Grundkunskap i Slackware Linux.
(Slackware Linux Essentials)

Officiell handledning till Slackware Linux.

av David Cantrell, Logan Johnson och Chris Lumens.
Översättning av Erik Jonsson.

Denna översättning är ännu ett pågående arbete och senast ändrad 10 november 2000. Det är lämpligt att kontrollera, huruvida det finns en nyare version, innan man tar kopior av denna bok.

Originalets titel: »*Slackware Linux Essentials.*«
av David Cantrell, Logan Johnson och Chris Lumens.

Originaltexten finns på `http://www.slackware.com/book/`

Svensk översättning / Swedish translation: »*Grundkunskap i Slackware Linux.*« utförd av / made by Erik Jonsson `<generaldepoten@yahoo.com>`.

Detta dokument är senast ändrat den 10 november 2000.

Innehåll

Förord.

Operativsystemet *Slackware Linux* är en kraftfull plattform för datorer, vilka bygger på Intel-processorer. Det är konstruerat så, att det skall vara stabilt, säkert och funktionellt både som server (nätverkstjänare) och som arbetsstation.

Avsikten med denna bok är, att man skall komma igång med operativsystemet Slackware Linux. Den avser inte skildra distributionen ur alla dess synvinklar utan vill snarare visa, vad distributionen duger till, och ge läsaren grundläggande kännedom om systemet.

Vi hoppas, att denna bok skall komma läsaren till nytta under insamlandet av erfarenheter tillsammans med Slackware Linux. Vi hoppas också, att läsaren skall låna ut den till alla sina vänner, när de kommer och frågar efter Slackware Linux.

Om än denna bok inte är någon spännande roman, så har vi förvisso försökt göra den så underhållande som möjligt. Har vi tur, får vi ett filmkontrakt. Men det är klart, att vi också hoppas, att läsaren skall kunna lära sig något av den och tycka, att den är till nytta.

Spelet kan börja.

I denna bok använda textkonventioner.

Författarna skriver i engelska originalets förord följande: »*Denna bok är skriven i SGML med användande av DTD:n DocBook 4.0. Vi har använt de i DocBook inbyggda elementen för filnamn, kommandon och filinnehåll. Detta ger ett konsekvent bruk av typsnitt och stilar för bokens alla aspekter. Man behöver göra sig förtrogen med några av dessa konventioner, innan man fortsätter läsa.*«

Översättarens anmärkning: Denna översättning är framställd i LaTeX och inte SGML, men textkonventionerna skiljer sig inte så våldsamt ifrån originalets. De räknas upp nedan.

Översättningens textkonventioner.

Textkonventionerna i översättningen är inte fullkomligt överensstämmande med originalets, men ligger likväl rätt nära:

- När författarna i den löpande texten nämner ett kommando, som kan köras, så återgives det med **`fet skrivmaskinsstil`**.
- Filnamn och kataloger återgives med `skrivmaskinsstil`.
- Längre kommandon tillsammans med text, som visas på skärm, återgives med `skrivmaskinsstil`. När sådana kommandon visas tillsammans med en prompt i form av dollartecknet `$`, innebär detta, att kommandot köres av en vanlig användare. När sådana kommandon visas tillsammans med en prompt i form av brädgårdstecknet `#`, innebär detta, att kommandot köres av överanvändaren **root**.
- Namn på användare och grupper visas vanligen i **fetstil**.

I

Inledning.

1

Introduktion till Slackware Linux.

1.1 Vad är Linux?

Linux påbörjades av Linus Torvalds år 1991 såsom ett personligt projekt. Han var ute efter ett sätt att kunna köra ett Unix-baserat operativsystem utan att behöva lägga ut stora pengar. Därtill ville han lära känna 386-processorn utan och innan. Linux släpptes ut kostnadsfritt till allmänheten under GNU General Public License,[1] så att vem som helst kunde hacka i det och göra förbättringar,.

Idag har Linux vuxit till att bli ett av de stora operativsystemen på marknaden. Det har porterats till flera datorarkitekturer, däribland Compaqs Alpha, Suns SPARC och UltraSPARC samt maskiner med Motorolas PowerPC-kretsar (från Apple Macintosh till IBM RS/6000 exempelvis). Linux utvecklas nu av hundratals (om inte tusentals) programmerare från hela världen. Det kör program såsom *sendmail*, *Apache* och *BIND*, vilka är några av de populäraste serverprogramvarorna på Internet.

Benämningen »*Linux*« hänför sig egentligen bara till kärnan — operativsystemets innersta beståndsdel, systemets hjärta. Det är denna del, som ansvarar för att styra processorn, minnet, hårddiskarna och andra periferienheter, som kan förekomma. Detta är allt, vad Linux gör. Linux styr datorns arbete och säkerställer, att alla program uppför sig hyfsat. Alla övriga program, som tillsammans gör linuxsystemet användbart, utvecklas av oberoende grupper. Olika företag och personer buntar ihop kärnan med programmen, så att resultatet blir ett operativsystem. Vi kallar detta för en *linuxdistribution.*

[1]se avsnittet *Öppen källkod och fri programvara* på s. 4 i kapitel 1 och *GNU General Public License* på s. 189 i bilagan.

1.2 Vad är Slackware?

Slackware var den första linuxdistribution, som nådde utbredd användning. Den påbörjades av Patrick Volkerding i slutet av 1992. Han kom i kontakt med Linux, när han till ett projekt behövde en LISP-tolk, som inte kostade för mycket. Vid den tiden fanns mycket få linuxdistributioner, och Patrick skaffade sig en ifrån *Soft Landing Systems* (SLS Linux).

Emellertid var SLS behäftat med en del skavanker, så att Patrick började rätta buggarna, efter hand som han hittade dem. Till sist beslöt han samla ihop alla buggfixarna i en egen distribution för sig själv och sina vänner. Denna privata distribution blev snabbt populär, och Patrick gjorde den tillgänglig för allmänheten under namnet *Slackware*.

Utefter vägen lade Patrick till nya saker och ting i distributionen, såsom t.ex. ett användarvänligt installationsprogram med menyer och begreppet pakethantering. Detta gör det enkelt för användarna att lägga till, ta bort eller förnya programvarupaket i sina system.

1.3 Öppen källkod och fri programvara.

Inom linuxgemenskapen verkar två stora ideologiska rörelser. Rörelsen för fri programvara (*Free Software*), som vi kommer till om en stund, arbetar med den målsättningen, att all programvara skall vara fri ifrån de inskränkningar i s.k. *intellectual property* (»intellektuell egendom«), vilka den anser hämmar teknisk förbättring och motarbetar samhällets bästa. Rörelsen för öppen källkod (*Open Source*) arbetar mestadels med samma mål, men intar en mer »pragmatisk« hållning gentemot dem, i det dess förfäktare föredrar att bygga sina argument hellre på de ekonomiska och tekniska fördelarna i att göra källkoden fritt tillgänglig än på de moraliska och etiska grundsatser, som utgör drivfjädrar i rörelsen för fri programvara.

Rörelsen för fri programvara anföres av *Free Software Foundation*, som verkar för att skaffa fram penningmedel till GNU-projektet. Fri programvara är snarast en ideologi. Ett ofta brukat uttryck är »*free speech, not free beer*« (»yttrandefrihet, inte gratis öl«). Väsentligen är rörelsen för fri programvara ett försök att garantera såväl användare som utvecklare bestämda rättigheter. Dessa rättigheter omfattar friheten att köra ett program av vilken som helst anledning, friheten att granska och förändra källkoden, friheten att vidareförmedla källkoden och friheten att dela med sig av alla ändringar, man kan tänkas införa däri. För att säkra dessa friheter är det, som *GNU General Public License* (GPL)[2] har skapats. I korthet går GPL ut på, att vem

[2]Se s. 189. Ö.a.

som helst, som distribuerar ett program enligt villkoren i GPL, också måste tillhandahålla källkoden och vidare tillåtas införa ändringar i programmet, förutsatt att dessa ändringar också blir tillgängliga i källkodsform. Detta garanterar, att när ett program väl »öppnats« för allmänheten, så kan det inte »stängas« igen förutom genom medgivande från varenda författare till vartenda stycke däri ingående kod (även förändringar). De flesta linuxprogram tillhandahålles på GPL-villkor.

Det är viktigt att lägga märke till, att GPL inte nämner någonting om pris. Hur märkligt det än må låta,[3] så får man ta betalt för fri programvara. Det »fria« med programvaran är de friheter, man har med källkoden, inte det pris, man betalar eller inte betalar. (Men när väl någon har sålt eller t.o.m. givit bort ett kompilerat program, som levererats på GPL-villkor, så är säljaren/givaren skyldig att tillhandahålla källkoden dessutom.)

I spetsen på den nyare rörelsen för öppen källkod (*Open Source*) står *Open Source Initiative* såsom en organisation, vilken har till enda uppgift att samla stöd för programvara med öppen källkod. Detta innebär, att programvarans källkod är tillgänglig tillika med det körklara programmet. Denna rörelse erbjuder ingen särskild licens utan stödjer i stället de olika förekommande licenserna för öppen källkod.

Tanken bakom OSI är den, att man skall få fler företag att ställa sig bakom öppen källkod genom att låta dem skriva sina egna licenser för öppen källkod och få dessa licenser certifierade genom *Open Source Initiative*. Många företag vill offentliggöra källkoden men inte använda GPL. Eftersom de inte får förändra GPL i grunden, så ger man dem i stället tillfälle att tillhandahålla sin egen licens och få den certifierad genom denna organisation.

Om än *Free Software Foundation* och *Open Source Initiative* arbetar för att hjälpa varandra, så är de likväl inte en och samma sak. *Free Software Foundation* tillämpar en särskild licens och levererar programvara under denna licens. *Open Source Initiative* eftersträvar stöd till alla licenser för öppen källkod inberäknat den ifrån *Free Software Foundation*. De grundvalar, utifrån vilka var och en av dem förfäktar källkodens fria tillgänglighet, skiljer ibland de två rörelserna åt; men själva den omständighet, att två ideologiskt olikartade grupper arbetar emot samma mål, förlänar trovärdighet åt bägges strävanden.

[3] Märkligt låter det kanske på engelska, men knappast på svenska, eftersom vi skiljer på *fri* och *gratis*. Ö.a.

2

Hjälp.

Det infinner sig stunder, då man behöver hjälp med ett bestämt kommando, med att rigga upp något program eller med att få någon maskinvaruenhet att fungera. Lyckligtvis finns en mängd olika sätt att skaffa hjälp. Om man har installerat paketen i programvaruserien F, så finns redan en hel mängd hjälpfiler installerade. Med programmen brukar också följa hjälp om deras valmöjligheter, inställningsfiler och bruk. Slutligen kan man ta en titt på Slackwares officiella WWW-sida för att få hjälp där.

2.1 I systemet inbyggd hjälp.

2.1.1 `man`.

Kommandot `man` (förkortning för »*manual*«) är av ålder den gängse inbyggda hjälpen i Unix- och Linux-system. Särskilt formaterade filer, »*man pages*«, finns skrivna för de flesta kommandon och distribueras tillsammans med programvaran. Om man kör `man någotkommando`, så visas manualsidan för (naturligtvis) kommandot eller programmet `någotkommando`.

Eftersom det finns så många manualsidor, så är de grupperade i numrerade avsnitt. Detta system har funnits så länge, att man ofta får se kommandon, program och t.o.m. programbiblioteksfunktioner omnämnda tillsammans med manualsidesavsnittets nummer. Till exempel kan man få se `man`(1). Detta utsäger, att kommandot `man` finns dokumenterat i avsnitt 1 (användarkommandon); vi kan ange, att vi vill se manualsidan i avsnitt 1 för »man« genom att kommendera `man 1 man`. Det kan vara lämpligt att ange avsnittets nummer i sådana fall, då det finns flera manualsidor med samma namn i olika avsnitt.

Avsnitt	**Innehåll**
Avsnitt 1	användarkommandon (endast inledning)
Avsnitt 2	systemanrop
Avsnitt 3	C-biblioteksanrop
Avsnitt 4	maskinvaruenheter (t.ex. hd, sd)
Avsnitt 5	filformat och protokoll (t.ex. `wtmp`, `/etc/passwd`, nfs)
Avsnitt 6	spel (endast inledning)
Avsnitt 7	konventioner, makropaket o.s.v. (t.ex. nroff, ascii)
Avsnitt 8	systemadministration (endast inledning)

Förutom `man` (1) finns kommandona `whatis` (1), och `apropos` (1), vars gemensamma syfte är att underlätta informationssökning i manualsystemet. Kommandot `whatis` ger en mycket kortfattad beskrivning av systemkommandon likt något slags lathund. `apropos` användes till att söka efter en manualsida, som innehåller ett givet nyckelord.

Ytterligare upplysningar om dessa kommandon finns på deras manualsidor. ;)

2.1.2 Katalogen `/usr/doc`.

Med källkoden till de programpaket, vi bygger, följer något slags dokumentation. Det kan vara `README`-filer, bruksanvisningar, licensvillkor... Vilken sorts dokumentation det än är, som medföljer källkoden, så installeras den i systemet under katalogen `/usr/doc`.

Om manualsidorna inte ger tillräckliga upplysningar, så bör `/usr/doc` bli nästa hållplats.

2.1.3 `HOWTOs` och `mini-HOWTOs`.

Det är i sann gemenskapsanda, som vi får samlingen med `HOWTO`/`mini-HOWTO`. Dessa filer är precis, vad det låter som — dokument, som beskriver, hur man gör olika saker. Om man installerar paketet med `HOWTO`-samlingen, så kommer `HOWTO`-dokumenten att installeras i `/usr/doc/Linux-HOWTOs` och `mini-HOWTO`-dokument i `/usr/doc/Linux-mini-HOWTOs`.

I samma paket ingår en samling FAQ:ar (*Frequently Asked Questions* — vanliga frågor med svar), som installeras på samma ställe.

Dessa filer förtjänar att läsas, närhelst man inte är riktigt säker på, hur man skall gå till väga med någonting. Här finns en häpnadsväckande spännvidd över olika ämnen, vilka stundom får en överraskande detaljerad behandling.

2.2 Hjälp på nätet.

Förutom den dokumentation, som levereras och kan installeras tillsammans med operativsystemet Slackware Linux, så finns hjälp att tillgå från flera ställen på Nätet.

2.2.1 WWW-sida och diskussionsforum.

www.slackware.com Detta är den officiella WWW-sidan för Slackware Linux och den är bräddfylld med smaklig slackwaredokumentation. Här finns introduktionshjälpsidorna, en installationsvägvisare, FAQ-listor med vanliga frågor och deras svar samt allehanda godsaker för nya (och ofta även för erfarna) användare.

På WWW-sidan finns också *Slackware Forum*, en avdelning där användare får diskutera sina erfarenheter av Slackware och hjälpa varandra med frågor och svårigheter. Den har visat sig vara en mycket omtyckt tillgång och har varit många till god hjälp, så att den förmodligen bör bli första hållplatsen, när man söker användarstöd. P.g.a. den stora läsekrets, man får för sina meddelanden, så har man ibland bättre tur med att få en snabb lösning här än någon annanstans; om man har tur, har någon annan redan lyckats lösa samma problem. Man bör först söka igenom *Slackware Forum* efter svar på frågor, som kanske redan har blivit både ställda och besvarade, innan man ställer gamla frågor på nytt.

Användarstöd genom epost Var och en, som köper en officiell CD med Slackware Linux, har rätt till gratis installationssupport per epost. Därmed är det sagt, att vi är av den gamla skolan. Vi gör vårt bästa för att hjälpa alla, som skriver supportfrågor till oss. Var dock vänlig att kontrollera dokumentationen och WWW-sidan (särskilt FAQ-sidorna och Slackware Forum) innan ni skickar epost hit; man kan mycket väl få snabbare svar på så vis; och ju mindre epost, vi måste besvara, desto snabbare kan vi hjälpa alla, vilket torde vara uppenbart.

Epostadressen för tekniskt användarstöd är `<support@slackware.com>`. Övriga epostadresser och annan kontaktinformation står förtecknad på WWW-sidan.

II

Installation.

3

Installera Slackware Linux.

Innan man kan använda Slackware Linux, måste man skaffa sig och installera detsamma. Att skaffa sig Slackware är lika enkelt som att köpa det eller ladda ner det gratis över Internet. Att installera det är också lätt, förutsatt att man har någon liten grundläggande kunskap om sin dator och är villig att lära sig en smula mer. Installationsprogrammet arbetar steg för steg. Därför kan man komma igång mycket snabbt.

3.1 Anskaffa Slackware Linux.

3.1.1 De officiella CD- och boxsatserna.

Det officiella CD-paketet med Slackware Linux kan anskaffas ifrån Slackware, Inc. Om man köper den officiella CD-satsen, så får man en bekväm installation ifrån CD, installationssupport via epost, ett installationshäfte på 30 sidor och mer. Slackwareboxen innehåller CD-satsen plus den officiella handboken för Slackware Linux. Det viktigaste är kanske det, att man genom köp av CD-satsen på ett utmärkt sätt ger direkt stöd till *Slackware Linux Project* (och hjälper oss köpa *nachos*[1]).

Kontaktsätt	**Information**
telefon	1-800-786-9907
WWW-sida	`http://www.slackware.com`
epost	`orders@slackware.com`
snigelpost	4041 Pike Lane, Suite F, Concord, CA 94520-1207

[1] Vad nu det är för något. Ö.a.

3.1.2 Över Internet.

Slackware Linux är också fritt tillgängligt över Internet. Även de, som laddat ner systemet på nätet, får sända in supportfrågor med epost; men de, som köpt den officiella CD-satsen, har företräde.

- Officiell WWW-sida för *Slackware Linux Project* är `http://www.slackware.com`
- Primär FTP-server för Slackware Linux är `ftp://ftp.slackware.com/pub/slackware/`
- Man kan även skaffa Slackware på närmare håll ifrån `ftp://ftp.sunet.se` och `ftp://ftp.funet.fi`

3.2 Systemkrav.

För att en installation av Slackware skall gå lätt, fordras åtminstone följande:

Maskinvara	Krav
Processor	386
RAM	16 MB
Hårddiskutrymme	500 MB
Diskettenhet	1.44 MB

Om man har en startbar CD-skiva och en dator, som kan starta på CD, så behöver man förmodligen ingen diskettenhet. För att man skall kunna göra en NFS-installation (över nätverket), fordras ett nätverkskort. Se avsnittet om NFS på s. 58 i kapitel 5 angående detta.

Kravet på hårddiskutrymme kan vara knepigt att fastställa. Rekommendationen 500 MB är vanligen på den säkra sidan; men om man gör en fullständig installation, så behöver man omkring en gigabyte tillgängligt hårddiskutrymme. Flertalet användare installerar inte allt. Man kan faktiskt klara sig med så litet som 100 MB hårddiskutrymme.[2]

Slackware kan installeras på system med mindre RAM och mindre hårddiskar, men det kräver då litet mer handarbete. Om man tänker göra sig den mödan, bör man ta en titt på filen `LOWMEM.TXT` i distributionskatalogträdet; där finns en del nyttiga tips.

[2] Detta gäller förstås begränsade användningar. Äldre utgåvor av Slackware kan klämmas in på mindre utrymme. Vi har själva fått plats med en starkt begränsad installation på en IBM PS/2 386SX 16 MHz med 4 MB RAM och 30 MB fastskiva, varav 5 MB användes till swap. Ö.a.

3.2.1 Programserier.

För enkelhets skull har Slackware av ålder indelat programvaran i serier. En gång i tiden kallades de för »diskettsatser«, eftersom de var avsedda att fördelas på de disketter, man dåförtiden installerade från. Nuförtiden brukas dessa programvaruserier främst till indelning av de program, som medföljer Slackware, i kategorier. Det är fortfarande möjligt att installera serie A och serie N ifrån disketter. Tabellen beskriver kortfattat innehållet i de olika serierna.

Serie	**Innehåll**
A	Grundsystem med enkel editor och kommunikationsprogram.
AP	Diverse program, som inte fordrar X-fönstersystemet.
D	Utvecklingsverktyg. Kompilatorer, avlusare, tolkar, manualsidor.
DES	Funktionen crypt() i GNU libc.
E	GNU Emacs.
F	FAQ:ar, HOWTO-dokument och annan blandad dokumentation.
GTK	GNOME Desktop Environment. GTK-biblioteket, GIMP.
K	Källkoden till linuxkärnan.
KDE	K Desktop Environment. Qt-biblioteket, som KDE är beroende av.
N	Nätverksprogram. Demoner, epostprogram, telnet o.s.v.
T	Typograferingssystemet teTEX.
TCL	Tool Command Language. Tk, TclX och filhanteraren TkDesk.
X	Grundläggande X-fönstersystem.
XAP	X-program, som inte ingår i KDE eller GNOME, t.ex. Netscape och Ghostscript.
XD	Programutveckling i X11. Bibliotek, PEX-stöd m.m.
XV	XView-biblioteken, OpenLook Virtual Window Manager m.m.
Y	Spel.

3.2.2 Installationsmetoder.

3.2.2.1 Diskett.

Om än det en gång var möjligt att installera hela Slackware Linux ifrån disketter,[3] så har programpaketens (och vissa enskilda programs) ökande omfång tvingat fram ett övergivande av diskettinstallation för alla programserier utom två. A-serien kan fortfarande helt och hållet installeras från disketter, och

[3]Det kunde — och kan — man t.o.m. Slackware 3.4. Ö.a.

likaså större delen av N-serien. Detta ger ett grundsystem, som kan användas till att installera resten av distributionen över nätverket.

Lägg märke till, att man fortfarande måste använda några disketter, om man inte har en startbar CD, liksom när man installerar via NFS.

3.2.2.2 CD-ROM.

Har man en startbar CD (ingår i den officiella CD-satsen från Slackware, Inc. — se avsnittet på s. 13 i kapitel 3 om att skaffa Slackware), är det enklare att installera från CD. Har man ingen sådan, får man starta på diskett.[4] Likaså kan det bli nödvändigt att använda särskilda disketter, om man har speciell maskinvara, som gör det svårt att starta den linuxkärna, som finns på CD-skivan.

Se om nödvändigt avsnitten ifrån *startdisketter* på s. 16 till *kompletteringsdisketter* på s. 17 beträffande upplysningar om vilka disketter, man bör använda, och hur man skapar dem.

3.2.2.3 NFS.

NFS (*Network File System*) är ett sätt att göra filsystem tillgängliga på fjärrdatorer. En NFS-installation möjliggör slackwareinstallation ifrån en annan dator i nätverket. Den maskin, varifrån man installerar, måste riggas upp så, att den exporterar slackwaredistributionens katalogträd till den maskin, man skall installera på. Detta förutsätter en del kunskaper om NFS, vilka meddelas i avsnittet om NFS på s. 58 i kapitel 5.

Det är möjligt att genomföra en NFS-installation genom PLIP (över parallellporten), SLIP och PPP (dock icke genom modemförbindelse). Vi anbefaller dock bruk av nätverkskort, om ett sådant är tillgängligt.

3.2.3 Startdiskett (boot disk).

En startdiskett (*boot disk*) är den diskett, man faktiskt startar systemet med för att påbörja installationen. Den innehåller en komprimerad avbild av linuxkärnan, vilken användes till att styra maskinvaran under installationen. Därför kan man inte klara sig utan denna diskett (såvida man inte startar på CD, såsom det skildras i avsnittet om CD-ROM på s. 16). Avbilder av olika startdisketter finns i katalogen `bootdsks.144/` i distributionskatalogträdet.

[4] Även om man startar på disketter, kan man installera ifrån CD, om bara linuxkärnan har stöd för den typ av CD-ROM-läsare, som kommer till användning. Så kan man göra, om datorn inte tillåter start på CD. Ö.a.

Det finns en mängd olika startdisketter för Slackware. En fullständig förteckning över dem finns tillsammans med deras beskrivningar i filen

```
bootdsks.144/WHICH.ONE
```

i slackwaredistributionsträdet. Emellertid klarar sig de flesta med någon av startdisketterna `bare.i` (för IDE-enheter) eller `scsi.s` (för SCSI-enheter). Se avsnittet på s. 18 om att skapa disketter.

När man har startat datorn på en sådan startdiskett, uppmanas man att stoppa in en rotdiskett (*root disk*). Vi anbefaller, att man därvidlag gör startdisketten till viljes.

3.2.4 Rotdiskett (root disk).

Rotdisketten innehåller programmet `setup` och ett filsystem, som användes under installationen. Denna diskett är nödvändig. Avbilderna av rotdiskar finns i katalogen `rootdsks` i distributionskatalogträdet.

Lyckligtvis finns betraktligt färre rotdiskettavbilder än startdiskettavbilder. Där finns faktiskt endast tre olika.

- `color.gz` är den, de flesta använder. Den visar menyer i färg, vilket kan vara trevligt.

- `text.gz` liknar `color.gz`, men visar menyerna i svart-vitt.

- `umsdos.gz` används för installation på en FAT-partition (DOS eller Windows), vilket vanligen endast anbefalles för experimentsyften. Är man intresserad av att prova Slackware på en windowspartition, rekommenderar vi bruk av ZipSlack eller BigSlack.

3.2.5 Kompletteringsdiskett (supplemental disk).

En kompletteringsdiskett (*supplemental disk*) behövs, om man skall installera över NFS eller till ett system med PCMCIA-enheter. Kompletteringsdisketterna finns i katalogen `rootdsks` i distributionskatalogträdet. De har filnamnen `network.dsk` och `pcmcia.dsk`.

Rotdiskettens ledtext ger instruktioner om, hur man skall förfara med kompletteringsdisketterna.

3.2.6 Skapa disketter.

När man väl har valt startdiskett, råskriver man den till en diskett. Denna process tillgår på olika vis beroende på operativsystem. I Linux (eller andra unixliknande operativsystem) använder man kommandot `dd`(1). Antag, att vi skall skriva diskettavbildsfilen `hejaz.dsk` till diskettenheten `/dev/fd0`. Kommandot blir då:

```
# dd if=hejaz.dsk of=/dev/fd0
```

Köres ett av Microsofts operativsystem, så får man använda programmet `RAWRITE.EXE`, vilket ingår i distributionskatalogträdet i samma kataloger som diskettavbilderna. Om vi antar, att vi skall skriva filen `hejaz.dsk` till enhet `A:`, så öppnar vi ett DOS-fönster och kommenderar:

```
c:\> rawrite a: hejaz.dsk
```

3.2.7 Partitionering.

När man har startat datorn med lämpligt startmedium, så behöver man partitionera hårddisken. Partitionen är en avdelning på hårddisken, där man skapar ett linuxfilsystem och installerar Slackware. Vi rekommenderar, att man skapar minst två partitioner: en för rotfilsystemet (`/`) och en för swaputrymme (virtuellt minne).

Efter att rotdisketten har laddats in, får man en inloggningsprompt. Vi loggar in som **root** (inget lösenord behövs än). Vid skalprompten kommenderar man endera `cfdisk`(8) eller `fdisk`(8). Programmet `cfdisk` har ett användarvänligare gränssnitt än det vanliga `fdisk` men saknar en del egenskaper. Vi skall nedan i korthet beskriva programmet `fdisk`.

Vi börjar med att köra `fdisk` för vår hårddisk. I Linux har hårddiskar inga enhetsbokstäver, utan representeras av filer. Den första IDE-hårddisken (*primary master*) är `/dev/hda`, *primary slave* är `/dev/hdb` o.s.v. SCSI-hårddiskar följer samma namngivningssystem men på formen `/dev/sdX`. Vi startar `fdisk` med kommandot

```
# fdisk /dev/hda
```

Liksom alla välartade unixprogram, så visar `fdisk` en prompt (och ingen meny). Först bör man undersöka vilka partitioner, som redan finns. Det gör vi genom att knappa in `p`:

```
Command (m for help): p
```

Detta visar all möjlig information om befintliga partitioner. De flesta väljer en ledig enhet att installera till och tar sedan bort befintliga partitioner för att bereda linuxpartitionerna plats.

Försiktigt! DET ÄR MYCKET VIKTIGT ATT SÄKERHETSKOPIERA ALL SÅDAN INFORMATION PÅ HÅRDDISKEN, SOM MAN VILL SPARA, INNAN MAN UTPLÅNAR DEN PARTITION, INFORMATIONEN IFRÅGA FINNS PÅ.

Det finns inget enkelt sätt att återställa en utplånad partition, så att man alltid skall säkerhetskopiera, innan man börjar leka med sådana.

