

KaHo Sint-Lieven - Studiegebied IWT  
Opleiding Bachelor ICT

Mario Wyns

**Labo Linux 1**



# Inhoudsopgave

<b>1 Inleiding</b>	<b>1</b>
1.1 Historiek . . . . .	1
1.1.1 Unix . . . . .	1
1.1.2 Linux . . . . .	1
1.2 Kernel . . . . .	2
1.3 Distributies . . . . .	2
1.4 *BSD . . . . .	3
1.5 GNU . . . . .	3
1.6 GPL / BSD licenties . . . . .	3
1.7 Voor- en nadelen van GNU/Linux tegenover Windows . . . . .	4
1.7.1 De voordelen . . . . .	4
1.7.2 Nadelen van GNU/Linux tegenover Windows . . . . .	5
<b>2 Waar en hoe uitgebreide informatie vinden</b>	<b>7</b>
2.1 The Linux Documentation Project (TLDP) . . . . .	7
2.1.1 HOWTO's . . . . .	7
2.1.2 Mini-HOWTO's . . . . .	9
2.1.3 Guides . . . . .	9
2.1.4 FAQ . . . . .	10
2.1.5 De man pagina's . . . . .	10
2.1.6 Linux gazette . . . . .	10
2.2 Nieuwsgroepen . . . . .	10
2.2.1 gnu.* . . . . .	10
2.2.2 comp.os.linux.* . . . . .	11
2.2.3 be.comp.os.linux . . . . .	11
2.3 Google . . . . .	11
2.4 Boeken . . . . .	11

2.5 LUGs . . . . .	11
<b>3 Installatie van Linux</b>	<b>13</b>
3.1 Inleiding . . . . .	13
3.2 Partitionering . . . . .	13
3.3 Gebruikers . . . . .	13
3.4 Software . . . . .	14
3.5 Bootloader . . . . .	14
3.6 Console toegang . . . . .	15
3.7 Netwerkinstallatie . . . . .	15
<b>4 Directories en bestanden</b>	<b>17</b>
4.1 Directories . . . . .	17
4.2 De directory ordening onder Linux . . . . .	18
4.3 Veranderen van directory . . . . .	18
4.4 Directory aanmaken . . . . .	19
4.5 Directory-gegevens bekijken . . . . .	19
4.6 Randapparaten als bestanden . . . . .	20
4.6.1 Hoe randapparaten voorgesteld worden . . . . .	20
4.6.2 Het mount bevel . . . . .	21
4.6.3 Het /etc/fstab bestand . . . . .	22
<b>5 Basisopdrachten</b>	<b>25</b>
5.1 De shell . . . . .	25
5.1.1 Samenstelling van opdrachten . . . . .	25
5.1.2 Opdrachten invoeren . . . . .	26
5.1.3 Informatieve opdrachten . . . . .	26
5.2 Creëren van bestanden . . . . .	26
5.3 Bekijken van bestanden . . . . .	27
5.4 Copy, Move en Link . . . . .	27
5.5 Directories en bestanden verwijderen . . . . .	28
5.6 Redirection . . . . .	28
5.6.1 Uitvoer omleiden . . . . .	28
5.6.2 Invoer omleiden . . . . .	29
5.6.3 Foutmeldingen omleiden . . . . .	29
5.7 Pipes . . . . .	30
5.8 Archiveren van bestanden . . . . .	31
<b>6 Rechten op bestanden en directories</b>	<b>33</b>
6.1 Opbouw . . . . .	33
6.1.1 Voorbeeld . . . . .	34
6.2 Bestanden en directories . . . . .	34
6.3 Rechten veranderen . . . . .	35

6.3.1	Symbolisch . . . . .	35
6.3.2	Numeriek . . . . .	35
6.4	Eigenaar veranderen . . . . .	36
6.5	Speciale rechten . . . . .	36
<b>7</b>	<b>Editors</b>	<b>37</b>
7.1	Inleiding . . . . .	37
7.2	Vi/Vim . . . . .	37
7.2.1	Verschillende modes . . . . .	38
7.2.2	Enkele standaard bevelen . . . . .	38
7.2.3	Extra opdrachten . . . . .	39
<b>8</b>	<b>Reguliere Expressies</b>	<b>41</b>
8.1	Wat zijn reguliere expressies? . . . . .	41
8.2	grep . . . . .	41
8.3	De mogelijke reguliere expressies met grep . . . . .	42
8.4	Enkele grep opties . . . . .	45
8.5	e-mailadressen ontleden en zoeken . . . . .	46
<b>9</b>	<b>Shells en shell scripting</b>	<b>49</b>
9.1	Shell . . . . .	49
9.2	Aliases . . . . .	50
9.3	Programmeren in de Shell . . . . .	50
9.4	Variabelen . . . . .	51
9.4.1	Systeemvariabelen . . . . .	51
9.4.2	Gebruikersgedefinieerde variabelen . . . . .	51
9.5	Speciale karakters . . . . .	53
9.6	Vergelijken van expressies . . . . .	54
9.6.1	Rekenkundige expressies . . . . .	54
9.6.2	Integer expressies . . . . .	54
9.6.3	String expressies . . . . .	55
9.6.4	Bestandsexpressies . . . . .	55
9.6.5	Logische expressies . . . . .	55
9.7	Conditionele expressies . . . . .	56
9.8	Iteraties . . . . .	59
9.8.1	for . . . . .	59
9.8.2	while . . . . .	61
9.8.3	until . . . . .	62
9.9	Functies . . . . .	62
9.10	Scripts onderbreken . . . . .	63
9.11	Dialogen . . . . .	63
9.11.1	Infobox . . . . .	63
9.11.2	Messagebox . . . . .	64

9.11.3 Ja/Nee box . . . . .	64
9.11.4 Inputbox . . . . .	65
9.11.5 Menu's . . . . .	65
<b>10 Gebruikersbeheer</b>	<b>67</b>
10.1 Inleiding . . . . .	67
10.2 Manueel nieuwe gebruikers toevoegen . . . . .	67
10.2.1 Nieuwe accounts definiëren . . . . .	68
10.2.2 Paswoorden toekennen . . . . .	69
10.2.3 Een home-directory aanmaken . . . . .	72
10.2.4 Login initialisatiebestanden . . . . .	72
10.2.5 Bestandspermissies instellen . . . . .	74
10.2.6 Een shell toekennen . . . . .	74
10.2.7 De gebruiker toevoegen aan andere systeemfaciliteiten . . . . .	74
10.2.8 Uittesten van de nieuwe account . . . . .	74
10.3 Automatisch gebruikers toevoegen . . . . .	75
10.4 Accounts inperken . . . . .	75
10.5 Accounts verwijderen of buiten gebruik stellen . . . . .	76
10.5.1 Gebruikersaccounts verwijderen . . . . .	76
10.5.2 Aanverwante zaken . . . . .	76
10.6 Groepen . . . . .	76
10.6.1 Het /etc/group bestand . . . . .	77
10.6.2 Lidmaatschap in secundaire groepen . . . . .	77
10.7 Standaard gebruikers en groepen . . . . .	77
10.7.1 Standaard gebruikers . . . . .	77
<b>11 Kernel</b>	<b>79</b>
11.1 Taken van de kernel . . . . .	79
11.2 Ontwikkeling . . . . .	79
11.3 Kernelversies . . . . .	79
11.4 Kernel informatie . . . . .	80
11.5 Een Linux kernel compileren . . . . .	80
11.6 Een kernel op maat . . . . .	80
11.7 Linux kernel configuratie . . . . .	81
11.8 Modules . . . . .	84
11.9 Kernel installatie . . . . .	85

## Inleiding

### 1.1 Historiek

#### 1.1.1 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.

#### 1.1.2 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.

## Wat is Linux

Linux is een Unix-achtig besturingssysteem. Als we het strikt nemen 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.

## 1.2 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](http://www.kernel.org).

## 1.3 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

**Fedora** : zoals RedHat maar dan zonder bedrijfssupport

**Mandriva** : Frans van oorsprong. Deze distributie is vooral gericht op gebruiksvriendelijkheid

**SUSE** : een Duitse versie van linux die populair is in Europa

**Ubuntu** : populaire distributie, vooral voor de desktop

**Debian** : gemaakt door vrijwilligers

**Slackware** : gelijkaardig met Debian

**Gentoo** : distributie waarbij alles gecompileerd wordt voor maximale prestatie

## 1.4 \*BSD

Vaak hoort men in linux nieuwsgroepen ook de namen van FreeBSD, OpenBSD en NetBSD vallen. De \*BSD's zijn net zoals linux vrij verkrijgbaar. Vaak is het verschil met een linuxsysteem heel klein omdat de \*BSD's gebruik maken van dezelfde programma's (de kernel is het enige grote verschil tussen deze twee systemen).

**FreeBSD** : gemaakt om als snelle server te dienen

**NetBSD** : gemaakt om op zoveel mogelijk verschillende systeemarchitecturen te draaien

**OpenBSD** : op een bepaald moment afgesplitst vanuit NetBSD na een "flamewar" tussen de NetBSD ontwikkelaars en Theo De Raadt. Deze laatste heeft dan OpenBSD opgericht, dat vooral gemaakt is met *security* in gedachte.

## 1.5 GNU

De commercialisering van UNIX stootte een zekere Richard Stallman zozeer tegen de borst dat hij besloot actie te ondernemen. Met de typisch Amerikaanse mengeling van naïviteit en dadendrang richtte hij in 1984 de *Free Software Foundation* op. Hij schreef samen met zijn geestesgenoten software die op praktisch elke Unix kon draaien. GNU, wat staat voor "Gnu is Not Unix", is de verzamelnaam van deze Unix-software. Toen de GNU programma's vervolledigd waren moest er nog een kernel worden geschreven om een volwaardig UNIX systeem te hebben. De GNU Hurd kernel was hiervoor bedoeld maar deze is nooit afgeraakt. De Linuxkernel kwam net op het juiste moment om deze leegte op te vullen.

## 1.6 GPL / BSD 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. Linux is niet enkel "free" in de zin van "free beer" (gratis), maar ook als "free speech" (vrij). 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).

## 1.7 Voor- en nadelen van GNU/Linux tegenover Windows

### 1.7.1 De voordelen

**Stabiliteit:** GNU/Linux is veel stabieler dan Windows. Ook in GNU/Linux kan een programma vastlopen, maar het zal niet ons hele besturingssysteem vullen. Er zijn veel mensen die GNU/Linux maanden en zelfs jaren draaiende houden op hun pc zonder één enkele reboot.

**Het rebooten:** Een Windows gebruiker die iets nieuws installeert (hard- of software) moet vaak het systeem rebooten. In GNU/Linux hoeft er niet ge-reboot te worden.

Opmerking: enkel indien we een andere kernel gaan gebruiken dienen we GNU/Linux te rebooten.

**Herinstallatie:** Als we veel programma's installeren en verwijderen gaat het register van Windows dichtslippen. We kunnen dan wachten tot alles rotsvast zit en een format c: uitvoeren. Of we kunnen Windows regelmatig herinstalleren. In GNU/Linux is hier geen sprake van (er bestaat simpelweg geen register). Als je een programma verwijdert kan dit zonder dat er ook maar één bit op de harde schijf achterblijft.

**De prijs:** Naast het feit dat we GNU/Linux gratis kunnen downloaden of kopiëren, kunnen we voor weinig geld een distributie aanschaffen. Dan krijgen we één of meerdere cd's boordevol software. Mochten we de Windows-equivalenten van deze software aanschaffen dan zouden we minstens Bill Gates moeten heten om alle licenties te kunnen betalen.

Opmerking: uiteraard hebben veel mensen Windows en aanverwanten illegaal gekopieerd maar in het geval van GNU/Linux is dit 100% legaal.

**Platform-onafhankelijk:** GNU/Linux is portable. Dit houdt in dat GNU/Linux werkt op merk pc's, klonen, Apple Macintosh, HP-Risc, Sun werkstations en Alpha computers. Het is dan ook nodeloos te zeggen dat twee verschillende computers waarop GNU/Linux draait geen probleem hebben om verbinding met elkaar te maken.

**Virussen:** Er bestaan zeer weinig virussen voor GNU/Linux.

**Vrije software:** Hier draait alles om bij GNU/Linux. We kunnen de software aan onze noden aanpassen. We hebben toegang tot alle broncode. Nadien moeten we natuurlijk de aangepaste broncode ter beschikking stellen voor anderen. Door het systeem van vrije software kunnen bugs heel snel opgespoord worden, tenslotte mag iedereen aan de software werken. Vaak wordt dit argument door tegenstanders verworpen met het argument dat niet alle gebruikers programmeurs zijn. Dit is natuurlijk waar, maar omdat de software vrij is, kunnen gebruikers andere mensen inhuren die de software naar hun noden aanpassen. Mensen die onvrije software gebruiken kunnen dit niet en zijn voor altijd gebonden aan de software producent. Met vrije software vermindert men deze lock in.

**Multi-tasking en multi-user:** GNU/Linux is een goed multi-tasking en multi-user besturingssysteem. Terwijl we een groot document afprinten zal GNU/Linux niet vertragen, wat niet kan gezegd worden van Windows. In GNU/Linux kunnen meerdere gebruikers terzelfdertijd van één computer gebruik maken. De configuratie- en persoonlijke bestanden staan per gebruiker in een eigen directory.

**Hardware:** GNU/Linux stelt geen al te hoge eisen aan de hardware. Als we een nieuwe versie van Windows aanschaffen hebben we meteen een reden om een nieuwe computer te kopen. Het geheugen- en processorgebruik neemt toe per versie die uitkomt.

Opmerking: aangezien we GNU/Linux kunnen gebruiken zonder een grafische omgeving, kunnen we reeds aan de slag met een 386 met 2 MB geheugen. Dit is natuurlijk niet wat het merendeel van de thuisgebruikers wensen te doen met hun computer. Maar met een 486 met 16MB geheugen is een grafische omgeving mogelijk.

**Netwerk mogelijkheden:** Met GNU/Linux kunnen we alles gaan doen wat de 'grooten' doen: een netwerk opstellen, een DNS-, proxy-, mail- of webserver opzetten, ... Niet in het minst omdat we alle software bij de distributie zullen aantreffen.

**Kennis:** Hoewel dit ook als nadeel kan bekijken worden, krijgen we aan de hand van GNU/Linux een beter inzicht in de werking van een computer. Dit omdat we veel moeten weten over de onderdelen die er in onze computer zitten. Is dit moeilijk? Nee. Het vraagt alleen wat tijd. Eenmaal we GNU/Linux leren kennen zijn de mogelijkheden onbeperkt. We kunnen alles configureren zoals we zelf willen.

**Programmeren:** GNU/Linux is eveneens een goed platform voor programmeurs.

## 1.7.2 Nadelen van GNU/Linux tegenover Windows

**Software:** Nog vele andere populaire softwarepakketten zijn voorlopig slechts beschikbaar in een Windows versie. Dit is vooral omdat deze softwarepakketten niet vrij zijn, waardoor andere programmeurs ze niet kunnen porten naar GNU/Linux.

Er bestaan ook softwarefabrikanten die een GNU/Linux versie van hun software uitbrengen, al dan niet als vrije software. Een voorbeeld hiervan is Adobe, die hun Acrobat Reader geport hebben naar GNU/Linux. Acrobat Reader is geen vrije software, maar draait wel op GNU/Linux.

Trouwens, er is reeds veel vrije software ter beschikking voor GNU/Linux die op zijn minst even goed is als de Windows variant. Denk maar aan OpenOffice dat een alternatief biedt voor Microsoft Office.

Oplossing: hier biedt een dualboot of virtuele machine een oplossing, dit tenminste als je geen genoegen neemt met het GNU/Linux alternatief van een (onvrij) populair software-pakket.

**Hardware:** Sommige fabrikanten van hardware maken nog steeds geen drivers voor GNU/Linux. Er is dus hardware die niet werkt onder GNU/Linux. Andere fabrikanten geven de specificaties niet vrij, waardoor de GNU/Linux programmeurs niet de kans krijgen eraan te werken.

