangelogd is op het systeem.

Probeer nu enkele commando's die argumenten en/of opties vereisen.

`ls -a`	Toon alle bestanden in een directory.
`ls -al`	Toon een lange lijst van alle bestanden in een directory.
`cat .bashrc`	De inhoud van .bashrc wordt getoond.
`mkdir dir1`	Maak een directory.
`cd dir1`	Wijzig huidige directory.
`cd ..`	Ga een directory terug.
`rm -r dir1`	Verwijder de directory.
`cp .bachrc b1`	Copieer ".bashrc" naar "b1".
`wc b1`	Tel het aantal lijnen, woorden en tekens in "b1".
`wc -l b1`	Tel enkel het aantal lijnen in "b1".
`rm b1`	Verwijder het bestand "b1".

Probeer verschillende commando's op één regel in te typen.
`cp .bashrc testfile; cat testfile`
Copieer een bestand en druk de inhoud af.
`ls -l testfile; rm testfile; ls -l testfile`
Geef de details van het bestand, verwijder het, en toon de details opnieuw.

Speciale tekens

Deze oefeningen zullen je vertrouwd maken met het gebruik van verschillende speciale tekens in Linux.

Zorg ervoor dat je in je home directory zit.
Je kan gemakkelijk naar de home directory gaan met het commando `cd`, zonder argumenten.

Gebruik de "*" joker om al de bestanden te tonen in de huidige directory. `ls -d *` (dit commando is gelijkwaardig aan ls zonder argumenten)

Gebruik het "?" teken om alle bestanden te tonen van vier karakters lang.

`ls ????`	Toon alle bestanden van vier karakters.
`touch test`	Maak een leeg bestand "test".
`ls ????`	Toon alle bestanden van vier karakters.

Concateneer (= voeg samen) drie bestanden tot één bestand. Toon deze dan op het scherm.
`echo alpha > a ; echo beta > b ; echo gamma > c`
Maak 3 bestanden aan.
`cat a b c > newfile ; cat newfile`
Concatenatie en het tonen op scherm in één commando.

Controletoetsen voor de terminal

Deze oefening zal je vertrouwd maken met de verschillende terminal controletoetsen die Linux gebruikt.

De commandoregel wissen. Typ een reeks willekeurige tekens aan de prompt. Druk *niet* op RETURN. Wis de volledige regel door op CTRL-U te typen.

Een opdracht afbreken. Geef het commando `sleep 300` in (doe niets gedurende 300 seconden). Nadat het gestart is, kan je eens proberen een ander commando in te geven. Stop nu het sleep proces door op CTRL-C te typen. Kan je nu terug commando's intypen?

Tekens wissen. Typ een reeks van willekeurige tekens aan de prompt. Druk *niet* op RETURN. Probeer nu CTRL-h te gebruiken om tekens te wissen. Vele toetsenborden laten ook toe om "DEL" of "BACKSPACE" te gebruiken.

Autocomplete. Je kan namen van bestanden automatisch laten vervolledigen. Typ bijvoorbeeld het commando `cat new`. Druk nu op TAB. Je zal merken dat Linux de bestandsnaam vervolledigd naar **newfile**.

Wijzigen van het wachtwoord

Wijzig je oorspronkelijk wachtwoord. Maak gebruik van het `passwd` programma. Antwoord, zoals gevraagd, met je oude wachtwoord en dan je nieuwe. Bevestig het nieuwe paswoord door het nogmaals in te typen. Merk op dat er op het scherm niets verschijnt wat je typt.

Aan informatie geraken

Gebruik het `man` commando om meer te weten te komen over een aantal Linux/Unix opdrachten.

```
man ls
man cp
man rm
man man
```

Gebruik de volgende opdrachten om iets te weten te komen over de gebruikers van het systeem.

```
who
who am i
whoami
finger
finger jouw-gebruikersnaam
```

Afmelden van het systeem

Met volgende opdrachten kan je afmelden van het systeem:

```
logout
exit
```

Indien je het systeem wil uitschakelen gebruik je één van volgende opdrachten:

```
poweroff = shutdown -h now
reboot = shutdown -r now
```

Oefeningen (bestandssysteem)

ls

Gebruik de `ls` opdracht zonder argumenten om de inhoud van directories te tonen. Hoeveel bestanden worden er getoond? Probeer ook eens de `dir` opdracht.

Gebruik nu `ls` met de `-a` optie. Hoeveel bestanden zie je nu? Merk op dat de bijgekomen bestandsnamen allemaal beginnen met een punt, het zijn dus verborgen bestanden.

Recentere linuxdistributies gebruiken kleuren om het onderscheid tussen directories en bestanden te tonen. Indien je niet beschikt over een kleurenterminal kan de `-F` optie nuttig zijn. Merk op hoe de uitvoer verschilt van de uitvoer van de opdracht zonder argumenten.