Om vi betraktar tabellen med partitionsinformation, så bör vi få se ett partitionsnummer, partitionens storlek och dess typ. Där finns ytterligare upplysningar, men dem behöver man inte bry sig om f.n. Vi skall utplåna alla partitioner på denna hårddisk, så att vi kan skapa nya för Linux. Vi väljer `d` för att utplåna dem:

```
Command (m for help): d
Partition number (1-4): 1
```

Denna procedur upprepas för var och en av de partitioner, vi vill ta bort. Efter att ha utplånat partitionerna är vi färdiga att skapa nya för Linux. Vi har beslutat skapa en partition för rotfilsystemet och en för swap. Man bör därvid lägga märke till, att partitioneringsupplägg i Unix är föremål för många *flame wars*,[5] så att varendaste en användare själv vet bästa sättet att partitionera. Vårt råd är, att man till en början gör två partitioner: en för rotfilsystemet och en för swap. Med tiden lär man sig en partitionering, som passar ens eget system.

Nu skall vi skapa partitionerna med kommandot `n`:

```
Command (m for help):n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4):1
First cylinder (0-1060, default 0):0
Last cylinder or +size or +sizeM or +sizeK (0-1060, default 1060):+64M
```

Man skall förvissa sig om, att det är *primära partitioner*, man skapar. Första partitionen skall vara swappartition. Vi meddelar `fdisk`, att partition 1 skall vara en primär partition. Vi börjar på cylinder 0 och slår in `+64M` i

[5] Hetsiga dispyter i diskussionsgrupper och andra fora på Internätet. Ö.a.

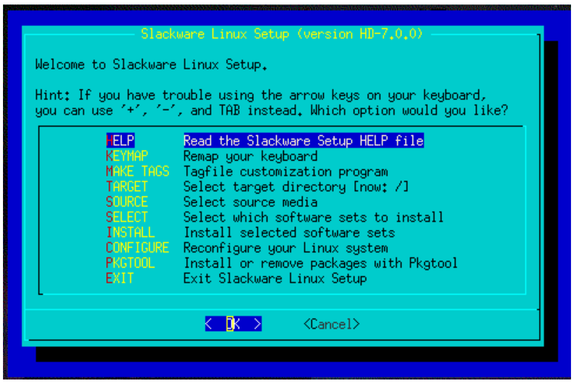
stället för numret på sista cylindern. Detta ger oss en partition på 64 MB till swap. (Den storlek på swappartitionen, man behöver, beror på hur mycket RAM, man har. Det är ett gammalt visdomsord, att man skall ha dubbelt så mycket swap som RAM.) Sedan avgränsar vi partition nummer 2 till att börja på första tillgängliga cylinder och fortsätta ända till hårddiskens slut.

```
Command (m for help):n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4):2
First cylinder (124-1060, default 124):124
Last cylinder or +size or +sizeM or +sizeK (124-1060, default 1060):1060
```

Vi är därmed nästan färdiga. Vi skall nu ändra typ på första partitionen ifrån 83 (*Linux native*) till 82 (*Linux swap*). Vi slår `t` för att kunna välja typ, väljer första partitionen och matar in `82`. Innan vi skriver ändringarna till hårddisken skall vi först kasta en sista blick på partitionstabellen. Vi slår `p` i `fdisk`, så visas partitionstabellen. Om allt ser ut att vara i ordning, så slår vi `w` för att skriva ändringarna till hårddisken och avslutar därmed `fdisk`.

3.2.8 Programmet `setup`.

När man väl har skapat sina partitioner, så är man redo att installera Slackware. Nästa steg är att köra programmet `setup`(8). Därför skriver vi `setup` vid skalprompten. `setup` är ett menysystem för installation av slackwarepaket och konfigurering av systemet.



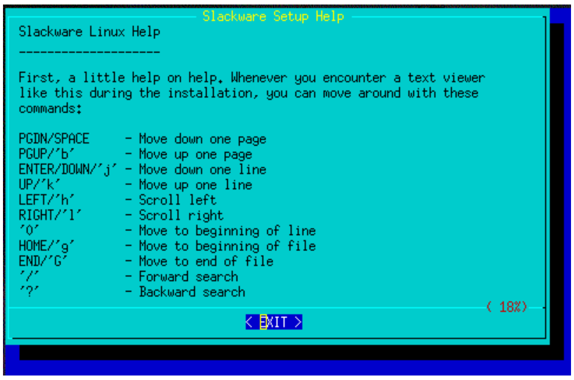
Figur 3.1: Slackwares installationsprogram `setup`.

Att köra `setup` går till ungefär så här: man stegar sig igenom vart val i `setup`-programmet i den ordning, de är förtecknade. (Man kan förstås göra allt i nästan vilken ordning, man vill, men då är det fara för, att det inte fungerar särskilt bra.) Menyalternativ väljes med pilknapparna upp och ner, knapparna »`Okay`« och »`Cancel`« med pilknapparna åt höger och vänster. Man kan också använda den tangent, vars bokstav är markerad i respektive menyalternativs namn. Flaggbara alternativ (sådana, som kan förkryssas) markeras och avmarkeras med mellanslagstangenten.

Givetvis står allt detta i `setup`-programmets »help«-avsnitt, men nu får läsaren veta det redan.[6]

3.2.8.1 HELP.

Om det är första gången, man installerar Slackware, så kan det löna sig att ta en titt på hjälptexten. Där beskrives de olika delarna av `setup` (ungefär som här, men mindre detaljerat), och där finns anvisningar för navigeringen igenom installationen.



Figur 3.2: Hjälpen i Slackwares `setup`.

3.2.8.2 KEYMAP.

Behöver man ett annat tangentbord än det amerikanska, så är det här, man väljer det.[7]

3.2.8.3 ADDSWAP.

Om man har skapat en swappartition för virtuellt minne (skildras i avsnittet om partitionering på s. 19), så kan man aktivera swap här. Swappartitionerna visas, så att man får välja, huruvida de skall formateras eller ej.

[6] Och läsaren får här dessutom läsa om det på svenska. Ö.a.

[7] Man behöver inte bläddra ända ner till `se-latin1.map` för att få svenskt tangentbord. Det går lika bra med det finska `fi-latin1.map`. Ö.a.

Figur 3.3: Välja tangentbord: man väljer lämpligen `fi-latin1.map`.

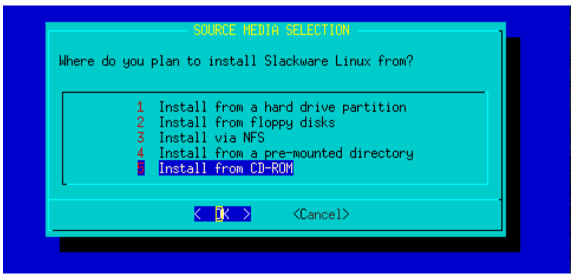
3.2.8.4 TARGET.

Här väljer man vilka partitioner (förutom swap), installationen skall göras till. För var partition får man välja, huruvida den skall formateras (och om formateringen skall kontrollera dåliga block), samt mellan ett antal olika inodstorlekar. För normalt bruk är det förvalda antalet inoder gott nog.

Första valet i avsnittet *target* gäller vilken partition, som rotkatalogen (/) skall ligga på. Därefter får man tillordna övriga partitioner till kataloger i filsystemet efter eget val. (Man kanske vill ha t.ex. tredje partitionen `/dev/hda3` till sitt hemfilsystem `/home`. Detta är endast ett exempel; man kan tillordna partitioner till kataloger efter eget gottfinnande.)

3.2.8.5 SOURCE.

Här väljer man källmedium för installationen, d.v.s. varifrån Slackware installeras. För närvarande kan man välja mellan fyra olika källor. Dessa är *floppy* (diskett), CD-ROM, NFS (*Network File System*) eller en i förväg monterad katalog.



Figur 3.4: Val av källa för installationen: varifrån, paketen skall installeras.

Alternativet *floppy* inleder en installation ifrån många, många disketter. Detta alternativ fordrar massvis med tid och tålamod, men det är dock fullt genomförbart. Tänk på, att man behöver skapa disketterna, innan man sätter igång `setup`-programmet.

Alternativet CD-ROM ger installation ifrån CD-ROM. Det erbjuder möjligheten att endera avsöka maskinen efter en CD-ROM-läsare eller också visa en lista, varifrån man får välja sin CD-enhet. Man skall ha CD-skivan med Slackware på plats i CD-läsaren innan man börjar avsöka enheten. När programmet har hittat en CD-ROM-enhet, frågar det, om man vill installera »slakware« eller »slaktest«. Förvalet är *slakware*, vilket är en standardinstallation. Valet *slaktest* genomför en minimal installation till hårddisken och behåller resten på CD. Man behöver då den s.k. *live*-CD:n i den officiella CD-satsen, för att detta skall fungera.

Alternativet NFS begär nätverksinformation för installationsmaskinen samt nätverksinformation för NFS-servern. NFS-servern måste ha riggats upp i förväg. Man kan inte använda värddatornamn, utan man måste använda IP-adresser (med siffror) för såväl den egna maskinen som för NFS-servern (det finns ingen namnresolver[8] — DNS-klient — på `setup`-disketten).

Störst flexibilitet erbjuder alternativet med förmonterad katalog (*premounted directory*). Man kan använda denna metod till att installera ifrån Jaz-diskar, genom NFS-montering över PLIP och ifrån FAT-filsystem. Man monterar först filsystemet till valfri katalog, innan man kör `setup`, så kan man sedan välja katalog där.

3.2.8.6 SELECT.

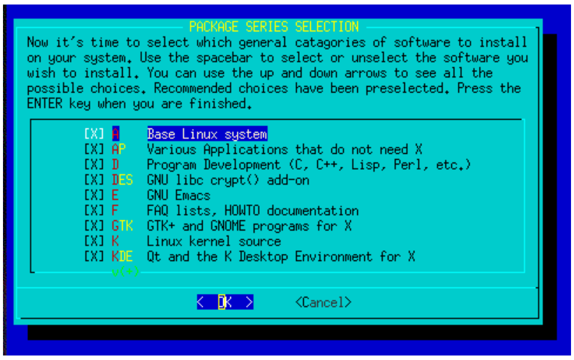
Alternativet *select* ger möjlighet att välja vilka programvaruserier, man vill installera. Dessa beskrives på s. 15. Man måste installera åtminstone A-serien för att få ett fungerande system. Alla andra serier är valfria.

3.2.8.7 INSTALL.

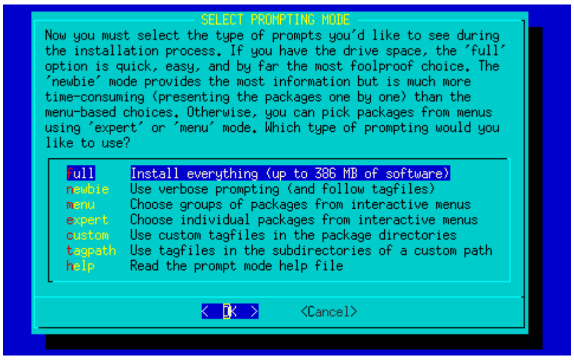
Om vi antar, att man har gått igenom alternativen under »target«, »source« och »select«, så kommer nu »install« att medge val av paket ur de redan valda programpaketserierna. Om inte, så blir man uppmanad att gå tillbaka och komplettera de tidigare avsnitten i `setup`. Detta alternativ erbjuder val emellan sex olika installationsmetoder: *full*, *newbie*, *menu*, *expert*, *custom* och *tag path*.

Alternativet *full* installerar alla paket i *de programserier man valt* under avsnittet »select«. Inga vidare uppmaningar visas. Detta är den enklaste

[8] Namnupplösare. Ö.a.



Figur 3.5: Val av programserier (kallades tidigare »diskettsatser«).



Figur 3.6: Här väljer man, hur installationen skall gå till.

installationsmetoden, eftersom man inte behöver fatta några beslut om installation av enstaka paket. Givetvis tar detta val mest hårddiskutrymme i anspråk.

Nästa alternativ är *newbie*. Detta val installerar alla erforderliga paket *i de valda serierna*. För icke erforderliga paket visas en fråga om installation med svarsalternativen »Yes«, »No« eller »Skip«. *Yes* och *No* väljer eller väljer bort ett visst paket, medan *Skip* går vidare till *nästa programserie*. Utöver detta visas en beskrivning och uppgift om behov av hårddiskutrymme för vart paket såsom hjälp vid valet, om sådan skulle behövas. Vi anbefaller detta alternativ för nya användare, eftersom det säkerställer, att man får de erforderliga paketen installerade. Emellertid går det en smula långsamt p.g.a. frågorna om, huruvida programmen skall installeras eller ej.

Alternativet *menu* är snabbare och en mer avancerad version av *newbie*. För var serie visas en meny, varifrån man får välja installation eller ej av alla paket, som inte är erforderliga. Erforderliga paket visas inte på denna meny.

För mer framskridna användare erbjuder `install` alternativet *expert*.

Detta val ger fullständig detaljstyrning av vilka programpaket, som installeras. Man kan då välja bort paket, som är ovillkorligen erforderliga, så att man får ett sprucket system. Å andra sidan kan man i detalj styra, vad som skall ingå i systemet. Man väljer paket ifrån var serie, som skall installeras. Detta anbefalles icke för nybörjare, eftersom det är lätt hänt, att man råkar skjuta sig själv i foten.

Alternativen *custom* och *tag path* är också avsedda för avancerade användare. Dessa val medger en installation, som bygger på skräddarsydda taggfiler (etikettfiler), man i förväg har skapat i distributionskatalogträdet. Detta är till nytta, när man snabbt vill göra identiskt lika installationer på många maskiner. Ytterligare upplysningar om taggfiler meddelas i avsnittet om att skapa taggar och taggfiler på s. 172.

När man har valt installationsmetod, kommer endera av flera olika saker att hända. Om man har valt *full* eller *menu*, så visas en menyskärm, där man kan välja vilka paket, som skall installeras. Om man har valt *full*, så börjar paketen därefter omedelbart installeras till vald målenhet. Om man har valt *newbie*, så börjar installationen och pågår fram till ett valfritt paket, då man får välja.

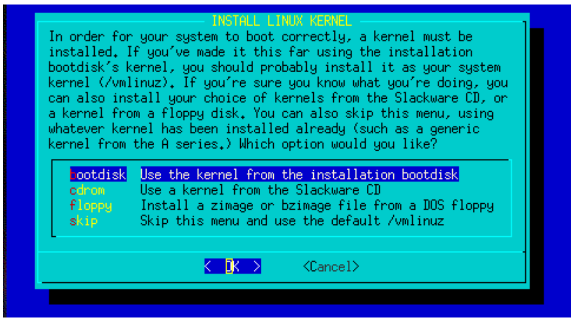
Lägg märke till, att man kan få brist på hårddiskutrymme, medan man installerar. Om man har valt för många paket i förhållande till hårddiskutrymmet, så får man bekymmer. Vid gränsfall är det säkrast att välja färre programpaket under den ursprungliga systeminstallationen och installera ytterligare paket senare. Detta är enkelt med pakethanteringsverktygen i Slackware. Om dem kan man läsa på s. 168.

3.2.8.8 CONFIGURE.

Avsnittet *configure* medger, att man genomför en del grundläggande systemkonfigurering redan under installationen, sedan paketen väl är installerade. Vad man får se här, beror till stor del på vilka program, man har installerat. Man får emellertid under alla förhållanden se följande:

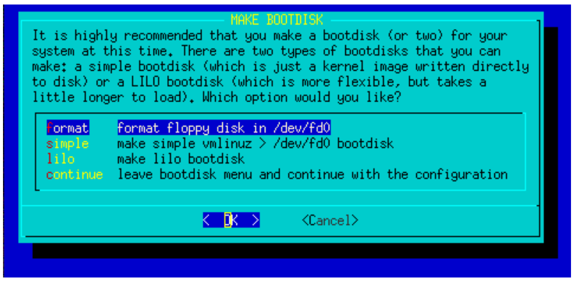
Kernel selection. Här tillfrågas man om vilken kärna, man vill installera. Man kan installera en kärna ifrån den startdiskett, man använt till installationen, från Slackware-CD:n eller ifrån någon annan diskett, man förberett (om man alltid tänker framåt). Eller också kan man hoppa över, i vilket fall en förvald kärna installeras.

Make a boot disk. Det är nog inte så dumt att skapa en startdiskett för framtida bruk. Man erbjudes att formatera en diskett och sedan skapa endera av två slags startdisketter. Den första typen, *simple*, skriver en linuxkärna till disketten. Ett flexiblare alternativ är *lilo* (anbefalles),



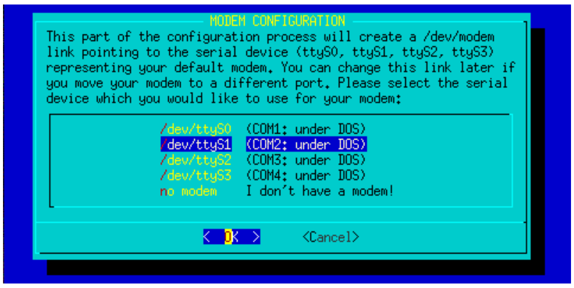
Figur 3.7: För att systemet skall starta efter installationen, behövs en linuxkärna.

vilket förstås skapar en startdiskett med LILO. Se avsnittet om LILO på s. 83. Man kan givetvis i stället välja att fortsätta (*continue*) utan att ha skapat någon startdiskett.



Figur 3.8: Skapa startdiskett: för säkerhets skull bör man skapa en. Om man inte tänker installera LILO eller Loadlin, är det nödvändigt att ha en startdiskett.

Modem. Här tillfrågas man om modemet: huruvida man har något modem, och vilken serieport, det är anslutet på.



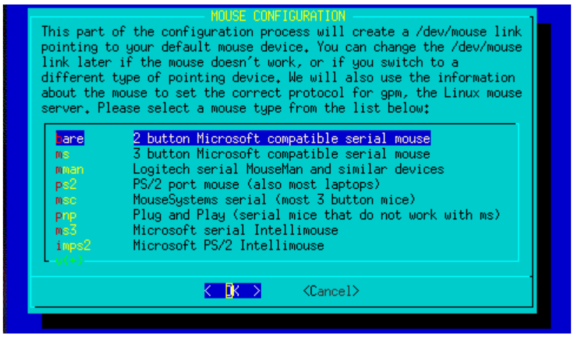
Figur 3.9: Modemkonfigurering: här anger man, var modemet är anslutet.

Följande underavdelningar i konfigureringen kanske visas, kanske inte; det beror på, om man installerat motsvarande programpaket (`setup` anpassar sig till detta).

Timezone Detta alternativ är tämligen enkelt och rakt på sak: man blir tillfrågad om vilken tidszon, man befinner sig i. Listan är lång, och man kan bara bläddra i den med piltangenterna, så det kan bli besvärligt för den, som använder Zulu-tid.

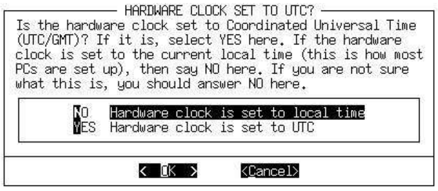
Figur 3.10: Tidszon: här kan man välja t.ex. »Europe/Copenhagen«.

Mouse Denna underavdelning frågar efter mustyp samt huruvida man vill starta `gpm` (8) (musstöd i den ickegrafiska miljön utanför X-fönstersystemet) vid systemstart.



Figur 3.11: Muskonfigurering: här anger man mustyp och hur, den är ansluten.

Hardware clock Här anger man, huruvida klockan på moderkortet följer världstiden *Coordinated Universal Time* (UTC eller GMT). Det gör de flesta PC-datorer inte, och i så fall svarar man nej.



Figur 3.12: Maskinvaruklockan ställes endera till världstid (UTC) eller ortstid.

Figur 3.13: Välja typsnitt för textmiljön utanför X-fönstersystemet.

Font Här kan man välja ett typsnitt för den textbaserade miljön utanför fönstersystemet X.

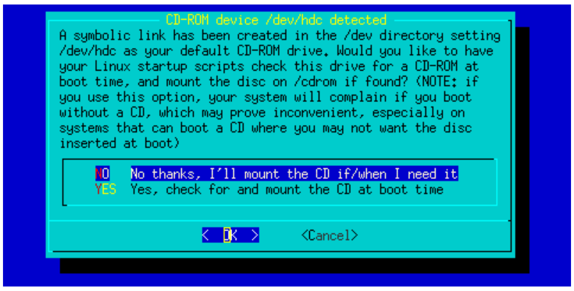
LILO Här erbjudes man att installera LILO (LInux LOader; se avsnittet om LILO på s. 83). Om Slackware skall vara ensamt operativsystem i datorn, så duger det med alternativet *simple*. Om man skall kunna välja mellan flera olika operativsystem vid start, så är alternativet *expert* ett bättre val. Se avsnittet om att starta flera operativsystem på s. 87.

Valet att inte installera LILO (*do not install*), anbefalles icke, såvida man inte säkert vet vad man gör och har ett mycket gott skäl att inte installera LILO. Om man installerar enligt alternativet *expert*, så får man välja, var LILO skall läggas. Man kan lägga LILO i MBR (Master Boot Record) på hårddisken, på linuxrotpartitionens *superblock* eller på en diskett.

Märk väl, att om man redan använder ett annat operativsystems bootladdare, så är det tillrådligt att installera LILO endera till linuxrotpartitionens *superblock* eller till diskett. I ett sådant fall kommer nämligen det andra systemets bootladdare att ställas åt sidan, när man installerar LILO till MBR, vilket kan göra livet synnerligen besvärligt.

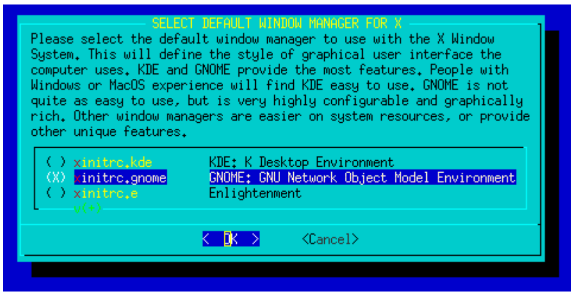
Network Underavdelningen för nätverksinställningar är lika med programmet `netconfig`. Se avsnittet om `netconfig` på s. 49 i kapitel 5.

CD-ROM Underavdelningen för CD-ROM frågar om, huruvida man vill, att systemet skall avkänna CD-läsaren och av sig själv montera tillgänglig CD-skiva under `/cdrom`.



Figur 3.14: Skall CD-läsaren monteras av sig själv vid systemstart?

X Window Manager I denna underavdelning får man ange förvald fönsterhanterare för X. Se s. 75 i kapitel 6 beträffande enskildheter om X och fönsterhanterare.



Figur 3.15: Välja fönsterhanterare för X.

Oavsett vilka paket, man har installerat, så avslutar `configure` med att fråga om, huruvida man vill välja ett **root**lösenord. Det är lämpligt av säkerhetsskäl, att man gör det. Såsom med nästan allt annat i Slackware, så får man göra som man vill i denna sak också.

3.2.8.9 EXIT.

Här lämnar man förstås installationsprogrammet. Att detta avsnitt finns med i boken är förstås en förolämpning emot läsarens förstånd. Vi ber ödmjukt om ursäkt.

3.3 Sammanfattning.

Man bör nu ha Slackware Linux installerat på sitt system. Därtill bör man ha blivit hemtam med partitionering av enheter, med programvarupaket, programmet `setup` och några enkla inställningar. Med dessa kunskaper bör man vara redo att skrida till verket och fullfölja konfigureringen av sitt system.

III

Inställningar.

4

Systeminställningar.

Innan man ger sig i kast med mer avancerad systemkonfigurering, är det förnuftigt att lära sig, hur systemet är upplagt, och vilka kommandon, som kan användas till att söka efter filer och program. Det är också bra att veta, huruvida man behöver kompilera en kärna och vilka mått och steg, man måste vidta för detta. I detta kapitel bekantar vi oss med, hur systemet är organiserat, samt med dess inställningsfiler. Därefter kan vi gå in på mer avancerade delar av systemet.

4.1 Systemöversikt.

Det är viktigt att förstå, hur ett linuxsystem är sammansatt, innan man dyker ner och fördjupar sig i konfigureringen och dess olika synvinklar. Ett linuxsystem skiljer sig avsevärt ifrån ett DOS- eller windowssystem (eller t.o.m en Macintosh), men följande avsnitt skall hjälpa läsaren bekanta sig med upplägget, så att man lätt skall kunna ställa in systemet så pass väl, att det tillgodoser ens egna behov.

4.1.1 Filsystem.

Den första märkbara skillnaden emellan Slackware Linux å ena sidan och DOS eller Windows å den andra är filsystemet. För det första använder vi inte enhetsbokstäver till att beteckna olika partitioner. I Linux finns en enda huvudkatalog. Man kan jämföra denna med enhet `C:` i DOS. Var partition i systemet monteras under en katalog i huvudkatalogen. Det kan liknas vid en hårddisk, som ständigt växer sig större.

Vi kallar huvudkatalogen för *rotkatalogen*, och den betecknas med ett enkelt, framåtlutat snedstreck (`/`). Detta kan synas underligt, men det gör faktiskt livet lättare för en, när man vill lägga till mer lagringsutrymme. Antag exempelvis, att lagringsutrymmet på den hårddisk, där katalogen `/home` är belägen, tar slut. De flesta installerar Slackware på en enda stor partition till rotkatalogen. Nå, eftersom nu en partition kan monteras under vilken som helst katalog, så kan man helt enkelt gå till affären och köpa en ny hårddisk, som man monterar under `/home`. Då har man så att säga »ympat« på litet mer utrymme till systemet. Allt utan att ha måst flytta omkring särskilt mycket.

Nedan följer beskrivningar av de viktigaste katalogerna omedelbart under rotkatalogen i Slackware.

`/bin` Grundläggande användarprogram förvaras här. Dessa utgör minsta möjliga uppsättning program, som behövs, för att man skall kunna använda systemet. Kommandoskal och filsystemskommandon (`ls`, `cp` o.s.v.) förvaras här. Katalogen `/bin` ändrar man vanligtvis inte på efter installationen. I så fall sker det genom paketuppgraderingar ifrån Slackware.

`/boot` Filer, som används av Linux Loader (LILO), förvaras här. Inte heller denna katalog förändras mycket efter installationen.[1]

`/cdrom` Som sagt, alla enheter måste monteras under en katalog under rotkatalogen. Katalogen `/cdrom` tillhandahålles här såsom monteringsställe för CD-ROM-läsaren.

`/dev` Allting i Linux behandlas såsom filer. Det gäller även maskinvaruenheter såsom serieportar, hårddiskar och bildläsare. För att man skall komma åt dessa enheter, måste en särskild sorts fil vara förhanden, vilken kallas för *device node*. Alla sådana förvaras i katalogen `/dev`. Detta överensstämmer med bruket i många unixliknande operativsystem.

`/etc` Denna katalog innehåller filer för systemkonfigurering. Allting ifrån X-fönstersystemet inställningsfil till användardatabasen och systemets startskript. Med tiden blir systemadministratören väl förtrogen med denna katalog.

`/home` Linux är ett fleranvändarsystem. Var användare i systemet får ett konto och en särskild katalog för personliga filer. Denna katalog kallas för användarens »*hemkatalog*«. Katalogen `/home` tillhandahålles såsom förval att ha användarnas hemkataloger i.

[1] Såvida man inte kompilerar om kärnan eller ändrar det meddelande, som skall visas vid LILO:s bootprompt. Ö.a

`/lib` Programbibliotek, som är nödvändiga, för att systemet skall kunna fungera, förvaras här. C-biblioteket, den dynamiska laddaren, ncurses-biblioteket och kärnmodulerna hör dit.

`/lost+found` När systemet startar, kontrolleras filsystemen med avseende på fel. Om några fel uppdagas, så köres programmet `fsck` för att undersöka, huruvida några av felen kan rättas till. De rättade delarna av filsystemet skrivs till katalogen `/lost+found`.

`/mnt` Denna katalog tillhandahålles som tillfälligt monteringsställe för arbete på hårddiskar eller löstagbara enheter.

`/opt` Katalogens namn står för *optional*. Den är avsedd för valfria programpaket. Tanken med `/opt` är den, att om man installerar programpaket under `/opt/<programpaket>`, så blir det lätt att ta bort dem senare. Slackware distribuerar en del program, som hamnar under `/opt` (såsom KDE i `/opt/kde`), men man kan lägga till vad man vill i `/opt`.

`/proc` Denna katalog är ensam i sitt slag. Den är egentligen inte en del av filsystemet, utan utgör ett virtuellt filsystem, som ger tillgång till information ifrån kärnan. Olika upplysningar, kärnan vill förmedla, uppenbaras genom »filer« i katalogen `/proc`. Man kan även sända information till kärnan genom några av dessa »filer«. Man kan försöka med att kommendera `cat /proc/cpuinfo`.

`/root` Systemadministratören kallas för **root** och har hemkatalogen `/root` i stället för `/home/root`. Skälet är enkelt. Vad skulle hända, om `/home` låge på en annan partition än `/` och denna partition inte kunde monteras? Då skulle förstås **root** vilja logga in och åtgärda problemet. Om då hans hemkatalog hade legat på det skadade filsystemet, så skulle det ha blivit svårt för honom att logga in.