Oplossing: Voordat een onderdeel voor een pc aangeschaft wordt zoeken we eerst eens op of GNU/Linux dit ondersteunt. Is dit niet het geval dan kopen we gewoon een ander merk. Bijvoorbeeld wat grafische kaarten betreft: Nvidia en 3DFx richten zich ook op hardware ondersteuning voor Linux, wat onmiddellijk maakt dat we genoeg keuze hebben aan grafische kaarten voor ons GNU/Linux besturingssysteem.

# 2

## Waar en hoe uitgebreide informatie vinden

### 2.1 The Linux Documentation Project (TLDp)

Een eerste startpunt voor informatie over GNU/Linux is **The Linux Documentation Project**

TLDp heeft als doel het ontwikkelen van vrije, kwalitatieve documentatie voor het GNU/Linux besturingssysteem. Dit houdt onder andere het creëren van “**HOWTOs**” en “**Guides**” in. Zo hopen ze een systeem van documentatie op te zetten voor GNU/Linux dat gemakkelijk te gebruiken en te doorzoeken is. Dit houdt ook de integratie in van de **manual pages**, **info docs** en andere documenten.

TLDp bestaat vooral uit vrijwilligers met een minimale centrale organisatie. Iedereen die wil meehelpen wordt dan ook uitgenodigd.

De Linux **HOWTOs** en **mini-HOWTOs** zijn gedetailleerde “how to” documenten over specifieke onderwerpen.

Wat volgt is een overzicht van de belangrijkste HOWTOs en mini-HOWTOs.

#### 2.1.1 HOWTO's

De **HOWTO's** kunnen we downloaden van, of bekijken op de TLDp webstek. Er zijn tevens nog verschillende andere webstekken die deze service aanbieden.

**3Dfx-HOWTO** Beschrijft de **3Dfx** graphics accelerator chip ondersteuning voor Linux.

**Adv-Routing-HOWTO** Een zeer handige stap voor stap uitleg over **iproute2**, trafiek controle en een stuk over netfilter.

**Apache-Compile-HOWTO** Beschrijving hoe we de **web server Apache** compileren met belangrijke modules zoals mod\_perl, mod\_dav, mod\_auth\_ldap, mod\_dynvhost, mod\_roaming, mod\_jserv en mod\_php.

**Apache-Overview-HOWTO** Overzicht van de Apache web server en aanverwante projecten.  
Bevat vooral verwijzingen naar andere documenten.

### **Belgian-HOWTO**

**Cable-Modem** Beantwoordt vragen over hoe we ons GNU/Linux systeem moeten aansluiten op een **kabelmodem**.

**CD-Writing-HOWTO** Legt uit hoe we CD-ROMs moeten schrijven onder GNU/Linux.

**Config-HOWTO** Probeerde uit te leggen hoe we ons zopas geïnstalleerde GNU/Linux systeem *fine tunen*. Geeft ook een verzameling van configuratie bestanden voor de meest voorkomende programma's en services.

**DNS-HOWTO** Hoe we een DNS service opzetten.

**DOS-Win-to-Linux-HOWTO** Speciaal geschreven voor alle DOS en Windowsgebruikers die beslist hebben om over te stappen naar GNU/Linux.

**Firewall-HOWTO** Hoe we een **firewall** opzetten.

**Hardware-HOWTO** Lijst van de meest door Linux ondersteunde **hardware** en hulp bij het lokaliseren van de nodige **drivers**.

**IP-Masquerade-HOWTO** Legt uit hoe we Linux **IP Masquerade** aanzetten op een gegeven Linux systeem.

**Kernel-HOWTO** Een gedetailleerde gids door de kernel configuratie en upgrades.

**LDAP-HOWTO** Informatie over de installatie, configuratie, in gebruikname en onderhoud van **LDAP** (Lightweight Directory Access Protocol) server op een GNU/Linux machine.

**Sound-HOWTO** Beschrijft de geluidsondersteuning voor Linux (hardware, configuratie, . . .).

**TeX-TeX-HOWTO** Omvat de basis installatie en gebruik van het **TeX** en **LaTeX** systeem, plus bijkomende pakketten zoals Ghostscript.

**Xinerama-HOWTO** Configuratie van XFree86 Version 4.0 met meerdere monitors en de **Xinerama** extensies.

### 2.1.2 Mini-HOWTO's

We geven weer een kort overzicht van de **mini-HOWTO's**, op de webstek van TLDP staan er nog veel meer.

**3-Button-Mouse** Hoe we een muis met drie knoppen kunt laten werken onder GNU/Linux.

**Advocacy** Suggesties voor de GNU/Linux community: hoe ze het GNU/Linux woord kunnen verkondigen.

**Alsa-sound** Beschrijft de installatie van de **ALSA** geluidsdrivers voor Linux.

**BackspaceDelete** Oplossingen om werkende Backspace en Delete toetsen te hebben op de console en onder X.

**DHCP** Beantwoordt de basisvragen over hoe we een **DHCP server** en **DHCP client** opzetten.

**Euro-Char-Support** Beschrijving hoe we ondersteuning voor het Euro-teken aanzetten onder GNU/Linux.

**Install-Strategies** Bespreking van enkele installatiestrategieën voor diegenen die willen **dual-booten** tussen GNU/Linux en Windows.

**LILO** Het gebruik van **LILO** (Linux Loader).

**Multiboot-with-LILO** Hoe te multibooten tussen Windows 9X, Windows NT en GNU/Linux.

**Partition** Hoe we het best onze harde schijf indelen.

### 2.1.3 Guides

Op de TLDP webstek vinden we ook **Guides**. Dit zijn grote brokken documentatie waar we heel veel informatie kunnen uit halen. Ze gaan ook dieper op de materie in.

**Advanced Bash-Scripting Guide** Dit document is tegelijkertijd een tutorial als een referentiewerk voor Bash shell scripting. Het vereist geen voorkennis van shell scripting maar bezorgt ons op korte tijd een degelijke onderbouw. Het staat ook vol goed becommentarieerde voorbeelden.

**Linux From Scratch** Dit boek beschrijft het proces om ons eigen GNU/Linux besturingssysteem te maken door alleen maar gebruik te maken van broncode.

**The Linux Cookbook: Tips and Techniques for Everyday Use** Meer dan 1500 tijdsbesparende tips en recepten voor moderne computergebruikers.

**The Linux System Administrators' Guide** Dit boek gaat uit van een degelijke voorkennis van de installatie en gebruik van GNU/Linux. Het legt uit hoe we een GNU/Linux systeem draaiend houden, hoe we software upgraden, systeem configuratie, backups, user accounts, ...

**The Linux Network Administrator's Guide, Second Edition** Eén van de betere referentierwerken voor netwerkbeheer onder GNU/Linux.

Op de TLDP webstek vinden we nog meer guides.

### 2.1.4 FAQ

Zoals reeds gezegd bevat de TLDP webstek ook **Frequently Asked Questions**. Enkele zijn: Linux Frequently Asked Questions with Answers, Linux-RAID FAQ, The Wine FAQ.

### 2.1.5 De man pagina's

**Man** pagina's zijn de hulppagina's die bij een programma horen. De meeste programma's bevatten een man pagina, hierin vinden we dan een korte uitleg wat het programma doet en hoe we het gebruiken. We vinden er meestal ook een opsomming van alle opties.

### 2.1.6 Linux gazette

**Linux Gazette** is een online magazine dat telkens voorzien is van GNU/Linux tips en trucs. Via de TLDP webstek kunnen we de huidige editie raadplegen en ook alle voorbije edities.

## 2.2 Nieuwsgroepen

Via onze favoriete nieuwslezer kunnen we de verschillende nieuwsgroepen lezen die handelen over GNU/Linux, GNU en Linux. Via <http://groups.google.com/> kunnen we de meeste van deze nieuwsgroepen on line raadplegen.

### 2.2.1 gnu.\*

De nieuwsgroepen die beginnen met gnu.\* handelen over de GNU. Hier vinden we onder andere discussies over bepaalde GNU software.

### 2.2.2 comp.os.linux.\*

**comp.os.linux.advocacy** Discussies over de voordelen van Linux versus andere besturingssystemen.

**comp.os.linux.announce** Linux aangaande aankondigingen.

**comp.os.linux.apps** Algemene discussies over Linux applicaties.

**comp.os.linux.development.system** Discussies specifiek over de Linux kernel, device drivers en laadbare modules.

**comp.os.linux.hardware** Algemene discussies over Linux hardware compatibiliteit.

**comp.os.linux.misc** Allerhande Linux discussies die niet in een andere nieuwsgroep thuishoren.

**comp.os.linux.networking** Netwerk discussies.

**comp.os.linux.setup** Algemene discussies over de installatie van Linux en systeembeheer.

**comp.os.linux.x** Specifieke discussies over het X Windows System onder Linux.

### 2.2.3 be.comp.os.linux

Onze eigenste Belgische Linux nieuwsgroep. Op <http://lugwv.be/~swift/bcol/> vinden we alvast de FAQ van deze nieuwsgroep. Iedereen die iets wil posten leest die FAQ best eens door.

## 2.3 Google

Als we op de TLDP webstek geen antwoord zouden vinden op onze vragen kunnen we nog altijd **Google** (<http://www.google.be>) raadplegen.

## 2.4 Boeken

Naast de talrijke online documentatie kunnen we natuurlijk nog altijd een boek kopen over GNU/Linux. De boeken die uitgegeven zijn bij O'reilly zijn altijd van degelijke kwaliteit en zijn vaak de enige boeken die over een bepaald onderwerp te vinden zijn.

## 2.5 LUGs

**Linux User Groups** zijn verenigingen waar Linux gebruikers samenkommen. In de Belgian HOWTO vinden we een overzichtje van de Belgische LUGs.



# 3

## Installatie van Linux

### 3.1 Inleiding

De installatie van Linux wordt met elke nieuwe release eenvoudiger. Bij de meeste distributies is de installatie min of meer gelijklopend.

Hieronder volgen een aantal aandachtspunten.

### 3.2 Partitionering

Voer deze indien mogelijk de partitionering handmatig uit, sommige distributies wissen automatisch de volledige schijf !

Het minimum noodzakelijke aan schijfruimte is:

- een swappartitie ter grootte van ongeveer twee keer het RAM-geheugen
- een linux partitie: grootte min. 5GB voor de software van moderne distributies, 200MB is nodig voor een minimale installatie

Let op: formateer enkel de bovenstaande aangemaakte partities, laat de eerste (Windows) partitie als aktief staan en formateer die niet.

### 3.3 Gebruikers

De administrator wordt root-gebruiker genoemd in Linux, kies voor deze account een veilig wachtwoord maar vergeet het niet.

Maak minstens ook één gewone gebruiker aan om je aan te melden op het systeem. Je aanmelden als root is een slechte gewoonte omdat dan alle opgestarte (grafische) programma's met administrator-rechten uitgevoerd worden.

## 3.4 Software

Een linux distributie is een verzameling van zowel het besturingssysteem zelf als allerlei soorten software. Wanneer je dus linux installeert kan je meteen ook de nodige software mee installeren.

De volgende groepen zijn meestal vorhanden:

- grafische omgeving: X-Window + KDE of Gnome
- internet clients
- kantoor / office
- grafische programma's
- servers
- ontwikkeling
- documentatie

## 3.5 Bootloader

Na de installatie moet linux kunnen opgestart worden, eventueel moet er ook voor een reeds bestaande windows installatie kunnen gekozen worden. Dit kan a.d.h.v. een bootloader.

Tegenwoordig kiezen de meeste distributies voor GRUB als bootloader.

De plaats van de installatie van de bootloader is belangrijk. Dit kan op de volgende plaatsen:

- in de MBR (/dev/hda), de eenvoudigste manier om het systeem te starten
- in de bootrecord van de partitie (bv. /dev/hda7), als je een andere bootloader laat doorverwijzen naar die van Linux
- op floppy of USB-stick, de veiligste manier maar omslachtig om het systeem op te starten

Verder is het bij de installatie van de bootloader ook nog belangrijk dat je het lijstje controleert van beschikbare besturingssystemen. Hier kan je meestal ook het standaard besturingssysteem instellen dat opgestart wordt.

## 3.6 Console toegang

Tijdens de installatie is er meestal toegang tot systeem. Dit wil zeggen dat je bijvoorbeeld de actieve processen kan nagaan of de netwerkverbinding kan controleren.

Activeer een console m.b.v. CTRL-ALT-F1 (of F2). Terug naar het (grafische) installatiescherm kan met CTRL-ALT-F7.

## 3.7 Netwerkinstallatie

Sommige distributies laten een netwerkinstallatie toe. Men gebruikt hiervoor een netinstall-CD of floppy. Enkel de basis staat op het installatiemedium, de rest wordt geïnstalleerd over het netwerk.

Op die manier heeft men onmiddelijk de laatste versies en patches van alle pakketten. Een ander voordeel is dat men enkel die pakketten download die men ook effectief installeert. Een nadeel is dan weer dat wanneer men meer dan één computer moet installeren de software meerdere malen zal gedownload worden.



# 4

## Directories en bestanden

De basis van Linux is gebaseerd op bestanden. Alles is eigenlijk een bestand. Er zijn vier soorten bestanden:

- gewone bestanden
- directories
- apparaatbestanden (devices)
- koppelingen of links

Extensies zijn niet nodig maar mogen wel gebruikt worden (zelfs meerdere).

### 4.1 Directories

Bestanden in Linux zijn hiërarchisch geordend. Dit wil zeggen dat zij in een directory boom staan, die te vergelijken is met de directory structuur van DOS<sup>1</sup>. Er zijn een aantal afwijkingen: ten eerste maakt Linux het onderscheid tussen hoofdletters en kleine letters ("case-sensitive") in de namen van directories en bestanden: Sample.txt is niet hetzelfde als sample.txt. Verder is het niet het "\\" teken dat directories van elkaar scheidt, maar "/". De subdirectory C van de directory src is dus src/C. UNIX (en dus ook Linux) systemen kennen ook geen drive aanduidingen zoals in DOS (zoals, bijvoorbeeld, A: en C:). Alle drives zijn toegankelijk via een subdirectory van de root (/) directory. Op de meeste systemen is de CD-ROM drive bijvoorbeeld toegankelijk via /cdrom/ of /mnt/cdrom/. Hoe dit praktisch gebeurt, vind je in sectie 4.6. De actieve directory vraag je op met het bevel pwd (pwd staat voor "print working directory").

---

<sup>1</sup>Disk Operating System.

/	de root directory
/etc	bevat data om het systeem op te starten, deze directory bevat ook de meeste configuratiebestanden
/etc/passwd	bevat de gebruikers database (is geen directory, wel een bestand)
/etc/rc.d	bevat systeem-initialisatiescripts
/lib	bevat functiebibliotheeken die gebruikt worden door de C compiler, evenals de gedeelde bibliotheken
/tmp	bevat tijdelijke bestanden
/var	bevat bestanden die aan verandering onderhevig zijn
/home	bevat de accounts van de gebruikers
/home/student	dit is de user account van de gebruiker <i>student</i>
/bin	bevat de meest noodzakelijke en gebruikte binaire en uitvoerbare bestanden
/sbin	bevat alle systeem-programma's
/usr	bevat al de rest niet aanwezig in de andere directories
/usr/bin	bevat binaire en uitvoerbare bestanden zoals /bin
/proc	bevat een pseudo-filesystem dat dient als een interface naar kernel data structuren. Het wordt ondermeer gebruikt om proces informatie uit het geheugen te lezen.
/dev	bevat bestanden die de hardware voorstellen van het systeem

Tabel 4.1: Linux directory structuur

## 4.2 De directory ordening onder Linux

Anders dan bij het Microsoft Windows besturingssysteem, waar er slechts enkele “verplichte” directories zijn zoals “Program Files”, “Windows” en “Windows/System” bijvoorbeeld, is de directory structuur op een UNIX (en dus ook Linux) systeem voor een groot deel vooraf vastgelegd. tabel 4.1 geeft een overzicht van de structuur van een Linux systeem.

## 4.3 Veranderen van directory

Het commando om van directory te veranderen is cd (change directory). Enkele voorbeelden zijn:

cd ..                          Ga naar de onmiddellijk hogere directory in de directorytree (“parent directory”);

cd /	Ga naar de hoogste directory (root);
cd /home/craymaek	Ga naar de home directory van de gebruiker craymaek;
cd ~craymaek	idem;
cd ~	Ga naar je eigen home directory;
cd dir	Ga naar de directory dir die onder de huidige directory staat;

Directory specificaties zijn niet steeds relatief ten opzichte van de huidige directory of absoluut (ten opzichte van de root directory).