Gebruik de vorm `ls -l` om een "lange" lijst van je bestanden te verkrijgen. Een voorbeeld van uitvoer met een verklaring van de getoonde informatie vind je hieronder.

```
-rw-r--r--    1 michel   users            6 Aug 10 13:57 a
-rw-r--r--    1 michel   users            6 Aug 10 13:58 b
drwxr-xr-x    2 michel   users           35 Aug 10 13:58 Desktop
-rw-r--r--    1 michel   users            7 Aug 10 13:58 g
-rw-r--r--    1 michel   users           18 Aug 10 13:58 newfile
-rw-r--r--    1 michel   users            0 Aug 10 13:58 test
    1         2   3        4             5       6      7
```

1 = toegangsmodi/permissies
2 = aantal koppelingen
3 = eigenaar
4 = groep
5 = grootte (in bytes)
6 = datum/tijdstip van de laatste wijziging
7 = naam van het bestand

Recursieve lijsten kunnen zeer nuttig zijn. Probeer volgende opdrachten. Wat merk je in de uitvoer?

```
ls -R /usr
ls -R1 /usr
```

more

Gebruik de more opdracht om een bestand te lezen: `more /etc/termcap`

Merk bij het lezen de "More" prompt op onderaan de bladzijde.

Probeer op de RETURN/ENTER toets te drukken. Wat gebeurt er?

Druk éénmaal op de SPATIEBALK. Wat gebeurt er?

Typ een letter b. Wat gebeurt er?

Gebruik de *voorwaarts zoeken* mogelijkheid om het woord clockwise te vinden met de opdracht / clockwise

Ga na welke mogelijkheden er nog zijn door hulp te vragen met het ?-teken.

More gaat verder tot het einde van het bestand of tot je q typt om het programma te verlaten. Typ een q om het programma te verlaten.

head & tail

Toon het begin van een bestand met de volgende opdracht. Hoeveel regels tel je?
```
head /etc/termcap
```

Probeer nu deze opdracht en tel het aantal regels dat nu wordt getoond.
```
head -5 /etc/termcap
```

Toon nu hetzelfde bestand met de `tail` opdracht. In hoeverre is hetgeen getoond wordt verschillend? Hoeveel regels tel je?
```
tail /etc/termcap
tail -5 /etc/termcap
```

Ga na waarom de `-f` optie in tail nuttig kan zijn. Vraag het eventueel aan de leraar indien je het niet vindt.

cat

Gebruik deze opdracht om de inhoud van een bestand te tonen. Wat gebeurt er?
```
cat /etc/termcap
```

Probeer nu volgende opdracht. Let op het verschil. Hoeveel regels bevat het bestand?
```
cat -n /etc/termcap
```

De `cat` opdracht wordt vaker voor andere doeleinden gebruikt dan het zuiver tonen van een bestand. Gebruik deze opdracht om twee bestanden te concateneren (aan elkaar te koppelen) in een derde bestand.
```
echo One > file1
echo Two > file2
```

`cat file1`	Toon inhoud file1.
`cat file2`	Toon inhoud file2.
`cat file1 file2 > newfile`	Voeg bestanden samen.
`cat newfile`	Toon inhoud nieuw bestand.

cp

Copieer een bestand in de configuratie directory naar een ander bestand in jouw home directory en lijst de inhoud om na te gaan of de opdracht gelukt is.
```
cp /etc/termcap mijnbestand
ls
```

Gebruik de copy opdracht met de "inquire" optie. Wat gebeurt er?
```
cp -i /etc/termcap mijnbestand
```

Gebruik de recursieve optie om een volledige subdirectory te copiëren naar een nieuwe subdirectory. Lijst de inhoud van beide directories om na te gaan of het heeft gewerkt. (Bij het copiëren kunnen er toegangsfouten optreden. Dit is normaal.)
```
cp -R /usr/bin mijndir
```

De copy opdracht aanvaardt jokers. Probeer volgende opdracht. Wat gebeurt er? (tip: doe een `ls *conf` in *mijndir*)
```
cp /etc/*conf mijndir
```

Indien je een bestand copieert van een andere locatie naar de huidige directory en je wil de bestandsnaam behouden, kun je "." gebruiken om de huidige directory aan te duiden.
```
cp /etc/issue .
```

mv

De `mv` opdracht kan gebruikt worden om bestanden te hernoemen. Gebruik deze opdracht en lijst dan de bestanden om te tonen dat de opdracht heeft gewerkt.
```
mv issue issue2
```

De `mv` opdracht wordt ook gebruikt om directories te hernoemen. Probeer deze opdrachten en controleer of deze gelukt zijn.
```
mv mijndir mijnmap
```