`/sbin` Väsentliga, grundläggande program, som **root** brukar köra och som körs under systemstart, förvaras här. Vanliga användare brukar inte köra program i denna katalog.

`/tmp` Katalog för tillfällig lagring. Alla användare har läs- och skrivrätt till denna katalog.

`/usr` Detta är den stora katalogen i ett linuxsystem. I stort sett allt övrigt hamnar här: program, dokumentation, källkoden till linuxkärnan och fönstersystemet X. Detta är den katalog, dit man troligast installerar program.

`/var` Systemets loggfiler, cachedata och programmens låsfiler förvaras här. Detta är en katalog för data, som ändras ofta.

Efter denna genomgång bör man ha en god känsla för vilka kataloger, som innehåller vad, i filsystemet. Nästa avsnitt skall visa, hur man lätt hittar bestämda filer, så att man inte behöver söka för hand.

4.1.2 Finna filer.

Nu vet vi, vad var katalog innehåller, men detta är likväl inte till så mycken hjälp, när man letar efter något. Man kan visst leta sig igenom alla katalogerna, men det finns snabbare sätt. Slackware har fyra viktiga sökkommandon.

4.1.2.1 `which`.

Det första är kommandot `which` (1). Det brukas vanligen till att snabbt leta upp ett program. Det genomsöker endast sökvägen (enligt variabeln PATH) och svarar med den första förekomst, det stöter på, tillsammans med sökvägen dit. Betrakta detta exempel:

```
$ which bash
/bin/bash
```

Som synes finnes `bash` i katalogen `/bin`. Detta är ett mycket begränsat sökkommando, eftersom det endast följer den sökväg, som står i variabeln PATH.

4.1.2.2 `whereis`

Kommandot `whereis` (1) fungerar på ett liknande sätt som `which`, men det kan även söka upp manualsidor och källkodsfiler. En sökning med `whereis` efter `bash` bör ge detta resultat:

```
$  whereis bash
bash: /bin/bash /usr/bin/bash /usr/man/man1/bash.1.gz
```

Detta kommando talar inte bara om för oss, var själva programmet finns, utan även var hjälpen (manualsidan) är lagrad. Likväl har detta kommando sina begränsningar. Tänk, om man skulle vilja söka efter en bestämd inställningsfil? Då duger inte `which` eller `whereis`.

4.1.2.3 `find`

Kommandot `find`(1) söker efter vad som helst. Antag, att vi vill söka efter systemets `xinitrc`-fil:

```
$ find / -name xinitrc
./var/X11R6/lib/xinit/xinitrc
```

`find` tar en stund på sig att köra, eftersom det måste genomlöpa hela rotkatalogträdet. Om man kör detta kommando såsom vanlig användare, så får man förmodligen en del felmeddelanden om vägrad åtkomst till kataloger, vilka endast **root** har sökrätt i. Men `find` har hittat vår fil, så det blev bra. Hade det bara varit en smula snabbare...

4.1.2.4 `locate`

Kommandot `locate`(1) genomsöker hela filsystemet, precis som `find`, men det letar i en databas i stället för att söka i det egentliga filsystemet. Denna databas är uppriggad så, att den av sig själv uppdateras dagligen klockan 04:40, för att man skall ha en någorlunda färsk fillista att söka i. Man kan uppdatera databasen för hand genom att köra `updatedb`(1) (först måste man använda kommandot `su` för att tillfälligt få rättigheter som **root** eller **nobody**). Här följer ett exempel:

```
$ locate xinitrc   # Vi behöver inte bli root först
/var/X11R6/lib/xinit/xinitrc
/var/X11R6/lib/xinit/xinitrc.fvwm2
/var/X11R6/lib/xinit/xinitrc.openwin
/var/X11R6/lib/xinit/xinitrc.twm
```

Nu fick vi mer än vi letade efter, och snabbt gick det. Med dessa kommandon skall man kunna finna vad man än letar efter i sitt linuxsystem.

4.1.3 Katalogen `/etc/rc.d`.

Systemets startfiler finns i katalogen `/etc/rc.d`. Slackware använder ett upplägg för systeminitieringsfilerna, vilket liknar BSD-systemens. Var uppgift (*task*) eller körnivå (*runlevel*) har sin egen `rc`-fil. Detta ger en välordnad uppbyggnad, som är lätt att underhålla.

4.1.3.1 Systemstart.

Det första program, som börjar köras under Slackware, förutom linuxkärnan, är `init` (8). Detta program läser filen `/etc/inittab` (5) för att se, hur systemet skall köras. Det kör skriptet `/etc/rc.d/rc.S` för att förbereda systemet innan det går in i önskad körnivå (*runlevel*). Filen `rc.S` gör det virtuella minnet verksamt, monterar filsystemen, rensar upp i vissa loggkataloger, initierar plug'n'play-enheter, laddar kärnmoduler, konfigurerar PCMCIA-enheter, ställer in serieportar och kör SystemV-initieringsskript (ifall sådana finnes). Uppenbarligen har `rc.S` mycket på tallriken, men här följer en lista över några av de skript, som `rc.S` anropar för att fullgöra sitt arbete:

`rc.S` Detta är det egentliga systeminitieringsskriptet.

`rc.modules` Laddar kärnmoduler. Saker och ting såsom nätverkskort, PPP-stöd, ljudkort, skrivarport m.m. laddas här. Om detta skript i samma katalog hittar en fil vid namn `rc.netdevice` och denna är körbar, så köres även det senare skriptet.

`rc.pcmcia` Avkänner och ställer in PCMCIA-enheter, som kan förekomma i systemet. Detta kommer särskilt väl till pass för användare av bärbara datorer, som förmodligen har ett PCMCIA-modem eller dito nätverkskort.

`rc.serial` Ställer in serieportarna genom att köra `setserial`-kommandon.

`rc.sysvinit` Letar upp SystemV-initieringsskript för önskad körnivå (*runlevel*) och kör dem. Detta diskuteras mer ingående nedan.

4.1.3.2 Initialiseringsskript för olika körnivåer.

Efter att systeminitieringen är färdig, fortsätter `init` med initiering för vald körnivå. En körnivå beskriver ett tillstånd, som maskinen skall köra i. Låter detta som något överflödigt? Nåja, körnivån talar om för `init`, huruvida man vill ta emot inloggningar av flera användare eller enbart en enda användare, huruvida man vill ha tillgång till nätverkstjänster och huruvida man vill låta X-fönstersystemet eller `agetty` (8) hantera inloggningarna. Nedan uppräknade filer definierar de olika körnivåerna i Slackware Linux.

`rc.0` Stanna systemet (*runlevel* 0). Såsom förinställning är denna fil länkad till filen `rc.6`.

`rc.4` Fleranvändarstart (*multiuser startup*, *runlevel* 4), men i fönstersystemet X11 med någon av de grafiska inloggningshanterarna KDM, GDM eller XDM.

`rc.6` Starta om systemet (*runlevel* 6).

`rc.K` Enanvändarstart (*single user mode*, *runlevel* 1).

`rc.M` Fleranvändarstart (*multiuser mode*, *runlevel* 2 och 3), men med den standardmässiga textbaserade inloggningen. Detta är förinställd körnivå i Slackware.

4.1.3.3 Nätverksinitiering.

Körnivåerna 2, 3 och 4 startar även nätverkstjänster. Följande filer ombesörjer nätverksinitiering:

`rc.inet1` Skapas av `netconfig`. Denna fil står för konfigurering av nätverksgränssnittet.

`rc.inet2` Köres efter `rc.inet1` och sätter igång grundläggande nätverkstjänster.

`rc.atalk` Sätter igång AppleTalk-tjänster.

`rc.httpd` Sätter igång webservern Apache.

`rc.samba` Sätter igång fil- och skrivardelning för Windows-maskiner.

`rc.news` Sätter igång newsservern.

4.1.3.4 Kompatibilitet med System V.

Kompatibilitet med initiering enligt System V har införts i Slackware 7.0. Många andra linuxdistributioner begagnar detta slag av systeminitiering i stället för den BSD-liknande, som Slackware använder. Vid SystemV-init är grunden den, att var körnivå (*runlevel*) får en egen *underkatalog* till sina initieringsskript, medan den BSD-liknande Slackware-initieringen har *ett enda initieringsskript* för var körnivå.

Skriptet `rc.sysvinit` letar upp SystemV-initieringsskript, som kan finnas i `/etc/rc.d`, och kör dem, ifall de passar in i körnivån. Detta är användbart i samband med vissa kommersiella programvarupaket, som installerar SystemV-initskript.

4.1.3.5 Andra filer.

Nedan beskrives de övriga systeminitieringsskripten. Deras körning utlöses vanligen ifrån något av de tidigare nämnda huvudskripten, så att man inte behöver göra något annat med dem än att redigera innehållet.

`rc.cdrom` Om detta skript är verksamt (filen körbar), så kommer det att söka efter en CD-ROM-skiva i CD-ROM-enheten och montera denna under `/cdrom`, om det hittar någon.

`rc.gpm` Startar `gpm`, *general purpose mouse services*, musdrivaren för textläge utanför X. Gör det möjligt att kopiera och klistra in med hjälp av musen även i textläge.

`rc.ibcs2` Startar stöd för *Intel Binary Compatibility.* Detta behövs endast, om man avser köra program, som kompilerats för SCO UNIX eller andra kommersiella UNIX för maskiner med intelprocessorer. Det behövs inte för linuxprogram.

`rc.font` Laddar ett skärmtypsnitt, man valt för textläget utanför X.[2]

`rc.local` Innehåller särskilda startkommandon för det egna systemet. Denna fil är tom omedelbart efter en färsk installation, eftersom den är förbehållen lokala systemadministratörer. Detta skript köres efter det, att all annan initiering redan har ägt rum.

Startskripten blir verksamma, genom att man gör dem körbara (exekverbara) med kommandot `chmod`. Omvänt görs de overksamma, genom att man avlägsnar exekveringsrätten från filerna. Upplysningar om, hur man använder `chmod`, står på s. 106.

4.2 Att välja kärna.

Kärnan är den del av operatsystemet, som ger maskinvaruåtkomst, processtyrning och systemstyrning över huvud. Kärnan innehåller stöd för olika maskinvaruenheter, så att det är ett viktigt steg att välja rätt kärna för sitt system.

Slackware levereras med omkring sextio förkompilerade kärnor att välja emellan. Var och en av dem innehåller förutom en standarduppsättning drivare även stöd för olika mer speciell maskinvara. Man kan köra en av de

[2] Kan vara olämpligt i samband med filhanteraren Midnight Commander, eftersom linjerna i Midnight Commander vid bruk av vissa typsnitt ersättes med bokstäver. Ö.a.

förkompilerade kärnorna eller också kan man bygga sin egen kärna ur källkoden. Hur man än gör, måste man förvissa sig om, att kärnan innehåller stöd för den maskinvara, man faktiskt har i systemet.

4.2.1 Katalogen `/kernels` på Slackware-CD:n.

De förkompilerade slackwarekärnorna finns i katalogen `/kernels` på Slackware-CD:n eller via FTP från Slackwares huvudkatalog. Vilka kärnor, som finns tillgängliga, växlar efter hand som nya kommer ut, så dokumentationen i denna katalog under huvudkatalogen är alltid den säkraste källan. Katalogen `/kernels` har underkataloger för var tillgänglig kärna. Underkatalogerna har samma namn som deras motsvarande startdiskett. I var och en av underkatalogerna skall man finna följande filer:

Fil	**Ändamål**
`System.map`	Systemtillordningsfilen för denna kärna
`bzImage` eller `zImage`	Själva kärnan
`config`	Källkodskonfigureringsfilen för denna kärna

När man valt en kärna, kopierar man `System.map` och `config` till katalogen `/boot` och kärnan till filen `/vmlinuz` Därefter installerar man LILO genom att köra `/sbin/lilo`(8) för den nya kärnan, och så startar man om systemet. Konstigare än så är det inte att installera en ny kärna.

De kärnor, vars namn slutar på »`.i`«, är IDE-kärnor. Det vill säga, att de inte innehåller något stöd för SCSI i själva kärnan. De kärnor, vars namn slutar på »`.s`«, är SCSI-kärnor. De innehåller allt IDE-stöd, som finns i kärnorna på »`.i`«, plus SCSI-stöd.

4.2.2 Att kompilera en kärna ur källkoden.

Frågan »bör jag kompilera en kärna för mitt system?« ställes ofta av nya användare. Svaret är ett bestämt kanske. Det finns få tillfällen, när man måste kompilera en för systemet skräddarsydd kärna. De flesta kan använda en förkompilerad kärna och laddbara kärnmoduler för att uppnå ett fullt fungerande system. Däremot kommer man att vilja kompilera en kärna för sitt system, om man förnyar kärnan till versioner, som vi f.n. inte erbjuder i Slackware, eller om man har lappat källkoden (anbringat en *patch* på den) för att få stöd för särskilda maskinvaruenheter, vilka inte ingår i den vanliga kärnkällkoden.

Det är inte så särskilt svårt att bygga en ny kärna. Såsom första steg skall man förvissa sig om, att källkoden till kärnan är installerad i systemet.

Man skall försäkra sig om, att man har installerat paketen i K-serien. Man behöver även ha D-serien installerad: nämligen C-kompilatorn, GNU `make` och GNU binutils. Rent allmänt är det klokt att ha hela D-serien installerad, om man avser ägna sig åt något slags programutveckling. Nu är vi redo att bygga vår kärna:

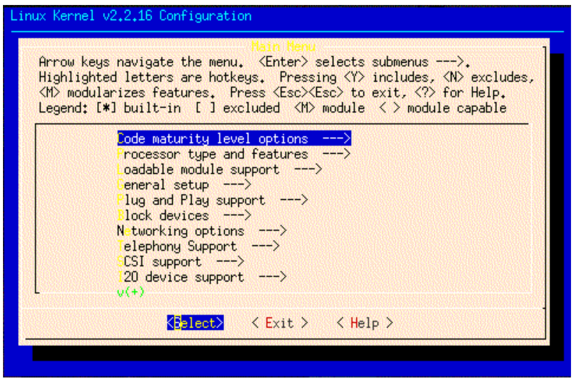
```
$ su -
Password:
# cd /usr/src/linux
```

Det första steget är att försätta kärnans källkod i sitt utgångsläge. Därför kommenderar vi:

```
# make mrproper
```

Nu kan man konfigurera kärnan för sitt eget system. Den nuvarande kärnan erbjuder tre sätt att åstadkomma detta. Det första är det ursprungliga textbaserade systemet med frågor och svar. Det ställer ett antal frågor och sedan bygger det en konfigureringsfil. Detta tillvägagångssätt medför den svårigheten, att man måste börja om igen ifrån början, i fall man väljer fel någonstans. Den i allmänhet föredragna metoden är den menystyrda. Slutligen finns även ett konfigureringsverktyg för X. Man väljer det, man själv föredrar, och använder motsvarande kommando:

```
# make config        (textversion, frågor och svar)
# make menuconfig    (textversion med menyer)
# make xconfig       (X-version, fungerar endast i X)
```



Figur 4.1: Menyversion av kärnkonfigureringsprogrammet.

Nya användare tycker förmodligen, att `menuconfig` är lättast att använda. Det har inbyggd hjälp, som förklarar olika delar av kärnan. När kärnan konfigurerats, stänger man konfigureringsprogrammet. Det skriver då de konfigureringsfiler, som behövs. Nu kan vi förbereda källkodskatalogträdet på att bygga en ny kärna:

```
# make dep
# make clean
```

Nästa steg är att kompilera kärnan (översätta programtexterna till maskinspråk och sätta ihop dem till en enda maskinbegriplig programfil). Först försöker man använda nedanstående kommando för att skapa en `zImage`. Det misslyckas, om kärnan blir för stor. Ingen fara, man kan ändå bygga kärnan med kommandot för att skapa en `bzImage`.

```
# make zImage  (försök detta först)
# make bzImage (försök detta, om make zImage misslyckas)
```

Detta kan ta en stund; hur lång stund beror på processorhastigheten. Under tiden visas några meddelanden ifrån kompilatorn. När kärnavbilden är byggd, skall sådana delar av kärnan byggas, man vid konfigureringen valt att kompilera såsom moduler:

```
# make modules
```

Nu kan vi installera kärnan och modulerna. Det gör man med följande kommandon:

```
# make modules_install
# mv /vmlinuz /vmlinuz.old
# cat arch/i386/boot/zImage > /vmlinuz
# mv /boot/System.map /boot/System.map.old
# cp System.map /boot/System.map
```

Man ersätter här `zImage` med `bzImage`, om man måst bygga en stor kärna. Man får redigera `/etc/lilo.conf` och lägga till ett avsnitt, som gör det möjligt att starta på den gamla kärnan, om den nya inte skulle fungera. Därefter installerar man det nya startblocket genom att köra `/sbin/lilo`. Sedan kan man starta om med den nya kärnan.

4.2.3 Att använda kärnmoduler.

Kärnmoduler är ett annat namn på sådana maskinvarudrivare, som kan stoppas in i en kärna under drift. De möjliggör utvidgande av maskinvarustödet i kärnan, utan att man behöver välja en helt annan kärna eller kompilera en egen.

Moduler kan dessutom laddas in och avlägsnas när som helst, även när systemet är i drift. Detta gör det lätt för systemadministratörer att förnya bestämda drivare. En ny modul kan kompileras, den gamla avlägsnas och en ny laddas, alltsammans utan att man behöver omstarta maskinen.

Modulerna förvaras i katalogen `/lib/modules/<kärnversion>`. De kan laddas vid start genom filen `rc.modules`. Denna fil är rikligt kommenterad och erbjuder exempel för de flesta maskinvarukomponenter. Om man vill se en förteckning över de moduler, som för närvarande är aktiva, kan man använda kommandot `lsmod` (1):

```
# lsmod
Module                  Size  Used by
parport_pc              7220   0
parport                 7844   0   [parport_pc]
```

Här ser man, att vi bara har parallellportsmodulen laddad. När man skall ta bort en modul, använder man kommandot `rmmod` (1). Moduler kan laddas med kommandot `modprobe` (1) eller `insmod` (1). `modprobe` (1) är vanligen säkrare, eftersom det även laddar sådana moduler, som den begärda modulen är beroende av.

Många användare behöver aldrig ladda eller avlägsna moduler för hand. De använder kärnans *autoloader* för modulhanteringen. För att kunna göra detta skall man ta bort kommentartecknet (#) före `/sbin/kerneld` (8) i filen `/etc/rc.d/rc.modules`, och så sätter autoladdaren igång vid systemstart. Den tar hand om laddning och avlägsnande av moduler, efter hand som man begär dem. En begäran är därvid det samma, som att man försöker använda enheten i fråga.

Ytterligare upplysningar om dessa kommandon finns på deras respektive manualsidor samt i filen `rc.modules`.

4.3 Sammanfattning.

Efter genomgång av detta kapitel bör man vara förtrogen med kommandon för genomsökning av filsystemet och med inställningsfilerna i katalogen `/etc`. Dessa färdigheter har man mycken nytta av, medan man lär sig mer om systemet. Vidare bör man nu veta, hur man skall konfigurera och kompilera en linuxkärna av källkoden.

5

Nätverksinställningar.

5.1 Nätverksmaskinvara.

I likhet med de flesta intressanta saker, man kan göra med en dator, fordras vid nätverksanslutning, att man använder särskild maskinvara. Man behöver ett *nätverkskort* (NIC) för att kunna ansluta sig till ett lokalt nätverk (LAN), kanske ett modem för att kunna ringa upp en Internet-leverantör, kanske flera av bäggedera eller ingendera.

Vad inställningarna beträffar, så kan vi indela sådan maskinvara i de två kategorierna *PCMCIA* (för bärbara datorer) och icke-PCMCIA. Bakgrunden till denna något skeva uppdelning är, att den färdiga linuxkärna, som medföljer Slackware, f.n. saknar stöd för PCMCIA. Detta stöd levereras i stället såsom ett särskilt paket, vilket innehåller nödvändiga drivare (i form av kärnmoduler) tillsammans med en del programvara för inställning och hantering av PCMCIA-enheter. Allt det övriga hanteras förstås av den linuxkärna, som medföljer distributionen.

5.1.1 Nätverksmoduler.

Drivrutinerna för sådana nätverksenheter, som kärnan understödjer, ingår i paketet *netmods* (`slakware/n3/netmods.tgz`). Om man ännu inte har installerat netmods, bör man göra det nu. Se s. 168 om hur, det går till att installera programpaket.

Kärnmoduler, som skall laddas vid systemstart, laddas ifrån `rc.modules` i katalogen `/etc/rc.d`. I den medlevererade `rc.modules` ingår ett block med överskriften »`Network device support`«. Om man öppnar `rc.modules` och letar upp detta avsnitt, skall man lägga märke till, att där först sökes efter en körbar fil vid namn `rc.netdevice` i katalogen `/etc/rc.d`. Denna `rc.netdevice` skapas, om `setup` vid systeminstallationen lyckas avkänna nätverkskortet. Om detta gått vägen, är det förmodligen onödigt att läsa vidare här, men annars bör man fortsätta läsa.

Nedanför detta block om `rc.netdevice` i `rc.modules`, vilket börjar med »`if`« och slutar med »`fi`«, står en lista över nätverkskort på rader, som samtliga börjar med kommandot `/sbin/modprobe`. Där letar man upp beteckningen för det egna nätverkskortet, tar bort kommentartecknet `#` (karttecknet för brädgård) från radens början och sparar filen. När man därefter (såsom **root**) kör filen `rc.modules`, så laddas drivaren för nätverkskortet (tillsammans med alla andra moduler, som inte är bortkommenterade med brädgårdstecknet `#` i denna fil). Lägg därvid märke till, att somliga moduler (såsom NE2000-drivaren) fordrar vissa parametrar;[1] det gäller att välja rätt rad med parametrar, som passar till kortet.

5.1.2 PCMCIA-nätverkskort.

PCMCIA-nätverkskort bör vara ännu enklare att ställa in än andra. Man skall förvissa sig om, att man har installerat stöd för pcmcia (som finns i paketet `slakware/a11/pcmcia.tgz`). Se s. 168 i kapitel 16 om hur man installerar programpaket. När man installerar paketet pcmcia, skapas en fil vid namn `rc.pcmcia` i katalogen `/etc/rc.d` och en katalog vid namn `/etc/pcmcia`. Drivare för PCMCIA-kort installeras samtidigt i katalogen `/lib/modules/<version>/pcmcia`. Finessen med paketet `pcmcia` är den, att det försöker av sig själv avkänna instoppande och avlägsnande av PCMCIA-kort. Man skall bara behöva stoppa i PCMCIA-nätverkskortet och lyssna efter det pip, som ljuder, när modulerna laddas. Om man drar ut kortet, skall motsvarande moduler av sig själva avlägsnas ur minnet.

Tyvärr måste man förmodligen kompilera om pcmcia-cs, om man kompilerar om linuxkärnan, så att modulerna med PCMCIA-drivare förnyas och passar till den nya kärnan. Givetvis medföljer källkoden; denna finns i katalogen `source/a/pcmcia` tillsammans med tillhörande dokumentation.

5.2 Nätverkstillbehör.

5.2.1 `ifconfig`.

När nu kärnan kan »tala« med nätverksmaskinvaran, behöver även programvaran kunna meddela kärnan, att den skall skicka iväg några upplysningar,

[1] Detta gäller modulen `ne` för NE2000-kompatibla ISA-kort. Drivaren `ne2k-pci` för NE2000-kompatibla PCI-kort klarar sig dock utan parametrar. Ö.a.

och omvänt. Vi måste därför rigga upp ett gränssnitt (*interface*). Då behöver vi `ifconfig` (8).

`ifconfig` lär man sig bäst genom exempel; det kan vara förnuftigt att ta en titt på befintlig `rc.inet1` (som behandlas i avsnittet med rubriken `rc.inet1` på s. 58) för att se, hur det användes där. Det enklaste och vanligaste fallet tar sig ut någonting i stil med detta:

```
# ifconfig eth0 192.168.1.10 broadcast 192.168.1.255 netmask 255.255.255.0
```

Denna rad öppnar `eth0` (första ethernetgränssnittet; till *token ring* använder man `tr0`, till PPP använder man `ppp0` o.s.v.) med IP-adressen `192.168.1.10`, *broadcast*adressen `192.168.1.255` för allmänt anrop (till hela undernätet `192.168.1`) och en nätmask på `255.255.255.0`. Nätmasken anger, att de tre första talen i notationen *dotted quad* (fyra tal avgränsade med punkter) betecknar nätverket, medan hela återstoden, »`.10`«, gäller vår särskilda värddator (*host*). Om man inte har särskilt finurliga knep för sig, så kan man nästan alltid använda `255.255.255.0` såsom nätmask. Om man har några särskilt finurliga knep för sig, så har man förmodligen redan tillräckliga kunskaper för att inte ha särskilt mycket nytta av denna del i vår bok.

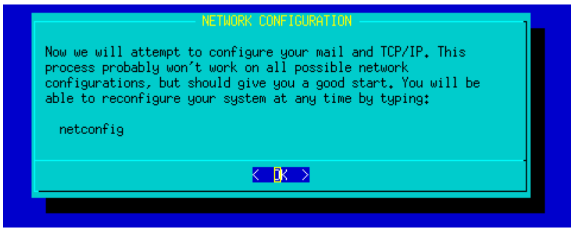
`ifconfig` kan även användas till att avläsa gällande inställningar. Om man kör det utan växlar (flaggor) eller parametrar, så visas en lista över samtliga nätverksgränssnitt och deras inställningar.

5.2.2 `route`.

För att veta, varthän data skall sändas, upprätthåller kärnan en *routingtabell*. Vi skall inte gå in på detaljer här, men man kan avläsa routingtabellen genom att köra `route` (8). Kommandot `route -n` visar en tabell med IP-adresser (siffror) i stället för namn; detta är användbart, om man har bekymmer med namntjänaren (DNS-servern) eller om man inte bryr sig om domännamnens skenvärld. Har man ett enkelt nätverk (vilket de flesta har), så för lyckligtvis 2.2-kärnorna in de nödvändiga posterna i routingtabellen, så att man inte själv behöver befatta sig med detta.

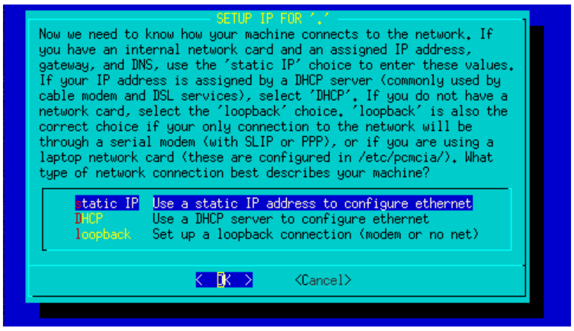
5.2.3 `netconfig`.

Programmet `netconfig` är en del av Slackwares `setup`-program, men liksom flertalet delar av `setup`, så kan det köras fristående. `netconfig` är mycket enkelt och går steg för steg igenom uppriggningen av en grundläggande nätverksförbindelse. Detta program är särskilt användbart, om man inte är förtrogen med `rc`-filerna för nätverksinställning. I `netconfig` hälsas man med skärmbild enligt första figuren.



Figur 5.1: Hälsningsbilden i `netconfig`.

Därefter blir man uppmanad att skriva in ett värddatornamn och ett domännamn för datorn. Man behöver förmodligen bara hitta på några lämpliga namn, såvida man inte riggar upp en server eller någon annan maskin, som många skall använda. Sedan blir man tillfrågad om, huruvida man skall använda statisk IP, DHCP eller bara *loopback*.

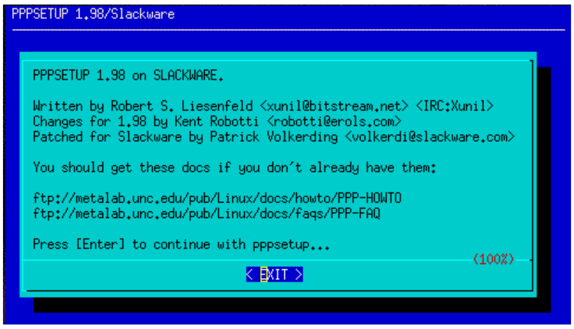


Figur 5.2: Sätta IP-adress på nätverksgränssnittet: statisk, dynamiskt tillordnad med DHCP eller endast *loopback*.

Om vi inte skall vara anslutna till något nätverk, så väljer vi *loopback*. Om man riggar upp en dator, som skall vara ansluten till ett universitets- eller stort kontorsnät, så bör man sannolikt välja DHCP. I annat fall väljer man statisk IP. Om vi nu inte väljer statisk IP, så är vi redan klara. Väljer vi däremot statisk IP, så skall vi nu även föra in datorns IP-adress, nätmask, *broadcast*adress och adress till en namntjänare (DNS-server). `netconfig` förklarar, hur man skall få reda på alla dessa nummer.