## 4.4 Directory aanmaken

Je kunt pas naar een directory gaan als deze directory bestaat. Om een directory te maken, moet je het commando `mkdir` (make directory) gebruiken met als parameter de directory die je wilt maken. Je kan alleen maar op de plaatsen waar je hiervoor toestemming hebt nieuwe directories aanmaken. Voor een gewone gebruiker is dat meestal alleen in de eigen home directory (`~/`) of de temp directory (`/tmp/`), voor de beheerder van het systeem (root of *super user*) is dat nagenoeg overal.

## 4.5 Directory-gegevens bekijken

Met het bevel `ls` (list) vraag je de inhoud van een directory op. Enkele voorbeelden zijn:

ls	Geeft de inhoud van de huidige directory;
ls oefeningen	Geeft de inhoud van de oefeningen directory;
ls h*	Geeft de lijst van alle bestanden waarvan de naam met h begint;
ls ?allo	Geeft de lijst van alle bestanden waarvan de naam met een willekeurig karakter begint en eindigt op "allo";

Vergeet niet dat ook hier directories relatief of absoluut kunnen zijn. Zoals je in de vorige voorbeelden kan zien is het ook mogelijk *joker*-tekens te gebruiken die door de shell ge-expand worden. Als je het teken \* gebruikt wil dit zeggen dat het met 0 of meer achtereenvolgende willekeurige tekens mag matchen. Het teken ? wil zeggen dat er met 0 of 1 willekeurig teken gematcht mag worden. Zo matcht ch\*ter1\* bijvoorbeeld met chapter1, chapter1, chter105689,... en ch?ter1? met chapter12, chapter1b,...

Het `ls` bevel heeft een reeks van parameters (switches). Enkele belangrijke zijn :

- a Geeft alle bestanden weer, ook de verborgen ". "bestanden;
- l Geeft de directory meer gedetailleerd weer;
- R Gaat ook alle subdirectories af;

Een bestand met een naam beginnende met een punt ".", is een speciaal bestand, meestal een configuratie bestand of iets dergelijks. Een gewone listing toont deze bestanden niet, we moeten hiervoor expliciet opgeven dat we alle bestanden willen zien met de optie -a.

Deze switches kunnen ook worden gecombineerd tot bv. ls -la. Indien de directory te groot is, zodat je niet alles in één keer op het scherm kan zien, kun je | more achter het ls bevel typen. Vb: ls -la /home | more. Dit geeft de directory listing pagina per pagina weer. Om naar de volgende pagina te gaan, moet je op de spatiebalk drukken. Bij sommige systemen moet je na de laatste pagina op "q" drukken (quit). Bijna alle bevelen die uitvoer op het scherm geven kunnen met | more worden gecombineerd. Meer informatie over het "|" symbool vind je in sectie 5.7. Een alternatief voor more is less. Hiermee kan je zowel voorwaarts als achterwaarts scrollen, waardoor less een meer ingewikkeld bevel is dan more. Na iedere pagina moet een bevel worden gegeven om het vervolg te zien. De belangrijkste zijn f (forward, soms ook de PageDown toets), b (backward, soms ook de PageUp toets) en q (quit). Meer informatie over less staat in de man pages.

## 4.6 Randapparaten als bestanden

### 4.6.1 Hoe randapparaten voorgesteld worden

Voor dat je de directories kan gebruiken om de randapparaten aan te spreken, moet je eerst het geviseerde "device" *mounten*. Dit wil zeggen dat je een directory en het apparaat logisch zal verbinden. Dit kan je bijvoorbeeld doen als volgt: mount -t iso9660 /dev/hdc /mnt/cdrom. *iso9660* duidt op het soort bestandssysteem dat men mount (andere mogelijkheden zijn *vfat* voor fat bestandssystemen en *extf2* voor Linux extended filesystem 2 bestandsystemen, en nog een hele hoop andere). */dev/hdc* is een voorbeeld van een bestand dat de verbinding maakt met de eigenlijke hardware; welk bestand voor welk randapparaat staat is afhankelijk van de hardware configuratie van je computer. Het hoeft bijvoorbeeld niet */dev/hdc* te zijn voor je cdrom; op een andere PC had dat ook */dev/hdb* kunnen zijn. Meestal kan je een floppy zelfs mounten zonder het type van bestandssysteem eraan mee te geven: mount /dev/fd0 /mnt/floppy bijvoorbeeld. */dev/fd0* stelt een bestand voor dat een abstractie van de hardware is (meer bepaald de floppy disk), en we willen deze mounten op de directory */mnt/floppy*. Let op: de directory */mnt/floppy* moet bestaan om dit bevel uit te voeren.

In UNIX en Linux worden randapparaten en andere hardware dus als bestanden voorgesteld. Deze speciale bestanden vind je terug in de directory */dev*. Om output naar deze randapparaten te sturen kan men dan simpelweg naar deze bestanden schrijven.

/dev/console	de monitor die aan de computer hangt
/dev/hd	de IDE randapparaten (harde schijven, cdrom), zo geeft /dev/hda2 de tweede partitie weer op de schijf hda
/dev/sd	de SCSI randapparaten
/dev/fd	de floppy drive
/dev/null	dit is de "vuilbak" van het systeem; alle data die men eraan toe schrijft is voor altijd verloren. Indien het gebruikt wordt als invoerbestand wordt er een bestand gecreëerd van lengte 0

Tabel 4.2: belangrijke device bestanden

Vb.: cat /usr/share/sounds/generic.wav > /dev/dsp stuurt de gegevens in het wave bestand naar de geluidskaart. We gebruiken hiervoor weer de redirectie operator >. Merk op dat het bestand waarnaar geschreven wordt in rechtstreeks contact staat met de eigenlijke hardware (via de kernel van het besturingssysteem om). Anderzijds kan men van deze bestanden ook informatie lezen. Zo geeft het bestand /dev/mouse<sup>2</sup> gegevens over de muis. Tabel 4.2 geeft een overzicht van de belangrijkste device bestanden.

### 4.6.2 Het mount bevel

We besteden nu wat meer aandacht aan de werking van het mount bevel. De algemene syntax van dit bevel wiet er als volgt uit:

```
mount [opties] device mountpoint
```

device staat voor het apparaat of de partitie die je wil mounten, mountpoint geeft aan waar je de data wil zien. Dat is een directory die al moet bestaan, en normaal gezien leeg is.

Enkele mogelijke opties zijn:

- **-a** mount alle filesystems in /etc/fstab (zie verder)
- **-t** Geeft het type filesystem aan (v.b. vfat, iso9660, ext2, msdos, ...)
- **-r** "read only": schrijven op het apparaat wordt volledig uitgeschakeld.
- **-o** gevuld door verdere opties voor het filesystem. Bijvoorbeeld:
  - 'ro' voor read only (zelfde als -r)

<sup>2</sup>Dit is meestal een link naar een ander bestand!

- ‘noexec’: zorgt ervoor dat programma’s op het gemounte filesystem niet kunnen uitgevoerd worden. Dit kan nuttig zijn voor beveiliging.

Zie `man mount` voor de overige opties.

Enkele voorbeelden van het gebruik van het mount bevel:

```
mount -t vfat /dev/hda1 /mnt/windows
```

Dit bevel zorgt ervoor dat de inhoud van de eerste partitie op de eerste IDE-hd toegankelijk wordt in de directory `/mnt/windows`. Het veronderstelt dat deze partitie een FAT/FAT32 filesystem bevat.

```
mount -t iso9660 -r /dev/hdc /mnt/cdrom
```

Dit zorgt ervoor dat je in `/mnt/cdrom` de inhoud van de CD in de CDROM drive aangesloten op secondary master ziet. Je kan ook met `mount -t iso9660 -o ro /dev/hdc /mnt/cdrom` hetzelfde effect bekomen.

```
mount -t vfat /dev/fd0 /floppy
```

Mount de DOS of MS Windows floppy die zich in de eerste floppydrive (de tweede drive is fd1) bevindt onder de directory `/floppy`. Je kan `/dev/fd0` vergelijken met A: onder MS Windows.

Als je de `-t` optie weglaat, tracht mount automatisch het type van filesystem te bepalen. Een verdere hulp in het automatiseren van mount bevelen is de configuratiefile `/etc/fstab`.

### 4.6.3 Het `/etc/fstab` bestand

Het configuratiebestand `/etc/fstab` bevat een aantal regels betreffende randapparaten die automatisch gemount kunnen worden. De regels die je terugvindt in fstab zien er als volgt uit:

```
device dir fstype opts df fs_passno
```

Het eerste veld is het te mounten apparaat of de partitie, bv. `/dev/hda1`. Het tweede veld is het mountpoint (zoals bij het mount commando). Het derde veld is het type filesystem, bvb. iso9660, ext2 , msdos,... Het vierde veld bevat een aantal opties. Hierin moet de manier van mounten vermeld staan, en eventueel opties die via `-o` doorgegeven worden aan het mount programma. Hier moet ook de optie ‘`noauto`’ vermeld worden: dit zorgt ervoor dat het filesystem in kwestie niet automatisch gemount wordt door ‘`mount -a`’ bij de systeemstart. Het vijfde veld heeft betrekking tot backups en het dump programma. Het zesde veld bepaalt de volgorde

waarin fsck (programma dat de integriteit van filesystems nagaat) de partities moet checken. Aan te raden is dit op 1 te zetten voor het / filesystem, 2 voor de andere filesystems.

Alle filesystems vermeld in /etc/fstab, behalve die met de noauto optie, worden bij het opstarten van het systeem automatisch gemount op de aangegeven mountpoints. Lijnen die beginnen met # worden als commentaar aanzien. Een paar voorbeelden zijn ook hier op hun plaats:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>  <type>  <options>          <dump> <pass>
/dev/hda2      /           ext2    defaults,errors=remount-ro 0       1
/dev/hda5      none        swap    sw                  0       0
proc          /proc        proc    defaults            0       0
/dev/hda1      /boot        ext2    rw                  0       2
/dev/hda6      /usr         ext2    rw                  0       2
/dev/hda7      /tmp         ext2    rw                  0       2
/dev/hda8      /var         ext2    rw                  0       2
/dev/hda4      /home        ext2    rw                  0       2
/dev/hdc1      /cdrom       iso9660 ro,user,noauto     0       0
```

Deze computer bevat dus een aantal ext2 partities. Ook zie je hier het gebruik van een aantal opties. Het plaatsen van regels voor vaak gebruikte filesystems in /etc/fstab heeft het voordeel dat je bijvoorbeeld niet telkens mount -t iso9660 /dev/hdb1 /mnt/cdrom hoeft te typen om de CDROM te mounten, mount /cd volstaat dan. Bij zo een bevel gaat mount zelf kijken in /etc/fstab voor de nodige info.



# 5

## Basisopdrachten

### 5.1 De shell

De shell, ook wel console, opdrachtprompt of commandline genoemd bestaat uit de volgende onderdelen:

- gebruikersnaam gevolgd door het @-teken
- computernaam (hostname)
- een dubbele punt
- de huidige directory ( $\sim$  = home-directory)
- een > of \$ -teken (root-gebruiker : #)

Een voorbeeld:

```
mwyns@bach:~>      (gewone gebruiker)  
bach:/home/mwyns # (root gebruiker)
```

#### 5.1.1 Samenstelling van opdrachten

De shell interpreteert in het eerste ingetikte woord een opdracht, na de opdracht volgen eventuele opties. Deze beginnen meestal met een minteken en bestaan uit één karakter. Langere equivalenten van opdrachten beginnen met twee streepjes :

```
ls -l          (de korte versie)  
ls --format=long (de lange versie)
```

Korte opties mogen meestal aan elkaar geschreven worden; bv. ls -lR. Na de opdracht en eventuele opties volgen dan de parameters:

```
mkdir --parents documenten/brieven
```

### 5.1.2 Opdrachten invoeren

Met de pijltjes-toetsen kan men door reeds ingegeven opdrachten navigeren. Een opdracht aanvullen kan met de TAB-toets, wanneer we bv. `moz` indrukken en op TAB dan wordt dit `mozilla`. Een opdracht onderbreken in dezelfde shell kan met CTRL-C. Radicaal een opdracht onderbreken kan via een andere shell met de opdracht `kill` :

```
kill -9 130          (onderbreekt proces nr. 130)
killall -9 mozilla   (breekt alle processen met de naam mozilla af)
```

### 5.1.3 Informatieve opdrachten

Met de volgende opdrachten kan men informatie opvragen:

- `date` geeft de huidige datum en tijd weer
- `who` vertelt wie er is aangemeld op het systeem
- `cal` drukt een kalender af van de huidige maand:

September 2004						
Su	Mo	Tu	We	Th	Fr	Sa
					1	2
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

## 5.2 Creëren van bestanden

Er is een simpel bevel om een *leeg* bestand te creëren in Linux, namelijk `touch`. Bijvoorbeeld `touch mijnbestand` maakt een leeg bestand aan met de naam `mijnbestand` indien dit bestand nog *niet* bestond. Anders verandert het gewoon de "modification time" van het bestand. Let op: een bestandsnaam mag geen karakters bevatten die een speciale betekenis hebben voor de shell. Concreet wil dit zeggen dat de volgende tekens niet mogen gebruikt worden in een bestandsnaam:

```
! @ # $ % ^ & * ( ) [ ] { } ' " \ / | ; < > `
```

De eenvoudigste manier om een bestand aan te maken met tekst erin is natuurlijk via een tekst editor. De meeste UNIX systemen hebben drie karaktergeoriënteerde tekst editors: `ed`, `vi` en `pico`. Hiervan is `pico` (onderdeel van het email en news programma `pine`) de eenvoudigste

om te gebruiken, maar de minst krachtige. Een andere veel gebruikte editor is *emacs*. Onder software ontwikkelaars zijn emacs en vi (of vim, “vi improved”) zeer populaire editors. Niet zozeer vanwege hun gebruiksvriendelijkheid (de leercurve mag wel steil genoemd worden) maar wel vanwege de efficiëntie die ermee kan bereikt worden. Een korte introductie tot vim kan je vinden in hoofdstuk 7.

## 5.3 Bekijken van bestanden

Een tekstbestand kan met een teksteditor worden bekeken. Maar indien je enkel een bestand wil bekijken is een tekst editor hiervoor niet altijd het beste hulpmiddel. Belangrijke bestanden<sup>1</sup> kan je beter niet editeren als je niet precies weet wat je aan het doen bent. Je loopt dan immers het gevaar dat je niet meer kan inloggen, als je per ongeluk een aantal parameters een verkeerde waarde geeft. Het kan ook zijn dat je wel het recht hebt om een bestand te bekijken maar niet om dit bestand te wijzigen. Het cat bevel is handig om een bestand te bekijken. De syntax is `cat bestandsnaam`. Ook `more` kan gebruikt worden samen met `cat` door middel van een pipe. Hierbij kan `cat bestandsnaam | more` op de meeste systemen worden afgekort tot `more bestandsnaam`. Dit is omdat `more` ook op bestanden kan werken, die dan per pagina op de standaard uitvoer (het scherm) te zien zijn.

`cat` wordt vaak gecombineerd met `head` of `tail`. Hiermee kan je respectievelijk het begin en het einde van een bestand bekijken. Vb.:

<code>cat readme   head</code>	Toont de 10 eerste regels van het bestand <code>readme</code> ;
<code>cat readme   head -n 5</code>	Toont de 5 eerste regels van het bestand <code>readme</code> ;
<code>cat log   tail</code>	Toont de 10 laatste regels van het bestand <code>log</code> ;
<code>cat log   tail -n 5</code>	Toont de 5 laatste regels van het bestand <code>log</code> ;

`cat` kan je ook gebruiken om twee of meerdere bestanden samen te voegen (concatenate), vb.: `cat bestand1 bestand2 > bestand3`. Dit wil zeggen dat de inhoud van `bestand1` en `bestand2` achter elkaar in `bestand3` moet worden geschreven. “`>`” is de redirectie operator en zegt dat de uitvoer niet naar de standaard uitvoer (dit is meestal de console) gaat, maar naar het vermelde bestand (`bestand3` in dit geval).

## 5.4 Copy, Move en Link

Een besturingssysteem is niet compleet als je bestanden niet kan kopiëren of verplaatsen. Het bevel `cp` (copy) kopiëert een bestand of directory. Om `bestand1` naar `bestand2` te kopiëren

---

<sup>1</sup>Onder “belangrijke bestanden” verstaan we configuratie bestanden zoals die bijvoorbeeld in de `/etc` directory voorkomen. Zonder een goede kennis van hun inhoud kan men deze beter niet zomaar veranderen.

moet je het volgende bevel geven: `cp bestand1 bestand2`. De volledige inhoud van een directory kan worden gekopieerd door gebruik te maken van het jokerteken “\*”. Het bevel `cp /etc/* .` zal de volledige inhoud van de `/etc` directory naar de huidige directory copiëren (doe dit niet: de `/etc` directory is zeer uitgebreid!!!). Door de `-r` switch te gebruiken kopiëer je een complete directory (inclusief de subdirectories), vb: `cp -r dir1 dir2`, dat `dir2` aanmaakt als kopie van `dir1`. De switch `-r` geeft aan dat het bevel `cp` hier recursief moet werken.

Het verplaatsen van bestanden en directories is analoog aan het kopiëren ervan. Hiervoor gebruik je het bevel `mv` (move). Met dit bevel hernoem je ook een bestand, vb.: `mv .pan .plan`.

In UNIX omgevingen kan men een bestand laten verwijzen naar een ander bestand; dit bestand wordt een link genoemd. De meest gebruikte links zijn symbolische links. Als men een symbolische link wil editeren, wordt het oorspronkelijke bestand geëditeerd als dat bestaat. Het bevel om een symbolische link te maken is `ln -s bestand1 bestand2`.

## 5.5 Directories en bestanden verwijderen

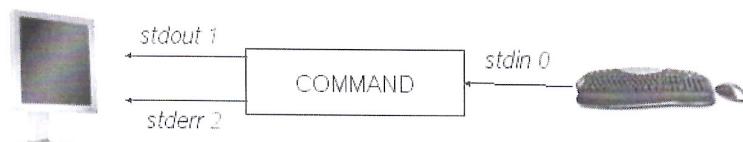
Bestanden kan je verwijderen met het bevel `rm` (remove), waarvan de syntaxis `rm bestandsnamen` is. Ook hier kan je jokerteken gebruiken. Om een (lege!) directory te verwijderen, gebruik je het bevel `rmdir` (remove directory). Het verwijderen van alle bestanden en subdirectories van een directory en het verwijderen van een directory zelf kan worden gecombineerd door de `-r` switch: `rm -r dirname`. Zoals bij het copy bevel geeft de `-r` switch ook hier aan dat `rm` recursief te werk moet gaan. Als je er zeker van bent dat je de bestanden wilt verwijderen kan je de optie `f` (force) gebruiken. Door `rm -f naam` in te typen zal het `rm` bevel dit bestand verwijderen zonder nog eens expliciet toestemming hiervoor te vragen aan de gebruiker.

## 5.6 Redirection

Veel opdrachten sturen standaard hun output naar het scherm. Men kan dit omleiden naar een bestand of een andere opdracht. Ook de input die normaal van het toetsenbord komt kan ook omgeleid worden. Verder bestaat er een tweede uitvoerkanaal voor eventuele foutmeldingen die de opdracht genereerd. De gewone output en de foutmeldingen hebben dus elk een onafhankelijk kanaal en kunnen dus onafhankelijk omgeleid worden. Zie figuur 5.1.

### 5.6.1 Uitvoer omleiden

Het gewone uitvoerkanaal wordt ook `stdout` genoemd. Om bijvoorbeeld de output naar een bestand te schrijven kan men de volgende opdracht gebruiken: `ls > ls.out`. Het bestand



Figuur 5.1: Omleiding of redirection.

`ls.out` wordt daarbij nieuw aangemaakt of overschreven als het reeds bestond. Output van een opdracht kan ook toegevoegd worden aan een bestand met bijvoorbeeld `ls >> ls.out`.

De onderstaande opdrachten:

```
date > ls.out
ls >> ls.out
```

zullen een bestand aanmaken met de huidige datum en directory inhoud.

### 5.6.2 Invoer omleiden

Het invoerkanaal wordt `stdin` genoemd, het invoer omleidingssteken is : <.

De volgende opdracht:

```
wc < ls.out
```

zal het aantal regels, woorden en tekens tellen in `ls.out`.

Men kan tegelijkertijd ook aan de output omleiden:

```
wc < ls.out > wc.out
```

dit schrijft het resultaat van de opdracht in `wc.out`.

### 5.6.3 Foutmeldingen omleiden

Het foutmeldingskanaal heeft `stderr` als naam. De fouten kunnen omgeleid worden met : 2>. De andere kanalen: input kan i.p.v. als < ook als 0< genoteerd worden, hetzelfde voor output: > staat voor 1>.

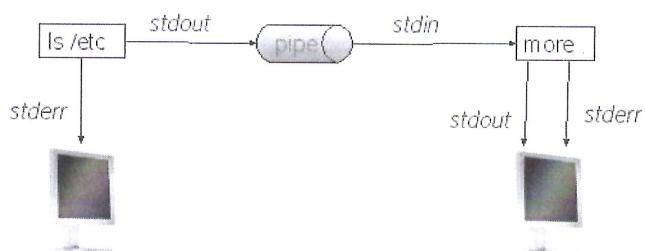
In het onderstaande voorbeeld:

```
rm images/ 2> fout.out
```

zal de foutmelding voor het verwijderen van een directory weggeschreven worden in `fout.out`.

## 5.7 Pipes

In het voorbeeldje van het gebruik van het `more` bevel zagen we het symbool “|”. Dit wordt een pipe (pijp) genoemd. In UNIX wordt het pipe symbol (|) gebruikt om twee of meerderen bevelen te combineren. Hierbij dient de uitvoer van een bevel als invoer voor het volgende bevel. De betekenis van dit symbool is makkelijk uit te leggen aan de hand van het voorbeeldje `ls /etc | more`. Het bevel voor de pipe, namelijk `ls /etc` genereert veel uitvoer, meer dan in één keer op het scherm kan. Het bevel na de pipe, namelijk `more`, zorgt ervoor dat als men het invoer geeft langer dan het scherm dit deel per deel kan bekijken worden op het scherm. Nu moeten we een manier hebben om de uitvoer van `ls /etc` als invoer aan `more` te geven. Hiervoor zorgt de pipe: “|”. Het neemt de uitvoer van het bevel dat voor het symbool | staat en voedt het als invoer aan het bevel dat achter het symbool | staat. Zie figuur 5.2.



Figuur 5.2: Gebruik van een pipe.

Een ander voorbeeldje van het gebruik van pipes is `ls /home | sort | less`. In het eerste deel, voor de eerste pipe wordt een listing gegeven van al de bestanden en subdirectories in `/home`. Deze listing is de invoer voor het bevel `sort`. Dit bevel sorteert de lijnen invoer die het krijgt in alfabetische volgorde. Als uitvoer van het bevel `sort` kan je dus een gesorteerde lijst regels verwachten. Deze uitvoer wordt door de tweede pipe doorgegeven aan het laatste bevel, nl. `less`. Dit zorgt ervoor dat de gebruiker al de uitvoer kan doornemen op het scherm.

Het bevel `grep` wordt ook regelmatig gebruikt in combinatie met andere bevelen. `grep` drukt lijnen af die aan een meegegeven patroon voldoen. Hierbij zoekt `grep` in de bestanden die het aangeduid krijgt via de argumenten of de standaard invoer als er geen bestanden opgegeven zijn. Als we bijvoorbeeld alle zinnen willen zien uit het bestand `/etc/rc.d/rc.local` waarin het woord “echo” voorkomt, dan kan dit door het bevel: `grep echo /etc/rc.d/rc.local`. Als eerste argument geeft men het patroon mee waarnaar men zoekt, en als tweede argument de plaats waar `grep` dit moet gaan zoeken (zie hoofdstuk 8 voor meer uitleg). Nu kunnen we bijvoorbeeld ook via een pipe een lijst tekstlijnen doorgeven en `grep` gebruiken om de ongewenste lijnen weg te filteren. Zo kunnen we bijvoorbeeld alle bestanden die als bestandsnaam het patroon “oef” bevatten (in de huidige directory) opvragen door het bevel `ls | grep oef`.

## 5.8 Archiveren van bestanden

Wanneer je met grote projecten werkt is het handig om de bestanden van dit project te archiveren. Hierbij worden de bestanden samengevoegd tot één bestand. Het bevel dat dit doet is tar (tape archiver). Historisch gezien wordt tar gebruikt om back-ups te maken op tape, maar het resultaat van tar kan eender welk bestand zijn. De syntax is: `tar [key] [bestand...]`. key kan de volgende waarden hebben:

c	creëert een nieuw archive (impliceert "r");
x	extract bestanden;
r	archiveert bestanden;
t	geeft een lijst van de bestanden in de archive;
v	verbose: geeft op de console weer wat er gebeurt;
f	de volgende parameter is de naam van de archive;
z of j	schakelt compressie in (resp. <i>gzip</i> en <i>bzip2</i> );

De `public_html` directory kan gearchiveerd worden met het bevel:

```
tar -cf pages.tar public_html
```

De switch c ("create") geeft aan dat er een tar-bestand wordt gecreëerd, f ("file") geeft aan dat we dit naar een bestand willen doen. Omdat de switch f aangeeft dat we naar een bestand gaan archiveren is het eerste argument dat verwacht wordt de bestandsnaam voor het resultaat, hier is dat `pages.tar`. Daarna verwacht tar al de directories en bestandsnamen die gearchiveerd moeten worden, hier is dat enkel de directory `public_html`.

De directorystructuur wordt op een relatieve manier ook mee gearchiveerd. Het gemaakte bestand is niet gecomprimeerd: de bestanden zijn gewoon in de archive geplaatst. Om een bestand te comprimeren kan je *gzip* gebruiken. Er zijn meerdere GNU compressie programma's beschikbaar, zo is *bzip2* een goed alternatief. Hiermee kan eender welk bestand worden gecomprimeerd. Het bevel *gzip* `pages.tar` verwijdert het bestand `pages.tar` en creëert het gecomprimeerde bestand `pages.tar.gz`. Het bevel *gunzip* dekomprimeert het bestand weer. Deze bevelen kunnen natuurlijk ook gecombineerd worden.

Om de `public_html` directory te archiveren en te comprimeren kan je het volgende intypen:

```
tar -c ./public_html/ | gzip > pubhtml.tgz
```

Door het tar bevel worden al de bestanden in de directory samengezet in een enkel bestand. De uitvoer van het tar bevel wordt rechtstreeks als invoer van het *gzip* bevel doorgegeven via de pipe. Meer uitleg over het gebruik van een pipe (`|`) vind je in sectie 5.7. *gzip* wijst

door > te gebruiken pubhtml.tgz als uitvoerbestand aan (dit wordt ook wel “redirectie naar een bestand” genoemd). Hierbij moet nog opgemerkt worden dat gzip slechts op individuele bestanden werkt. Als men geprobeerd had gzip ./public\_html/ in te typen zouden alle bestanden in de directory in aparte bestanden met .gz als extensie gecomprimeerd worden.

Om het bestand pubhtml.tgz nu terug uit te pakken kan men

```
tar -xvzf pubhtml.tgz
```

intypen (waarbij x aangeeft dat het om extractie gaat, v aangeeft dat er informatie over de operatie moet gegeven worden<sup>2</sup>, z aangeeft dat het bestand eerst gedecomprimeerd moet worden met gunzip en f aangeeft dat het om een bestand gaat).

---

<sup>2</sup>de optie v is beschikbaar voor de meeste bevelen om informatie te geven over hetgeen het bevek aan het doen is

# 6

## Rechten op bestanden en directories

### 6.1 Opbouw

In Linux behoort iedere gebruiker tot een *groep*. Aan de hand hiervan kan de beveiliging van het systeem worden opgezet. Ieder bestand in Linux heeft namelijk een eigenaar (user or owner), een groep (group) en een beveiligingsmodus van de volgende vorm TUUUGGGOOO. Dit kan men in 4 delen opsplitsen

T	type van het bestand, vb.: directory (d), gewoon bestand (-);
UUU	user read (r), write (w) en execute (x);
GGG	group read (r), write (w) en execute (x);
OOO	others read (r), write (w) en execute (x);

Symbolisch	Binair	Numeriek	Recht
---	000	0	elk type toegang is verboden
--x	001	1	alleen uitvoerrecht
-w-	010	2	alleen schrijfrecht
-wx	011	3	schrijf- en uitvoerrecht
r--	100	4	alleen leesrecht
r-x	101	5	lees- en uitvoerrecht
rw-	110	6	lees- en schrijfrecht
rwx	111	7	lees-, schrijf- en uitvoerrecht

Tabel 6.1: Symbolische en numerieke rechten

Bij iedere entiteit komen de rechten rwx voor; het lezen, schrijven en uitvoeren. Wanneer de letter er staat dan is het recht toegekend, staat er een streepje dan is het recht niet toegekend.

Naast de symbolische notatie bestaat er ook een numerieke (octale) notatie: aan de rechten wordt er nul toegekend als er een streepje staat, zoniet een één. Het bekomen binaire getal wordt dan in een octaal (of decimaal) getal omgezet. Zie tabel 6.1.

### 6.1.1 Voorbeeld

Als we de opdracht `ls -l` uitvoeren in onze home directory krijgen we bv. de volgende output:

```
-rwxr-xr-x 1 wynn user 32 Feb 9 8:01 test.txt
||-||-||-| | |--| |--| |-| |-----| |-----|
1 2 3 4 5 6 7 8 9 10
```

Deze lijn kunnen we als volgt opdelen:

1 = type	6 = eigenaar
2 = rechten v/d eigenaar	7 = groep
3 = rechten v/d groep	8 = grootte (in bytes)
4 = rechten v/d rest	9 = datum en tijd
5 = aantal namen	10 = bestandsnaam

Het type van het bestand is een gewoon bestand. De eigenaar van dit bestand mag het lezen, (be)schrijven en uitvoeren (als het bv. een script is). Anderen (de groep en alle andere gebruikers van het systeem) mogen het bestand lezen en uitvoeren, maar hebben geen schrijfrechten.

Alleen de eigenaar van een bestand (of directory) en de root gebruiker hebben het recht om deze informatie te veranderen.

## 6.2 Bestanden en directories

De rechten `rwx` hebben een andere betekenis als het gaat over bestanden of directories. Voor bestanden zijn de rechten als volgt:

**read** : het bestand kan ingekijken worden

**write** : men kan een bestand veranderen en er in schrijven, het wisrecht wordt bepaald door de directory !

**execute** : het bestand kan uitgevoerd worden, enkel voor binaire bestanden en shell-scripts, hiervoor is steeds leesrecht nodig !

Bij directories daarentegen zijn de rechten als volgt:

**read** : men kan een overzicht krijgen van de bestanden in de directory  
(de ls opdracht uitvoeren)

**write** : men mag bestanden toevoegen en verwijderen in de directory  
(een bestand kan zelfs verwijderd worden zonder leesrecht)

**execute** : men mag bestanden lezen of uitvoeren in de directory  
(toegang tot de directory)

## 6.3 Rechten veranderen

De opdracht om de rechten, ook wel permissies genoemd, te veranderen is: chmod (change modus). Deze kan op twee manieren worden gebruikt:

**symbolisch** : de rechten worden symbolisch genoteerd

**numeriek** : de rechten worden numeriek genoteerd

### 6.3.1 Symbolisch

De eerste manier zorgt ervoor dat voor de user (u), group (g), others (o) of iedereen (a, all) een bepaalde permissie (Read, Write, eXecute) wordt toegevoegd (+) of afgenaomen (-). Ook kunnen de permissies aan een bepaalde permissie worden gelijkgesteld (=). Om bijvoorbeeld andere gebruikers van dezelfde group, waar je zelf inzit, permissie te geven om het text.txt bestand te lezen moet chmod g+r text.txt worden uitgevoerd.

De algemene notatie van de opdracht is:

```
chmod [ugo] \{+,-,=\} [rwx] bestandsnaam
```

### 6.3.2 Numeriek

De tweede manier bekijkt de permissie op een binaire wijze. De permissie rw- r-- r-- kan binair worden gezien als 110 100 100. Decimaal is dit 644. Om nu het text.txt bestand deze permissie te geven, kan het volgende worden ingetypt: chmod 644 text.txt.

De algemene notatie van de opdracht is:

```
chmod mode bestandsnaam
```

## 6.4 Eigenaar veranderen

Om de eigenaar van een bestand of directory te veranderen, gebruik je het bevel chown (change owner). Deze opdracht gebruik je alleen maar als je 100% zeker bent van wat je doet. De syntax is chown user bestandsnaam. Alleen de nieuwe eigenaar van het bestand of directory en de systeembeheerder (super user) kunnen je opnieuw de eigenaar maken van deze bestanden. Meestal kan enkel de superuser chown toepassen.

De groep verander je met het bevel chgrp (change group): chgrp group bestandsnaam.

## 6.5 Speciale rechten

Naast de gewone lees-, schrijf- en uitvoerrechten bestaan er nog andere rechten. Eén daarvan is het SUID-recht (of setuid). Met dit recht toegekend kan een (uitvoerbaar) bestand uitgevoerd worden met de rechten van de eigenaar. Het x-recht wordt dan in de notatie weergegeven door een s. Een bestand kan bv. de volgende rechten hebben:

```
-rwsr-xr-x 1 root root ... smbmnt
```

Met deze rechten kan het uitvoerbaar bestand smbmnt uitgevoerd worden door een gewone gebruiker met root-rechten. Hierdoor is het mogelijk om (sommige) administratieve taken door een gewone gebruiker te laten uitvoeren zonder dat hij daarvoor root-rechten moet krijgen. In verband met de veiligheid van het systeem is het aan te raden om het SUID-bit zo weinig mogelijk te gebruiken.

## 7.1 Inleiding

Er zijn zeer veel verschillende editors beschikbaar voor Linux. De juiste keuze is dan ook meestal een kwestie van smaak. De meeste beginnende Linux gebruikers zullen pico gebruiken om tekstbestanden te editeren. Pico is een editor die bij de emailclient pine hoort.

Pico is echter verre van een krachtige editor. Hij kan enkel de basisbewerkingen doen. Dit hoofdstuk bespreekt één van de krachtigste editors die er te vinden zijn namelijk *vim*<sup>1</sup>.

## 7.2 Vi/Vim

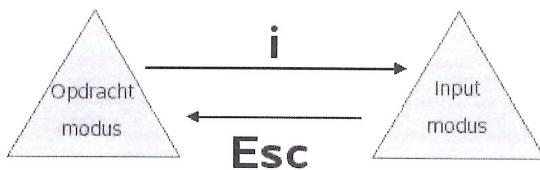
Vi was de eerste editor voor UNIX die niet meer lijn per lijn werkte, zoals ed, maar wel pagina-georiënteerd was. Dit feit, en het gebrek aan alternatieven onder UNIX gedurende de jaren 70 en 80, maakten vi tot de “standaard” editor van UNIX. Dit wil zeggen: de enige editor waarvan iedere leverancier van programma’s voor UNIX *altijd* mag verwachten dat hij aanwezig is.

Vim staat voor *Vi improved*, en is als vrije software project opgestart door de Nederlander Bram Molenaar. Vim is een uitbreiding van Vi, met meer flexibiliteit en meer opties. Eerst behandelen we de basis, namelijk Vi, waarna we verder ingaan op Vim. De tekst die je nu aan het lezen bent is trouwens geschreven met behulp van Vim!

Je kan vim opstarten en een bestand laten laden door het commando: `vim test.txt`. Dit zorgt ervoor dat het bestand `test.txt` geladen wordt, of, indien het bestand nog niet bestond, dat het gecreëerd wordt. Je sluit vim af door `:q` in te drukken (let wel, je moet hiervoor in command mode zijn, zie sectie 7.2.1). Om ervoor te zorgen dat vim de aangebrachte wijzigingen vergeet druk je `:q!`; om ze te bewaren gebruik je `:wq`.

---

<sup>1</sup><http://www.vim.org>



Figuur 7.1: Vi modes

### 7.2.1 Verschillende modes

Beginnende Vi gebruikers hebben nogal eens last van de verschillende moden waarin Vi toets-aanslagen interpreteert (figuur 7.1):

**Command mode** Deze dient voor het uitvoeren van commando's, zoals: het bestand opslaan, vi afsluiten, of een tekst inplakken. Standaard zit je in de command mode bij opstarten. En de [ESC] toets brengt vi vanuit elke mode naar de command mode.

**Insert mode** (of *input mode*, of *editeer mode*). Je gaat van command mode naar insert mode door `i` in te drukken. In deze mode kan je tekst intypen.

**Search mode** Een / vanuit command mode brengt vi naar de laatste lijn van het editeerscherms, waar je nu een zoekterm kan intypen. De [ENTER] toets is het signaal om het zoeken te beginnen.

### 7.2.2 Enkele standaard bevelen

Tabel 7.1 geeft een lijstje van vi bevelen voor de command mode. Sommige van deze bevelen brengen vi naar zijn insert mode. Vi kan werken met een heleboel "buffers" waarin je tekst editeert. Die buffers zijn zowel gehele bestanden, als tijdelijke opslagplaatsen waarmee je stukken tekst kan "knippen en plakken." Naast de bevelen uit tabel 7.1 zijn er nog vele andere; raadpleeg hiervoor de documentatie, of gebruik de on-line documentatie, via :help.

Een belangrijk aspect van deze commando's is dat men ze kan combineren, door de verschillende "codes" simpelweg na elkaar te plaatsen vooraleer op [ENTER] te drukken. Door een getal *x* voor een commando te zetten kan je dit ook *x* keer laten herhalen. Enkele voorbeelden hiervan zijn:

**d5\$** verwijdert tot op het einde van de huidige lijn en verwijdert ook de volgende vier lijnen.

**y10w** copieert de volgende 10 woorden na de cursor.

**d10w** cut de volgende 10 woorden (copieert en verwijdert).

**y\$** copieert tot het einde van de huidige lijn.

**yG** copieert tot het einde van het huidige bestand.

### 7.2.3 Extra opdrachten

Ten slotte nog enkele veel gebruikte opdrachten die de weergave beïnvloeden:

**syntax highlighting** : :syntax on

**regelnummers** : :set number

**auto insprong** : :set autoindent

Deze instellingen kunnen ook als standaard toegepast worden in het bestand .exrc.

een opgegeven patroon voorkomt. De syntax is als volgt:

```
grep [options] <PATROON> [BESTAND(EN) ...]
```

Ter illustratie, stel dat we alle lijnen die het woord “Linux” bevatten in de ( $\text{\LaTeX}$  bestanden van de) hoofdstukken “Programmeren onder Linux” en “Een device driver in Linux” willen zoeken. We zouden dan het volgende bevel intypen:

```
grep 'Linux' shellprogr.tex devicedr.tex
```

Het resultaat hiervan is:

```
devicedr.tex:\textbackslash chapter\{Een device driver in Linux\}
devicedr.tex:over de Linux kernel is te vinden op \textbackslash
url\{http://www.kernelnotes.org/\}.
devicedr.tex:Al de devices waarvan je Linux systeem gebruik kan maken staan
opgesomd in de /dev directory, zoals reeds vermeld
devicedr.tex:Er is een duidelijk verschil in aanpak wanneer we voor de kernel
moeten programmeren. Het besturingssysteem Linux kent
devicedr.tex:we er in Linux mee rekening houden dat lezen van en schrijven naar
een randapparaat dezelfde
linprogr.tex:\textbackslash chapter \{Programmeren onder Linux\}
linprogr.tex:Op zowat elk Linux systeem is er een \textbackslash
href\{http://gcc.gnu.org/\}\{GNU C compiler\} aanwezig,
linprogr.tex:omdat deze de meest gebruikte compilers zijn op Linux systemen.
linprogr.tex:\%voor het Linux platform. Daarnaast zijn er (net als op het
Microsoft platform)
```

Telkens wordt eerst de naam van het bestand afgebeeld, en daarna de regel waarin het woord “Linux” voorkomt. Men kan ook het lijnnummer mee laten afbeelden door de optie `-n` mee te geven aan grep. Raadpleeg de man-pagina’s voor alle opties.

*Opmerking:* het is altijd veiliger om quotes rond het patroon te zetten, anders probeert de shell het patroon te expanderen.

### 8.3 De mogelijke reguliere expressies met grep

Naast de voorbeelden uit het begin van dit hoofdstuk biedt grep nog heel wat andere mogelijkheden om reguliere expressies te vormen. We zetten deze hier op een rijtje met telkens enkele voorbeelden. Als voorbeeld bestand gebruiken we een Makefile.

[ ]: de character class; matcht een karakter uit een verzameling van karakters tussen [ en ], zoals reeds getoond in de inleiding van dit hoofdstuk (zie sectie 8.1). Binnen in [ en ] kan de verzameling karakters voorafgegaan worden door ^. Dit betekent dan dat deze karakters *niet* mogen gematcht worden. Zo zal **b[^A-Da-d]l** ervoor zorgen dat de letters a, b, c, d, A, B, C en D niet mogen voorkomen als letter tussen b en l. De volgende woorden zouden dus mogelijk wel matchen: *bol*, *bOl*, *b5l*, *bil* en *bwl*.

. : matcht eender welk karakter;

bevel	grep '[iI]...[xX]' Makefile
output	<pre>BIB = bibtex # for generating the BibTeX entries INDEX = makeindex # for generating the index \$(INDEX) \$(FILE) \$(INDEX) \$(FILE)</pre>

Dit voorbeeld matcht elk woord dat begin met *i* of *I*, gevolgd door *juist* 3 karakters en afgesloten door *x* of *X*. Zo matcht **k.st** ondermeer de woorden *kast*, *bst*, *kost*, *k3st*,...

\* : het voorgaande karakter zal nul, één of meer keer herhaald worden. Let op hierbij: als je \* gebruikt in de shell zal dit ge-expand worden naar alle mogelijke opeenvolgingen van karakters. Bij grep is dit anders; zo matcht **hoo\*fd** de woorden *hofd*, *hoofd*, *hooofd*, *hoooofd*, *hooooofd*,...

bevel	grep 'hoo*l' Makefile
output	<pre># de hoofdfile van de tekst # de hoofdregel van deze Makefile</pre>

bevel	grep 'ho*l' Makefile
output	

**-w** : toon enkel de regels waarin een heel woord gematcht wordt.

## 8.5 e-mailadressen ontleden en zoeken

Gegeven is een bestand (`grmail.txt`) met als inhoud een lijst met namen geassocieerd aan email-adressen (zie listing 8.1). Sommige namen worden geassocieerd met meerdere email-adressen, omdat ze meer dan één keer voorkomen in het bestand. We voeren enkele zoekacties uit in dit bestand, met behulp van grep.

Listing 8.1: `grmail.txt`; bestand met emailadressen

drs. Chris Raymaekers	chris.raymaekers@luc.ac.be
Jori Liesenborgs	jori.liesenborgs@luc.ac.be
Jori Liesenborgs	jori@lumumba.luc.ac.be
drs. Tom Van Laerhoven	tom.vanlaerhoven@luc.ac.be
drs. Tom Van Laerhoven	tom@lumumba.luc.ac.be
prof. dr. Karin Coninx	karin.coninx@luc.ac.be
prof. dr. Eddy Flerackers	eddy.flerackers@luc.ac.be
drs. Kris Luyten	kris.luyten@luc.ac.be
drs. Kris Luyten	kris@lumumba.luc.ac.be
Jo Segers	jo.segers@luc.ac.be
drs. Jan Van den Bergh	jan.vandenbergh@luc.ac.be
student endroew eldritsj	endroew.eldritsj@student.luc.ac.be
student sjef vanknutesel	sjef.vanknutesel@student.luc.ac.be
student soja boon soja	boon@luk.ac.be
student Jozef Lumumba	lumumba@congo.org

- Geef alle emailadressen in de lijst die een emailadres hebben op de server *lumumba*. Het is duidelijk dat grep 'lumumba' `grmail.txt` onvoldoende is. We krijgen dan als output:

```
Jori Liesenborgs      jori@lumumba.luc.ac.be
drs. Tom Van Laerhoven tom@lumumba.luc.ac.be
drs. Kris Luyten kris@lumumba.luc.ac.be
student Jozef Lumumba lumumba@congo.org
```

De laatste hit is geen email adres op de server *lumumba*. We kunnen grep ook laten weten dat *lumumba* moet voorafgegaan worden door een @!

grep '@lumumba' `grmail.txt` geeft ons:

## Shells en shell scripting

### 9.1 Shell

Een shell is een programma dat zorgt voor de “command line interface” (CLI) tussen de gebruiker en het besturingssysteem (d.w.z., de kernel en de GNU tools). De shell interpreteert elk bevel van de gebruiker, en voert het uit; d.w.z., het zet het bevel om in een reeks van “system calls” voor het besturingssysteem. UNIX heeft verschillende soorten shells. De belangrijkste (historisch gesproken) zijn: de Bourne shell, de Korn shell en de C shell.

**De Bourne shell** (sh) is de originele UNIX shell en is daarom beschikbaar in alle UNIX versies. Deze shell is zeer geschikt voor shell programmeren, maar is niet erg gebruiksvriendelijk.

**De C shell** (csh) is krachtiger dan de Bourne shell. Hoewel veel mensen deze shell niet zo goed vinden als de Bourne shell, wordt deze shell door veel C programmeurs gebruikt omdat de syntax sterk lijkt op de C syntax.

**De Korn shell** (ksh) combineert de voordelen van de Bourne shell en de C shell en is compatibel met de Bourne shell.

Modernere shells die op deze shells zijn gebaseerd, zijn o.a. de *Bourne again shell* (bash) en tcsh. Elke shell heeft een aantal initialisatie-bestanden. Dit wil zeggen, bestanden met shell-bevelen die altijd uitgevoerd worden wanneer de gebruiker een nieuwe shell start. Typisch zijn de namen van die initialisatie-bestanden gevormd uit de naam van de shell met daarachter “rc” (wat oorspronkelijk stond voor “resource configuration”). Zo heeft een typische Linux-installatie voor de bash shell de bestanden /etc/bash.bashrc (voor “system wide” initialisatie) en een verborgen bestand .bashrc in de home directory van elke gebruiker. Het system wide initialisatie-bestand wordt eerst uitgevoerd, en daarna de .bashrc van de gebruiker.

Dit hoofdstuk is gebaseerd op sh, bash en ksh. Vergelijkbare technieken bestaan ook voor csh en tcsh, soms met kleine verschillen o.a. in de syntax.

Listing 9.5: Toekenning aan variabelen.

```
een=1
twee='expr $een + 1'
echo $twee
datum='date'
echo $datum
```

De back tick wordt ook gebruikt om een string *uit te voeren*, d.w.z., de shell interpreteert de string als een shell bevel. Bijvoorbeeld, het bevel `info='ls'` stelt de variabele `info` niet gelijk aan "ls", maar aan de inhoud van de huidige directory.

## 9.6 Vergelijken van expressies

### 9.6.1 Rekenkundige expressies

De syntax van een rekenkundige expressie is als volgt:

```
expr operand1 math-operator operand2
```

Enkele voorbeelden:

```
expr 1 + 3      # som
expr 2 - 1      # verschil
expr 10 / 2     # gehele deling
expr 20 % 3     # rest v/d deling
expr 10 \* 3     # vermenigvuldiging
echo 'expr 6 + 3' # let op de back-quotes en spaties
```

### 9.6.2 Integer expressies

Met integer expressies kunnen we vergelijkingen maken tussen twee waarden:

- `-eq` : equal , vb. `3 -eq 5` geeft vals
- `-ge` : greather then or equal
- `-gt` : greather then
- `-le` : less then or equal
- `-lt` : less then

- -ne : not equal

Ze kunnen als volgt gebruikt worden:

```
if test 1 -gt 0
```

### 9.6.3 String expressies

Deze expressies laten ons toe tekst (strings) te vergelijken:

- = : equal , vb str1 = str2
- != : not equal
- str1 : is niet null
- -n str1 : lengte van str1 > 0
- -z str1 : lengte van str1 = 0

### 9.6.4 Bestandsexpressies

Hiermee kunnen we nagaan wat het type van een bestand is en wat de rechten zijn:

- -d bestandsnaam : directory ?
- -f bestandsnaam : gewoon bestand ?
- -r bestandsnaam : leesbaar ?
- -s bestandsnaam : lengte > 0 ?
- -w bestandsnaam : beschrijfbaar ?
- -x bestandsnaam : uitvoerbaar ?
- -e bestandsnaam : bestaat ?

### 9.6.5 Logische expressies

Ten slotte zijn er nog de logische expressies:

- !expr1 : negatie van expr1
- expr1 -a expr2 : expr1 en expr2
- expr1 -o expr2 : expr1 of expr2

## 9.7 Conditionele expressies

De Bourne shell kent twee conditionele expressies: *if* en *case*. De syntax van de *if* expressie is:

Listing 9.6: Syntax van de if expressie.