Gebruik nu de mv opdracht om een bestand te verplaatsen.
```
mv mijnbestand $HOME/mijndir
```

rm

Gebruik de `rm` opdracht om een bestand te verwijderen. Lijst de directory uit om na te gaan of het commando geslaagd is.
```
rm newfile
```

Verwijder de directory "mijnmap". Waarom gebruiken we de optie `-f`?
```
rm -Rf mijnmap
```

Verwijder nu alle bestanden in jouw homedirectory.
```
rm *
```

Voer een `ls -al` opdracht uit. Waarom is niet alles weg? Willen we alles wel weg? Welke opdracht moeten we maken om alles weg te vegen?

file

Gebruik de `file` opdracht om het type van volgend bestanden te bepalen:
```
file /bin/su
file /etc/xinetd.conf
file /bin/*
```

find

Gebruik de opdracht `find` om het bestand *xinetd.conf* te vinden. Doe dit in de */etc* directory.
```
cd /etc
find . -name xinetd.conf
```

Zoek nu naar bestanden waarvan de naam eindigt op "conf".
```
find . -name "*conf"
```

Zoek enkel directories die beginnen met de letters "rc".
```
find . -name "rc*" -type d
```

Leg uit waarom `find` nuttig kan zijn in plaats van `ls`?

ln

Creëer een koppeling tussen 2 bestanden. Typ daarna de `ls -l` opdracht om te controleren of de opdracht gelukt is. (Vergeet niet naar de home directory terug te keren.)
```
ln -s /etc/issue new.issue
```

Gebruik de `rm` opdracht om de koppeling te verwijderen. Merk op dat de originele bestanden

hierdoor niet werden beïnvloed.
```
rm new.issue
```

sort

Gebruik de `cat` opdracht om te kijken naar een ongesorteerde namenlijst. Merk op dat er verschillende kolommen zijn en dat de lijst niet alfabetisch geordend is.
```
cp /etc/passwd my.passwd
cat my.passwd
```

Gebruik nu de sort opdracht om een elementaire sortering van het bestand uit te voeren.
```
sort my.passwd
```

Voer dezelfde sortering uit, maar stuur de uitvoer naar een bestand. Gebruik daarna de `cat` opdracht om het bestand te bekijken.
```
sort my.passwd > sorted.passwd
cat sorted.passwd
```

mkdir

Wees er zeker van dat je in je home directory bent. Creëer dan een nieuwe directory met de `mkdir` opdracht.
```
cd
mkdir newdir
```

Creëer een aantal subdirectories in *newdir*.
```
mkdir newdir/sub1 newdir/sub2 newdir/sub3
```

Probeer een directory te creëren in een directory waar je geen rechten hebt. Wat gebeurt er?
```
mkdir /etc/mydir
```

Oefeningen (toegangsrechten)

Verzeker je ervan dat je in je home directory bent. Voer volgende commando's uit (toegangsfouten mag je negeren):

```
cp -r /etc ~/etc
cd etc
ls -al
```

Let op de toegangsrechten toegewezen aan elk bestand. Kun je er uit afleiden:

- Welke bestanden directories zijn?
- Of een bestand uitvoerbaar is?
- Wie schrijfrechten heeft op elk bestand?
- Wie leesrechten heeft op elk bestand?
- Wie eigenaar is van het bestand?
- Tot welke groep de eigenaar behoort?
- Welke rechten de groep heeft?
- Welke rechten de gebruikers hebben?

Gebruik de `chmod` opdracht om de rechten te wijzigen op enkele bestanden. Lijst de bestanden uit voor en na elke wijziging. Ga na hoe de toegangsrechten op de bestanden zijn aangepast.

```
ls -l yp.conf
chmod o-r yp.conf
ls -l yp.conf
chmod ugo+wx yp.conf
ls -l yp.conf
chmod g-x yp.conf
ls -l yp.conf
```

De chown opdracht dient om de eigenaar van een bestand te wijzigen. Probeer volgende opdracht.

```
chown root.root yp.conf
```

Waarom lukt dit / lukt dit niet?

VI

Inleiding

VI, de VIsual editor, is aanwezig op vrijwel ieder serieus UNIX-platform. VI werd ontworpen om de toen bestaande line-editors op te volgen, omdat terminals met beeldschermen in gebruik werden genomen, als vervanging van teletypes e.d.
VI is gebaseerd op de line-editor EX, waarvan ook vrijwel alle commando's over genomen zijn. Dit zijn de commando's die met een ':' beginnen.