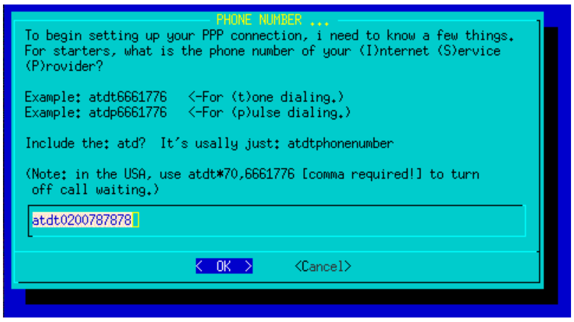
5.2.4 `pppsetup`.

I Slackware ingår tillbehöret `pppsetup` för inställning av uppringd förbindelse till en ISP (internetleverantör). Detta tillbehör ingår i programpaketet `ppp.tgz` i programvaruserie N. `pppsetup` använder samma gränssnitt som `setup`. Kommer man inte ihåg, hur detta gränssnitt användes, så kan man läsa om `setup` på s. 20. `pppsetup` ställer en rad frågor och skapar därefter ett antal inställningsfiler i katalogen `/etc/ppp`. Man kör `pppsetup` som **root**. Här skall vi gå igenom frågorna i tur och ordning.



Figur 5.3: Ordna uppringd internetförbindelse med `pppsetup`.

Telefonnummer (phone number) Första frågan gäller telefonnumret till internetleverantören (ISP). Numera använder man tonval. Om ISP:ns telefonnummer är 0200-787878, så skriver man in `atdt0200787878` i dialogrutan.

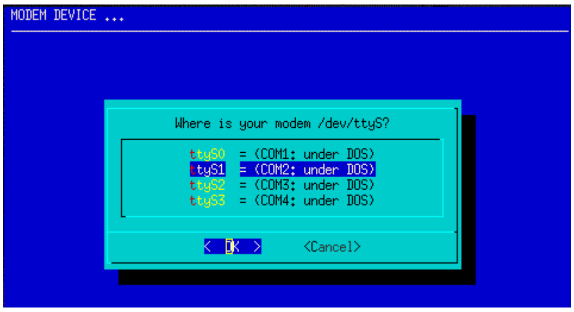


Figur 5.4: Ange telefonnummer till internetleverantören.

Om man har »samtal väntar« påslaget, så bör man se till att stänga av det, när man ringer upp ISP. Koden för att stänga av »samtal väntar« kan läggas in i uppringningskoden.[2] Mellan koden för att stänga av »samtal väntar« och själva telefonnumret skall man sätta in ett komma för att få en paus där under uppringningen.

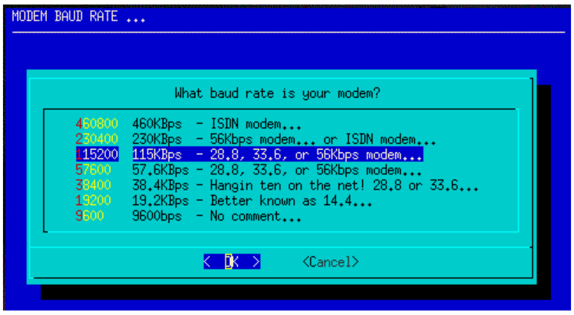
[2]Huruvida detta fungerar i Sverige med Telias kod #43# är inte känt. I så fall skulle man i vårt exempel kunna skriva telefonnumret `atdt#43#,0200787878`. Den som brukar ha »samtal väntar« påkopplat, kan ju prova detta. Ö.a.

Modem device Här anger man, hur modemet är anslutet. Därvid betecknar `ttyS0` den första serieporten (`COM1` i DOS/Windows) och `ttyS1` den andra (`COM2` i DOS/Windows). Om man inte vet, kan man börja med `ttyS0` och prova sig fram.



Figur 5.5: Ange var modemet är anslutet.

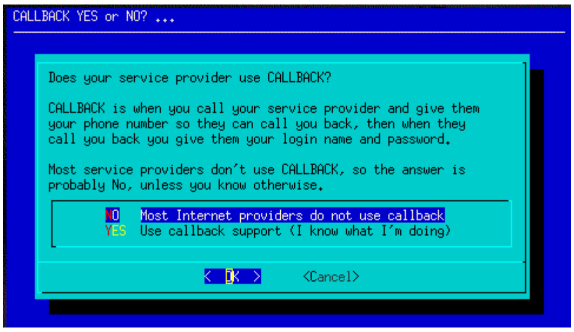
Modem baud rate (Modemhastighet) Man väljer här den porthastighet (hastighet mellan dator och modem), som ligger närmast den, modemet klarar. Har man äldre serieportar på datorn, kan man behöva välja en lägre hastighet. Vart val har flera exempel, som man kan vägledas av.



Figur 5.6: Ange porthastighet.

Callback Det är få internetleverantörer, som erbjuder *callback*, så man kan i normalfallet välja `NO`. *Callback* innebär, att man ringer upp sin ISP, som sedan ringer tillbaka, så att man kan logga in.

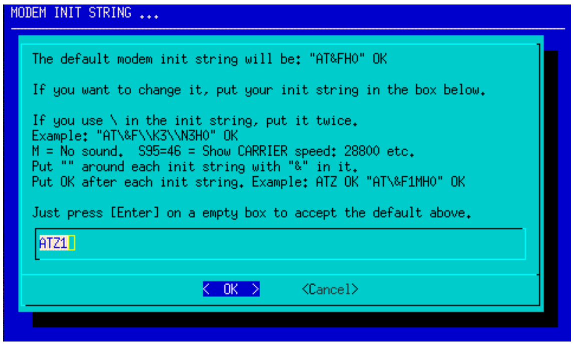
Om man måste använda *callback*, väljer man `YES`. Sedan blir man uppmanad att skriva in sitt telefonnummer, inloggningsnamn och lösenord. Det är inte säkert, att man måste föra in inloggningsnamn och lösenord. Till sist blir man tillfrågad om, vilket slags behörighetskontroll, internetleverantören använder. Om denne använder CHAP eller



Figur 5.7: Ringer internetleverantören tillbaka till oss?

PAP, väljer man `YES`. Man får välja endera senare. Se avsnittet om behörighetskontroll på s. 54.

Modem init string (Inledande kommando till modemet) Här skriver man en initieringssträng till modemet. Man kan i de flesta fall godkänna förvalet »`AT&FH0`« genom att trycka på `Retur`.[3] Se för övrigt den dokumentation, som medlevererats modemet.



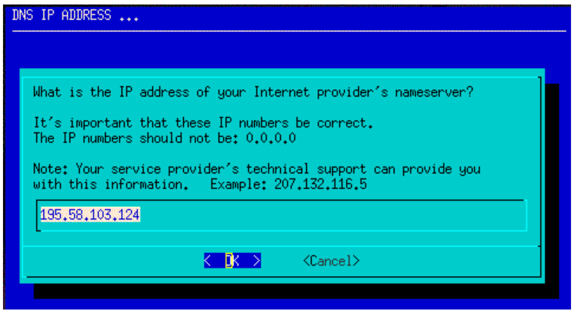
Figur 5.8: Initieringssträng för modemet.

Domain name (Domännamn) Nu skall man skriva in internetleverantörens domännamn. Detta brukar vara någonting i stil med »exempel.net«, »slackware.com« eller liknande. Nå, »slackware.com« är det bestämt *icke*. :)

DNS IP address (IP-adress till namntjänare) Här skriver man in IP-adressen till internetleverantörens namnserver. Denna uppgift får man ifrån sin ISP.

[3] Om modemet har lagrat lämpliga inställningar i ett eller flera minnen, kan man hämta dessa inställningar genom att här välja `ATZ`, `ATZ1` eller liknande. Ö.a.

Figur 5.9: Internetleverantörens domännamn.



Figur 5.10: Internetleverantörens IP-adress för DNS.

Authentication method (Behörighet) Här skall man ange, vilken sorts behörighetskontroll, internetleverantören använder.[4] Det enklaste sättet kan vara, att fråga internetleverantören. Annars får man prova sig fram. Om man emellertid ställes inför inloggnings- och lösenordsprompter, när man ringer upp, så bör man sannolikt välja »`SCRIPT`«.

Figur 5.11: PAP eller CHAP.

[4] De flesta använder PAP (t.ex. Telenordia och Swipnet) eller CHAP (t.ex. Utfors). Ö.a.

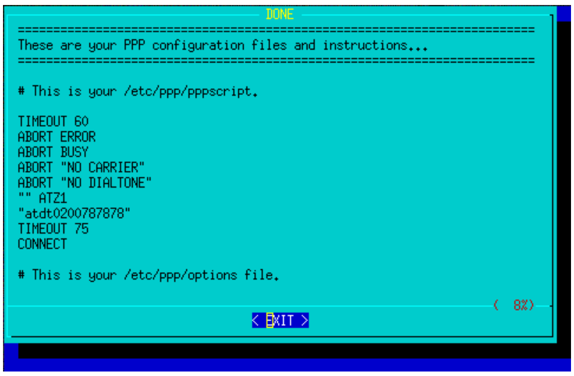
PAP eller CHAP Om man har valt »`PAP`« eller »`CHAP`«, så blir man uppmanad att skriva in ett användarnamn. Man har fått ett användarnamn av internetleverantören. Annars är något misstag begånget, och man får begära ett användarnamn av sin ISP.

Därefter skriver man in det lösenord, som hör ihop med det användarnamn, man har hos internetleverantören.

Chat script (Uppringningsskript) Om man har valt »`SCRIPT`«, så blir man förelagd en rätt lång utläggning om vad, ett *chat script* är för något. Det är viktigt, att man läser igenom detta, eftersom denna text beskriver saken mycket bra. Enkelt uttryckt gäller det att tala om, vilka upplysningar internetleverantören sänder, och vad ens egen dator då skall sända tillbaka, för att man skall kunna logga in.

Done (Färdigt) Till sist får man se, hur de färdiga inställningsfilerna för PPP ser ut. Man kan inte påverka deras innehåll i detta läge, men man kan åtminstone kontrollera dem. När man trycker på `Retur`, sparas filerna och man lämnar `pppsetup`.

Denna skärmbild innehåller även en mängd upplysningar om, hur man skall koppla upp den uppringda förbindelsen och hur man skall koppla ner den igen. Grundtanken är denna: såsom **root** kör man `ppp-go`, så kopplas förbindelsen upp. När man får ett *local IP* och ett *remote IP* är man uppkopplad och har förbindelse med Internet. När man är färdig, kör man `ppp-off` som **root**, så kopplas förbindelsen ner.[5]



Figur 5.12: Innan PPP-förbindelsen provas, får man granska resultatet av att köra `pppsetup`.

[5] Det finns förstås också sådana knep, som gör det möjligt för vanliga användare att ringa upp ISP utan att logga in som **root**. Ö.a.

5.3 Filerna under `/etc`.

5.3.1 `/etc/inetd.conf`.

I ett system, som ställer nätverket i centrum, är det inte ovanligt att ha flera tjänster igång. För var tillgänglig tjänst behövs ett program, som lurpassar på förbindelser. Detta kan bli en belastning för ett system, som kör många sådana tjänare. Det är för att minska denna, som `inetd` har tillkommit. `inetd` är »internetövertjänaren«, »*the internet super-server*« — den sitter och lurar på förbindelser, som kan komma in på många olika »socklar« (*sockets*). När en sådan infinner sig, så är det `inetd`, som startar lämplig server för att hantera uppgiften. På så vis kan behovet av en mängd väntande tjänare (*servrar*) begränsas till en enda.

`/etc/inetd.conf` är inställningsfilen för `inetd`. Däri anges vilka servrar, som skall köras för bestämda anslutningar. Manualsidan för `inetd`(8) innehåller närmare upplysningar om detta. Vi skall likväl kasta ett öga på, hur en rad för en tjänst kan se ut:

```
ftp  stream  tcp  nowait  root  /usr/sbin/tcpd  wu.ftpd -l -i -a
```

Detta är sista raden för FTP-servern. Lägg märke till protokollnamnet (»`ftp`«) först och till sist det kommando, som skall köras vid svar på en begäran om anslutning. I detta är det programmet `/usr/sbin/tcpd`, som skall köras; detta är ett »omslag« (*wrapper*), som tillhandahåller några grundläggande säkerhetsinställningar för den server, det utgör »omslag« omkring. `wu.ftpd` är vår egentliga FTP-server, men `tcpd` kör den i vårt ställe. Ytterligare upplysningar om `tcpd` finns i avsnittet om *tcp_wrappers* på s. 59.

Liksom i många andra systemfiler, så kommenteras raderna i `inetd.conf` med brädgårdstecknet `#`; man kan lägga till och ta bort tjänster i `inetd.conf` genom att helt enkelt lägga till eller ta bort kommentartecken och därefter starta om `inetd`.

5.3.2 `/etc/resolv.conf`.

Detta är den fil, som talar om för övriga systemet, var det skall få tag i DNS-upplysningar, d.v.s. hur namn på värddatorer (med bokstäver) skall översättas till IP-adresser (med siffror). De namntjänare, man använder, skall stå här, liksom den egna värddatorns domännamn. Här är ett exempel på `resolv.conf` (från den bärbara, jag skriver detta på, som kallas för **ninja.tdn**):

```
domain tdn
```

```
nameserver 192.168.1.1
search tdn. slackware.com
```

På första raden anges **ninja**s domännamn; detta är allt, som står efter värddatornamnet i vår adress. Andra raden anger DNS-server (namntjänare) för vårt hemmanätverk. Man kan ha så många sådana, man behöver; de tillfrågas i den ordning, de står, när ett program behöver slå upp den IP-adress (med siffror), som motsvarar ett visst domännamn (med bokstäver).

Sista raden är litet intressantare. Den anger domännamn, som skall antagas tillhöra vårt system. Antag, att vi t.ex. har maskinerna **zuul.tdn** och **hejaz.slackware.com**.

Då kan vi helt enkelt pinga **zuul** och **hejaz** utan att ange domännamn. Detta fungerar därför, att `ping` först försöker lägga »**.tdn**« till **zuul**, hittar motsvarande värddatornamn och är nöjd med det. I fallet »**hejaz**« försöker det först med **hejaz.tdn**

Den maskinen finns inte, så då försöker det med **hejaz.slackware.com** i stället, och träffar då rätt. Märk, att alla de uppräknade domänerna efter `search` måste sluta med en punkt (».«) förutom den sista; om där bara står en enda domän, så är denna den sista och fordrar ingen avslutande punkt.

5.3.3 /etc/hosts.

Filen `hosts` erbjuder det allra enklaste sättet att slå upp domäner. Det är en förteckning över några värddatornamn (*hostnames*) och deras motsvarande IP-adresser. Detta är användbart i ett smärre nätverk, där det inte är lönt att ha någon DNS. Det är också brukbart i sådana fall, när DNS är otillförlitlig o.s.v. Slutligen användes denna fil när systemet startar, då i alla fall inga namntjänare är tillgängliga. Våra skulle kunna se ut så här:

```
127.0.0.1     localhost
192.168.1.32  ninja.tdn ninja
```

Första raden torde vara självförklarande. Andra raden är kanske inte lika självklar. Man kan räkna upp så många namn och alias, man önskar, och då avdelar man dem med mellanslag. Vi har alltså IP-adressen »`192.168.1.32`«, som översättes till »**ninja.tdn**« (och omvänt), men vi kan alltså även använda »**ninja**« som alias, när vi är för lata att skriva även »**.tdn**« (vilket vi för det mesta är).

5.4 rc.inet1.

I filen `/etc/rc.d/rc.inet1` riggar man upp nätverkets »infrastruktur« — där initieras enheterna, där fastställes adresser och vägar (*routes*). Den med Slackware levererade `rc.inet1` innehåller tämligen rikligt med kommentarer, så vi komme enbart att upprepa oss, om vi skulle gå igenom alltsammans här igen.

5.5 rc.inet2.

Filen `/etc/rc.d/rc.inet2` är till för övriga delar av nätverkandet: för att anordna tjänster, starta demoner och för övrigt hantera intressanta nätverksinställningar. Låt oss betrakta ett block som exempel:

```
# Start the NAMED/BIND name server:
if [ -f ${NET}/named ]; then
   echo -n " named"
   ${NET}/named -u daemon -g daemon
fi
```

Här är det rad fyra, som är den viktiga; det är den, som faktiskt startar `named` (8). Återstoden är pynt: »`if`«-satsen förvissar sig om, att det verkligen finns ett program vid namn `named` där, det förväntas finnas, och raden med »`echo`« talar när systemet startar om, att `named` sättes igång. Man skall upptäcka, att de flesta servrar, som startas genom `rc.inet2`, köres ifrån block i stil med dessa; en enkel test, som kontrollerar, ifall det skulle finnas en uppenbar orsak till att servern inte kan köras, ett meddelande om att tjänsten är igångsatt, och därefter kommandon, som startar själva tjänsten. Även `rc.inet2` är ymnigt kommenterad; här kan man ägna en stund åt att läsa kommentarerna.

5.6 NFS (Network File System).

Network File System använder man, såsom namnet säger, till att dela filer mellan flera maskiner i ett nätverk. Finessen med NFS är den, att man kan montera enheter på andra datorer transparent, såsom funnes de på t.ex. en hårddisk i den egna datorn.

För att detta skall kunna äga rum, måste emellertid ett och annat ordnas. För det första måste motsvarande tjänster vara igång på servermaskinen:

dessa är `portmap`(8), `nfsd`(8) och `mountd`(8). För det andra måste servern uttryckligen »exportera« (dela ut) ett filsystemsträd till klienten, vilket åstadkommes medelst filen `exports`(5) i `/etc`.

Första delen av detta räknestycke hanteras därigenom, att man installerar programpaketet `tcpip.tgz` (från programserie N) och låter `rc.inet2` sköta sina saker. `/etc/exports` är litet roligare.

Antag, att vi på datorn **battlecat.tdn** har en katalog med bilder, vilken vi vill montera på **ninja.tdn**. På **battlecat** behövs en rad i `/etc/exports` i stil med denna:

```
/var/media/images   ninja.tdn(ro)
```

Därefter kan man helt enkelt montera denna katalog med bilder i under katalogen `/mnt` på maskinen **ninja**:

```
# mount -t nfs battlecat.tdn:/var/media/images /mnt
```

Tråkigt nog har vi förbjudit oss själva att skriva till denna delade katalog — det sista »`(ro)`« i **battlecat**s `/etc/exports` är ett val, som betyder »endast läsbar« (»*read-only*«). Sådana val måste stå efter klientens namn och inom parentes i en kommaseparerad lista. Ett exempel:

```
/var/media/images   ninja.tdn(rw,no_root_squash)
```

Valet »`rw`« står uppenbarligen för »*read-write*« (»läs- och skrivbar«) — användare på **ninja** får skriva i den delade katalogen. Se manualsidan `exports`(5) beträffande en förklaring av »`no_root_squash`«.

5.7 tcp_wrappers.

tcp_wrappers är ett system för att förhindra (eller uttryckligen tillåta) tillgång till tjänster hos oss ifrån vissa värddatorer. I ett nötskal fungerar det så här:

`inetd` (*the internet super-server*) kör en mängd servrar, varav många är »inslagna« i `tcpd`. Med andra ord är `tcpd` den, som faktiskt kör dessa servrar, men `inetd` vet inte om det (eller bryr sig om det, för den delen). `tcpd` loggar anslutningsförsök och kontrollerar därefter i filerna `/etc/hosts.allow` och `/etc/hosts.deny`, huruvida en anslutning bör tillåtas.

De regler, dessa filer innehåller, kan vara något invecklade, men låt oss förmoda, att **pyramid.tdn** är riktigt påstridig och inte vill låta lilla **mojo.tdn** vara ifred. **mojo.tdn** kan då lägga in en rad i `/etc/hosts.deny`, som ser ut så här:

```
ALL: pyramid.tdn
```

Denna rad torde vara tydlig nog; den hindrar pyramid ifrån att använda några som helst utav de tjänster på **mojo**, som skyddas av `tcpd`. Skulle vi störas av en hel domän, så skulle raden kunna se ut så här:

```
ALL: pyramid.tdn, .stoerande.domaen
```

Men vänta litet! Min kompis Hobbes sitter fast på en maskin i den där **.stoerande.domaen**en, men jag vill låta honom få tillgång till min maskins tjänster (bara inte hans »stoeriga« vänner). Det är en smal sak. Vi lämnar `hosts.deny` som den är, men följande rad i `hosts.allow` släpper in Hobbes:

```
ALL:hobbes.stoerande.domaen
```

Närmare upplysningar finner man under `tcpd` (8), `hosts_access` (5), och `hosts_options` (5). Systemet med *tcp_ wrappers* är mycket smidigare än så här, och det lönar mödan väl att tränga djupare in i det.

5.8 Sammanfattning.

I detta kapitel har vi lärt oss att konfigurera systemet för anslutning till ett nätverk; man bör ha lärt sig, hur man skruvar ihop inställningsfilerna och man bör ha lärt sig några av säkerhetens grunder. Därtill bör man ha lärt sig använda nätverksfilsystemet (NFS) och hur man får det att fungera i sitt system. När man har gjort sitt system till en del av ett nätverk, får man tillgång till flera olika tjänster såsom epost, diskussionsgrupper (news) och WWW-sidor. Se s. 141 om hur man använder några grundläggande nätverksprogram.

6

Fönstersystemet X.

Fönstersystemet X är det vanliga grafiska användargränssnittet (GUI) för alla UNIX-system, vilket även inbegriper Linux. I motsats till vad fallet är i Windows och MacOS, så är det grafiska användargränssnittet i Linux och Unix skilt ifrån operativsystemskärnan. Detta bidrar till systemets stabilitet: om det grafiska användargränssnittet störtar, så drar det inte med sig hela systemet i fallet.

Ett bekymmer med X är, att det länge varit besvärligt att konfigurera för ett visst bestämt system. Emellertid har Slackware version 7 infört en konfigureringsfri X-inställning, som använder *framebuffer*-drivaren. Detta innebär, att man inte nödvändigtvis måste gå igenom avsnittet om `xf86config` på s. 62 eller avsnittet om `XF86Setup` på s. 73. Denna s.k. *framebuffer* fungerar med vilket som helst grafikkort (bildskärmskort), som innehåller normen *VESA 2.0*.[1] Detta betyder, att nästan vilket som helst modernt bildskärmskort fungerar under X. Emellertid är denna *framebuffer* märkbart långsammare än en för systemet och dess grafikkort skräddarsydd X-inställning.

Om man väljer att använda *framebuffer*-servern för X, så skall man installera paketet `xxfb.tgz` (`slakware/x1/xxfb.tgz` ur programvaruserien för X-programvara). Man skall också välja en upplösning för textläge (konsolläge) under installationen. Det för X anbefallna valet är förmodligen det lämpligaste för de flesta.

Om man väljer att konfigurera X särskilt för sitt eget system, så måste man följa anvisningarna i avsnittet om `xf86config` på s. 62 eller avsnittet om `XF86Setup` på s. 73. Det första av följande två avsnitt skildrar bruket av `xf86config` (1), som är ett kommandoradsbaserat program för konfigurering av X. Det därpå följande avsnittet skildrar `XF86Setup` (1), som är en grafisk version av konfigureringsprogrammet.

[1] Det fungerar dock icke med VESA 1.2. Ö.a.

6.1 xf86config.

Programmet `xf86config` är ett av två, vilka kan användas till att konfigurera X för ett visst bestämt system. Dess grundtanke är enkel: man får en rad flervalsfrågor. Av de olika svaren på var fråga väljer man det för systemet närmast träffande. Efter att hela programmet `xf86config` är genomgånget, skrives filen `/etc/XF86Config` (5) och fönstersystemet X är sedan färdigt att användas. Skulle man göra fel under körningen av `xf86config`, så får man avbryta programmet med `Ctrl-c` och börja om ifrån början igen.

Det är till god hjälp, om man vet så mycket som möjligt om bildskärmen och grafikkortet *innan* man startar `xf86config`. Man kan skaffa sig en del upplysningar om bildskärmskortet med hjälp av programmet `SuperProbe`:

```
# SuperProbe
```

Därvid får man en varning om, att programkörningen möjligtvis kan leda till, att systemet hänger sig. Den som skrämmes av denna varning, kan avbryta genom tryckning på `Ctrl-c` inom fem sekunder. I annat fall kan man få en del upplysningar om grafikkortet:

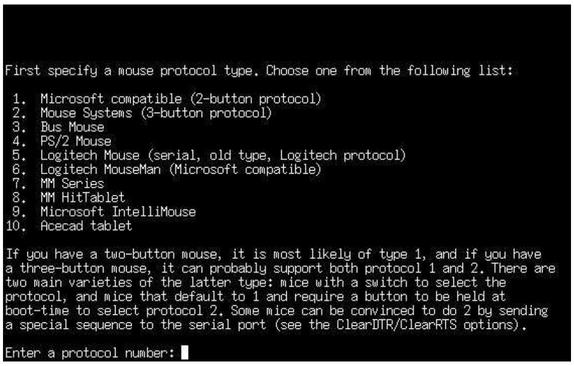
```
First video: Super-VGA
Chipset: ATI 264GT3 (3D Rage Pro) (Port Probed)
Memory: 4096 Kbytes
RAMDAC: ATI Mach64 integrated 15/16/24/32-bit
        DAC w/ clock
        (with 8-bit wide lookup tables)
        (programmable for 6/8-bit wide lookup tables)
Attached graphics coprocessor:
        Chipset: ATI Mach64
        Memory: 4096 Kbytes
```

Så kan det se ut, om man har ett *ATI Rage Pro.* Man skriver ner de upplysningar, `SuperProbe` lämnat, eller också växlar man till en annan virtuell terminal med en tangentkombination i stil med `Alt-F2` o.s.v. och kör `xf86config` där i stället. Upplysningarna om grafikkortet behövs senare, så att skärmbilden med resultatet av `SuperProbe` skall lämnas orörd, om man inte skrivit av dem. Programmet `xf86config` måste köras av **root**, eftersom det skall skriva filer och skapa symboliska länkar i kataloger, där endast **root** har skrivrätt:

```
# xf86config
```

När man väl har startat `xf86config`, så visar det en skärmfull text om vad, det skall göra. Man skall därvid tänka på, att det ingen återvändo finns till föregående skärmbild, om man skulle begå något fel; så det gäller att hela tiden välja med omsorg och tänka sig för, innan man väljer. Eljest kan man bli tvungen att gå igenom hela programmet flera gånger. Efter första skärmbilden med text skall man trycka på `Retur`, såsom man får besked på.

6.1.1 Mouse protocol (musprotokoll)



Figur 6.1: Ange musprotokoll.

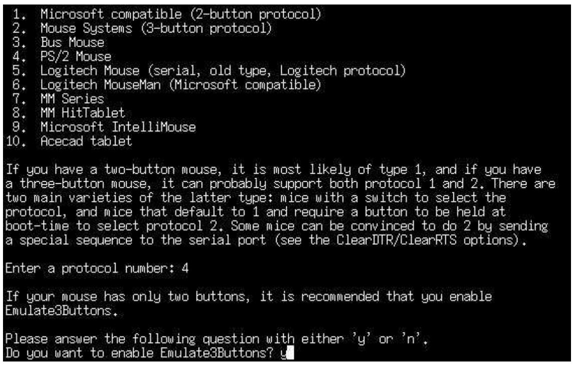
Här väljer man mus ifrån en lista. Nuförtiden är möss i allmänhet endera av typen *PS/2* eller *Microsoft Intellimouse*. Äldre möss kan fordra, att man väljer någon av de andra typerna på listan. Om man har en *Logitech*-mus med PS/2-kontakt, är den troligen PS/2-kompatibel, så att man väljer *PS/2*.[2]

6.1.2 Emulate3Buttons (emulera tre knappar)

Om musen bara har två knappar, så kan man välja att emulera en tredje knapp.[3] När sedan vänster och höger musknapp trycks ner tillsammans, så tolkas det som en tryckning på den mittknapp, vilken egentligen saknas. Eftersom många program drar nytta av tredje (mellersta) knappen, så anbefalles det, att man gör treknappsemuleringen verksam.

[2]Hjulet på en logitechmus kan man få att fungera genom att i efterhand redigera `/etc/XF86Config` och ändra protokollet `PS/2` till `MouseManPlusPS/2` samt lägga till `ZAxisMapping 4 5` på en ny rad. Till att börja med väljer man emellertid endast *PS/2* i konfigureringsprogrammet. Ö.a.