```
if [ expressie ]
then
    statements
elif [ expressie ]
then
    statements
else
    statements
fi
```

Een voorbeeldje van het gebruik van een *if*-statement in een shell script vind je in listing 9.7.

Listing 9.7: Voorbeeld van een if expressie.

```
if [ $var = "Yes" ]
then
    echo "Value is Yes"
elif [ $var = "No" ]
then
    echo "Value is No"
else
    echo "Invalid value"
fi
```

De *case* expressie is vergelijkbaar met het *switch* bevel van C. De syntax is beschreven in listing 9.8.

Listing 9.8: Syntax van de case expressie.

```
case word in
    str1)
        statements;;
    str2 | str3)
        statements;;
    *)
        statements;;
esac
```

Een voorbeeldje van het gebruik van een case-expressie in een shell script vind je in listing 9.9.

Listing 9.9: Voorbeeld van een case expressie.

```
case $1 in
  01 | 1) echo "Month is January";;
  02 | 2) echo "Month is February";;
  03 | 3) echo "Month is March";;
  04 | 4) echo "Month is April";;
  05 | 5) echo "Month is May";;
  06 | 6) echo "Month is June";;
  07 | 7) echo "Month is July";;
  08 | 8) echo "Month is August";
  09 | 9) echo "Month is September";;
 10) echo "Month is October";;
 11) echo "Month is November";;
 12) echo "Month is December";;
 *) echo "Invalid Parameter";;
esac
```

### Voorbeeld: polariteit testen

Het script in listing 9.10 test een meegegeven argument op zijn polariteit.

Listing 9.10: Polariteit testen

```
#!/bin/sh
#
# Script om te testen of een argument positief is
#
if test $1 -gt 0
then
  echo "$1 nummer is positief"
fi
```

Een uitgebreidere versie die ook test of er een argument is meegegeven zie je in listing 9.11.

Listing 9.11: Argument en polariteit testen

```
#!/bin/sh
#
# Script to see whether argument is positive or negative
#
```

```

if [ $# -eq 0 ]
then
    echo "$0 : Je moet 1 getal ingeven!"
    exit 1
fi
if test $1 -gt 0
then
    echo "$1 nummer is positief"
else
    echo "$1 nummer is negatief"
fi

```

### Voorbeeld: keuze besturingssysteem

Een voorbeeld van een geneste if-structuur zie je in listing 9.12.

Listing 9.12: Keuze besturingssysteem

```

osch=0

echo "1. Unix"
echo "2. Linux"
echo -n "Kies je besturingssysteem [1 of 2]: "
read osch

if [ $osch -eq 1 ] ; then
    echo "Je koos Unix"

else ##### nested if i.e. if within if #####
    if [ $osch -eq 2 ] ; then
        echo "Je koos Linux"
    else
        echo "Je koos niet voor Unix/Linux."
    fi
fi

```

## 9.8 Iteraties

Een iteratie in een shell script kan op drie verschillende manieren worden geïmplementeerd: for, while en until.

### 9.8.1 for

De syntax van de for expressie is:

Listing 9.13: Syntax van de for expressie.

```
for var in list
do
    statements
done
```

Een voorbeeld van het gebruik van een for expressie in een shell script vind je in listing 9.14. Het script in listing 9.14 schrijft de getallen 1 t.e.m. 5 tweemaal op het scherm.

Listing 9.14: Voorbeeld 1 van een for expressie.

```
#!/bin/sh
for i in 1 2 3 4 5
do
    echo "Eerste $i"
done

for (( i = 1 ; i <= 5; i++ ))
do
    echo "Tweede $i"
done
```

Een ander voorbeeld vind je in listing 9.15. Het script schrijft de huidige directory uit.

Listing 9.15: Voorbeeld 2 van een for expressie.

```
#!/bin/sh
dirlist='ls'
for file in $dirlist
do
    echo $file
done
```

Stel dat je in een directory heel wat tar.gz files staan hebt die je allemaal wil decomprimeren en uitpakken, dan kan je daarvoor een soortgelijk script schrijven. Een voorbeeldje hiervan zie je in listing 9.16.

Listing 9.16: untarren en gunzippen m.b.v. een for lus

```
#!/bin/sh
for file in `ls *tar.gz`
do
    tar -xvzf $file
done
```

Merk op dat ‘ls \*tar.gz’ een verzameling bestanden teruggeeft waarover geïtereerd wordt. Je kan in plaats van het ls bevel uit te voeren, het te expanden patroon meegeven: for file in \*tar.gz. De shell zal dit dan eerst expanden tot alle tar.gz bestanden in de huidige directory. In feite hoeft dit helemaal niet in een bestand gezet te worden. We zouden ook het

script kunnen intypen achter de shell prompt, waarbij we de enters vervangen door ;. Zo zou je in plaats van listing 9.16 in een bestand te plaatsen ook kunnen intypen:

```
for file in `ls *tar.gz`; do tar -xvzf $file ; done
```

Dit heeft hetzelfde effect.

### Voorbeeld: maaltafel

Het voorbeeld in listing 9.17 zal van een meegegeven argument de maaltafel op het scherm afdrukken.

Listing 9.17: Maaltafel afdrukken

```
#!/bin/sh
#
if [ $# -eq 0 ]
then
    echo "Fout - geen getal ingevuld"
    echo "Syntax : $0 getal"
    echo "Gebruik om de maaltafel af te drukken"
    exit 1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
    echo "$n * $i = `expr $i \* $n`"
done
```

### Voorbeeld: geneste for-lus

Het voorbeeld in listing 9.18 zal een matrix op het scherm afdrukken.

Listing 9.18: Geneste for-lus

```
for (( i = 1; i <= 5; i++ ))      ##### buitenste for-lus #####
do

    for (( j = 1 ; j <= 5; j++ )) ##### binneste for-lus #####
    do
        echo -n "$i "
    done
    echo "" ##### nieuwe lijn afdrukken #####
done
```

### 9.8.2 while

De syntax van de while expressie is:

Listing 9.19: Syntax van de while expressie.

```
while expressie
do
    statements
done
```

Een voorbeeldje van het gebruik van een while expressie in een shell script vind je in listing 9.20.

Listing 9.20: Voorbeeld van een while expressie.

```
#!/bin/sh
i=1
while [ $i -le 10 ]
do
    echo $i
    i='expr $i + 1'
done
```

Het script in listing 9.20 telt tot 10 en schrijft de getallen uit.

### 9.8.3 until

Tenslotte is er nog de until expressie waarvan de syntax beschreven wordt in listing 9.21. De until expressie voert de code tussen do en done uit tot de conditie die achter until staat voldaan is. Als deze conditie voldaan is voordat delus voor de eerste keer uitgevoerd is, wordt de lus gewoon niet uitgevoerd.

Listing 9.21: Syntax van de until expressie.

```
until expressie
do
    statements
done
```

Een voorbeeldje van het gebruik van een until expressie in een shell script vind je in listing 9.22.

Listing 9.22: Voorbeeld van een until expressie.

```
#!/bin/sh
i=1
until [ $i -gt 10 ]
do
    echo $i
    i='expr $i + 1'
done
```

Ook het voorbeeld uit listing 9.22 telt tot 10 en schrijft de getallen uit.

## 9.9 Functies

Scripts voor de Bourne shell kunnen ook functies bevatten. De syntax is voorgesteld in listing 9.23.

Listing 9.23: Syntax voor script functies.

```
functienaam () {
    statements
}
```

De functie kan worden opgeroepen met de functienaam. Ook kunnen er parameters worden doorgegeven: functienaam param1 param2 .... Deze parameters kunnen dan in de functie worden aangeroepen door middel van de positionele parameters (\$1, \$2, ...). Een voorbeeldje van een functie die argument1 argument2-keer met zichzelf optelt (zoals een gewone vermenigvuldiging dus) vind je in listing 9.24. Merk op dat de functie voor de aanroep moet gedefinieerd worden.

Listing 9.24: Voorbeeld van een script functie.

```
#!/bin/sh
telkeerop(){
    result=$1
    teller=$2
    until [ $teller -eq 1 ]
    do
        result='expr $result + $1'
        teller='expr $teller - 1'
    done
    echo $result
}

telkeerop 5 7
```

## 9.10 Scripts onderbreken

Een iteratie (for, while en until) kan worden onderbroken door middel van het bevel `break`. Het hele script kan beëindigd worden met het bevel `exit`.

Een laatste bevel is `shift`. Hiermee worden de positionele parameters één plaats opgeschoven. Een voorbeeld hiervan vind je in listing 9.25. Dit voorbeeld schrijft alle parameters uit. De parameters kunnen ook meerdere plaatsen (`n`) plaatsen worden opgeschoven door `shift n`.

Listing 9.25: Voorbeeld van een shift opdracht.

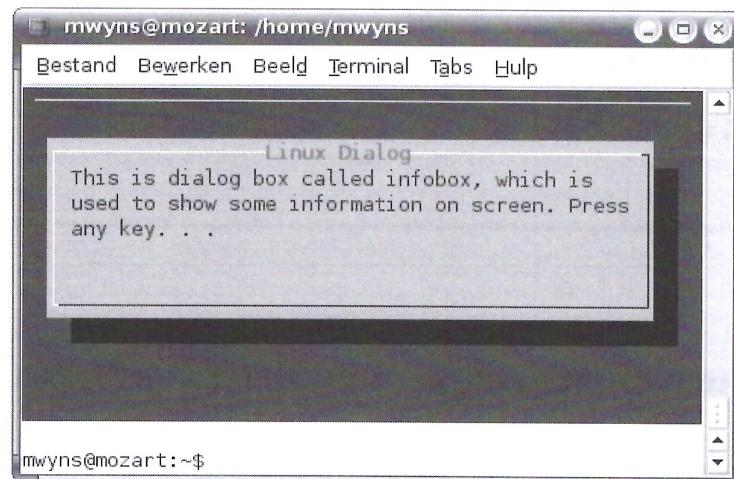
```
#!/bin/sh
while [ $# -ne 0 ]
do
    echo $1
    shift
done
```

## 9.11 Dialogen

Met de tool `dialog` kunnen we snel gebruikersinterfaces maken.

### 9.11.1 Infobox

```
dialog --title "Linux Dialog" --backtitle "Shell Script" --
infobox "This is dialog box called infobox, which is used
to show some information on screen. Press any key. . ." 7
      50
read
```



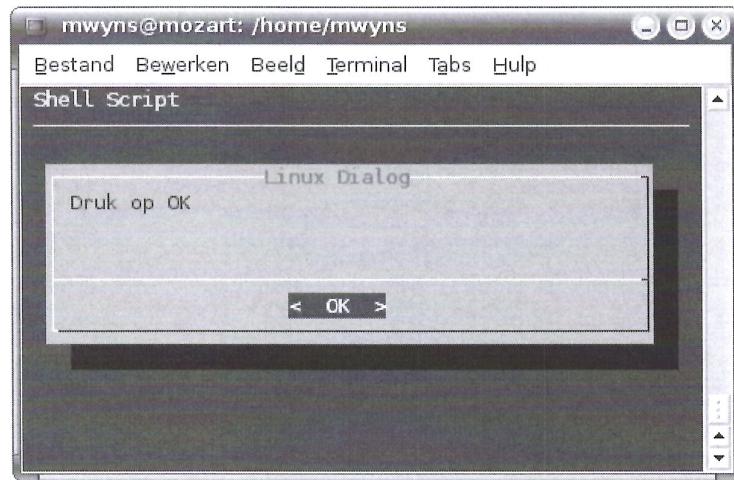
Figuur 9.1: Infobox.

### 9.11.2 Messagebox

```
dialog --title "Linux Dialog" --backtitle "Shell Script" --
msgbox "Druk op OK" 7 50
```

### 9.11.3 Ja/Nee box

```
dialog --title "Bestand wissen ?" --backtitle "Shell Script" --
yesno "\nHet bestand '/usr/letters/jobapplication' wissen ?" 7
      60
sel=$?
case $sel in
  0) echo "Wissen geselecteerd";;
  1) echo "Niet wissen geselecteerd";;
  255) echo "Geannuleerd met [ESC] toets";;
esac
```



Figuur 9.2: Messagebox.

### 9.11.4 Inputbox

```
dialog --title "Inputbox" --backtitle "Shell Script" --inputbox "
Uw naam AUB:" 8 60 2> input.txt

sel=$?
na='cat input.txt'
case $sel in
  0) echo "Hallo $na" ;;
  1) echo "Geannuleerd" ;;
  255) echo "[ESCAPE] toets ingedrukt" ;;
esac

rm -f input.txt
```

### 9.11.5 Menu's

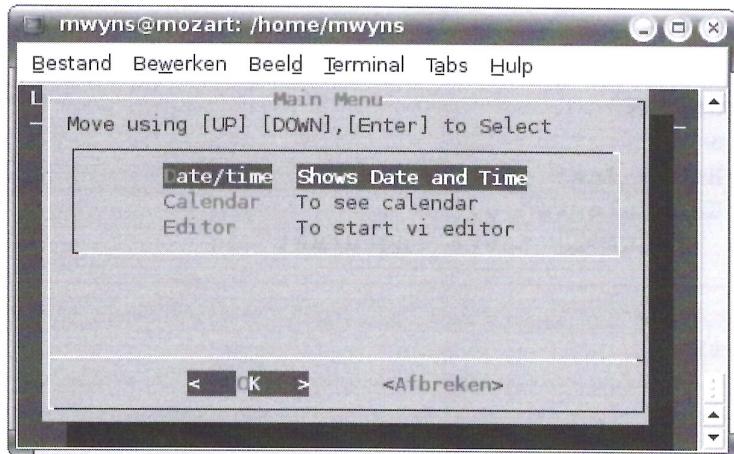
```
dialog --backtitle "Linux Shell Script Tutorial" --title "Main
Menu" --menu "Move using [UP] [DOWN],[Enter] to Select" 15 50
3 Date/time "Shows Date and Time" Calendar "To see calendar"
Editor "To start vi editor" 2> menu.txt

menuitem='cat menu.txt'
case $menuitem in
  Date/time) date;;
  Calendar) cal;;
```



Figuur 9.3: Ja/Nee box.

```
Editor) vi;;  
esac
```



Figuur 9.4: Menu's.

Jori Liesenborgs jori@lumumba.luc.ac.be  
drs. Tom Van Laerhoven tom@lumumba.luc.ac.be  
drs. Kris Luyten kris@lumumba.luc.ac.be

Dit is wat we zochten!

- Verdergaand op de vorige vraag: we willen zoeken welke mensen een emailadres hebben op de server lumumba en aan het doctoreren zijn (*drs.*). We kunnen dit op twee manieren doen:

- De output van de vorige oplossing pipen naar een nieuwe grep:

```
grep '@lumumba' grmail.txt | grep 'drs\.'
```

- Het grep bevel uitvoeren met een wat moeilijkere expressie:

```
grep 'drs\..*@lumumba' grmail.txt
```

Let op de `.*`: deze betekent dat een willekeurige reeks karakters kan gematcht worden.

- Tel de lege lijnen in de file met behulp van grep. Een lege lijn is een lijn zonder inhoud: er komt niets voor tussen het begin van de lijn (^) en het einde van de lijn (\$). Met behulp van het patroon `^$` kunnen we dus lege lijnen vinden. Het aantal lijnen dat je vindt met behulp van grep kan je laten zien door de optie `-c` te gebruiken.