Het belangrijkste wat je moet weten, is dat VI twee hoofdmodes heeft, nl. een edit mode en een command mode. Als je VI opstart, kom je automatisch in de command mode terecht. Door een tekstbewerkingscommando uit te voeren kom je in de edit mode. Om weer terug in command mode te komen volstaat het indrukken van <esc>. Als je niet meer weet in welke mode je je bevindt, druk je dus gewoon op <esc> en je zit weer in de command mode. De <esc> is hiermee een van de meest gebruikte toetsen in VI. Je kunt overigens ook zeggen dat men na het typen van ':' in de command mode in een aparte, derde mode terechtkomt, nl. in de line-edit mode.

Er bestaan veel varianten van VI, maar die ondersteunen over het algemeen alle commando's die 'standaard' VI heeft. Varianten zijn bijvoorbeeld Elvis en Vim. In deze twee varianten is het bijvoorbeeld ook mogelijk de pijltjes-toetsen te gebruiken, zelfs tijdens het editen, zodat je minder vaak tussen de edit en de command mode heen en weer hoeft te schakelen.

Starten van de VI editor

Met VI kun je nieuwe of bestaande files editeren.
vi start de editor en later (bij het saven) wordt de naam van de file opgegeven.
vi fileke.txt start met het editeren van file fileke.txt. Indien de file niet bestaat, wordt ze straks aangemaakt.
Bij bestaande files wordt op de onderste regel van het scherm de lengte van de file in aantal regels en aantal karakters gegeven, bij nieuwe files gewoon de aanduiding dat het om een nieuwe file gaat.

Cursor beweging

*n*h *n* tekens naar links.
*n*j *n* regels naar beneden.
*n*k *n* regels naar boven.
*n*l *n* tekens naar rechts. Een spatie doet hetzelfde.
0 (nul) Naar het begin van de huidige regel.
n| Naar de *n*-de kolom van de huidige regel.
n$ Naar het einde van de *n*de regel vanaf de cursor. Als *n* ontbreekt of gelijk is aan 1: naar het einde van de huidige regel.
n^ Naar het eerste teken (niet blanco) op de huidige regel.
n+ Naar het eerste teken (niet blanco) *n* regels lager. <return> in plaats van '+' kan ook.
n- Naar het eerste teken (niet blanco) *n* regels hoger.
*n*fc find. Tot en met de *n*de *c* vooruit op de huidige regel. N.B. *c* is willekeurig.
*n*Fc Find. Tot en met de *n*de *c* terug op huidige regel.
*n*tc till. Tot de *n*de *c* vooruit.
*n*Tc Till. Tot de *n*de *c* terug.
n; Herhaal de laatste `f`, `F`, `t` of `T` opdracht *n* keer.
n, Idem in de tegenovergestelde richting.
*n*w word. *n* woorden verder.

*n*W Word. *n* woorden verder.
*n*b back. *n* woorden terug. Als *n* ontbreekt of gelijk is aan 1: naar het begin van het huidige woord.
*n*B Back. *n* woorden terug.
*n*e end. Naar het einde van het *n*de woord vooruit. Als *n* ontbreekt of gelijk is aan 1: naar het einde van het huidige woord.
*n*E End. Naar het einde van het *n*de woorden vooruit.
*n*G Goto. Naar de *n*de regel. Als *n* ontbreekt: naar het einde van het file.
n) *n* zinnen verder.
n(*n* zinnen terug.
n} *n* paragrafen verder.
n{ *n* paragrafen terug.
/regexp/ Zoek vooruit naar *regexp*, een *regular expression*. Op het intypen van de eerste slash verhuist de cursor naar de onderste regel van het scherm. De tweede slash is optioneel. Dit commando moet met <return> afgesloten worden.
?*regexp*? Zoek achteruit naar *regexp*. Het tweede vraagteken is optioneel. Dit commando moet met <return> afgesloten worden.
n next. Herhaal de laatste `/` of `?` opdracht.
N Next. Idem, maar dan in de tegenovergestelde richting.
% Ga naar de tegenhanger van de eerstvolgende ronde haak, rechte haak of accolade.
`*m* Naar merkteken (*mark*) *m*. Een merkteken *m* wordt aangeduid met één letter uit de reeks `a` t/m `z`. Het commando *mm* "merkt" de huidige cursor positie met de letter *m*. Let op de backquote!
" Naar de laatste cursorpositie vóór de meest recente absolute sprong (door een `G` of `/` opdracht bijvoorbeeld).

Voor alle commando's hierboven, behalve `G`, met een *n* ervoor geldt: als *n* gelijk is aan 1 kan hij weggelaten worden.