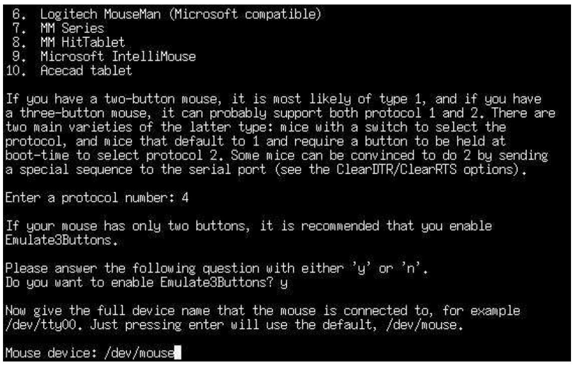
[3]I X har man nytta av en treknapparsmus och använder ofta mittknappen. Ö.a.



```
 1.  Microsoft compatible (2-button protocol)
 2.  Mouse Systems (3-button protocol)
 3.  Bus Mouse
 4.  PS/2 Mouse
 5.  Logitech Mouse (serial, old type, Logitech protocol)
 6.  Logitech MouseMan (Microsoft compatible)
 7.  MM Series
 8.  MM HitTablet
 9.  Microsoft IntelliMouse
10.  Acecad tablet

If you have a two-button mouse, it is most likely of type 1, and if you have
a three-button mouse, it can probably support both protocol 1 and 2. There are
two main varieties of the latter type: mice with a switch to select the
protocol, and mice that default to 1 and require a button to be held at
boot-time to select protocol 2. Some mice can be convinced to do 2 by sending
a special sequence to the serial port (see the ClearDTR/ClearRTS options).

Enter a protocol number: 4

If your mouse has only two buttons, it is recommended that you enable
Emulate3Buttons.

Please answer the following question with either 'y' or 'n'.
Do you want to enable Emulate3Buttons? y
```

Figur 6.2: Emulera tre knappar på tvåknapparsmus.

6.1.3 Mouse device name (musens enhetsnamn)



```
 6.  Logitech MouseMan (Microsoft compatible)
 7.  MM Series
 8.  MM HitTablet
 9.  Microsoft IntelliMouse
10.  Acecad tablet

If you have a two-button mouse, it is most likely of type 1, and if you have
a three-button mouse, it can probably support both protocol 1 and 2. There are
two main varieties of the latter type: mice with a switch to select the
protocol, and mice that default to 1 and require a button to be held at
boot-time to select protocol 2. Some mice can be convinced to do 2 by sending
a special sequence to the serial port (see the ClearDTR/ClearRTS options).

Enter a protocol number: 4

If your mouse has only two buttons, it is recommended that you enable
Emulate3Buttons.

Please answer the following question with either 'y' or 'n'.
Do you want to enable Emulate3Buttons? y

Now give the full device name that the mouse is connected to, for example
/dev/tty00. Just pressing enter will use the default, /dev/mouse.

Mouse device: /dev/mouse
```

Figur 6.3: Musens enhetsfil i katalogen `/dev`.

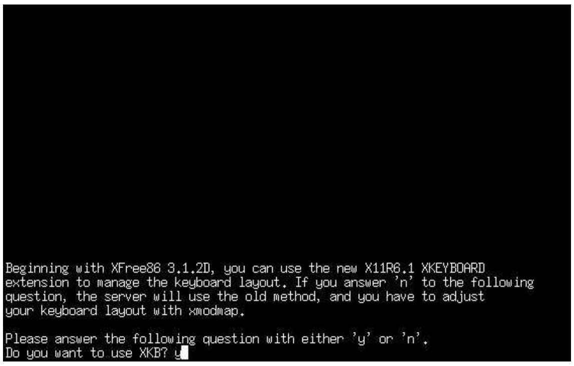
Förvalet `/dev/mouse` är vanligen det riktiga. Har man emellertid anslutit musen på något ovanligt sätt, så kan man behöva ändra detta. Med flertalet seriemöss och PS/2-möss är dock förvalet det bästa valet.[4]

6.1.4 XKEYBOARD extension (XKB)

Man vill förmodligen använda XKB. Om man inte väljer detta, kan det hända, att vissa tangenter (t.ex. *Backspace* och *Delete*) uppför sig på oväntat sätt, eller att vissa tangentkombinationer inte fungerar.[5]

[4]Förutsatt att man vid slackwareinstallationen valt rätt port för musen, så att `/dev/mouse` verkligen är en symbolisk länk till `/dev/psaux` för en PS/2-mus eller till `/dev/ttyS0` för en seriemus på första serieporten (som kallas `COM1` i DOS).

[5]T.ex. i *Gnome* och *Enlightenment*. Om man vill komponera särskilda tangentbord, kan man senare kopiera en tangentbordsfil ur katalogen `/usr/share/xmodmap` till en fil

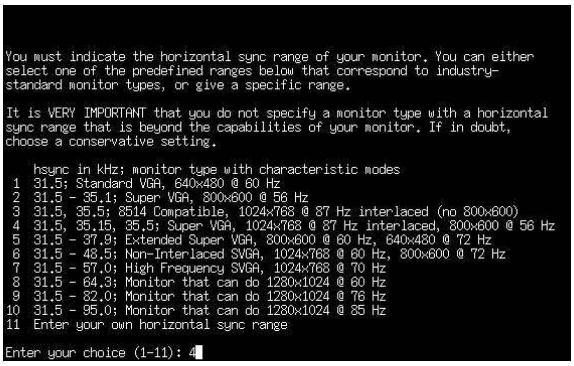


Figur 6.4: Välja XKB för tangentbordshantering.

6.1.5 Bindings for alt keys (Alt-tangenterna)

Om man vill kunna få fram ytterligare tecken förutom dem, man får med eller utan tryckning på omskiftartangenten, så skall man godta det val, som erbjudes här. Det innebär nämligen bl.a., att högra *Alt*-tangenten kommer att fungera just så, som man förväntar sig att en *AltGr*-tangent skall fungera. Den är ju också mycket riktigt märkt »*AltGr*«.

6.1.6 Horizontal sync range (linjefrekvens)



Figur 6.5: Ange gränser för bildskärmens möjliga linjefrekvenser.

Detta är den första fråga, som har med bildskärmen att göra. Det är viktigt, att man gör ett klokt val här. Man skall akta sig för att välja ett område för horisontell synkronisering (linjefrekvens), vilket ligger utanför det område, bildskärmen är avsedd att arbeta med. Detta är av mindre betydelse för

---

vid namn `.Xmodmap` i sin hemkatalog och redigera denna fil efter egen smak. Ö.a.

någorlunda nya bildskärmar, eftersom de inte ens försöker arbeta utanför dessa gränser. Äldre skärmar kan man emellertid förstöra, om man försöker driva dem utanför föreskrivna gränser. Är man osäker, bör man välja försiktigt och nöja sig med lägre linjefrekvens hellre än fördärva sin bildskärm.

För nyare skärmar kan man förmodligen välja 31.5–48.5 eller 31.4–57.0. Till högvärdigare skärmar jan man välje större frekvensområden. Om man vet bildskärmens gränser för linjefrekvensen (horisontella frekvensen), kan man välja att föra in sina egna värden.

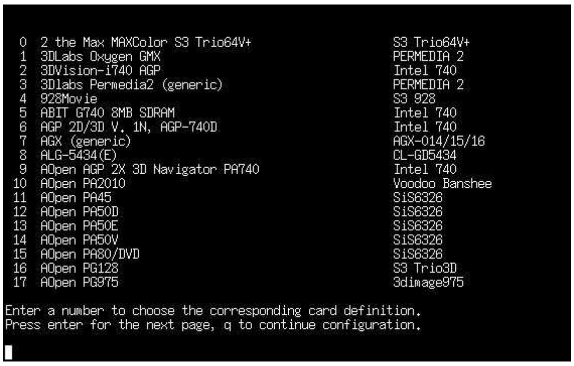
6.1.7 Vertical sync range (bildfrekvens)

Figur 6.6: Ange gränser för bildskärmens möjliga bildfrekvenser.

Återigen är det nödvändigt att känna till bildskärmens specifikationer, för att man skall kunna besvara denna fråga om vertikal synkronisering (bildfrekvens). Skulle man hysa tvivel, är det bäst att välja försiktigt, d.v.s. ett smalt frekvensområde. Man håller sig förmodligen på den säkra sidan, om man väljer 50–90 eller 50–100. Om man känner till bildskärmens bildfrekvens (vertikala frekvensen), kan man välja att föra in sina egna värden.

6.1.8 Identification strings (bildskärmsbeskrivning)

Man får här i tur och ordning skriva in tre uppgifter om bildskärmens typ, tillverkare och modellbeteckning. Detta är inte särskilt viktigt. Det går bra att skriva in vad som helst här, eller också kan man helt enkelt trycka på `Retur` utan att skriva något. Det, man skriver in här, användes i konfigureringsfilen som kännemärken i olika avsnitt.



Figur 6.7: Ange grafikkort eller grafikkrets.

6.1.9 Video card database (grafikkortsdatabas)

Nästa betydande avsnitt i X-konfigureringen handlar om grafikkortet (bildskärmskortet). Den dokumentation, som är tillgänglig för kortet, kommer nu väl till pass tillsammans med de upplysningar, `SuperProbe` har lämnat. För att få se grafikkortsdatabasen måste man svara med bokstaven »`y`« och först därefter trycka på `Retur`. Trycker man bara på `Retur` utan att skriva »`y`«, så går man förbi kortdatabasen och vidare till nästa avsnitt.

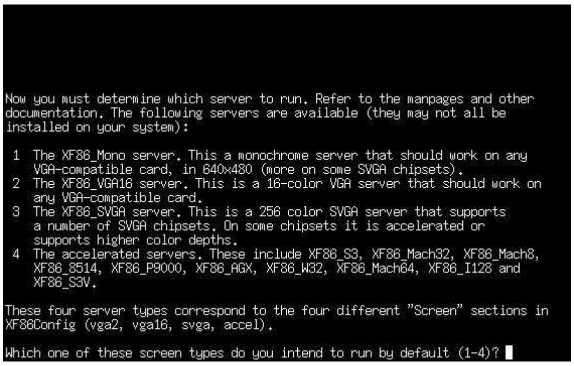
Där finns mer än 800 kort i databasen. Vänstra spalten innehåller ett nummer för vart kort tillsammans med kortets namn. Högra spalten innehåller namnet på kortets kretsuppsättning (*chipset*). Man trycker gång på gång ner tangenten `Retur`, tills det kort, man själv har, visas på listan. Först då matar man in kortets nummer (står i vänstra spalten) och trycker på `Retur`. Kan man inte hitta kortet på listan, finns ytterligare några möjligheter. För det första kan man använda `SuperProbe`:s uppgift om *chipset* och på listan leta upp ett kort, som använder samma *chipset* (kretsuppsättning). Eller också kan man välja en generell SVGA-drivare. Många kort, som ingen egen X-server har, fungerar med SVGA-servern, så att detta möjligen kan vara ett ofarligt val.

När man har bestämt sig för ett kort, så får man en del ytterligare upplysningar. Om vi fortsätter på exemplet med *ATI Rage Pro* ovan, så skulle vi få följande upplysningar:

```
Your selected card definition:

   Identifier: ATI Mach64
   Chipset:    ATI-Mach64
   Server:     XF86_Mach64
   Do NOT probe clocks or use any Clocks line.
```

Nu är det dags att förvissa sig om, att motsvarande serverpaket verkligen är installerat. X-servern XF86_Mach64 finns i paketet `xma64.tgz`. Rätt serverpaket måste vara installerat, eljest kommer X icke att köra igång.

6.1.10 Which server to run? (Vilken server skall köras?)



Figur 6.8: Välja X-server.

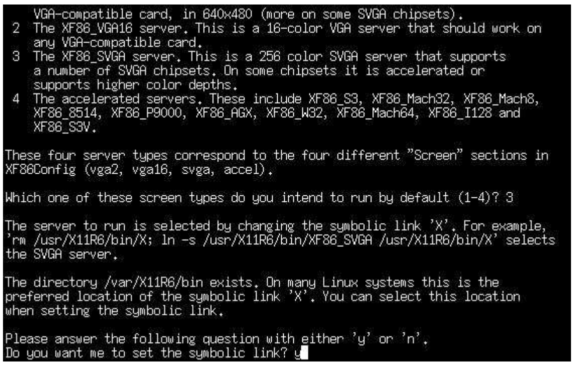
Nästa fråga förelägger användaren ett val emellan flera olika möjliga X-servrar. Om man har gjort rätt vid valet av grafikkort, så kan man utan vidare trycka på `Retur`. Därigenom väljes nämligen den server, som motsvarar det angivna kortet. I annat fall kan man välja mellan *Mono*-servern, *VGA16*-servern, *SVGA*-servern eller en *accelererad* server. Lämpligaste val är att använda den server, som motsvarar angivet kort.

6.1.11 Setting the symbolic link (sätta den symboliska länken)

Välj »`y`«, så sättes den symboliska länken. Detta innebär, att filnamnet `X` förknippas med filnamnet på vald X-server genom att `X` skapas såsom länk till serverns filnamn.

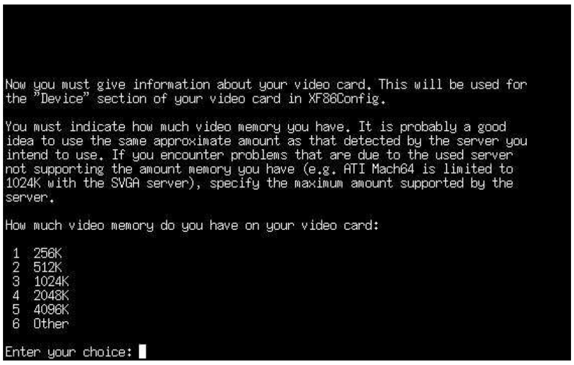
6.1.12 Video memory (grafikkortsminne)

Här anger man, hur mycket minne, som sitter på grafikkortet. Ibland kan man få motsvarande information genom att köra `SuperProbe`. Har man mer än förvalen, väljer man »`Other`« och skriver in rätt minnesstorlek i *kilobyte* (kB); 8 MB (megabyte) är t.ex. lika med 8192 kB.



```
    VGA-compatible card, in 640x480 (more on some SVGA chipsets).
 2  The XF86_VGA16 server. This is a 16-color VGA server that should work on
    any VGA-compatible card.
 3  The XF86_SVGA server. This is a 256 color SVGA server that supports
    a number of SVGA chipsets. On some chipsets it is accelerated or
    supports higher color depths.
 4  The accelerated servers. These include XF86_S3, XF86_Mach32, XF86_Mach8,
    XF86_8514, XF86_P9000, XF86_AGX, XF86_W32, XF86_Mach64, XF86_I128 and
    XF86_S3V.

These four server types correspond to the four different "Screen" sections in
XF86Config (vga2, vga16, svga, accel).

Which one of these screen types do you intend to run by default (1-4)? 3

The server to run is selected by changing the symbolic link 'X'. For example,
'rm /usr/X11R6/bin/X; ln -s /usr/X11R6/bin/XF86_SVGA /usr/X11R6/bin/X' selects
the SVGA server.

The directory /var/X11R6/bin exists. On many Linux systems this is the
preferred location of the symbolic link 'X'. You can select this location
when setting the symbolic link.

Please answer the following question with either 'y' or 'n'.
Do you want me to set the symbolic link? y
```

Figur 6.9: Sätta symbolisk länk.



```
Now you must give information about your video card. This will be used for
the "Device" section of your video card in XF86Config.

You must indicate how much video memory you have. It is probably a good
idea to use the same approximate amount as that detected by the server you
intend to use. If you encounter problems that are due to the used server
not supporting the amount memory you have (e.g. ATI Mach64 is limited to
1024K with the SVGA server), specify the maximum amount supported by the
server.

How much video memory do you have on your video card:

 1  256K
 2  512K
 3  1024K
 4  2048K
 5  4096K
 6  Other

Enter your choice:
```

Figur 6.10: Ange grafikkortets minnesstorlek.

6.1.13 Identification strings (grafikkortsbeskrivning)

Här får man tillfälle att skriva in ytterligare tre textrader, som denna gång är förknippade med grafikkortet. Liksom fallet var med bildskärmstexterna, så kan man skriva vad som helst här eller bara trycka på `Retur`, ifall man inte vill ha något namn på grafikkortet i inställningsfilen.

6.1.14 RAMDAC (DA-omvandlare på grafikkortet)

Det är enbart vid vissa X-servrar för S3-, AGX- och W32-kort, som man behöver ange RAMDAC.[6] Genom `SuperProbe` kan man få reda på vilken RAMDAC, som sitter på grafikkortet. Man bläddrar igenom listan, tills rätt krets visas; då skriver man in motsvarande nummer och trycker på `Retur`. Behöver man inte ange RAMDAC, så skriver man »`q`« och trycker på `Retur`.

[6] Inte för alla S3-kort heller; t.ex. inte för S3 Trio64 eller S3 Trio64V+.

6.1.15 Clockchip setting (programmerbar taktkrets)

Figur 6.11: Ofta behöver man inte välja *clockchip*, utan kan gå vidare.

Om kortet har en programmerbar taktkrets, väljer man en ifrån listan. Flertalet kort har ingen sådan *clockchip*, så att man för det mesta helt enkelt kan trycka på `Retur` och gå vidare. Om en sådan klockkrets finnes på grafikkortet, så bör man få reda på det vid körning av `SuperProbe`.

6.1.16 Clocks line

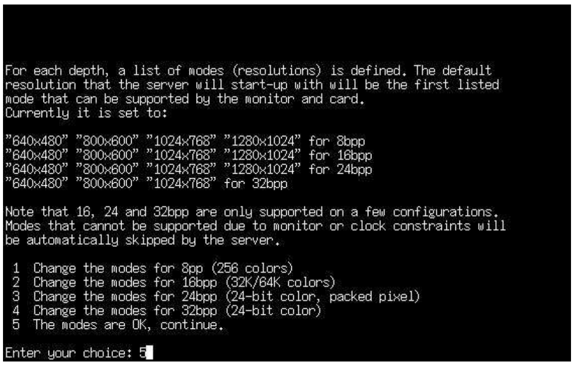
Nästa skärm berättar om, vad en *clocks line* är för någonting. Såsom det där förklaras, så behöver man ingen klockrad till moderna utrustningar. Därefter blir man tillfrågad om, huruvida programmet skall försöka avkänna förekomst av klocka. Det talar även om, huruvida kortet behöver avkännas eller ej. I fallet med *ATI*-kortet, så säger `xf86config` t.ex.:

```
The card definition says to NOT probe clocks.
```

Om det säger något i den stilen, så väljer man »`n`« såsom svar på frågan om att avkänna (*probe for*) klockor. Mycket gamla grafikkort behöver avkännas, men `xf86config` talar om vilketdera, som behöver göras.

6.1.17 Video modes (upplösningar och antal färger)

Nu är det dags att välja vilka upplösningar, X-servern skall använda vid de givna färgdjupen 8bpp, 16bpp, 24bpp och 32bpp. Var och en av dem uppvisar en lista över de upplösningar, som är möjliga vid färgdjupet ifråga. När man startar X, så börjar det med att visa ett förvalt färgdjup (t.ex. 8bpp lika med 256 färger) och använda den upplösning, som står först på listan för detta färgdjup (t.ex. 640x480 bildpunkter, om man inte har valt något annat). Om



Figur 6.12: Välja upplösningar för olika färgdjup i X.

man vill, att X skall starta med någon annan upplösning än den förvalda, så är det dags nu att ändra på förinställningen.

Om man godkänner den ursprungliga ordningen på upplösningarna, så kan man välja »OK«, vilket leder vidare till nästa steg i konfigureringen. I annat fall väljer man det färgdjup, man vill ändra på. Antag exempelvis, att man får följande val:

```
"640x480" "800x600" "1024x768" "1280x1024" for 8bpp
"640x480" "800x600" "1024x768" "1280x1024" for 16bpp
"640x480" "800x600" "1024x768" "1280x1024" for 24bpp
"640x480" "800x600" "1024x768" for 32bpp
```

Om man då vill, att X såsom förval skall starta med någon annan upplösning, så väljer man först siffran som står först på raden med det färgdjup, man vill ändra upplösningen för. Man slår t.ex. in siffran 1 för att ändra på upplösningarna för 8bpp och trycker på `Retur`. Därefter följer man de anvisningar, `xf86config` ger. Ville man bara kasta om ordningen på upplösningarna, så skulle man kunna svara som så:

```
Which modes? 5432
```

Här kan man även ta bort upplösningar. Om grafikkortet inte klarar av upplösningen 1280x1024, så finns ingen anledning att försöka ens. Man skulle då kunna ta bort denna upplösning genom att svara:

```
Which modes? 432
```

När man har valt upplösningar för ett visst färgdjup, så blir man tillfrågad om, huruvida man vill ha en virtuell skärmbild, som är större än den fysiska skärmbilden. En virtuell skärmbild är en skärmbild, som är större än den bild, som kan visas på den verkliga bildskärmen. När man flyttar omkring muspekaren på den virtuella skärmen, så rullar den skärmbilden ett stycke innan den når fram till kanten. Där försiggår således en panorering i höjd- och sidled över skärmbilden, så att den verkliga skärmen visar en bit i taget av den virtuella skärmbilden. Just därför, att man inte ser hela skärmbilden på en gång, kan en sådan virtuell skärm vara tämligen irriterande. Det kan likväl vara en intressant leksak, så att somliga kanske har lust att prova på.

Efter detta val kommer man tillbaka till listan över upplösningar vid olika färgdjup. Sedan upplösningarna för färgdjupet 24bpp ändrats, skulle denna lista kunna se ut på följande vis:

```
"640x480" "800x600" "1024x768" "1280x1024" for 8bpp
"640x480" "800x600" "1024x768" "1280x1024" for 16bpp
"1280x1024" "1024x768" "800x600" "640x480" for 24bpp
"640x480" "800x600" "1024x768" for 32bpp
```

Man fortsätter med att ändra ordningen på upplösningarna, tills man är nöjd. När man är färdig med detta avsnitt, väljer man »OK« och fortsätter.

6.1.18 Write the config file (skriva inställningsfilen)

Nu skall X vara färdigkonfigurerat. Då frågar `xf86config` om det skall skriva inställningsfilen till `/etc/XF86Config`. Vill man kunna köra X, så skall man svara »`y`« på denna fråga, eftersom detta är den inställningsfil, X kommer att söka efter.

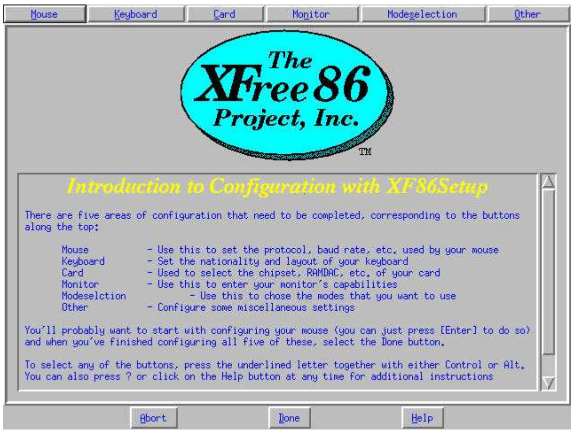
Om vi antar, att alla frågor besvarats riktigt, och motsvarande X-serverpaket installerats, så skall man nu kunna starta X med följande kommando:

```
$ startx
```

Har man installerat KDE eller Gnome, så skall någon av dessa skrivbordsmiljöer uppenbara sig nu. I annat fall får man köra `xwmconfig` och välja den fönsterhanterare, man vill ha såsom förval. Fönsterhanterare (*window managers*) skall behandlas senare i detta kapitel. Programmet `xwmconfig` ställer in en viss fönsterhanterare såsom förval enbart för den användare, som kör `xwmconfig`. Om man har flera användare i systemet, så får var och en av dem välja fönsterhanterare själv.

Det finns några tangentkombinationer, som kan komma väl till pass, när man använder X. Om man någon gång måste lämna X men inte kan stänga det på riktigt sätt, så finns för detta ändamål en tangentkombination, som *tvingar* ner X. Om man trycker på `Ctrl`+`Alt`+`Backspace`, så dödas X och man kommer tillbaka till kommandoraden. Man kan även växla mellan X och kommandoradsterminalerna utan att stänga X. Det gör man genom att trycka på `Ctrl`+`Alt` tillsammans med någon av funktionstangenterna `F1` till och med `F6`, vilket kan jämföras med växling mellan de virtuella terminalerna utanför X, där man använder tangentkombinationer med `Alt` och någon funktionstangent från `F1` till och med `F6`, således utan `Ctrl`-tangenten. Själva X-körningen finns på terminal 7, så att man kan komma tillbaka in i X genom tryckning på `Alt`+`7`. Slutligen kan man även ändra upplösning med X igång. Tryckning på `Ctrl`+`Alt` tillsammans med `+` på räknemaskinknappsatsen ger närmast högre upplösning, medan tryckning på `Ctrl`+`Alt` tillsammans med `-` på räknemaskinknappsatsen ger närmast lägre upplösning. När högsta eller lägsta upplösning redan är nådd, leder nästa tryckning till, att varvet startas om ifrån början.

6.2 XF86Setup.



Figur 6.13: Startbilden i `XF86Setup`.

Ett annat sätta att konfigurera X är att använda `XF86Setup`, som är ett grafiskt inställningsprogram och ingår i paketet `xset.tgz`. För att det skall fungera, behöver man även installera paketet `xvg16.tgz`.

Man kör XF86Setup genom att logga in som **root** och skriva:

```
# XF86Setup
```

Om man redan har en fil vid namn `/etc/XF86Config` (t.ex. om man redan har konfigurerat X en gång), så blir man tillfrågad om, huruvida man vill använda inställningarna i befintlig `XF86Config` till förval. I annat fall går programmet direkt till grafikläge.

`XF86Setup` är väldigt likt `xf86config`. Det ställer samma slags frågor, men det lägger fram dem i en grafisk miljö. Om man undrar över, vad frågorna betyder, så kan man upplysa sig om det i föregående avsnitt. `XF86Setup` har även en mängd inbyggd hjälp, så att man inte skall ha några svårigheter med att komma underfund med det.

6.3 Inställningar för X och X-program.

6.3.1 `xinitrc` och `~/.xinitrc`.

`xinit` (1) är det program, som egentligen sätter igång X; det anropas utav `startx` (1), så att man kanske inte märker detta (och förmodligen inte behöver märka det). Dess inställningsfil avgör emellertid vilka program (inberäknat och i synnerhet fönsterhanteraren), som skall köras, när X startar. `xinit` söker först efter en `.xinitrc`-fil i användarens hemkatalog. Om denna fil finnes, så användes denna, eljest systemets `/var/X11R6/lib/xinit/xinitrc`. Här är ett exempel på en `xinitrc`:

```
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps
if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# start some nice programs
twm &
xclock -geometry 50x50-1+1 &
xterm -geometry 80x50+494+51 &
xterm -geometry 80x20+494-0 &
exec xterm -geometry 80x66+0+0 -name login
```

Alla dessa »`if`«-block finns med för att infoga inställningar ifrån andra filer. Vi skall om ett ögonblick gå vidare till `.Xresources`, men vi lämnar `.Xmodmap` för sig själv. Filens intressanta del finns framemot slutet, där olika program startas. Denna X-körning börjar med fönsterhanteraren `twm`(1), en klocka och tre terminalfönster. Lägg märke till `exec` före det sista `xterm`. Det ersätter körande skal (det, som exekverar detta `xinitrc`-skript) med kommandot `xterm`(1). När användaren avslutar detta `xterm`, så avslutas även X-körningen.

6.3.2 `.Xresources` och `.Xdefaults`.

Många X-program använder ett system, som kallas för *X Resource Database* för att inhämta diverse användarförval (typsnitt, färger o.s.v.). Denna databas underhålles genom programmet `xrdb`(1), som man sannolikt aldrig behöver köra direkt. I stället köres det i Slackware från `xinitrc`. Den fil, som `xinitrc` anger såsom källa för `xrdb` att söka olika val i, är `~/.Xresources`. `xrdb` laddar även `.Xresources`, så endera av dessa filnamn fungerar. En minimal `.Xresources` ser ut så här:

```
xterm*background: black
xterm*foreground: gray
xterm*scrollBar: true
xterm*font: -*-lucidatypewriter-*-r-*-*-15-*-*-*-*-*-*-*
```

Dessa fyra rader anger inställningar för programmet `xterm`. En X-resurs förtecknas i en av dessa filer på följande vis:

```
program*val: inställning/värde
```

Sålunda torde exemplet med `.Xresources` ovan vara tämligen självförklarande. Man skall inte bli konfys av »`font`«-raden; typsnitt anges i X alltid på detta sätt.

6.4 X-servrar och fönsterhanterare.

X-fönstersystemet konstruerades ursprungligen för att fungera transparent över ett nätverk. En stor server skulle sköta om det faktiska körandet av X-programmen, men de skulle visas på olika klientmaskiner någon annanstans i nätverket. Förmågan att kunna visa program på fjärrdatorer kan vara till stor fördel. Den största nackdelen med detta nätverkskoncept är, att det är mindre säkert än att köra tillämpningsprogram på en lokal maskin, och det tar upp mycket bandbredd i nätet. Detta diskuteras längre fram i avsnittet *Exportera skärmvisning* på s. 78.