```
grep -c "^$" grmail.txt geeft als output: 2
```



# 10

## Gebruikersbeheer

### 10.1 Inleiding

Een gebruiker of user is niet altijd een individuele persoon, het is een entiteit dat programma's kan uitvoeren of bestanden bezitten. Dit kunnen dus computersystemen zijn die automatisch opdrachten uitvoeren of groepen van personen met een gelijklopende functie.

Een gebruiker identificeert zichzelf aan de hand van een gebruikersnaam (=login). Bij het toekennen van een nieuwe gebruiker wordt een unieke UID toegekend en wordt daarbij ook tot een bepaalde groep ingedeeld. Een groep kan bijvoorbeeld zijn: mensen die aan een bepaald project werken. Ook de groepen hebben een uniek nummer: GID. De UID en GID samen bepalen de rechten van een gebruiker.

De gegevens van een gebruiker worden bewaard in het paswoordbestand (/etc/passwd), de paswoorden zijn gecodeerd. De groepsgegevens kan men terugvinden in het bestand /etc/group. Beide bestanden zijn publiek maar kunnen enkel door de beheerder veranderd worden. Paswoordbestanden worden soms beschouwd als een veiligheidslek en daarom worden meestal shadow-paswoordbestanden gebruikt (zie verder).

### 10.2 Manueel nieuwe gebruikers toevoegen

Om een nieuwe gebruiker toe te voegen moeten er verscheidene stappen ondernomen worden:

- het toekennen van: username, ID-nummer en een primaire groep (eventueel andere)
- de data ingeven in paswoord- en groepsbestand
- een paswoord toekennen
- andere parameters instellen vb. vervaldag, privileges, ...

- een home-directory aanmaken
- de initialisatiebestanden plaatsen in de gebruiksdirectory
- de gebruiker toegang geven tot zijn home-directory en initialisatiebestanden
- andere faciliteiten toekennen, vb: mail, printer
- andere initialisatietaken
- het uittesten van de nieuwe gebruiker

Tegenwoordig gebeuren al deze stappen automatisch maar het is belangrijk de verschillende stappen te kennen en te begrijpen die gebeuren bij een automatisch proces.

### 10.2.1 Nieuwe accounts definiëren

Dit gebeurt door het paswoordbestand te editeren. Het is aan te raden een kopie te maken alvorens dit te doen.

Een voorbeeld van een typische lijn in /etc/passwd:

```
mwyns:x:503:503:Mario Wyns:/home/mwyns:/bin/csh
```

De opbouw van het paswoordbestand ziet er als volgt uit:

**mwyns** : gebruikersnaam (login), publiek. Bij hoge veiligheid is de naam opgebouwd uit willekeurige strings.

**x** : geïncrypteerd paswoord. Als de account niet gebruikt wordt om in te loggen vult men een sterretje in. Indien het veld blanco gelaten wordt dan is er geen paswoord nodig om in te loggen, dit verlaagt aanzienlijk de veiligheid van het systeem. Een x verwijst naar /etc/shadow (zie shadow). Bij het definiëren van een account is het gebruikelijk van een sterretje in te vullen en daarna het paswoord te veranderen met de opdracht passwd.

**503** : user identification number. Dit nummer is belangrijk voor het bijhouden van activiteiten bij bv. identieke accounts. Om een overzicht te hebben worden de nummers gerangschikt in het paswoordbestand:

- kleiner dan 100: systeem accounts
- 200-250: scheikunde
- 251- 299: fysica

**503** : group identification number. Dit is de primaire groep van een gebruiker en geeft de gebruiker toegang tot de bestanden van die groep.

- kleiner dan 10 of 20: systeem groepen
- 50 tot ...

**Mario Wyns** : volledige naam van de gebruiker. Dit wordt gebruikt door o.a. mailsystemen. Andere gegevens kunnen toegevoegd worden (vb. telefoonnummer) maar denk hierbij aan de veiligheid.

**/home/mwyns** : bij het inloggen komt de gebruiker terecht in deze directory.

**/bin/csh** : shell dat het systeem zal gebruiken voor die gebruiker. Vb: Bash, C-shell, Korn-shell.

Nieuwere systemen raden het manueel aanpassen van het paswoordbestand af, gebruik in plaats de toegekende opdrachten.

### 10.2.2 Paswoorden toekennen

Elke gebruiker heeft een paswoord behalve wanneer het gaat over volledig geïsoleerde systemen. Zoals reeds vermeld kan het paswoord veranderd of toegekend worden met de opdracht `passwd`. Om de veiligheid te verhogen moet het paswoord aan het volgende basisprincipe voldoen: het paswoord moet gemakkelijk kunnen onthouden worden maar moeilijk te raden of te kraken zijn. Gebruik dus geen moeilijke willekeurige paswoorden, anders bestaat de kans dat men het opschrijft. Verder moet het volgende vermeden worden:

- Gedeeltes van eigen naam of familie.
- Nummers : telefoon, geboortedatum,...
- Namen van dingen die verwantschap houden: artiest, film, plaats, sportclub,...
- Elke verwantschap met het bedrijf: nummers, producten, personen,...

Ook kraakprogramma's moet men te slim af zijn, vermijd dus het volgende:

- Correct gespelde woorden te vinden in een woordenboek (kan men controleren met specifiek commando).
- Bekende personen.
- Gepubliceerde paswoordvoorbereelden.

- ...

De keuze van de paswoorden zijn dus zeer belangrijk: één slecht paswoord kan een toegang zijn tot het volledige systeem voor eventuele krakers.

Enkele goede keuzes bestaan uit al dan niet een combinatie van de volgende richtlijnen:

- Toevoegen van speciale karakters aan gewone woorden.
- Verkeerd spellen van gewone woorden.
- Hoofdlettergebruik op niet gewone plaatsen.
- Samenstelling van (gedeeltelijke) woorden of het tussenvoegen ervan.
- Tussen elkaar weven.

Slecht	Goed	Nog beter
StarTrek	sTarteRk	\$TarkEr^T
stArtrEk	stIrTruk	st^IrTre#
StarDrek	JeTrekdi	Yetr^Ekd^I
trekstar	sttraerk	st^TraeRk

Bij het kiezen van verschillende paswoorden mag men geen herkenbaar patroon gebruiken. Nieuwere systemen verplichten hierbij het gebruik van enkele regels.

Nog enkele algemene richtlijnen:

- Gebruik geen onbeschermd accounts.
- Een minimum paswoordlengte verplichten (indien mogelijk). Een goede keuze is 8 karakters of beter 10 tot 12.
- De paswoorden moeten veranderd worden in volgende gevallen:
  - Als iemand anders het paswoord begint te kennen (meekijken over schouder, ...).
  - Als de gebruiker het bedrijf verlaat.
  - Als de beheerder weggaat: het root-paswoord. Verander eventueel ook alle andere gebruikers.
  - Bij een aanval en het niet gebruiken van een schadow-paswoordbestand: verander alle gebruikersaccounts.
  - Bij een aanval op de beheerder: alle paswoorden.

- Als het paswoordbestand ingekken is via het netwerk of inbellijn.
- Root-paswoorden periodisch veranderen, in alle gevallen.
- Gebruikerspaswoorden enkel veranderen indien toegang tot buitenwereld. Het onnodig verplichten leidt tot de keuze van te gemakkelijke paswoorden.
- Gebruikers inlichten over een goede keuze.

### Shadow-paswoordbestanden

Het gebruik van een shadow-paswoordbestand (/etc/shadow) beschermt de paswoorden tegen niet root gebruikers. Dit bestand dat niet leesbaar is door iedereen bevat de geïncrypteerde paswoorden, het gewone paswoordbestand (/etc/passwd) dat wel leesbaar is door iedereen bevat enkel een verwijzing hiernaar door middel van een "x". Dit gebruik is nodig omdat op de meeste systemen het paswoordbestand toegankelijk moet zijn voor de gebruikers. Men kan eenvoudig nagaan of er een shadow-paswoordbestand gebruikt wordt, alle paswoorden in /etc/passwd staan dan op "x".

Een typische lijn in /etc/shadow ziet er als volgt uit:

```
student:10F6LE0db$1RQEs5teaNnyFvgHvEWL00:12698:0:99999:7:::
```

**student** : gebruikersnaam

**10F6LE...** : geïncrypteerd paswoord

**12698** : aantal dagen sedert 01/01/1970 dat het paswoord niet veranderd is

**0** : aantal dagen voordat paswoord **kan** veranderd worden (min. lifetime)

**99999** : aantal dagen voordat paswoord **moet** veranderd worden (max. lifetime)

**7** : aantal dagen voordat waarschuwing gegeven wordt

### Permissies van het paswoordbestand

De eigenaar van het paswoordbestand is de root (UID 0) en systeemgroep (GID 0). Op nieuwe systemen heeft de eigenaar meestal geen rechtstreekse schrijftoegang, er moeten dan bijgeleverde tools gebruikt worden.

### Restricties op paswoorden

Dit houdt bijvoorbeeld in de gebruikers te verplichten hun paswoord te veranderen om luiheid tegen te gaan. Dit kan aan de hand van "Password aging". Men moet de restricties voorzichtig instellen om geen averechts effect te creëren.

- max. lifetime: levensduur van een paswoord
  - niet te kort instellen om geen averechts effect te creëren
- min. lifetime: tijd dat een paswoord niet mag veranderd worden
  - wordt gebruikt om al te vlug overschakelen naar oud paswoord tegen te gaan

Dit instellen op verschillende systemen wordt hier niet in detail besproken.

### Paswoorden achterhalen

Met een tool zoals bv. nutcracker is het mogelijk om paswoorden te achterhalen. De tool gebruikt een woordenboekbestand en zal één voor één de geëncrypteerde versies eruit vergelijken met de records in het paswoordbestand. Dit is zeer tijdsrovend bij een uitgebreide lijst.

user name	status	password
root	unable to crack	X
daemon	disabled	-
mwyns	unable to crack	X
sshd	disabled	-
student	CRACKED	student

### 10.2.3 Een home-directory aanmaken

Dit gebeurt eenvoudig met:

```
# mkdir /home/chavez
```

### 10.2.4 Login initialisatiebestanden

Men moet de gebruiker voorzien van verschillende initialisatiebestanden zoals .profile bij het gebruik van een Bourne shell of .login en .cshrc bij een C shell. Deze bestanden moeten in de home-directory staan. Ze worden uitgevoerd bij het inloggen of als er een nieuwe shell

opgeroepen wordt. Normaal maakt de beheerder standaardbestanden aan en bewaart die op een standaardlocatie, bv.: /home/root. Bij het creëren van een nieuwe home-directory kunnen deze dan daar naar toe gekopieerd worden. Daarna kan de gebruiker ze dan aanpassen naar eigen behoeftte.

Andere initialisatiebestanden zijn er voor bijvoorbeeld mail- en editorprogramma's. Bij het uitloggen wordt .logout uitgevoerd. Wanneer X-windows gebruikt wordt zijn er ook daarvoor initialisatiebestanden.

Als voorbeeld wordt het doel van .login en .profile aangehaald:

- zoekpad zetten.
- default bestandsbescherming instellen (umask).
- terminal type instellen en bepalen.
- andere omgevingsvariabelen.
- andere functies.

Voorbeeld van een .profile bestand:

```
...
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi
...
...
```

### Systeemomvattende initialisatiebestanden

Sommige initialisatiebestanden worden uitgevoerd alvorens dat de persoonlijke initialisatiebestanden uitgevoerd worden. Zo wordt /etc/profile uitgevoerd voor de persoonlijke .profile.

### De directory /etc/skel

skel verwijst naar *skelleton* of *geraamte*. In deze directory staan voorbeeld initialisatiebestanden die gebruikt worden bij het aanmaken van een nieuwe account. Wijzigingen die voor nieuw aan te maken gebruikers gelden kunnen daarin doorgevoerd worden, deze hebben geen effect op reeds bestaande gebruikers.

### 10.2.5 Bestandspermissies instellen

Na het kopiëren van de initialisatiebestanden kan men de permissies instellen met de opdracht chown. Een voorbeeld ziet er als volgt uit:

```
# chown -R student:student /home/student
```

### 10.2.6 Een shell toekennen

Het laatste veld in het paswoordbestand specificeert de gebruikte shell van de gebruiker. Een overzicht van de mogelijke shells vindt men terug in het bestand /etc/shells. Opgelet met de root account, een verandering van de root shell kan problemen opleveren.

### 10.2.7 De gebruiker toevoegen aan andere systeemfaciliteiten

Enkele faciliteiten kunnen zijn:

- veiligheidsfaciliteiten:
  - harddiskruimte.
  - mail alias.
  - printertoegang.
- toevoegen aan andere groepen in het bestand /etc/group.
- andere taken, bijvoorbeeld voor specifieke software.

### 10.2.8 Uittesten van de nieuwe account

Het minimum dat men kan doen om de account te testen is inloggen als de nieuwe gebruiker. Zo wordt reeds de gebruikersnaam, paswoord en home-directory uitgetest. Verder kan men de initialisatiebestanden uittesten. Men kan de omgevingsvariabelen bekijken en aliases uittesten. Gebruik de opdracht su (= substitute user) om de gebruikersomgeving te hercreëren. Deze

opdracht samen met een andere account als optie laat toe het inloggen onder die andere gebruiker te simuleren. In het voorbeeld wordt men de gebruiker harvey:

```
# su - harvey
```

## 10.3 Automatisch gebruikers toevoegen

De meest gebruikte opdracht is van de vorm adduser. Deze is echter systeemafhankelijk met vele opties. Een overzicht van de verschillende opdrachten voor de verschillende systemen:

- adduser
- useradd
- admintool
- mkuser
- cpcople

Het voordeel van deze automatische opdrachten is dat men de opeenvolgende stappen niet meer moet onthouden. Deze tools zijn pas echt goed na aanpassing, daarna is het bijmaken van accounts zeer eenvoudig. Een nadeel is dan weer dat na de basisstappen de tool er mee ophoudt. Een toevoeging aan bv. een mail-systeem zit er meestal niet in. Voor een volledige automatisatie naar wensen kan men de tool van de verkoper aanpassen of hem zelf schrijven.

## 10.4 Accounts inperken

Wanneer een gebruiker slechts één programma hoeft te gebruiken kan men bijvoorbeeld het programma automatisch opstarten bij het inloggen en bij het verlaten van het programma automatisch uitloggen. Niet alle programma's laten dit echter toe. In dat geval kan men een beperkte (restricted) shell gebruiken (rsh). Gebruikers kunnen dan bijvoorbeeld niet:

- cd (change directory) gebruiken.
- PATH variabelen instellen.
- opdrachten uitvoeren met een / (slash) erin.
- output redirection gebruiken (> of >>).

Om een gebruiker zijn rechten in te perken moet men zeer nauwkeurig te werk gaan. Via bijvoorbeeld een mail-programma kan een gebruiker een editor opstarten en zo in een shell terechtkomen via een shell-escape.

## 10.5 Accounts verwijderen of buiten gebruik stellen

Wanneer een persoon het bedrijf verlaat is het raadzaam de account af te sluiten, zeker bij een vertrek in minder aangename omstandigheden. Dan moet het zelfs zeer vlug gebeuren. Het buiten gebruik stellen kan eenvoudig door een sterretje toe te voegen in het paswoordbestand op de plaats van het paswoord. Men kan ook een tool gebruiken meegeleverd met het systeem om de account af te sluiten. Een account enkel afsluiten en niet verwijderen kan problemen voorkomen met sommige bestandspermissies. Pas wanneer men zeker is dat de account niet meer nodig is kan men hem volledig verwijderen. Sommige veiligheidsvoorschriften raden dit echter af, zo kan de UID niet opnieuw gebruikt worden.

### 10.5.1 Gebruikersaccounts verwijderen

Meestal is ook daarvoor een tool beschikbaar, bv. userdel, rmuser, removeuser, .... De home-directory wordt meestal behouden.

### 10.5.2 Aanverwante zaken

- andere paswoorden veranderen, bv. het dialin-paswoord.
- processen die werden opgestart door de gebruiker beëindigen.
- gebruiker verwijderen uit secundaire groepen.
- de gebruiker zijn mail-bestanden en eventuele aliasses wissen.
- kijken of de gebruiker geen cron of at jobs achtergelaten heeft.
- eventuele printopdrachten verwijderen.
- backup maken van de home-directory en nadien wissen.

## 10.6 Groepen

Een groep van gebruikers heeft als doel bestanden en andere bronnen te delen. Men kan een groep definiëren op twee manieren:

**implicit** : elke nieuwe voorkomende GID in het paswoordbestand is een nieuwe groep.

**expliciet** : men kan ook de naam en de GID van de groep ingegeven in /etc/group.

### 10.6.1 Het /etc/group bestand

De inhoud kan er als volgt uitzien:

```
group name::GID:additional users
```

Waarbij:

- group-name: de naam van de groep is.
- GID: het groepsidentificatienummer.
- additional-users: additionele gebruikers en groepen die ook toegang tot de groep hebben.

### 10.6.2 Lidmaatschap in secundaire groepen

Meestal is er geen onderscheid tussen primaire en secundaire groepen. De verschillende lidmaatschappen kan men nagaan met de opdracht groups. Om te veranderen van actieve groep na het inloggen gebruikt men newgrp en om de actieve groep na te gaan id.

## 10.7 Standaard gebruikers en groepen

### 10.7.1 Standaard gebruikers

- root: gebruiker 0 of super user, deze gebruiker heeft onbeperkte toegang tot het systeem. Opgelet, elke gebruiker met UID 0 heeft root toegang.
- daemon: gebruikt voor systeemprocessen.
- bin: eigenaar van uitvoerbare bestanden.
- sys: eigenaar van systeembestanden.
- cron, mail, news, usenet, lp of lpd: eigenaars van de respectievelijke bestanden en processen.
- nobody: gebruikt door NFS.
- www-data: gebruikt door de webserver.

Op de root gebruiker na wordt de rest meestal niet gebruikt om in te loggen. De verschillende Linux distributies kunnen additionele gebruikers toevoegd hebben.



# 11

## Kernel

### 11.1 Taken van de kernel

De Linux kernel is het eigenlijke besturingssysteem. De kernel regelt de interactie tussen de soft- en hardware. De kernel verdeelt het geheugen en processortijd tussen de programma's. Stuurbestanden (device drivers) in de kernel regelen de toegang tot hardware componenten. Zonder kernel kunt u uw computerhardware niet met uw software besturen.

U kunt onder Linux stuurbestanden in de vorm van kernelmodulen on the fly laden en verwijderen. Als een als module gecompileerd stuurbestand aangeroepen wordt, laadt de kernel het stuurbestand in het geheugen. Als het stuurbestand niet meer gebruikt wordt verdwijnt het uit het geheugen.

### 11.2 Ontwikkeling

De Linux kernel ([kernel.org](http://kernel.org)) wordt via het internet ontwikkeld. Het werk wordt in taakgroepen verricht. Linux is in C+ geschreven. C+ is een minder gecompliceerde taal dan C++, wat het compileren versnelt en het debuggen vergemakkelijkt. Verbeteringen van de officiële kernel worden aan Linus Torvalds doorgegeven. Als ze goed genoeg zijn komen ze in de volgende stabiele kernel. Geheel nieuwe onderdelen worden aan een development kernel toegevoegd.

Linus Torvalds heeft een soort vetorecht over de officiële kernel. Hij verricht vooral coördinerend werk en wordt hierbij ondersteund door andere mensen.

### 11.3 Kernelversies

Kernel versies worden aangegeven door een major, minor en patchlevel nummer. Een voorbeeld is versie 3.2.16. Major numbers (de 3) veranderen zelden. Minor versions (3.0, 3.2) veranderen

om de enkele maanden, patchlevels van in ontwikkeling zijnde kernels (development releases) soms dagelijks. Het laatste nummer (de 16 in 3.2.16) is het patchlevel.

## 11.4 Kernel informatie

Wat flitste er over het scherm direct na de Linux boot? Het zijn de berichten van de kernel. Ze worden onderschept en opgeslagen door klogd, de kernel log daemon. Direct na het booten (en voor de login) zijn ze zichtbaar te maken met Shift-PageUp. Behalve de kernel berichten laat Shift-PageUp u ook de resultaten van de deamons uit de init scripten zien.

Na de login zijn ze zichtbaar te maken met het opdracht dmesg:

```
dmesg | less (tekst op het scherm tonen met less)  
dmesg > dmesg.txt (naar bestand dmesg.txt in de huidige dir)
```

Het is zeker de moeite waard om deze teksten eens te bestuderen. Het geeft veel informatie over het besturingssysteem en de hardware die u draait.

Een andere bron van (dynamische) systeeminformatie is te vinden in de (pseudo) directory /proc:

```
mwyns@bach:~$ cat /proc/version  
Linux version 3.2.16 (root@zombie) (gcc version 4.6.3 (Debian ...
```

cat /proc/version geeft de Linux kernel versie en de hiervoor gebruikte c-compiler weer.

## 11.5 Een Linux kernel compileren

Onder Linux is het vrij gebruikelijk om een zelf kernel aan te maken. Het programmeerwerk is al voor u gedaan. Wat u in feite doet is de samenstelling van de kernel bepalen: de ondersteuning voor hardware, netwerk en bestandssystemen. U stelt dus het menu samen, daarna geeft u de kok (de GNU c compiler) de opdracht het eten te klaar te maken. Of het ook opgediend wordt, of dat alles bij het oude blijft, bepaalt u wederom zelf.

Dat is natuurlijk alleen mogelijk als u precies weet wat u wilt en u uw hardware kent.

## 11.6 Een kernel op maat

Goede redenen om voor uzelf een nieuwe kernel te compileren zijn:

- U wilt een hogere kernelversie i.v.m. nieuwe of verbeterde eigenschappen
- U wilt een kernel op maat i.p.v. de door uw distributie geleverde standaard kernel.
- U wilt eens wat met uw hardware experimenteren.

Een kernel op maat is een aan uw behoeften aangepaste kernel: onnodige stuurbestanden laat u weg, zelden gebruikte stuurbestanden kunt u als laadbare modulen compileren. Door onnodige stuurbestanden weg te laten bespaart u geheugen en laat u bronnen van potentieel negatieve interacties weg.

Uw hardware en de door u benodigde stuurbestanden bepalen welke opties u in het kernel configuratie menu plaatst: bestandssystemen, netwerkopties, geluidskaart. Alles wat maar door Linux ondersteund wordt kunt u zo instellen.

Met een kernel optimalisatie voor de processor kan bijv. al een behoorlijke snelheidswinst te behalen zijn. Anderzijds zijn er ook experimentele kernelopties voor bijvoorbeeld nieuwe hardware.

Als u geen netwerk gebruikt, kunt de netwerk ondersteuning uit de kernel laten. Als u de computer later op een netwerk aansluit, kunt u alsnog een aan uw netwerk aangepaste kernel compileren, maar nu is het balast.

Vergeet echter nimmer de TCP/IP netwerkondersteuning, want die hebt u bijvoorbeeld nodig om te printen (loopback), voor de X server en voor het internet (en om contact te houden met de andere gebruikers van uw systeem).

## 11.7 Linux kernel configuratie

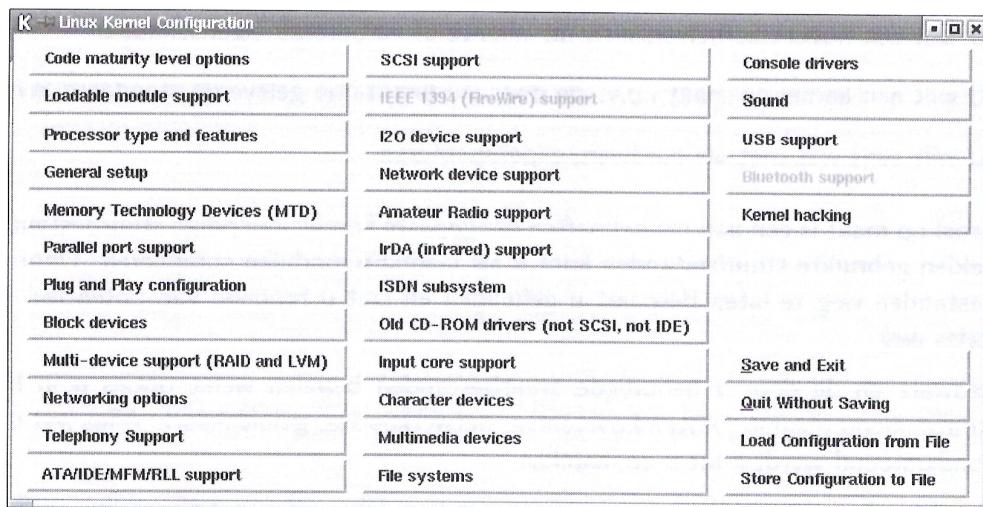
De linux kernel is meestal een gezippte kernel (zImage of bzImage). GRUB, de bootloader zorgt ervoor dat hij wordt geladen. U mag er zoveel versies van aanmaken als u wilt. Zo is het gebruikelijk om bij aanmaak van een nieuwe kernel de oude achter de hand te houden, zodat als de nieuwe kernel tegenvallt u de oude kernel weer kunt laden.

De broncode van de Linux kernel is vrij. Hij hoort bij iedere Linux distributie te zitten. De normale plaats ervoor is in /usr/src/linux . In de praktijk is /usr/src/linux een symbolische verwijzing naar de actuele broncode (source, src): /usr/src/linux-3.2.16.SuSE bijvoorbeeld. Dit is ook de plaats waar u de kernel moet compileren.

U hebt een GNU c compiler nodig (gcc), het make utility en een geschikte c bibliotheek (libc). In principe zullen ze onderdeel uitmaken van uw distributie.

De /usr/src/linux/README en de /usr/src/linux/Documentation/ map geven belangrijke achtergrondinformatie.

Er zijn drie programma's om een nieuwe kernel te configureren. Ze worden door de gebruiker root in /usr/src/linux opgestart:



Figuur 11.1: make xconfig

- make config
- make menuconfig
- make xconfig

Hier voor start u een terminalvenster op. U logt in als root (su) en gaat naar /usr/src/linux. Dit is een symlink naar de te compileren kernelversie (hier kernelversie 3.2.16). Het kan geen kwaad dit met ls -l nog eens te controleren in /usr/src:

```
root@bach:/usr/src$ ls -l
total 4
drwxr-xr-x 14 root root 4096 Aug 10 2011 kernel-source-3.2.16
lrwxrwxrwx 1 root src 20 Nov 24 2011 linux -> kernel-source-3.2.16

root@bach:/usr/src > cd linux
```

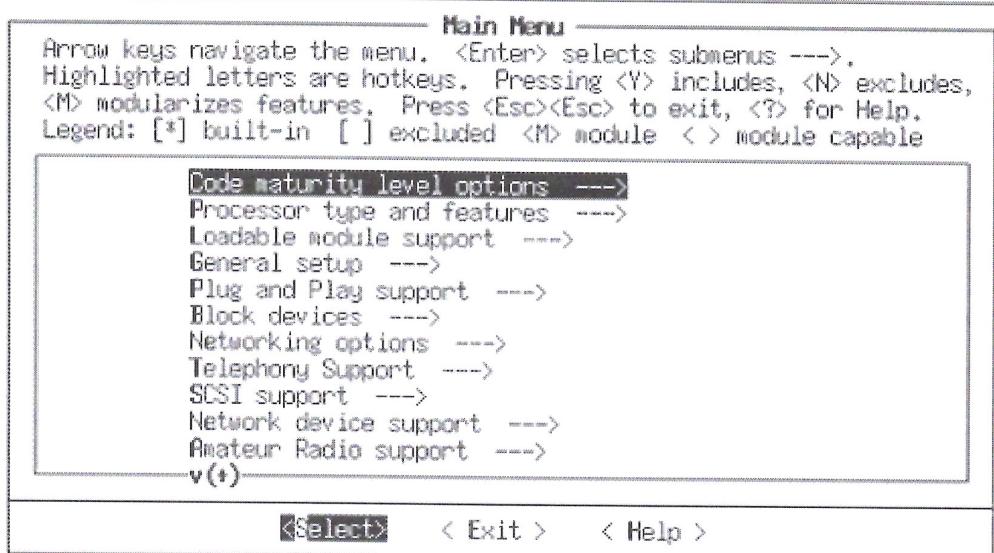
In /usr/src/linux typt u make xconfig (figuur 11.1) als u in X zit of make menuconfig (figuur 11.2) zonder X server.

```
root@visser:/usr/src/linux >make xconfig
```

```
root@bach:/usr/src/linux >make menuconfig
```

Wanneer geen van beide opties werkt kan men nog altijd het spartaanse make config gebruiken. Alle opties worden dan één voor één afgelopen:

## Linux Kernel v2.2.16 Configuration



Figuur 11.2: make menuconfig

```

bach:/home/mwyns/uml/linux-3.2.16# make config
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
/bin/sh scripts/Configure arch/i386/config.in
#
# Using defaults found in .config
#
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers [Y/n/?] y
*
* Loadable module support
*
Enable loadable module support (CONFIG_MODULES) [Y/n/?] n
*
* Processor type and features

```

make xconfig produceert onder het X Window systeem een soortgelijk scherm met instellingen van alle kernelopties. Wat u invoert wordt na *Save and Exit* in het verborgen bestand

/usr/src/linux/.config opgeslagen.

Met de optie *Store configuration to File* kunt u een kernelconfiguratie bibliotheek aanmaken (aanrader, net zo goed als het verstandig is een backup van oude config.sys bestanden te bewaren). Als u na een minder geslaagde configuratie van de regen in de drup komt kunt u met *Load configuration from File* een beproefde kernelconfiguratie als uitgangspunt nemen.

Ga als root naar /usr/src/linux. Deze directory is een symlink naar de door u te gebruiken broncode van de kernel.

Met het commando make dep clean zImage wordt de gezipte kernel gecompileerd. Het bestaat uit drie afzonderlijke commando's:

```
make dep
make clean
make zImage
```

Dit commando maakt een kernel image. De z geeft aan dat het om een gezipte versie gaat.

Krijgt u daarna foutmeldingen, dan kunt u niet verder gaan. De foutmelding *the kernel is too big* betekent dat uw kernel te groot is. U hebt drie opties om de kernel te verkleinen:

- Minder features in de kernel inbouwen (met N in plaats van Y).
- Minder vaak gebruikte delen als modulen inbouwen (met M in plaats van Y).
- Een sterk gecomprimeerde bzImage aanmaken (make bzImage i.p.v. make zImage).

Daarna moet u de kernelmodulen maken en installeren.

## 11.8 Modules

Als u de kernel modular (aanbevolen) opbouwt moet u vervolgens de volgende twee opdrachten ook uitvoeren in /usr/src/linux:

```
make modules
```

maakt de kernelmodules aan en

```
make modules_install
```

installeert specifieke modules uit /usr/src/drivers/.. in de map /lib/modules/kernel-versie.  
U kunt het ook in een regel doen:

```
make modules modules_install
```

Deze opdracht eindigt met iets als:

```
make[1]: Entering directory '/usr/src/linux-3.2.16.SuSE/arch/i386/lib'
make[1]: Nothing to be done for 'modules'.
make[1]: Leaving directory '/usr/src/linux-3.2.16.SuSE/arch/i386/lib'
Installing modules under /lib/modules/3.2.16/block
Installing modules under /lib/modules/3.2.16/net
Installing modules under /lib/modules/3.2.16/fs
Installing modules under /lib/modules/3.2.16/fs
Installing modules under /lib/modules/3.2.16/misc
root@bach:/usr/src/linux >
```

De modulen zijn hier in bovengenoemde subdirectory van /lib/modules geïnstalleerd.

## 11.9 Kernel installatie

Als slot moet de gecompileerde kernel nog naar de /boot directorie gekopieerd worden en de bootloader aangepast worden:

- maak een backup van de /boot directorie
- kopieer de nieuwe kernel en het bestand System.map naar de /boot directorie
- System.map bevat info over de benodigde modules
- stel *grub* voor de nieuwe kernel in
- laat ook de oude kernel als optie in de bootmanager staan