Bladeren

n^F Forward. *n* pagina's (beeldschermen) vooruit.
n^B Backward. *n* pagina's terug.
n^D Down. *n* regels naar beneden. Zonder *n* geldt het aantal regels van de vorige Control-D opdracht. In het begin is *n* een halve pagina.
n^U Up. *n* regels naar boven. Zonder *n* geldt het aantal regels van de vorige Control-U opdracht. In het begin is *n* een halve pagina.
z+ Positioneer de huidige regel bovenaan het scherm.
z. Positioneer de huidige regel in het midden van het scherm.
z- Positioneer de huidige regel onderaan het scherm.
^L Druk het scherm opnieuw af zoals het er volgens `vi` uit moet zien.
^R Op langzame terminal s of modemverbindingen worden weggegooide regels aangegeven door een '`@`' Dit is gedaan om niet onnodig veel tijd te verliezen met het iedere keer opschuiven van aangrenzende regels. Control-R verwijdert zulke regels van het scherm.
^G Laat op de onderste regel van het scherm de status van het file zien. Er wordt aangegeven wat de naam van het file is, of er veranderingen zijn gemaakt, wat het huidige regelnummer is en op welke relatieve positie in het file de cursor op dat moment staat.

Toevoegen

*n*a... append. Voeg ... *n* maal toe achter de cursor.
*n*A... Append. Voeg ... *n* maal toe aan het einde van de regel.
*n*i... insert. Voeg ... *n* maal toe vóór de cursor.
*n*I... Insert. Voeg ... *n* maal toe aan het begin van de regel.
o... open. Ga over in input-mode op een nieuwe regel onder de huidige.

O... Open. Ga over in input-mode op een nieuwe regel boven de huidige.
n. (het commando "puntje".) Herhaal de laatste toevoeging *n* keer.

Verwijderen

*n*x Verwijder *n* tekens vanaf de cursor. Als *n* ontbreekt of gelijk is aan 1: verwijder het teken onder de cursor.
*n*X Verwijder *n* tekens vóór de cursor. De tekst erachter schuift mee naar links.
*n*d↔ delete. Verwijder alle tekens vanaf de cursor tot de positie waar het commando *n* ↔ de cursor naar toe zou brengen. Voor ↔ kan elk `vi` commando van de cursor beweging ingevuld worden. Voorbeelden: `3dw` verwijdert drie woorden en `dG` alle tekst vanaf de huidige regel tot aan het einde van het file.
*n*dd Verwijder *n* regels.
D Delete. Verwijder de rest van de huidige regel inclusief het teken onder de cursor.
n. (het commando "puntje".) Herhaal de laatste verwijdering *n* keer.

Soms zijn line-mode commando's geschikter, met name voor het verwijderen van grote stukken tekst. Zie buffer manipulatie.

Wijzigen

*n*r*c* replace. Vervang vanaf de cursor *n* tekens door hetzelfde aantal *c*'s. Dit commando moet niet met <escape> worden afgesloten. Het wordt meestal gebruikt als `rc` om het teken onder de cursor te vervangen.
*n*R... Replace. Vervang bestaande tekst *n* keer door .... Als er *m* tekens overschreven worden totdat er op <escape> wordt gedrukt, dan komen er in totaal *n* . *m* tekens voor in de plaats.
*n*s... substitute. Vervang vanaf de cursor *n* tekens door ....
*n*S... Substitute. Vervang vanaf de huidige regel *n* regels door ....
*n*c↔... change. Vervang alle tekens vanaf de cursor tot de positie waar het commando *n* ↔ de cursor naar toe zou brengen door .... Voor ↔ kan elk `vi` commando van de cursor beweging ingevuld worden. Voorbeeld: `ct.` vervangt alle tekens tot de eerstvolgende punt op de huidige regel.
*n*cc... Vervang *n* regels. Gelijk aan `n`S.
*n*C... Change. Vervang de rest van de huidige regel en de volgende *n*-1 regels door ....
*n*J Join. Plak de huidige en de volgende *n* regels aan elkaar. Merk op dat er twee regels aan elkaar worden geplakt als *n* ontbreekt of gelijk is aan 1. Dit gebeurt overigens ook als *n* gelijk is aan 2. Is dit een bug of een feature?
n. (het commando "puntje".) Herhaal de laatste wijziging *n* keer.

De laatst gemaakte verandering kan ongedaan gemaakt worden met het u (*undo*) commando. Merk op dat twee maal "undo" achter elkaar de oorspronkelijke situatie herstelt; m.a.w. het commando u maakt ook een voorafgaande "undo" ongedaan. De hoofdletter variant U doet alle veranderingen die op de huidige regel zijn gemaakt teniet, maar alleen als de cursor in de tussentijd niet van die regel af is geweest.

Gewiste regels worden tijdelijk opgeslagen in een aantal buffers en kunnen daaruit opgeroepen worden. Zie Copiëren en verplaatsen voor een uitgebreide uitleg.