T.o.m när man kör X på sin egen maskin, så har man likväl att göra med en klient-server-modell. Servern är den del av X, som är knuten till grafikkortet. När vi konfigurerade X och talade om, vilket slags grafikkort vi hade, så bestämde vi, vilket serverprogram för X, som skulle komma till användning. Klientdelen är alla andra program, man kör under X. En särskild klient, som kallas *fönsterhanterare*, ansvarar för utseende och handhavandestil i en viss X-körning. Fönsterhanteraren diskuteras mer ingående senare.

Fönsterhanterarens uppgift är att hantera uppritandet av fönster på skärmen med program inuti dessa fönster, liksom att hantera inmatning ifrån mus och tangentbord. De första fönsterhanterarna gjorde detta och inte så mycket därutöver. Dagens fönsterhanterare är mycket mer invecklade program och kan skräddarsys på nästan alla upptänkliga sätt och vis. De har allehanda finurliga valmöjligheter, som kan få ens eget skrivbord att se annorlunda ut än alla andras.

Förekomsten av flera olika fönsterhanterare är förvisso en skiljelinje mellan Linux och Windows vad skrivbordet beträffar. I Windows är man i grund och botten hänvisad till en enda fönstermiljö. I Linux kan man köra en av många olika fönsterhanterare, var och en med olika utseende och olika egenskaper. Somliga är benägna att kalla detta för en svaghet, eftersom det inte finns något enhetligt, överensstämmande utseende. Emellertid håller flertalet linuxanvändare detta för en styrka, eftersom var och en kan rigga upp sitt system helt och hållet efter egen smak.

6.5 Att välja skrivbordsmiljö.

I flera år har Unix varit nästan uteslutande ett serveroperativsystem med undantag för några professionella arbetsstationer. Endast de tekniskt intresserade har kunnat tänkas använda ett unixliknande operativsystem, och användargränssnittet har återspeglat detta. Grafiska användargränssnitt har brukat vara tämligen påvra och konstruerade för att köra ett fåtal nödvändigtvis grafiska tillämpningar såsom CAD-program och bildbehandlingsprogram. Fil- och systemhantering har vanligen utförts på kommandoraden. Flera leverantörer (Sun Microsystems, Silicon Graphics o.s.v.) har sålt arbetsstationsinstallationer med ansatser till sammanhängande, följdriktigt utseende och konsekvent handhavandestil, men det rika utbud av programmeringsverktyg för framställning av grafiska användargränssnitt, som stått till programutvecklarnas förfogande, har oundvikligen medfört en upplösning av skrivbordets enhetlighet. En blädderlist kanske inte har sett likadan ut i två olika tillämpningsprogram. Menyer kanske har dykt upp på olika ställen. Programmen har kunnat ha olika knappar och kryssrutor. Färgskalorna har varit vidsträckta och vanligen hårdkodade i var särskild programmeringsverktygssats. Så länge användarna främst har varit tekniska yrkesmän, så har inget utav detta haft så särskilt stor betydelse.

Med de fria unixliknande operativsystemens intåg och de grafiska tillämpningarnas växande antal har X på senare tid erhållit en vidsträckt skrivbordsanvändarbas. Flertalet användare är förstås vana vid den konsekvent enhetliga stil, som erbjudes dem i Microsofts Windows eller Apples MacOS; avsaknaden av sådan enhetlighet i X-tillämpningar har blivit ett hinder för, att de skall anammas i bredare kretsar. Som svar på detta har två projekt, som arbetar med öppen källkod, startats: *K Desktop Environment* (KDE) och *GNU Network Object Model Environment* (GNOME). Vardera har ett brett urval av tillämpningar ifrån programstartarpaneler och filhanterare till spel och kontorsprogramsviter, vilka är skrivna i samma programmeringsverktygssatser för grafiska gränssnitt (*GUI toolkit*) och snävt integrerade för att erbjuda en enhetlig, sammanhängande skrivbordsmiljö.

Skillnaderna mellan KDE och GNOME är på det hela taget tämligen små. De ser olika ut, därför att de använder olika *GUI toolkit* (verktygslådor för grafiska gränssnitt). KDE bygger på biblioteket Qt ifrån Troll Tech AS, medan GNOME använder GTK — en verktygslåda, som ursprungligen utvecklats för *GNU Image Manipulation Program* (eller GIMP kort och gott). I egenskap av ifrån varandra skilda projekt har KDE och GNOME var för sig sina egna konstruktörer och programmerare med olika utvecklingsstilar och tänkesätt. Resultatet är i vardera fallet emellertid i grunden detsamma: en konsekvent, snävt integrerad skrivbordsmiljö och uppsättning tillämpningsprogram. Funktionaliteten, användbarheten och det rena skönhetsvärdet i KDE såväl som GNOME kan mäta sig med vad som helst, som är tillgängligt i andra operativsystem.

Den bästa biten är dock, att dessa avancerade skrivbordsmiljöer är gratis.[7] Detta innebär, att man kan ha endera eller bägge (ja, på samma gång). Valet är ens eget.

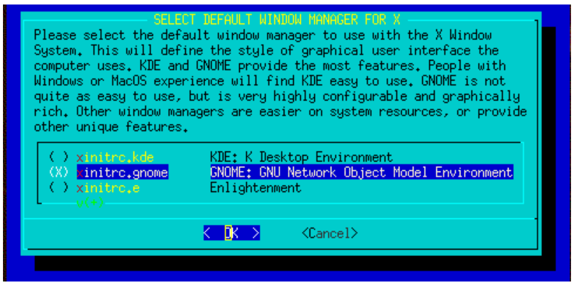
Förutom skrivbordsmiljöerna GNOME och KDE ingår i Slackware en stor samling fönsterhanterare. Somliga är formgivna så, att de efterliknar andra operativsystems grafisk miljöer, andra är konstruerade för att skräddarsys efter personlig smak och personliga behov, andra åter för att vara så snabba

[7] GNOME är dessutom fri enligt FSF-definitionen, i det GTK tillhandahålles på licensvillkor enligt GPL. Qt, som KDE bygger på, lär även komma att tillhandahållas enligt villkoren i GPL. Ö.a.

som möjligt. Urvalet är stort. Självfallet kan man installera så många man önskar, leka med allihop och själv bestämma, vilken man tycker bäst om.

För att underlätta val av skrivbordsmiljö ingår i Slackware även ett program, som heter `xwmconfig`, vilket kan användas till att välja skrivbordsmiljö eller fönsterhanterare. Det köres på följande vis:

```
$ xwmconfig
```



Figur 6.14: Välja skrivbordsmiljö eller fönsterhanterare med `xwmconfig`.

En lista över alla installerade skrivbordsmiljöer och fönsterhanterare visas. Man väljer en på listan. Var användare i systemet får själv köra detta program, eftersom olika användare får använda olika skrivbordsmiljöer, och alla kommer inte att vilja ha den, som valts vid systemets installation.

Därefter startar man X:

```
$ startx
```

6.6 Exportera skärmvisning.

Såsom tidigare omnämnts, så är det möjligt att köra X-program på en dator och visa dem på en annan. Detta är otroligt bandbreddsintensivt, så att man förmodligen inte vill göra detta genom modemförbindelse eller över mycket långa sträckor. Därtill måste man överväga säkerheten; att exportera en skärmbild strider emot god säkerhet, eftersom man kan låta hela nätverket titta på, vad man gör. Likväl kan det komma till nytta i ett lokalt nät.

En viktig sak att lägga märke till i detta sammanhang är bruket av orden »*klient*« och »*server*«. När man exporterar skärmbilden, så kan man bli förvillad ifråga om vad, som är klient, och vad, som är server. Vi omnämner den maskin, som faktiskt kör X-programmen och sänder ut information om skärmbilden, såsom »server«. Den maskin, man använder till att visa det program, som köres på fjärrdatorn, kallas för »klient«. När vi diskuterar, hur

X är konstruerat, förhåller det sig omvänt. Det program, som visar någonting, kallas för »server«, medan det körande programmet kallas för »klient«. Detta är inte särskilt svårt att hålla reda på, men det förtjänar att påpekas.

I detta exempel skall vi använda två datorer: **golf** är en tämligen kraftfull server och anbragt under ett skrivbord i ena änden av ett tätt befolkat rum. Den är bestyckad med en nätt liten processor, som förfogar över rikligt med RAM. Därtill har den mängder av X-program installerade men ingen bildskärm. I rummets andra ände står en soffa och datorn **soffa**, en gammal maskin med ont om RAM och rätt litet hårddiskutrymme. Den är alldeles för klen för att kunna köra sådana resursintensiva program som Netscape. **soffa** har dock två stora fördelar: den är försedd med bildskärm och den står intill soffan, så att man kan använda den liggande. Det bästa vore, om man kunde köra Netscape utan att resa sig ifrån soffan. Lösningen kallas för *export*.

Först loggar man in på **soffa** och sätter igång X. Sedan startar man ett terminalprogram (xterm, rxvt, eterm, aterm eller något av alla de andra). Som första steg för att visa X-program på en annan dator ställer man in klientmaskinen så, att andra datorer får visa sina program på dess skärm. Detta inbegriper bruk av programmet `xhost` för åtkomststyrning. Om man befinner sig i ett säkrat internt nät, bryr man sig förmodligen inte om vilka, som får fjärrvisa programfönster. I så fall kan man helt enkelt släppa in vem som helst i nätverket på sin skärm:

```
soffa$ xhost +
access control disabled, clients can connect from any host
```

Å andra sidan kanske man vill göra detta med maskiner i ett ickesäkrat nät (Internet, ett skolnät eller vad som helst, man inte kan styra). Då vill man säkerligen inte låta vem som helst ansluta sig:

```
soffa$ xhost + golf.foc
golf.foc being added to access control list
```

Nu kan **golf.foc** (tidigare nämnd server) visa program på **soffa**. Man kan se vilka, som har åtkomst till skärmen, genom att köra `xhost` utan argument:

```
soffa$ xhost
access control enabled, only authorized clients can connect
INET:golf.foc
INET:localhost
INET:soffa.foc
LOCAL:
```

Detta behöver man göra på klientsidan. Såsom nästa steg riggar man upp servern så, att den vet om, att den kan visa program någon annanstans än på sin egen bildskärm. Eftersom servern saknar bildskärm (och följaktligen inte har X igång), så måste den få reda på, var den kan visa programmen.

Att rigga upp servern är heller inte särskilt besvärligt. När man anslutit till den, sätter man miljövariabeln `$DISPLAY` till fjärrdatorns namn tillsammans med en siffra, som betecknar vilken X-skärm, bilderna skall visas på. Man har så gott som säkerligen endast en X-körning igång, så att den variabeln vanligen inte föranleder tveksamhet.

På följande vis sätter man variabeln `$DISPLAY` i vårt exempel under `bash`. Andra kommandoskal fordrar annorlunda syntax, men värdet skall vara detsamma.

```
golf$ export DISPLAY=soffa.foc:0.0
```

Svårare är det inte att rigga upp servern. Nu gäller det bara att fortsätta vara inloggad på servern igenom detta terminalfönster och köra programmen därifrån. All skärmvisning ifrån programmet sändes över nätverket till klientmaskinen, även om det köres på en dator i lokalens andra ände.

```
golf$ netscape &
```

Detta kommando startar Netscape på servern, men eftersom variabeln `DISPLAY` pekar på **soffa**, så visas allting där. Man behöver inte köra några stora X-program på sin terminal. En viktig anmärkning i detta sammanhang är dock, att servern måste ha alla X-bibliotek och andra stödfiler, som fordras för att köra programmet. Emellertid behöver den varken ha X-server eller någon `XF86Config`-fil, eftersom ingenting skall visas på servern.

Efteråt kanske man vill stänga av skärmexporten genom att ta bort servern från klientens åtkomstlista:

```
soffa$ xhost - golf.foc
golf.foc being removed from access control list
```

Man inser lätt, att detta är ett strålande sätt att dela på datorresurser. Men man skall vara försiktig, eftersom man kan få vara värd för många X-program på fjärrdatorer utan att ens veta om det.

6.7 Sammanfattning.

I detta kapitel har vi lärt oss att ställa in X-fönstersystemet genom användning av `xf86config` och `XF86Setup`. Man bör nu ha klart för sig, vad en skrivbordsmiljö (*desktop environment*) och en fönsterhanterare (*window manager*) är för någonting, samt hur man växlar emellan de olika valmöjligheterna. Man vet nu också, hur man kan exportera X-körningen till en annan dator. Vid detta laget skall man ha sin grafiska miljö igång.

7

Starta systemet.

Det kan ibland vara lätt men ibland åter svårt att starta linuxsystemet. Många användare installerar Slackware på sin dator, och så är den saken klar. De bara sätter igång maskinen, och så är den färdig för användning. Somliga måste emellertid använda något annat operativsystem för vissa uppgifter, så att de behöver ha bägge operativsystemen tillgängliga i maskinen.

7.1 LILO.

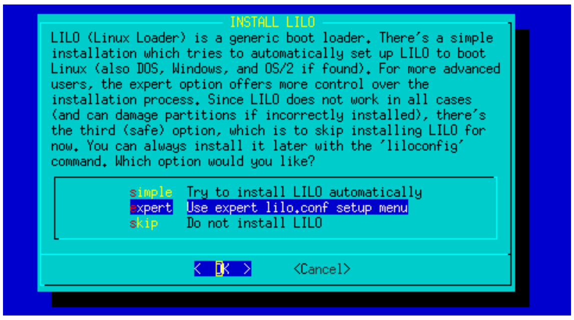
Linux Loader — eller LILO — är den omtycktaste systemladdaren (*bootladdaren*)[1] på linuxsystem. Den är i hög grad konfigurerbar och kan lätt användas till att starta även andra operativsystem.

Med Slackware Linux följer `liloconfig`, som är ett menystyrt inställningstillbehör för LILO. Detta program köres först under `setup` i samband med att man installerar Slackware, men man kan även starta det senare genom att skriva `liloconfig` på kommandoraden.

LILO läser sina inställningar ifrån filen `/etc/lilo.conf` (5). Denna fil läses inte var gång, man startar systemet, utan endast var gång, man installerar LILO. LILO måste återinstalleras till hårddiskens *bootsektor* var gång, man har ändrat någon av dess inställningar, t.ex. när man har kompilerat en ny linuxkärna och vill kunna starta på den nya kärnan. Programmet `liloconfig` hjälper en att skapa inställningsfilen, så att man kan installera LILO i systemet. Den som föredrar att redigera `/etc/lilo.conf` för hand får även själv ominstallera LILO genom att kommendera `/sbin/lilo`, sedan inställningsfilen redigerats.

När man först startar `liloconfig`, ser det ut så här:

[1]En bootladdare är ett program, som köres när man startar datorn, och som gör det möjligt att starta ett eller flera operativsystem.



Figur 7.1: Välja sätt att konfigurera LILO.

Är detta första gången, man skall rigga upp LILO, så bör man välja »*simple*«. Annars kan man välja »*expert*«, vilket går fortare, om man känner väl till LILO och Linux. När man väljer »simple«, börjar LILO-konfigureringen.

Är linuxkärnan kompilerad med *framebuffer*-stöd, så frågar `liloconfig` vilken skärmupplösning, man vill ha. Detta är samma upplösning, som skall användas av *framebuffer*-servern för XFree86. Vill man inte ha något särskilt grafikläge i textkonsollen (utanför X), så väljer man *normal* och får behålla det vanliga textläget 80x25.

Figur 7.2: Välja skärmupplösning för textläge.

I LILO-konfigureringens följande avsnitt gäller det var, man vill installera LILO. Detta är kanske det allra viktigaste steget. Nedanstående lista förklarar, var LILO installeras:

Root Detta val installerar LILO i början av Linux' rotpartition. Detta är det säkraste valet, om man har ytterligare ett operativsystem i datorn. Det säkerställer, att andra bootladdare inte blir överskrivna. Nackdelen är, att LILO endast kan laddas på detta vis, om den skivenhet, Linux är installerat på, är systemets *första* skivenhet.

Floppy Detta tillvägagångssätt är t.o.m. säkrare än föregående. Det skapar en startdiskett, varmed man kan starta linuxsystemet. Detta håller bootladdaren helt och hållet borta ifrån hårddisken, så att man endast bootar på denna diskett, när man vill starta Slackware.

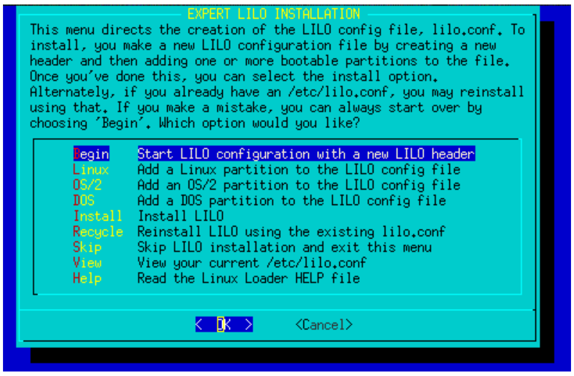
MBR Detta tillvägagångssätt använder man helst, om Slackware är ensamt operativsystem i datorn, eller om man skall använda LILO till att välja mellan flera olika operativsystem.

VARNING!

Väljer man att lägga LILO i MBR, kommer vilken som helst annan bootladdare, som redan ligger i MBR, att skrivas över!

När man har valt, var LILO skall installeras, kommer `liloconfig` att skriva inställningsfilen `lilo.conf` i katalogen `/etc` och installera LILO.

Om man väljer »*expert*«, får man i stället en särskild meny. Denna meny medger, att man mer i detalj kan påverka innehållet i `/etc/lilo.conf`, lägga till ytterligare operativsystem till startmenyn och ge direktiv till LILO om att ladda linuxkärnan med särskilda parametrar vid start. Expertmenyn ser ut på följande vis:



Figur 7.3: Expertens sätt att konfigurera LILO.

Vilken systemkonfiguration man än har, så är det lätt att rigga upp en fungerande bootladdare. Med `liloconfig` är det en bagatell att rigga upp den. Emellertid finns särskilda fall, när LILO inte är ett gott val för ett visst system. Lyckligtvis finns då andra möjligheter.

7.2 LOADLIN.

Med Slackware Linux följer även ett annat sätt att starta systemet, nämligen LOADLIN. LOADLIN är ett DOS-program, som gör det möjligt att starta Linux ifrån ett DOS-system. Det fordrar, att linuxkärnan är kopierad till DOS-partitionen, så att LOADLIN kan ladda den och starta systemet i vederbörlig ordning.

Under slackwareinstallationen kopieras LOADLIN till **root**s hemkatalog såsom `.ZIP`-fil. Det finns ingen självverkande uppriggningsprocedur för LOADLIN. Man måste kopiera linuxkärnan (`/vmlinuz`) och LOADLIN-filen från **root**s hemkatalog `/root` till DOS-partitionen.

LOADLIN är bra att ha, om man skulle vilja göra en startmeny på DOS-partitionen. En sådan meny kan man lägga till i sin `AUTOEXEC.BAT`, så att man får välja mellan Linux och DOS, när man startar DOS. Vid val av Linux skulle då LOADLIN köras och starta slackwaresystemet. En sådan `AUTOEXEC.BAT` under Windows 95 skulle kunna se ut på följande vis:

```
@ECHO OFF
SET PROMPT=$P$G
SET PATH=C:\WINDOWS;C:\WINDOWS\COMMAND;C:\
CLS
ECHO Please Select Your Operating System:
ECHO.
ECHO [1] Slackware Linux
ECHO [2] Windows 95
ECHO.
CHOICE /C:12 "Selection? -> "
IF ERRORLEVEL 2 GOTO WIN
IF ERRORLEVEL 1 GOTO LINUX
:WIN
CLS
ECHO Starting Windows 95...
WIN
GOTO END
:LINUX
ECHO Starting Slackware Linux...
CD \LINUX
LOADLIN C:\LINUX\VMLINUZ ROOT=<root partition device> RO
GOTO END
:END
```

I kommandot för att köra LOADLIN anger man rotpartition för Linux på samma sätt, som man anger partitioner i Linux, t.ex. `/dev/hda2`. LOADLIN kan även användas på kommandoraden, när DOS redan är igång. Kommandot köres på samma sätt, som det visas i exemplet ovan. Dokumentationen till LOADLIN visar flera exempel på, hur detta kan utföras.

7.3 Använda mer än ett operativsystem.

Många användare riggar upp sina datorer för att kunna starta endera Slackware Linux eller något annat operativsystem. Vi skildrar här nedan några typiska scenarier för sådan »tvestart« (*dual boot*) av två olika operativsystem, i fall någon skulle ha svårigheter med att rigga upp systemet för detta.

7.3.1 Windows 9x/DOS.

Att rigga datorn för att starta endera Windows 9x eller Linux är förmodligen det vanligaste scenariet för tvestart (*dual boot*). Det finns ett antal olika sätt att rigga sådan systemstart; detta avsnitt skildrar två.

Ofta lyckas den, som skall rigga upp start av två olika system, snida till en fulländad plan för slutresultatet, men trasslar sedan likväl till det ifråga om åtgärdernas ordningsföljd. Det är mycket betydelsefullt att fatta vikten av, att operativsystem måste installeras i en viss ordning, för att en sådan tvestart skall fungera. Linux erbjuder alltid användaren fulla styrmöjligheter beträffande vad, som skall skrivas till MBR (*Master Boot Record*). Därför är det tillrådligt att installera Linux till sist. Windows bör i förekommande fall installeras först, eftersom det vid installation alltid skriver över MBR med sin egen bootladdare.

7.3.1.1 Använda LILO.

De flesta vill förmodligen använda LILO vid val emellan Linux och Windows. Såsom ovan sagts, så bör man först installera Windows och därefter Linux.

Låt oss anta, att vi har en IDE-disk på 47 GB såsom enda hårddisk i systemet. Låt oss vidare anta, att vi vill använda hälften av utrymmet till Windows och hälften till Linux. Detta ställer till med ett bekymmer, när man skall starta Linux. Vi känner inte så nog till hårddiskens särskilda geometri, men det kan mycket väl vara så, att vi någonstans förbi 23,5 GB på disken passerar den 1024:e cylindern. En bättre uppdelning av denna hårddisk vore då:

1. 1 GB Windows boot (startpartition, enhet `C:`)
2. 1 GB Linux rotpartition (`/`)
3. 22.5 GB Windows övrigt (enhet `D:`)
4. 22.5 GB Linux /usr (`/usr`)

Man vill nog också lägga undan lämpligt utrymme till en swappartition för Linux. En oskriven lag är, att man skall ha dubbelt så mycket swap (för virtuellt minne) som man har RAM. Ett system med 64 MB RAM ger man i så fall 128 MB swap o.s.v.

När partitionerna skapats,[2] så kan man fortsätta med att installera Windows. När detta är installerat och fungerar, så kan man installera Linux. Installation av LILO fordrar särskild uppmärksamhet. Man skall i ett fall som detta välja »expert« i `liloconfig`, när man riggar upp LILO.

Så vidtar skapandet av en ny LILO-konfiguration. Vi vill installera systemladdaren LILO till MBR (*Master Boot Record*), så att den kan användas till att välja mellan två operativsystem. Från menyn väljer man först att lägga till en linuxpartition och sedan att lägga till en windowspartition (eller DOS-partition). När det är färdigt, väljer man att installera LILO.

Därefter startar man om datorn. LILO bör nu laddas och väntar på användarens inmatning. Man kan trycka på `Alt` för att få fram prompten `boot:`. Där skriver man in namnet på det system, man vill ladda (namnen kan vara olika beroende på vad, man valt, då man riggade upp LILO). Om man har glömt namnen, så trycker man på `Tab` och får därmed en lista över operativsystem, som kan startas.

Man kan ställa in LILO genom att redigera filen `/etc/lilo.conf` på linuxpartitionen. Vi vill t.ex. att LILO skall visa följande:

```
System Boot Menu
================

1 - Linux
2 - Windows

LILO boot:
```

Då skulle vår `/etc/lilo.conf` se ut så här:

```
# LILO configuration file

boot = /dev/hda
vga = normal
```

[2] För att få rätt ordning på partitionerna, så kan det vara lämpligt att först skapa allihop med Linux' `fdisk`, sedan ta bort windowspartitionerna även med Linux' `fdisk` för att slutligen skapa windowspartitionerna i det sålunda skapade lediga utrymmet. I allmänhet skapas partitioner med `fdisk` i det system, som skall använda partitionerna, således DOS-partitioner i DOS' `fdisk` och linuxpartitioner i Linux' `fdisk`. Ö.a.

```
message = /boot/message

image = /vmlinuz
   root = /dev/hda2
   label = 1
   read-only

other = /dev/hda1
   label = 2
   table = /dev/hda
```

Och vår `/boot/message` skulle se ut så här:

```
System Boot Menu
================
1 - Linux
2 - Windows
```

LILO är en högst konfigurerbar systemladdare. Den är inte enbart begränsad till att kunna ladda Linux eller DOS. Den kan ladda nästan vad som helst. Manualsidorna för `lilo` (8) och `lilo.conf` (5) innehåller närmare upplysningar om detta.

Om nu LILO inte skulle fungera? Det finns fall, när LILO inte fungerar på en viss maskin. Lyckligtvis finns ett annat sätt att starta datorn med växelvis Linux och Windows.

7.3.1.2 Använda LOADLIN.

Detta tillvägagångssätt kan tillgripas i fall LILO inte fungerar i systemet, eller om man inte vill rigga upp LILO. Denna metod är således lämplig för användare, som ofta installerar om Windows. Var gång, man ominstallerar Windows, skriver det över MBR (*Master Boot Record*) och ödelägger därigenom LILO-installationen. Med LOADLIN är man inte utsatt för detta bekymmer. Den främsta nackdelen är den, att LOADLIN endast kan användas till att starta Linux.

När man använder LOADLIN, kan operativsystemen installeras i vilken ordning som helst. Var dock försiktig, så att ingenting installeras till MBR (*Master Boot Record*). LOADLIN är beroende av, att windowspartitionen är startbar. Därför måste man hoppa över LILO-konfigureringen under installation av Slackware.

Sedan operativsystemen installerats, kopierar man filen `lodlinX.zip` (där »`X`« är ett versionsnummer, såsom »`16a`«) från **root**s hemkatalog till windowspartitionen. Man skall även kopiera linuxkärnan till windowspartitionen. Man måste befinna sig i Linux för att detta skall gå att utförbara. Följande exempel visar, hur detta kan gå till:

```
# mkdir /win
# mount -t vfat /dev/hda1 /win
# mkdir /win/linux
# cd /root
# cp loadlin* /win/linux
# cp /vmlinuz /win/linux
# cd /win/linuz
# unzip loadlin16a.zip
```

Detta skapar en katalog vid namn `C:\LINUX` på windowspartitionen (vi förmodar, att den är `/dev/hda`) och kopierar det, som LOADLIN behöver. När detta är gjort, måste man starta om datorn till Windows och rigga upp en startmeny.

När man väl har Windows igång, så skall man se till att få en DOS-prompt. Först måste man se till, så att systemet inte startar med det grafiska gränssnittet.

```
C:\>cd \
C:\>attrib -r -a -s -h MSDOS.SYS
C:\>edit MSDOS.SYS
```

I filen `MSDOS.SYS` lägger man till följande rad:[3]

```
BootGUI=0
```

Filen sparas och vi stänger editorn. Sedan redigeras `C:\AUTOEXEC.BAT` så att den förses med en startmeny. Följande är ett exempel på, hur ett startmenyblock i `AUTOEXEC.BAT` skulle kunna ta sig ut:

```
cls
echo System Boot Menu
echo.
echo 1 - Linux
echo 2 - Windows
echo.
```

[3]OBS! Detta gäller om man kör Windows 95 eller Windows98! Om man kör MS-DOS före version 7, så är `MSDOS.SYS` ingen textfil, så att den inte kan (och inte behöver) redigeras på detta vis! Ö.a.

```
choice /c:12 "Val? -> "
if errorlevel 2 goto WIN
if errorlevel 1 goto LINUX
:LINUX
cls
echo "Startar Linux..."
cd \linux
loadlin c:\linux\vmlinuz root=/dev/hda2 ro
goto END
:WIN
cls
echo "Startar Windows..."
win
goto END
:END
```

Den viktigaste raden är den, som kör LOADLIN. Vi talar om för LOADLIN vilken kärna, det skall ladda, vilken partition som är rotpartition för Linux, samt att vi till en början vill montera denna partition såsom endast läsbar (*read-only*).

Verktygen för dessa två tillvägagångssätt ingår i Slackware Linux. Det finns åtskilliga andra bootladdare på marknaden, men dessa två torde duga i flertalet situationer, när man vill kunna välja mellan två operativsystem.