Tekst in de edit-buffer kan ook rechtstreeks met andere UNIX programma's gewijzigd worden. Het edit-mode commando `n!`↔*cmd* geeft alle tekens vanaf de cursor tot de positie waar het commando *n*↔ de cursor naar toe zou brengen als stdin door aan het UNIX commando *cmd* en zet vervolgens de output daarvan op dezelfde plaats in de edit-buffer terug. Hierbij kan gedacht worden aan filters als fmt (format) en sort. Het edit-mode commando `!}fmt` formatteert bijvoorbeeld één alinea - d.w.z. de tekst tot de eerstvolgende lege regel - door de regels zo af te breken dat ze allemaal ongeveer 72 tekens lang zijn, overigens zonder te proberen een rechte rechter kantlijn te maken; de

onderlinge spatiëring van woorden blijft ongewijzigd.

De vorm `n!!cmd` geeft op dezelfde manier *n* regels vanaf de huidige door aan *cmd*.

Tenslotte is er nog het exotische commando '~' (tilde), dat het teken onder de cursor vertaalt van hoofdletter naar kleine letter of omgekeerd, al naar gelang de situatie. De cursor wordt bovendien naar het volgende teken op die regel gezet.

Copiëren en verplaatsen

*n*y↔ yank. Plaats alle tekens vanaf de cursor tot de positie waar het commando *n* ↔ de cursor naar toe zou brengen in de undo-buffer. Voor ↔ kan elk `vi` commando van de cursor beweging ingevuld worden.

*n*yy Plaats *n* regels in de undo-buffer.

*n*Y Yank. Equivalent met `yy`. Dit is een inconsistentie van `vi`; gezien de werking van `C` en `D` zou `Y` eigenlijk gelijk moeten zijn aan `y$`.

*n*p put. Zet de inhoud van de undo-buffer n keer achter de cursor. Als de inhoud uit één of meerdere regels bestaat, heeft *n* geen effect en komt de tekst onder de huidige regel terecht.

*n*P Put. Zet de inhoud van de undo-buffer *n* keer voor de cursor. Als de inhoud uit één of meerdere regels bestaat, heeft *n* geen effect en komt de tekst boven de huidige regel terecht.

n. Herhaal het laatste put commando *n* keer. Hier is echter iets speciaals mee aan de hand. Als het laatste put commando van de vorm `"bp` was, met *b* één van de buffers `1` t/m `9`, dan voert de punt operator een "put" uit met de inhoud van de *volgende* genummerde buffer. De commandoreeks `"1p..` (bestaande uit vijf tekens) zet dus de laatste *drie* "line deletes" terug.

Willekeurige stukken tekst kunnen heel eenvoudig naar een buffer worden gekopieerd door aan het einde een *mark* te zetten (zie het `m` commando in cursor beweging voor details) en vervolgens aan het begin van het gewenste gedeelte `y`*m* (let op de backquote), waarbij *m* de naam van het merkteken is, in te typen, eventueel vooraf gegaan door een bufferspecificatie.

Buffer manipulatie

:q quit. Stop `vi`. Dit kan alleen als er nog geen veranderingen zijn gemaakt of als de buffer net is weggeschreven.

:q! Quit. Stop `vi` zonder de wijzigingen definitief te maken. Eventuele veranderingen zijn hierna voorgoed verdwenen. Gevaarlijk dus, maar soms noodzakelijk.

:w [*file*] write. Schijf de edit-buffer weg naar het huidige file. Er mag ook een ander *file* worden opgegeven.

:w! [*file*] Write. Schrijf de edit-buffer weg naar het huidige file of naar het opgegeven *file*. Als het al bestaat wordt het overschreven.

:w >> [*file*] Write. Voeg de edit-buffer toe aan het opgegeven *file*. Als *file* nog niet bestaat, geeft `vi` een foutmelding tenzij de optie `writeany` is gezet. In dat geval en ook als de vorm `:w! >> file` wordt gebruikt, wordt er wel automatisch een nieuw file gecreëerd, mocht dat nodig zijn.

:x [*file*] exit. Schrijf de edit-buffer weg als er veranderingen zijn gemaakt en stop daarna `vi`. Als het wegschrijven niet lukt (door diskruimtegebrek bijvoorbeeld) wordt de editor *niet* beëindigd. In zo'n geval kan met het commando `:x /tmp/myfile` tijdelijk de gewijzigde versie veilig worden gesteld, totdat er weer ruimte op de disk is.

:x! [*file*] eXit. Gelijk aan `:x` maar een bestaand *file* wordt nu overschreven.

:e [*file*] edit. Om een ander *file* te editen zonder `vi` te stoppen en zonder de "yank buffers" (behalve de undo-buffer) te verliezen. Op deze manier kunnen via de buffers stukken tekst van het ene file naar een ander file worden gekopieerd. `Vi` geeft een foutmelding als er in de huidige edit-buffer veranderingen zijn gemaakt. Deze moet dan eerst weggeschreven worden, met `:w` bijvoorbeeld. Als *file* ontbreekt leest `vi` het huidige file opnieuw in.