7.3.2 Windows NT.

Detta är det näst vanligaste fallet med val av två operativsystem vid start. Windows NT erbjuder många fler svårigheter än fallet med Windows 9x och Linux. Det största problemet är, att NT inte lyckas starta, om MBR (*Master Boot Record*) blir överskriven av LILO. Därför måste vi använda den *OS Loader*, som medföljer Windows NT. Följande steg visar, hur man kan rigga upp ett system med val emellan Windows NT och Linux.

1. Installera Windows NT.
2. Installera Linux med LILO på linuxpartitionens *superblock* (och *icke* i MBR).
3. Ta de första 512 byten på Linux' rotpartition och lagra dem på windowspartitionen
4. Redigera `C:\BOOT.INI` i Windows NT, så att val av Linux lägges till där.

Att installera Windows NT bör vara tämligen enkelt, liksom att installera Linux. Därefter blir det en smula knepigare. Att ta ut de första 512 byten från linuxpartitionen är enklare än det låter. Man måste vara i Linux för att lyckas med detta. Om vi antar, att det är `/dev/hda2`, som är Linux' rotpartition, så kommenderar man följande:

```
# dd if=/dev/hda2 of=/tmp/bootsect.lnx bs=1 count=512
```

Så är det gjort. Nu skall man kopiera `bootsect.lnx` till windowspartitionen. Då står vi inför nästa svårighet. Linux har inte stabilt understöd för skrivning till filsystemet NTFS. Om man har installerat Windows NT och formaterat partitionen som NTFS, så måste man kopiera filen till en FAT-diskett[4] och sedan läsa den därifrån i Windows NT. Om man har formaterat windowsenheten såsom FAT, så kan man helt enkelt montera den i Linux och kopiera filen dit. Vilketdera man än gör, så vill man få över filen `/tmp/bootsect.lnx` från linuxpartitionen till `C:\BOOTSECT.LNX` på partitionen för Windows NT.

Sista steget är att lägga till ett menyval i startmenyn för Windows NT. I Windows NT öppnar man en kommandoprompt.

```
C:\WINNT>cd \
C:\>attrib -r -a -s -h boot.ini
C:\>edit boot.ini
```

Denna rad lägges till i filens slut:

```
C:\bootsect.lnx="Slackware Linux"
```

7.3.3 Linux.

Ja, folk gör faktiskt det.[5] Detta är det allra enklaste scenariet för att starta flera operativsystem. Man lägger helt enkelt till fler val i filen `/etc/lilo.conf`. Svårare än så är det inte.[6]

[4] DOS-formaterad diskett, Ö.a.

[5] T.ex. vill man kanske ha flera olika linuxdistributioner i datorn. Då kan man använda LILO till att välja endera vid start. Ö.a.

[6] Efter ändringar i filen `/etc/lilo.conf` måste man förstås köra `lilo` igen. Ö.a.

7.4 Sammanfattning.

Detta kapitel har behandlat start av systemet genom endera LILO eller Loadlin. Det har även behandlat möjligheterna att välja mellan Linux och andra operativsystem vid start. Man bör nu vara i stånd att konfigurera startmetoden på riktigt sätt och starta valfritt operativsystem, när så önskas.

IV

Att använda Slackware Linux.

8

Kommandoskalet.

I en grafisk miljö tillhandahålles gränssnittet av ett program, som skapar fönster, blädderlister, menyer o.s.v. I en kommandoradsmiljö tillhandahålles användargränssnittet såsom ett »*skal*« (kallas även *kommandotolk*), vilket tolkar kommandon och i största allmänhet gör saker och ting användbara. Omedelbart efter inloggning (som även skildras i detta kapitel), kommer användaren in i ett skal och tillåtes sköta sina angelägenheter där. Detta kapitel tjänar som introduktion till kommandoskalet och till det allra vanligaste skalet bland linuxanvändare — *Bourne Again Shell* (`bash`). Närmare upplysningar om det, som behandlas i detta kapitel, finns på manualsidan `bash`(1).

8.1 Användare.

8.1.1 Inloggning.

Jaha, då har vi fått igång datorn, och nu stirrar vi på något i stil med detta:

```
Welcome to Linux 2.2.16
darkstar login:
```

Hmm… ingen har sagt något om att logga in. Vad är förresten en *darkstar*? Ingen fara; det är antagligen inte så, att vi har råkat upprätta en radiolänk igenom hyperrymden till Imperiets konstgjorda måne. (Tyvärr stödjer linuxkärnan f.n. inte något protokoll för radiolänkar igenom hyperrymden.) Nej, *darkstar* är bara namnet på en av våra datorer, och alla nyinstallationer får namnet *darkstar*, om inget annat valts. Har man angivit något namn på sin dator, när man körde `setup`, så får man se detta namn i stället för *darkstar*.

Vad inloggningen anbelangar... Om detta är första gången, så får man logga in såsom överanvändaren **root**. Har man valt ett lösenord för **root**, när man körde `setup`, så skriver man sedan in detta vid lösenordsprompten. Har man inte valt något sådant, kommer man in utan lösenord och behöver bara trycka på `Retur`. Saken är klar, nu är vi inne!

8.1.2 root: överanvändaren.

Jaha, vem eller *vad* är »**root**«? Och vad gör den/det med vårt system?

Nå, i Unix-världen och i unixliknande operativsystem (såsom Linux) är det skillnad på användare och användare. Vi skall gå närmare in på detta senare, men det, som nu är viktigt att känna till, är att **root** är en användare, som står över alla användare; **root** är allsmäktig och allvetande, och *ingen* trotsar **root**. Det är helt enkelt inte tillåtet. **root** är vad vi kallar en »överanvändare«, och det med all rätt. Och det bästa av allt är, att man själv får vara **root**.

Fint, vad?

Ifall det nu skulle råda något tvivel om den saken: ja, det är mycket bra. Haken är bara den, att **root** också kan knäcka vad som helst, om han så önskar. I detta sammanhang kan det nog vara nyttigt att hoppa fram till s. 129 och läsa om, hur man lägger till användare; sedan kan man skapa en användare, logga in såsom denne och arbeta som denne. Den nedärvda klokskapen i detta är, att man helst bara skall bli överanvändare, när det är fullkomligt nödvändigt, så att man gör risken för att oavsiktligen råka knäcka någonting i systemet så liten som möjligt.

Förresten, om man skulle komma på tanken att vilja bli **root**, när man redan har loggat in som någon annan, så ställer det inte till med några besvärligheter. Man använder då kommandot `su` (1). Man avkräves därvid ett lösenord och skriver då in lösenordet för **root**; därefter får man vara **root** tills man lämnar rootinloggningen med kommandot `exit` eller `logout`. Man kan också tillfälligt uppträda som en annan användare med hjälp av kommandot `su`, under förutsättning att man vet denne användares lösenord: `su logan` skulle t.ex. göra läsaren till mig.[1]

[1] D.v.s. till Logan Johnson, en av bokens författare. Ö.a.

8.2 Kommandoraden.

8.2.1 Köra program.

Det är svårt att åstadkomma särskilt mycket utan att köra något program; kanske skulle man kunna ha datorn till bokstöd eller dörrhållare, och en del datorer avger ett vackert, surrande läte, men mycket mer än så får man inte ut av den. Vi tror, att vi alla kan vara överens om, att datorns användning såsom surrande dörrhållare inte är precis det, som har givit persondatorn den popularitet, den numera åtnjuter.

Vi påminner oss, att allting i Linux är filer. Nå, detta gäller även program. Vartenda kommando, man kör (som inte är inbyggt i skalet), ligger i en fil någonstans. Man kör ett program genom att ange hela dess sökväg.

Ett exempel på detta är kommandot `su` från föregående avsnitt. Det finns faktiskt i katalogen `/bin`: `/bin/su` utför detta.

Vad är det då som gör, att det räcker med att knappa in `su`? Vi har ju inte talat om, att det ligger i `/bin`. Det skulle kunnat vara i `/usr/local/share`, eller hur? Hur kunde det veta, att vi kallade på det? Svaret ligger i miljövariabeln `PATH`; de flesta skal har endera `PATH` eller någonting, som är mycket likt `PATH`. Det innehåller en förteckning över kataloger, som skall genomsökas efter program, man försöker köra. Så när vi körde `su`, genomsökte skalet sin kataloglista och letade därvid efter en körbar fil vid namn `su`; den första med detta namn, det träffade på, kördes. Detta inträffar var gång, man kallar på ett program utan att ange dess fullständiga sökväg; om man får felmeddelandet `Command not found`, finns det sökta programmet inte i `PATH`. (Det gäller förstås även, om programmet inte alls finns till. . .) Vi skall gå djupare in på miljövariabler i avsnittet om *The Bourne Again Shell* (`bash`) på s. 101.

Man skall även komma ihåg, att »`.`« är en förkortning för »den katalog, jag befinner mig i«, så om man hade råkat befinna sig i `/bin`, så skulle `./su` ha fungerat som fullständig sökväg.

8.2.2 Jokertecken (wildcards).

Nästan alla skal erkänner vissa tecken såsom ersättningar eller förkortningar för »vad som helst kan stå här«. Sådana tecken kallas lämpligt nog för »jokertecken« (»*wildcards*«); de vanligaste är `*` och `?`. Frågetecknet `?` står vanligen för ett enstaka tecken. Vi kan t.ex. anta, att en katalog innehåller tre filer: `ex1.txt`, `ex2.txt` och `ex3.txt`. Alla dessa filer vill vi kopiera (med kommandot `cp`, som vi behandlar i avsnittet `cp` på s. 118) till en annan katalog, låt oss säga `/tmp`. Att slå `cp ex1.txt ex2.txt ex3.txt /tmp` är alldeles för arbetsamt. Det är mycket lättare att slå `cp ex?.txt /tmp`; frågetecknet `?` passar in på vart och ett av tecknen »`1`« »`2`« och »`3`«, så att de i tur och ordning ersätter frågetecknet.

Är detta också för arbetsamt? Riktigt. Det är upprörande; vi har väl arbetarskyddslagar till att bevara oss ifrån dylikt. Lyckligtvis har vi även stjärnan (asterisken) `*`. Såsom tidigare nämnts, träffar `*` »vilket antal tecken som helst« inberäknat noll. Vore de tre filerna ensamma i katalogen, så skulle vi ha kunnat kommendera `cp * /tmp` och hade då svept allesamman i ett drag. Antag emellertid, att katalogen även innehållit filen `exempel.txt` och filen `hejaz.txt`. Vi ville kopiera `exempel.txt` men inte `hejaz.txt`; `cp exempel* /tmp` skulle utföra detta.

8.2.3 Omdirigering av inmatning/utmatning; rörledning.

(Här kommer något finurligt.)

```
$ ps > blargh
```

Här kör vi `ps` för att se vilka processer, som är igång; `ps` behandlas på s. 123. Nu är det inte det, som är så listigt. Det finurliga är `> blargh`, vilket ungefär betyder »ta utmatningen ifrån `ps` och skriv den till en fil vid namn blargh«. Vänta bara, det blir ännu finurligare.

```
$ ps | less
```

Detta kommando tar utmatningen ifrån `ps` och »trattar« in det i en rörledning (pipe) till `less`, så att vi bekvämt kan bläddra framåt och bakåt i den.

```
$ ps >> blargh
```

Denna omdirigering kommer på tredje plats ibland de vanligaste; den gör samma sak som »`>`« förutom det, att »`> >`« lägger till utmatningen ifrån `ps` på slutet i filen `blargh`, om nämnda fil finns till. Om den inte redan finns, så skapas den, liksom i fallet med »`>`«. (»`>`« skulle skriva över innehållet i `blargh` med nytt innehåll.)

Det finns även ett »`<`«, som betyder »hämta inmatning ifrån det, som följer här efter«, men detta tecken användes inte så ofta.

```
$ fromdos < dosfile.txt > unixfile.txt
```

Omdirigeringen blir riktigt skojig, när man radar upp kommandon och argument:

```
$ ps | tac > > blargh
```

Detta kör `ps`, kastar om inmatningen och lägger den till filen `blargh`. Man kan stapla så många sådana led ovanpå varandra, som man önskar; man skall bara vara noga med att komma ihåg, att de tolkas ifrån vänster till höger.

Se manualsidan för `bash` (1) beträffande närmare upplysningar om omdirigering.

8.3 Bash (Bourne Again Shell).

8.3.1 Miljövariabler.

Ett linuxsystem är en invecklad skapelse; där är mycket att hålla reda på och en mängd smådetaljer, som sättes igång vid vår vanliga samverkan med olika program (varav man inte ens är medveten om en del). Ingen människa vill väl använda en rad växlar (flaggor) till vartenda program, som skall köras, eller behöva tala om för det vilken slags terminal, man använder, datorns värdnamn, hur prompten skall se ut…

För att hantera detta har användaren det, som kallas omgivning eller miljö (*environment*). Miljön avgränsar de villkor, varunder programmen skall köras, och en del av denna avgränsning är variabel; användaren kan ändra den och leka med den, som sig bör i ett linuxsystem. Nästan vartenda skal har miljövariabler (i annat fall är det förmodligen inget särskilt användbart skal). Här skall vi lägga fram en översikt över de kommandon, som `bash` tillhandahåller för manipulering av sina miljövariabler.

```
$ set
```

`set` utan flaggor (växlar) visar alla miljövariabler, som är satta, såväl som deras värden. Såsom det är vanligt med inbyggda kommandon i `bash` kan det även mycket annat (med parametrar); vi överlämnar dock åt manualsidan för `bash` (1) att täcka det. Ett utdrag ur utmatningen ifrån `set` på en av våra datorer ser ut så här:

```
PATH=/usr/local/lib/qt/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:
/usr/openwin/bin:/usr/games:.:/usr/local/ssh2/bin:/usr/local/ssh1/bin:
/usr/share/texmf/bin:/usr/local/sbin:/usr/sbin:/home/logan/bin
PIPESTATUS=([0]="0")
PPID=4978
PS1='\h:\w\$ '
PS2='> '
```

```
PS4='+ '
PWD=/home/logan
QTDIR=/usr/local/lib/qt
REMOTEHOST=ninja.tdn
SHELL=/bin/bash
```

Lägg märke till den tidigare omtalade variabeln `PATH`; Vi kan köra vad som helst i de angivna katalogerna genom att enbart knappa in filnamnet.

```
$ unset VARIABEL
```

`unset` avlägsnar variabler och tar därvid bort såväl variabel som dess värde; `bash` glömmer, att variabeln någonsin funnits till. (Ingen fara. Om det inte gäller någon variabel, man uttryckligen definierat i under körningen av skalet, så kommer den variabeln förmodligen att definieras på nytt i vilken annan körning som helst.)

```
$ export VARIABEL=något_värde
```

Det är behändigt att exportera. När man gör detta, så ger man variabeln `VARIABEL` värdet »`något_värde`«; om `VARIABEL` inte redan finns, så skapas den. Om `VARIABEL` redan hade haft ett värde, så är det gamla värdet nu borta. Det är inte så bra, om man försöker lägga till en katalog i variabeln `PATH`. I så fall vill man förmodligen hellre göra något i stil med detta:

```
$ echo $PATH
/usr/local/lib/qt/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:
/usr/openwin/bin:/usr/games:.:/usr/local/ssh2/bin:/usr/local/ssh1/bin:
/usr/share/texmf/bin:/usr/local/sbin:/usr/sbin:/home/logan/bin
```

8.3.2 Tabulatorkomplettering.

(Här kommer ytterligare något finurligt.)

1. Ett kommandoradsgränssnitt medför, att man måste skriva en hel del.

2. Det är arbetsamt att skriva.

3. Ingen tycker om att arbeta.

Av punkterna 3 och 2 drar vi slutsats 4, att ingen tycker om att skriva. Lyckligtvis räddar oss `bash` ifrån punkt 5 (att ingen tycker om kommandoradsgränssnitt).

Hur fullgör `bash` denna underbara bedrift, frågar man sig? Svaret är, att `bash` förutom det ovan skildrade utnyttjandet av jokertecken dessutom innehåller »tabulatorkomplettering«.

Tabulatorkomplettering fungerar ungefär på följande vis: Man skriver ett filnamn. Kanske finns det i `PATH`, kanske skriver man ut hela namnet med fullständig sökväg. Man behöver då endast skriva tillräckligt mycket av filnamnet, för att det skall vara unikt identifierbart. Därefter slår man an tabulatortangenten. Då funderar `bash` ut, vad man vill, och skriver ut resten av filnamnet!

Exempel: `/usr/src` innehåller två underkataloger: `/usr/src/linux` och `/usr/src/sendmail`. Vad finns i då i `/usr/src/linux`?

Vi skriver `ls /usr/src/l` och slår an tabulatortangenten, så ger `bash` oss `ls /usr/src/linux`.

Antag nu, att det finns två underkataloger, som heter `/usr/src/linux` och `/usr/src/linux-old`; Om vi skriver `/usr/src/l` och slår an tabulatortangenten, så fyller `bash` i återstoden, så att vi får `/usr/src/linux`. Vi kan stanna där, eller också kan vi slå an tabulatortangenten en gång till, så att `bash` visar en förteckning över de kataloger, vars namn passar in på det, som hittills matats in.

Mindre knappande, således (och följaktligen kan folk tycka om kommandoradsgränssnitt).

8.4 Virtuella terminaler.

Man är mitt uppe i arbetet och inser, att man behöver göra något annat. Man skulle kunna släppa, vad man har för händer, och börja på något annat, men vi arbetar ju i ett fleranvändarsystem, eller hur? Då kan väl en viss användare också logga in flera gånger, om han så vill? Varför skulle man då behöva göra endast en sak åt gången?

Det behöver man inte. Vi kan inte alla ha flera tangentbord, möss och bildskärmar till varenda maskin; antagligen vill de flesta av oss heller inte ha det så. Lösningen ligger således inte i maskinvara. Återstår programvaran, och Linux visar en av sina starka sidor här genom att tillhandahålla »virtuella terminaler« (VT).

Genom att trycka på tangenten `Alt` och någon av funktionstangenterna, kan man växla mellan virtuella terminaler; var funktionstangent motsvarar en virtuell terminal. Slackware är förinställt för inloggningar på 6 VT. Med `Alt` + `F2` kommer man till den andra, `Alt` + `F3` till den tredje o.s.v. upp till `F6`.

Återstående funktionstangenter är förbehållna X-sessioner. Var X-körning använder en VT med början på den sjunde (`Alt` + `F7`) och uppåt. När man är inuti X, så ersättes därvid kombinationen `Alt` + `F`unktionstangent med `Ctrl` + `Alt` + `F`unktionstangent; om man är i X och vill komma tillbaka till textinloggningen (utan att stänga X-körningen), så kommer man till tredje textinloggningen med `Ctrl` + `Alt` + `F3`. Med `Alt` + `F7` kommer man in i X igen, förutsatt att det är första X-sessionen, som användes.

8.5 Sammanfattning.

Detta kapitel har behandlat användare, skalet, kommandoraden och virtuella terminaler. Man bör nu finna sig väl till rätta med arbete på kommandoraden, med att köra program samt använda rörledningar (*piping*) och omdirigeringar till att kombinera kommandon. Slutligen bör man ha en uppfattning om **root**användarens makt, och varför det är dåligt att alltid köra som **root**.

9

Filsystemets uppbyggnad.

Vi har redan diskuterat katalogstrukturen i Slackware Linux. Vi kan nu hitta de filer och kataloger, vi behöver ha reda på. Men filsystemet är mer än en katalogstruktur blott och bart.

Linux är ett fleranvändarsystem. Systemet är i alla sina vinklar och vrår präglat av det, att det är avsett för flera användare. Systemet lagrar upplysningar om vem, som äger en viss fil, och om vem, som får läsa den. Det finns även andra unika beståndsdelar i filsystemet, såsom länkar och NFS-montering. Detta avsnitt skall förklara dessa, liksom även vad det innebär, att filsystemet är gjort för flera användare.

9.1 Ägarskap.

Filsystemet lagrar uppgift om tillhörighet för var fil och katalog i systemet. Detta omfattar vilken ägare och vilken grupp, en viss fil tillhör. Det lättaste sättet att ta reda på detta är att använda kommandot `ls`:

```
$ ls -l /usr/bin/wc
-rwxr-xr-x   1 root     bin    7368 Jul 30   1999  /usr/bin/wc
```

Det är tredje och fjärde kolumnerna, som intresserar oss. De innehåller namn på den användare och den grupp, filen tillhör. Vi ser, att det är användaren »**root**« och gruppen »**bin**«, som äger denna fil.

Vi kan lätt ändra ägare på filen med kommandot `chown`(1) (vilket står för »*change owner*«, ändra användare) och kommandot `chgrp`(1) (vilket står för »*change group*«, ändra grupp). Vi kan ändra ägaren till »**daemon**« med kommandot `chown`:

```
# chown daemon /usr/bin/wc
```

Vi kan ändra gruppen till »**root**« med `chgrp`:

```
# chgrp root /usr/bin/wc
```

Vi kan även använda `chown` till att på en gång ange användar- och grupptillhörighet för en fil:

```
# chown daemon.root /usr/bin/wc
```

9.2 Befogenheter.

Befogenheter/rättigheter (*permissions*) är en annan viktig del av filsystemet sett ur ett fleranvändarperspektiv. Med användning av sådana kan man bestämma, vem som skall få läsa, skriva och köra filer.

Informationen om rättigheterna lagras i ett fyrställigt oktaltal, där var siffra anger en särskild uppsättning rättigheter. Dessa är ägarens (*owner*), gruppens (*group*) och alla övrigas (*world*). Fjärde oktala siffran brukas till att lagra information såsom sättande av användar-ID (UID), sättande av grupp-ID (GID) och den »klibbiga« biten (*"sticky" bit*). De oktala värden, som tillordnas befogenheterna, är (de har även tillordnade bokstäver, som kan visas av program såsom `ls` och kan användas av `chmod`):

Rättighetstyp	**Oktalt värde**	**Bokstav**	**Rättighetstyp**
"sticky" bit	1	t	»klibbig« bit
set user ID	4	s	sätta användar-ID
set group ID	2	s	sätta grupp-ID
read	4	r	läsa
write	2	w	skriva
execute	1	x	köra

Man lägger ihop de oktala värdena för var rättighetsgrupp. Om man till exempel vill, att gruppen skall få läsa och skriva, så sätter man »6« i rättighetsinformationens gruppdel.

De förinställda rättigheterna för `bash` är följande:

```
$ ls -l /bin/bash
-rwxr-xr-x   1 root     bin  477692 Mar 21 19:57 /bin/bash
```

Första bindestrecket hade ersatts med ett »`d`«, om detta hade varit en katalog (*directory*). De tre rättighetsgrupperna (ägare, grupp och hela världen) visas därefter. Vi ser, att ägaren har rätt att läsa, skriva och exekvera (`rxw`). Gruppen har endast läs- och exekveringsrätt (`r-x`). Alla andra har endast läs- och exekveringsrätt (`r-x`).

Hur skulle vi kunna sätta rättigheterna på en annan fil på samma sätt som för `bash`? Först skall vi skapa en exempelfil:

```
$ touch /tmp/example
$ ls -l /tmp/example
-rw-rw-r---  1 david    users    0 Apr 19 11:21 /tmp/example
```

Sedan använder vi `chmod`(1) (vilket betyder *change mode*) till att sätta rättigheterna på exempelfilen. Vi lägger ihop de oktala siffrorna, så att vi får önskade rättigheter. För att ägaren skall få läsa, skriva och exekvera, så får vi värdet 7. Läsa och exekvera ger värdet 5. Sätter man ihop dessa med `chmod`, så får vi

```
$ touch /tmp/example
$ ls -l /tmp/example
-rw-rw-r---  1 david    users    0 Apr 19 11:21 /tmp/example
```

Speciella rättigheter sätter man genom att lägga ihop dem och placera dem i första kolumnen. Till exempel sätter vi UID och GID med talet 6 i första kolumnen:

```
$ chmod 6755 /tmp/example
$ ls -l /tmp/example
-rwsr-sr-x   1 david    users    0 Apr 19 11:21 /tmp/example
```

Om de oktala värdena verkar förvirrande, så kan man använda bokstäver tillsammans med `chmod`. Grupperna visas då med följande bokstäver:

Ägare	u
Grupp	g
Världen	o
Alla ovanstående	a

För att åstadkomma samma sak som tidigare med siffror, så skulle vi behöva skriva flera kommandorader med bokstäver:

```
$ chmod a+rx /tmp/example
$ chmod u+w /tmp/example
$ chmod ug+s /tmp/example
```

Somliga föredrar bokstäver framför siffror. Vilket man än använder, så resulterar det i samma uppsättning rättigheter.

Vi har tidigare flera gånger nämnt sättande av rättigheter för användar-ID och grupp-ID. Läsaren undrar måhända, vad detta kan vara för något. När man kör ett program, arbetar det vanligen under användarkontot för den användare, som har startat programmet. Programmet har följaktligen samma rättigheter, som en vanlig användare. Det samma gäller för gruppen. När man kör ett program, så exekveras det under den grupp, användaren tillhör. När man sätter UID-rättigheter, så kan man tvinga programmet att alltid köra som om det startats av programmets ägare (t.ex. såsom »**root**«). När man sätter GID inträffar motsvarande för gruppen.

Detta skall man vara försiktig med, eftersom sättande av UID och GID kan öppna stora säkerhetsluckor i systemet. Om man ofta använder UID-program, som ägs av »**root**«, så tillåter man vem som helst att köra programmet ifråga och köra det som **root**. Eftersom **root** inte har några begränsningar i systemet, så inser man lätt, att detta utgör ett stort säkerhetsproblem. Kort sagt, det är inte dåligt att sätta UID eller GID vid körning, men man får därvid även använda sunda förnuftet.

9.3 Länkar.

Länkar är pekare mellan filer. Med länkar kan man låta filer finnas närvarande på många ställen och vara tillgängliga under många namn. Det finns två slags länkar: hårda och mjuka.

Hårda länkar är namn på vissa filer. De kan endast finnas till inom en enda katalog och försvinner endast, om man tar bort det riktiga namnet ifrån systemet. De är användbara i vissa fall, men många användare håller mjuka länkar för mångsidigare än hårda.

En mjuk länk, som även kallas för symbolisk länk (symlänk), kan peka på en fil utanför sin egen katalog. Den är i verkligheten en liten fil, som innehåller de upplysningar, den behöver. Man kan lägga till och avlägsna en mjuk länk utan att påverka den verkliga filen.

Länkar har ingen egen uppsättning rättigheter och ägarskap utan återspeglar i stället dem, som gäller för de filer, de pekar på. Slackware använder för det mesta mjuka länkar. Här är ett vanligt exempel:

```
$ ls -l /bin/sh
lrwxrwxrwx   1 root     root      4 Apr  6 12:34 /bin/sh -> bash
```

Det skal, som i Slackware ibland kallas för `sh`, är i själva verket `bash`. Man kan ta bort länkar med `rm`. Kommandot `ln` användes till att skapa länkar. Dessa kommandon behandlas mer på djupet på s. 118 i kapitel 10.

9.4 Montera enheter.

Såsom det tidigare diskuterats i avsnittet om filsystem på s. 33 i kapitel 4, så utgör alla skivenheter i datorn ett enda stort filsystem. Olika hårddiskpartitioner, CD-ROM-läsare och diskettenheter inordnas i ett och samma katalogträd. För att kunna ympa på alla dessa enheter vid filsystemet, så att de blir åtkomliga, måste man använda kommandona `mount` (1) och `umount` (1).

En del enheter monteras av sig själva, när man startar datorn. Det är sådana, som står förtecknade i filen `/etc/fstab`. Allt, som man vill skall monteras av sig själv, skall föras in i denna fil. Vad andra enheter beträffar, så måste man ge ett kommando var gång, enheten ifråga skall användas.

9.4.1 `fstab`.

Låt oss betrakta ett exempel på en `/etc/fstab`:

```
/dev/sda1     /            ext2      defaults          1  1
/dev/sda2     /usr/local   ext2      defaults          1  1
/dev/sda4     /home        ext2      defaults          1  1
/dev/sdb1     swap         swap      defaults          0  0
/dev/sdb3     /export      ext2      defaults          1  1
none          /dev/pts     devpts    gid=5,mode=620    0  0
none          /proc        proc      defaults          0  0
/dev/fd0      /mnt         ext2      defaults          0  0
/dev/cdrom    /cdrom       iso9660   ro                0  0
```

Första kolumnen är enhetsnamnet. I detta fall är enheterna fem partitioner, som är utspridda över två SCSI-hårddiskar, vidare två speciella filsystem, som inte behöver någon enhet, en diskettenhet och en CD-ROM-enhet. Andra kolumnen visar enhetens monteringsställe. Detta måste vara ett katalognamn, förutom i fallet med swappartitionen. Tredje kolumnen visar enhetens filsystemstyp. För vanliga linuxfilsystem är detta `ext2` (*second extended filesystem*). CD-ROM-enheter är `iso9660` och windowsbaserade enheter endera `msdos` eller `vfat`.