:e! [*file*] Edit. Hetzelfde als `:e` zonder eventuele veranderingen definitief te hoeven maken. Alle wijzigingen sinds het laatste "write" commando gaan zo verloren! Als *file* wordt weggelaten leest `vi` het huidige file opnieuw in.

:n [*file*] next. Edit het volgende file uit de lijst van files die bij het aanroepen van `vi` waren opgegeven, of een willekeurig ander *file*. `Vi` geeft een foutmelding als er in de huidige edit-buffer veranderingen zijn gemaakt.

:n! [*file*] Next. Hetzelfde als `:n` zonder eventuele veranderingen definitief te hoeven maken. Alle wijzigingen sinds het laatste "write" commando gaan zo verloren!

Wijzigen en verwijderen

:*x*,*y*d [*b*] Delete. Wis regel *x* t/m *y* en plaatst ze in buffer *b*. Als er echter geen buffer is opgegeven, kan zo'n wijziging wel ongedaan gemaakt worden met "undo", maar *niet* met put ergens anders terug worden gezet!

:*x*,*y*s/*str1*/*str2*/[gc] Substitute. Vervang op regel *x* t/m regel *y* alle (met `g` van global) of alleen de eerste (zonder `g`) string *str1* (een *regular expression*) door *str2*. Als er geen regelaanduiding *x*,*y* is opgegeven, dan werkt `:s` alleen op de huidige regel. Met de `c` (check) optie wordt voor elke *str1* gevraagd of hij veranderd moet worden, door één voor één de bewuste regels te geven met "pijltjes" wijzend naar *str1*. Typ `y` gevolgd door <return> om te veranderen. Elke andere reactie laat *str1* ongemoeid. `^C` breekt de "substitute + check" operatie voortijdig af. Een '`&`' in *str2* staat voor de gevonden *str1* en een '`%`' voor de laatstgebruikte substitutiestring *str2*. Heel handig zijn de speciale operatoren `\U` (upper case) en `\l` (lower case), die de tekst erachter naar hoofdletters c.q. kleine letters vertalen, dus `:s/./\l&/g` vertaalt op de huidige regel elk teken (.) naar zijn kleine letter (`\l`) equivalent (`&`).

:*x*,*y*g/*str*/*cmd* Global. Voer het `ex` commando *cmd* uit op elke regel waarop *str* (een *regular expression*) voorkomt. Als er geen regelaanduiding *x*,*y* is opgegeven, dan werkt `:g` op de gehele edit-buffer. Voor *cmd* kan bijvoorbeeld `d` of `p` worden gekozen, zodat elke regel waarop *str* voorkomt gewist c.q. afgedrukt wordt. Heel aardig is `:g/.*/m0`, waarbij alle regels (de reguliere uitdrukking '`.*`' staat voor elke willekeurige tekenreeks, inclusief de lege string) naar het begin van het file verplaatst worden (d.m.v. het `ex` commando `m0`, "move after line 0"). Het effect is dat de volgorde van de regels in het file wordt omgekeerd. Let op dat met "undo" (`u`) naderhand alleen het *totale* "global" commando ongedaan gemaakt kan worden.

:*x*,*y*v/*str*/*cmd* inVerse (vergelijk met de `-v` optie van `grep`). Als `:g`, maar dan voor alle regels die niet aan de voorwaarde voldoen.

Ex en shellcommando's

:sh Start een subshell op. Deze kan op de gebruikelijke manieren worden beëindigd.

:!*cmd* Voer het commando *cmd* uit in een subshell. In *cmd* mag een '`%`' staan, waarvoor door `vi` de huidige filenaam wordt ingevuld. Op deze manier kan bijvoorbeeld vanuit de editor de syntax van een C programma gecheckt worden (met `lint`) `:!lint %` De output komt op de terminal terecht.

:!! Herhaal het laatste subshell commando.

:*x*,*y*w !*cmd* Geef de regels *x* t/m *y* als stdin aan het commando *cmd*. De edit-buffer verandert niet. Let op de spatie tussen de `w` en het uitroepteken. Zonder deze spatie is dit het `:w!` commando, die (een deel van) de edit-buffer in een file wegschrijft!

:*x*r *file* read. Voeg de inhoud van het opgegeven *file* onder regel *x* (als *x* ontbreekt: onder de huidige regel) toe aan de tekst. In *file* mogen *wildcards* ('`*`' en '`?`' met dezelfde betekenis als in de shell) voorkomen, maar de expansie moet wel precies één filenaam opleveren.

:*x*r!*cmd* Read. Voeg de uitvoer van het commando *cmd* onder regel *x* (of de huidige regel) toe aan de tekst.

(http://www.eng.hawaii.edu/Tutor/vi.html)

Regular Expressions

Een RegEx is in simpele woorden: een opeenvolging van tekens die een bepaalde groep tekens voorstelt die wordt gebruikt bij 't zoeken en/of vervangen van tekst. Deze tekst kan natuurlijk ook namen van files of directories zijn. In heel veel Unix / Linux programma's zijn deze RegEx ingebakken, vb. `vi`, `grep`, `awk`, `sed`,...

Een RegEx is gewoon een serie tekens waarvan de meeste tekens voor zichzelf staan. De RegEx *boek* komt dan overeen met *boek*, *boek*enkast, ge*boek*t, draai*boek*. Sommige tekens hebben echter een specifieke functie:

Symbool	**Betekenis**
.	Komt overeen met alle tekens, behalve het symbool voor nieuwe regel.
^	Komt overeen met een item aan het begin van de regel.
$	Komt overeen met een item aan het einde van de regel.
\<	Komt overeen met een item aan het begin van een woord.
\>	Komt overeen met een item aan het einde van een woord.
*	Komt overeen met 0 of meer keren het voorgaande item.
+	Komt overeen met 1 of meer keren het voorgaande item.
?	Komt overeen met 0 of 1 keer het voorgaande item.
[ ]	Herkent de letters die tussen de haakjes vermeld staan. Met het koppelteken kan een bereik opgegeven worden, vb. [a-z].
[^]	Komt overeen met alle niet tussen de haakjes staande letters. (Tegenovergestelde van vorige).
\	Het volgende teken wordt letterlijk overgenomen en dus ontdaan van een speciale functie.
\(\)	Slaat het patroon tussen \(en \) op. Dit patroon (maximaal 9 stuks) wordt vervangen door \1 t.e.m. \9.
|	Komt overeen met de vorige of volgende RegEx. Dit is dus een OR functie.
( )	Herkent de groep reguliere expressies die tussen de haakjes staan.
\{i\}	Het voorgaande item moet exact i keren voorkomen.
\{i,\}	Het voorgaande item moet minstens i keren voorkomen.
\{i,j\}	Het voorgaande item moet tussen de i en de j keren voorkomen. (i en j inbegrepen)

Voorgaande symbolen kan men beginnen combineren. ^boek zoekt het woord boek aan het begin van een regel, boek$ zoekt het woord boek aan het einde van de regel, ^boek$ zoekt de regels met enkel het woord boek.

In de `vi` editor kan men bijvoorbeeld naar teksten gaan zoeken:

`^bloem`	Zoekt naar het woord bloem aan het begin van een regel.
`^[Bb]loem`	Zoekt naar het woord Bloem of bloem aan het begin van een regel.
`\.$`	Zoekt naar een punt op het einde van een regel.
`[0-9]`	Zoekt alle cijfers.
`[A-Z]`	Zoekt alle hoofdletters.
`[a-zA-Z]`	Zoekt alle kleine en hoofdletters.
`X*`	Zoekt 0, 1, 2, ... X-en.
`XX*`	Zoekt 1, 2, 3, ... X-en. (zelfde als X+)

of men kan teksten vervangen:

`s/a/e`
Zoekt op de huidige regel naar een letter a en vervangt deze door de letter e. (s = substitute)

```
s/i/o/g
```
Zoekt op de huidige regel naar alle i's en vervangt deze door een o. (g = global)

```
1,$s/x/y/g
```
Zoekt in de ganse tekst (van regel 1 t.e.m. einde (= $)) de letter x en vervangt deze door een y.

```
1,$s/p.o/zzz/g
```
Zoekt in de ganse tekst naar p<om het even wat>o en vervangt dit door zzz.

```
1,$s/^/>>/
```
Vervangt het begin van elke regel door >>.

```
1,$s/..$//
```
Wist de 2 laatste tekens van elke regel.

```
1,$s/e.*e/+++/g
```
Vervangt overal e...e door +++.

```
1,$s/[aeiouyAEIOUY]//g
```
Niet zo praktisch om te lezen, maar wist alle klinkers uit de tekst.

Een paar handige tekst-vervangingen op basis van RegEx die ik wel interessant vind:

```
1,$s/  */ /g
```
Vervangt overal meerdere spaties door 1 spatie.

```
1,$g/^$/d
```
Wist alle lege regels uit een tekst.

```
1,$s/\(.*\)^I\(.*\)/\2^I\1/g
```
(^I = tab): Zoekt <om het even wat parm1><tab><om het even wat parm2> en vervangt dit door <parm2><tab><parm1>.