Fjärde kolumnen är en lista med möjliga tillval för filsystemets d monterande. Värdet »`defaults`« duger till det mesta. Emellertid bör man ge enheter, som endast kan läsas, flaggan `ro`. Det finns en mängd tillgängliga valmöjligheter. Fler upplysningar om detta finns på manualsidan för `fstab` (5). De sista två kolumnerna användes av `fsck` och andra kommandon, som behöver manipulera enheterna. Se manualsidan.

När man installerar Slackware Linux, bygger `setup` upp mycket av `fstab`. Det är enbart, när man lägger till diskar eller vill ha några enheter automatiskt monterade vid start, som man behöver införa ändringar i denna fil.[1]

9.4.2 `mount` och `umount`.

Det är enkelt att ympa på en ny enhet vid filsystemet. Man använder då `mount` tillsammans med några tillval. Det kan även göras enklare att använda `mount`, genom att man lägger till en rad för enheten i `/etc/fstab`. Antag exempelvis, att man vill montera sin CD-ROM-enhet och att `fstab` ser ut så som i föregående avsnitts exempel. Man skulle då kunna montera den med följande kommando:

```
# mount /cdrom
```

Eftersom det finns en rad i `fstab` för detta monteringsställe, så »vet« `mount` om, vilka växlar det skall använda. Om ingen sådan rad funnits, skulle man ha behövt förse `mount` med en hel rad växlar:

```
# mount -t iso9660 -o ro /dev/cdrom /cdrom
```

Detta kommando innehåller samma information som exemplet med `fstab`, men vi skall likväl gå igenom det. Växeln `-t iso9660` anger filsystemstyp för den enhet, som skall monteras. I detta fall är det ett filsystem av typen iso-9660, vilket vanligen användes av CD-ROM-skivor. Växeln `-o ro` anger, att enheten skall monteras för läsning enbart. Enhet, som skall monteras, anges med `/dev/cdrom`, och `/cdrom` är dess monteringsställe i filsystemet.

Innan man får avlägsna en diskett, CD-skiva eller annan löstagbar enhet, som för tillfället är monterad, måste man avmontera den. Det gör man med kommandot `umount`. Fråga inte var bokstaven »`n`« har blivit av; det säger vi inte. Man kan använda endera den monterade enheten eller dess monteringsställe såsom argument till `umount`. Om vi t.ex. ville avmontera CD-ROM-skivan i föregående exempel, så skulle bägge dessa kommandon fungera:

```
# umount /dev/cdrom
# umount /cdrom
```

[1] Man kan även behöva ändra i den, om man vill ge vanliga användare rätt att montera vissa enheter. Ö.a.

9.5 NFS-montering.

NFS står för *Network File System*. Det ingår egentligen inte i det riktiga filsystemet, men det kan användas till att ympa på nätverksenheter i det monterade filsystemet.

I stora unixmiljöer delar man ofta på samma program, hemkataloger och spolkataloger för eposten. Problemet att få kopior av katalogerna i var maskin löses med NFS. Vi kan använda NFS till att dela en uppsättning hemkataloger mellan samtliga arbetsstationer. Arbetsstationerna monterar sedan denna NFS-enhet, såsom vore den en enhet i var och en av deras egna maskiner.

Se avsnittet om NFS (*Network File System*) på s. 58 i kapitel 5 samt manualsidorna för `exports`(5), `nfsd`(8) och `mountd`(8).

9.6 Sammanfattning.

I detta kapitel bör man ha samlat en del kunskap om ägarskap och rättigheter. Man bör nu veta varför, sådana finns, och hur man sätter dem. Man bör även känna till länkar emellan filer, hur man monterar skivenheter och hur man monterar NFS-enheter. Dessa tre aspekter på filsystemet är viktiga. Man bör nu ha en grundläggande kunskap om, hur sådana kan användas.

10

Att hantera filer och kataloger.

Slackware Linux siktar emot att bli så Unix-likt, det kan bli. Sedan gammalt har Unix-systemen varit inriktade på användning av kommandoraden. Vi har ett grafiskt användargränssnitt i Slackware, men kommandoraden är likväl det viktigaste redskapet för att styra systemet. Därför är det viktigt att ha grepp om några grundläggande filhanteringskommandon.

Följande avsnitt förklarar några vanliga filhanteringskommandon och visar exempel på hur, de kan användas. Det finns många andra kommandon, men de följande räcker, för att man skall komma igång. Närmare upplysningar finns på manualsidan till vart kommando.

10.1 `ls`.

Detta kommando visar en lista över filerna i en katalog. Windows- och DOS-användare lägger märke till likheten med kommandot `dir`. Utan flaggor (växlar) ger kommandot `ls` en förteckning över filerna i aktuell katalog. För att få se, vad som finns i rotkatalogen, kan man kommendera:

```
$ cd /
$ ls
bin   cdr    dev  home  lost+found  proc  sbin   tmp  var
boot  cdrom  etc  lib   mnt         root  suncd  usr  vmlinuz
```

Det är ett bekymmer för somliga, att man av denna förteckning inte så lätt kan utläsa, vad som är kataloger och vad som är filer. En del användare föredrar att lägga en typidentifierare till förteckningen genom att lägga en flagga till `ls` på följande vis:

```
$ ls -FC
bin/   cdr/    dev/  home/  lost+found/  proc/  sbin/   tmp/  var/
boot/  cdrom/  etc/  lib/   mnt/         root/  suncd/  usr/  vmlinuz
```

Katalogerna får här ett snedstreck vid namnets slut, körbara (exekverbara) filer får en asterisk (stjärna) vid namnets slut o.s.v.

Kommandot `ls` kan även användas till att visa filstatistik. Exempelvis kan man för att få se, när filen skapats, ägarens användarnamn och vilka rättigheter, som är förknippade med filen, välja lång lista:

```
$ ls -l
drwxr-xr-x   2 root     bin          4096 May  7  1994 bin/
drwxr-xr-x   2 root     root         4096 Feb 24 03:55 boot/
drwxr-xr-x   2 root     root         4096 Feb 18 01:10 cdr/
drwxr-xr-x  14 root     root         6144 Oct 23 18:37 cdrom/
drwxr-xr-x   4 root     root        28672 Mar  5 18:01 dev/
drwxr-xr-x  10 root     root         4096 Mar  8 03:32 etc/
drwxr-xr-x   8 root     root         4096 Mar  8 03:31 home/
drwxr-xr-x   3 root     root         4096 Jan 23 21:29 lib/
drwxr-xr-x   2 root     root        16384 Nov  1 08:53 lost+found/
drwxr-xr-x   2 root     root         4096 Oct  6  1997 mnt/
dr-xr-xr-x  62 root     root            0 Mar  4 15:32 proc/
drwxr-x---x  12 root     root         4096 Feb 26 02:06 root/
drwxr-xr-x   2 root     bin          4096 Feb 17 02:02 sbin/
drwxr-xr-x   5 root     root         2048 Oct 25 10:51 suncd/
drwxrwxrwt   4 root     root       487424 Mar  7 20:42 tmp/
drwxr-xr-x  21 root     root         4096 Aug 24  1999 usr/
drwxr-xr-x  18 root     root         4096 Mar  8 03:32 var/
-rw-r---r---   1 root     root       461907 Feb 22 20:04 vmlinuz
```

Antag, att man vill se en förteckning över dolda filer i aktuell katalog. Det kan man åstadkomma med följande kommando:

```
$ ls -a
.              bin   cdrom  home        mnt   sbin   usr
..             boot  dev    lib         proc  suncd  var
.pwrchute_tmp  cdr   etc    lost+found  root  tmp    vmlinuz
```

Filer, vars namn börjar med punkt, är »dolda«, när man kör `ls` utan växlar (flaggor). Man får se dem, om man använder flaggan `-a` till `ls`.

10.2 `cd`.

Kommandot `cd` användes till att växla katalog. Man skriver `cd` åtföljt av den sökväg, man vill växla till. Här följer några exempel:

```
darkstar:~$ cd /bin
darkstar:/bin$ cd usr
bash: cd: usr: No such file or directory
darkstar:/bin$ cd /usr
darkstar:/usr$
```

Lägg märke till, att utan föregående snedstreck (`/`) försöker det växla till en katalog direkt under befintlig katalog.

Kommandot `cd` är inte som andra kommandon. Det är ett inbyggt skalkommando. Inbyggda skalkommandon diskuteras i avsnittet om miljövariabler på s. 101 i kapitel 8. Detta kanske verkar obegripligt än så länge. Det innebär emellertid, att det inte finns någon manualsida för detta kommando. I stället får man använda hjälpen till skalet. Man skriver så här:

```
$ help cd
```

Det visar växlarna till `cd` samt hur, man använder dem.

10.3 more.

`more` (1) är vad som kallas en *pager* (ett program, som visar text en sida i taget). Ofta är utmatning ifrån ett bestämt kommando alltför omfattande för att rymmas på en skärmsida. Enstaka kommandon vet inte, hur de skall dela upp sin utmatning på flera skärmsidor. De överlämnar denna uppgift till en *pager*.

Kommandot `more` bryter ner utmatningen i enstaka skärmbilder och väntar på, att man skall trycka ner mellanslagstangenten, innan det går vidare till nästa skärmbild. När man trycker på returtangenten, matas texten fram en rad. Här är ett exempel:

```
$ cd /usr/bin
$ ls -l
```

Det torde rulla över skärmen en stund. För att dela upp utmatningen i flera skärmsidor, så kan man skicka den igenom en rörledning (*pipeline*) till `more`:

```
$ ls -l | more
```

Där ser vi rörledningstecknet »`|`« (`AltGr` + `<`). Rörledningen (*pipeline*) innebär i detta fall »tag utmatningen ur `ls` och mata in den i `more`«. Man kan skicka nästan vad som helst i en rörledning till kommandot `more` och inte bara utmatning ifrån `ls`. Sådan *piping* behandlas även i avsnittet om rörledning och omdirigering på s. 100 i kapitel 8.

10.4 `less`.

Kommandot `more` är behändigt, men ofta upptäcker man, att den skärmsida, man ville läsa, redan är förbibläddrad. Dessvärre kan man inte bläddra bakåt med `more`. Däremot kan man det med `less` (1). Det användes på samma sätt som `more`, så att föregående exempel äger giltighet även här. Således är `less` mer än `more`.

10.5 `cat`.

`cat` (1) är en förkortning av »*concatenate*« Det har ursprungligen skapats för att slå ihop flera textfiler till en enda, men det kan användas till många andra ändamål.

Om man räknar upp ett antal filer (två eller fler) efter kommandot `cat` och omdirigerar utmatningen till en ny fil, så slås filerna ihop till denna enda fil. `cat` arbetar med *standard input* och *standard output*, så att man måste använda skalets omdirigeringstecken. Exempel:

```
$ cat fil1 fil2 fil3 > storfil
```

Detta kommando tar innehållet i `fil1`, `fil2` samt `fil3` och sätter ihop alltsammans. Den nya utmatningen skickas till *standard out*.

Man kan även begagna `cat` till att visa innehållet i filer. Många »cattar« textfiler genom kommandona `more` eller `less` på följande vis:

```
$ cat fil1 | more
```

Detta visar innehållet i filen `fil1` genom att mata in det i kommandot `more`, så att endast en skärmsida visas åt gången.

Ytterligare en användning för `cat` är kopiering av filer. Man kan kopiera vilken som helst fil med `cat` på följande vis:

```
$ cat /bin/bash > ~/minbash
```

Programmet `/bin/bash` har därmed kopierats till hemkatalogen under namnet »`minbash`«

`cat` har många användningar och de här uppräknade är bara ett fåtal av dem. Eftersom `cat` begagnar *standard input* och *standard output*, så är det mycket användbart i skalskript eller delar av andra, sammansatta kommandon.

10.6 touch.

`touch` användes till att ändra tidsstämpeln på en fil. Man kan ändra tidsstämplar för åtkomst och ändring med detta kommando. Om angiven fil inte är förhanden, så kommer `touch` att skapa en tom fil med angivet namn. Vill man märka en fil med närvarande systemtid, kan man befalla följande:

```
$ touch fil1
```

Det finns flera växlar för `touch` inberäknat växlar för att ange vilken tidsstämpel, som skall ändras, vilken tid, som skall stämplas på och ytterligare många fler. Manualsidan redogör för detaljerna.

10.7 echo.

Kommandot `echo` visar angiven text på skärmen. Man anger efter kommandot den textsträng, som skall visas. Såsom förinställning visar `echo` textsträngen och lägger därefter till tecknet *newline* (`\n`) för radbrytning efteråt. Man kan lägga till växeln `-n` för att undertrycka radbrytningen. Flaggan `-e` får `echo` att söka efter escapetecken i strängen och exekvera dem.

10.8 mkdir.

`mkdir` (1) skapar en ny katalog. Man anger vilken katalog, som skall skapas. Följande kommando skapar katalogen `hejaz` såsom underkatalog till aktuell katalog:

```
$ mkdir hejaz
```

Man kan även ange en sökväg, nämligen så här:

```
$ mkdir /usr/local/hejaz
```

Växeln `-p` talar om för `mkdir`, att det vid behov skall skapa moderkataloger. Ovanstående exempel skulle misslyckas, om katalogen `/usr/local` inte funnes till. Med växeln `-p` skapas `/usr/local` och `/usr/local/hejaz`:

```
$ mkdir -p /usr/local/hejaz
```

10.9 `ln`.

`ln` användes till att skapa länkar emellan filer. Dessa länkar kan vara endera hårda länkar eller mjuka (symboliska) länkar. Skillnaden emellan de två sorternas länkar har behandlats i avsnittet om länkar på s. 108 i kapitel 9. Skulle man vilja skapa en symlänk ifrån katalogen `/var/media/mp3` till sin egen hemkatalog, så skulle man kommendera:

```
$ ln -s /var/media/mp3 ~/mp3
```

Flaggan `-s` bestämmer, att `ln` skall skapa en *symbolisk* länk. Nästa argument är befintlig fil (eller katalog) och till sist står den skapade länkens namn. I detta fall skapas i hemkatalogen en fil vid namn `mp3`, vilken pekar på `/var/media/mp3`. Man kan kalla en länk vadhelst, man vill, genom att ändra sista argumentet.

Det är lika lätt att skapa en hård länk. Man låter då bli att använda växeln `-s`. Följande kommando skapar således en hård länk i stället:

```
$ ln /var/media/mp3 ~/mp3
```

10.10 `cp`.

`cp`(1) kopierar filer. DOS-användare lägger märke till likheten med kommandot `COPY`. Det finns många växlar till `cp`, så man bör läsa manualsidan, innan man använder det.

Det är vanligt att med `cp` kopiera en fil ifrån ett ställe till ett annat. Till exempel:

```
$ cp hejaz /tmp
```

Detta kopierar filen `hejaz` från aktuell katalog till katalogen `/tmp`.

Många användare vill helst bibehålla tidsstämplar, såsom i detta exempel:

```
$ cp -a hejaz /tmp
```

Detta säkerställer, att tidsstämplarna inte förändras vid kopieringen.

Om man vill kopiera innehållet i en katalog rekursivt till en annan katalog, så kommenderar man:

```
$ cp -R enkatalog /tmp
```

Detta kopierar `enkatalog` till katalogen `/tmp`.

`cp` har många fler växlar, vilka skildras mer ingående på manualsidan.

10.11 mv.

`mv` flyttar filer ifrån ett ställe till ett annat. DOS-användare märker likheten med kommandot `MOVE`. Man anger källan och målet efter `mv`. Detta exempel visar en vanlig användning av `mv`:

```
# mv myfile /usr/local/share/hejaz
```

`mv` har många växlar, vilka dokumenterats ingående på manualsidan.

10.12 rm.

`rm` utplånar filer och katalogträd. DOS-användare märker likheten med såväl `DEL` som `DELTREE`. `rm` kan vara ett mycket farligt kommando, om man inte passar sig. I motsats till vad fallet är i DOS eller Windows, så finns i Linux inget sätt att återställa utplånade filer.[1]

Man raderar en enstaka fil genom att ange dess namn efter `rm`:

```
$ rm fil1
```

Om skrivrättigheterna är borttagna ifrån filen, så får man ett felmeddelande om vägrad åtkomst. Man kan tvinga fram (*force*) utplånande av filen med flaggan `-f` på följande vis:

```
$ rm -f fil1
```

Om man vill ta bort en hel katalog, så kan man använda flaggorna `-r` och `-f` tillsammans. Här följer ett fint exempel på, hur man kan utplåna allt innehåll från hårddisken. Det vill man vanligen inte göra. Här är kommandot i alla fall:

```
# rm -rf /
```

Man får vara försiktig med `rm`; eljest kan man lätt komma till att skjuta sig själv i foten. Det finns åtskilliga växlar, som behandlas ingående på manualsidan.

[1] D.v.s. det ingår inte i filsystemet, men det går att ordna med tilläggsprogramvara, som kan användas igenom exempelvis Midnight Commander. Vidare är skrivbordsmiljön KDE försedd med en soptunna, som osäkra användare kan ta för vana att »slänga« filer i i stället för att radera dem direkt. Det bästa är förstås att bara radera filer, man verkligen vill bli av med. Ö.a.

10.13 `rmdir`.

`rmdir`(1) avlägsnar kataloger ifrån filsystemet. Katalogen måste vara tom för att kunna raderas med `rmdir`. Syntax är enkel:

```
$ rmdir <katalog>
```

Detta exempel tar bort underkatalogen `hejaz` ifrån aktuell arbetskatalog:

```
$ rmdir hejaz
```

Om katalogen ifråga inte finns, talar `rmdir` om detta. Man kan även ange fullständig sökväg till den katalog, som skall utplånas, såsom följande exempel visar:

```
$ rmdir /tmp/hejaz
```

Detta exempel försöker ta bort katalogen `hejaz` under katalogen `/tmp`.

Man kan även utplåna en katalog tillsammans med samtliga dess moderkataloger genom att använda flaggan `-p`:

```
$ rmdir -p /tmp/hejaz
```

Detta försöker först ta bort `hejaz` under `/tmp`. Om det lyckas, så fortsätter kommandot med att avlägsna `/tmp`. `rmdir` fortsätter ända tills ett fel tillstöter, eller hela det angivna katalogträdet utplånats.

10.14 Sammanfattning.

Detta kapitel har behandlat en mängd program, som hanterar filer och kataloger. Man bör nu veta, hur man skall skapa, utplåna och flytta nästan vad som helst i filsystemet. Man bör även veta, hur man skall förteckna och ändra tidsstämplar på filer, när så erfordras. Slutligen bör man ha klart för sig, varför det är mycket illa tänkt att kommendera `rm -rf /`.

11

Att styra processer.

Alla program, som är igång, kallas för *processer*. Dessa processer kan t.ex. vara fönstersystemet X eller olika systemprogram (demoner), som startas, när man startar datorn. Var process köres av någon särskild användare. Sådana processer, som sättes igång vid systemstart, körs vanligen av **root** eller **nobody**. Sådana processer, man själv startar i egenskap av vanlig användare, körs av den användare, man är inloggad som.

Man får styra alla processer, man själv har startat. Därtill får **root** styra samtliga processer i systemet inbegripet dem, som andra användare har satt igång. Processer kan styras och övervakas såväl genom flera olika program som genom några kommandon i skalet.

11.1 Lägga process i bakgrunden.

Sådana program, som igångsättes ifrån kommandoraden, startar i *förgrunden*. Detta gör, att man kan se programmets utdata och samverka med programkörningen. Emellertid förekommer tillfällen, då man hellre vill ha programmet till att köra utan att lägga beslag på terminalen. Detta benämnes att *köra programmet i bakgrunden*, och det kan man åstadkomma på flera olika sätt.

Första sättet att lägga en process i bakgrunden är att lägga till et-tecknet »`&`« efter programmets namn, när man startar det på kommandoraden. Antag t.ex., att vi skulle vilja använda `amp`, som är en mp3-spelare för kommandoraden, och låta den spela en hel katalog full med mp3-filer, men att vi behöver göra något annat på samma terminal. Följande kommando sätter då igång `amp` i bakgrunden och laddar samtliga mp3-filer i katalogen:

```
$ amp *.mp3 &
```

Programmet kör som vanligt, men man får tillbaka prompten »`$`«.

Man kan även lägga en process i bakgrunden, när den redan är igång och kör. Först startar vi ett program. När det är igång och kör, trycker vi på `Ctrl` + `z`. Detta lägger processen i pausläge (suspenderar processen), så att den tillfälligt upphör med att köra.[1] Det kan dock när som helst startas igen. Har man pausat ett program, så får man prompten tillbaka. Då kan man lägga programmet i bakgrunden genom att kommendera

```
$ bg
```

Nu kommer den tidigare pausade processen att starta igen, men denna gång i bakgrunden.

11.2 Lägga process i förgrunden.

Om man behöver samverka med en process, som lagts i bakgrunden, så kan man få fram den i förgrunden igen. Har man bara en enda process i bakgrunden, kan man få fram den med följande kommando:

```
$ fg
```

Om programmet inte har slutat köra, kommer det att att överta terminalen, och man får då inte tillbaka prompten. Ibland händer det, att något program avslutar körningen, medan det ligger i bakgrunden. I så fall visas ett meddelande i stil med följande:

```
[1]+  Done                  /bin/ls $LS_OPTIONS
```

Detta talar om, att bakgrundsprocessen (i detta fall `ls`) har kört färdigt.

Det är möjligt att ha flera bakgrundsprocesser igång samtidigt. När detta inträffar, behöver man veta vilka processer, man skall hämta fram till förgrunden. Att enbart slå `fg` skulle hämta fram den process, som sist lades i bakgrunden. Men tänk, om man nu skulle ha en hel lista med processer i bakgrunden? Lyckligtvis innehåller `bash` ett kommando för att förteckna alla processer. Det heter `jobs` och ger en utmatning som denna:

```
$ jobs
[1]   Stopped                  vim
[2]-  Stopped                  amp
[3]+  Stopped                  man ps
```

[1] Samma tangenttryckning i X minimerar programmet till en ikon. Ö.a.

Detta visar en lista över alla processer, som skjutits i bakgrunden. Som synes är samtliga stannade. Detta innebär, att processerna är suspenderade (pausade). Siffran före är en sorts ID för bakgrundsprocesser. Ett sådant ID med plustecken framför (`man ps`) betecknar den process, som skall läggas i förgrunden, om man kommenderar enbart `fg`.

Om man vill hämta fram `vim` till förgrunden, så kan man skriva:

```
$ fg 1
```

så far `vim` upp på skärmen. Det kan vara mycket användbart att lägga processer i bakgrunden, när man har endast en terminal öppen över en uppringd förbindelse. Man kan på så vis ha flera program igång samtidigt på denna enda terminal, medan man då och då växlar fram och tillbaka mellan dem.

11.3 ps.

Nu vet vi alltså, hur vi skall kunna växla fram och tillbaka mellan flera processer, som vi har startat på kommandoraden. Vi vet också, att mängder av processer är igång samtidigt. Hur skall man då kunna få se en förteckning över alla dessa program? Det gör man med kommandot `ps`(1). Detta kommando har många växlar, så att vi bara skall behandla de viktigaste här. En fullständig förteckning finns på manualsidan för `ps`. Manualsidor behandlas ingående i avsnittet om `man` på s. 7 i kapitel 2.

Genom att enbart kommendera `ps` får man en lista över program, som kör på terminalen. Ofta är detta en mycket kort lista:

```
$ ps
   PID TTY          TIME CMD
  7923 ttyp0    00:00:00 bash
  8059 ttyp0    00:00:00 ps
```

Även om detta inte är så många processer, så är informationen typisk. Man får samma kolumner, då man använder `ps`, oavsett hur många processer, som är igång. Vad betyder då detta?

PID står för process-ID. Alla processer, som är igång, får ett unikt identifieringsnummer. Med linuxkärnor i 2.2.x-serien kan detta PID vara vad som helst emellan 1 och 32767. Var process tilldelas nästa lediga PID. När en process avslutas (eller dödas, såsom vi skall få se i nästa avsnitt), kommer nästa lediga nummer att gå tillbaka till det lägsta lediga numret. Detta skall sannolikt ändras i den kommande 2.4-serien av linuxkärnan och införandet av 32-bit-PID.

Kolumnen TTY visar vilken terminal, processen kör på. Ett enkelt `ps` visar endast en lista över program, som kör på aktuell terminal, så att alla processer ger samma information i TTY-kolumnen. Som synes köres bägge de här förtecknade processerna på `ttyp0`. Detta visar, att de endera kör på en fjärrdator eller på en X-terminal av något slag.

Kolumnen TIME visar, hur mycket CPU-tid (processortid), processen har tillryggalagt. Denna avviker ifrån den verkliga tid, processen har varit igång. Man påminner sig, att Linux är ett operativsystem, som kan köra flera program samtidigt. Hela tiden är många processer igång och dessa processer får vardera en liten del av processortiden. Därför bör kolumnen TIME visa mycket mindre tid för var process, än den verkligen har hållit på att köra. Om man ser mer än några minuter i kolumnen TIME, betyder det, att någonting är i olag.

Slutligen visar kolumnen CMD vad programmet egentligen är för något. Denna kolumn förtecknar endast programmets namn, inga växlar eller dylika upplysningar. För att få sådana upplysningar måste vi använda en av de många flaggor, som hör till `ps`. Vi skall snart diskutera detta.

Man kan få en fullständig förteckning över processer, som är igång i systemet, om man använder en lämplig uppsättning växlar. Det kommer förmodligen att utmynna i en lång lista över processer (femtiofem i vår egen bärbara, medan vi skriver denna mening), så att vi förkortar återgivandet av utmatningen:

```
$ ps -ax
PID TTY      STAT   TIME COMMAND
  1 ?        S      0:03 init [3]
  2 ?        SW     0:13 [kflushd]
  3 ?        SW     0:14 [kupdate]
  4 ?        SW     0:00 [kpiod]
  5 ?        SW     0:17 [kswapd]
 11 ?        S      0:00 /sbin/kerneld
 30 ?        SW     0:01 [cardmgr]
 50 ?        S      0:00 /sbin/rpc.portmap
 54 ?        S      0:00 /usr/sbin/syslogd
 57 ?        S      0:00 /usr/sbin/klogd -c 3
 59 ?        S      0:00 /usr/sbin/inetd
 61 ?        S      0:04 /usr/local/sbin/sshd
 63 ?        S      0:00 /usr/sbin/rpc.mountd
 65 ?        S      0:00 /usr/sbin/rpc.nfsd
 67 ?        S      0:00 /usr/sbin/crond -l10
 69 ?        S      0:00 /usr/sbin/atd -b 15 -l 1
 77 ?        S      0:00 /usr/sbin/apmd
 79 ?        S      0:01 gpm -m /dev/mouse -t ps2
 94 ?        S      0:00 /usr/sbin/automount /auto file /etc/auto.misc
106 tty1     S      0:08 -bash
108 tty3     SW     0:00 [agetty]
109 tty4     SW     0:00 [agetty]
```

```
   110 tty5     SW     0:00 [agetty]
   111 tty6     SW     0:00 [agetty]
[utmatning avklippt]
```

Flertalet av dessa processer startas vid systemstart på de flesta maskiner. Vi har dock gjort några förändringar i vårt system, så att erfarenheterna härvidlag förmodligen går isär. Emellertid kan man se de flesta av dessa processer i vilket system som helst. Som synes visar dessa växlar även de körande processernas växlar. Vi får här också några fler kolumner och en del annan intressant utmatning.

För det första lägger man märke till, att de flesta av dessa processer uppges köra på en tty vid namn »?«. Detta är processer, som startats ifrån en numera icke längre aktiv terminal. Därför är de heller inte längre bundna till någon särskild terminal.

För det andra finns här en ny kolumn: STAT. Den visar processens status (tillstånd). S står för sovande (*sleeping*); processen väntar på, att någonting skall hända. Z står för zombieprocess. En zombieprocess är en process, vars förälder har dött, så att barnprocessen är övergiven. Detta är inte bra.

Om man vill ha ännu mer information om processer, som är igång, kan man försöka med detta kommando:

```
   $ ps -aux
   USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
   root         1  0.0  0.0   344   80 ?        S    Mar02   0:03 init [3]
   root         2  0.0  0.0     0    0 ?        SW   Mar02   0:13 [kflushd]
   root         3  0.0  0.0     0    0 ?        SW   Mar02   0:14 [kupdate]
   root         4  0.0  0.0     0    0 ?        SW   Mar02   0:00 [kpiod]
   root         5  0.0  0.0     0    0 ?        SW   Mar02   0:17 [kswapd]
   root        11  0.0  0.0  1044   44 ?        S    Mar02   0:00 /sbin/kerneld
   root        30  0.0  0.0  1160    0 ?        SW   Mar02   0:01 [cardmgr]
   bin         50  0.0  0.0  1076  120 ?        S    Mar02   0:00 /sbin/rpc.port
   root        54  0.0  0.1  1360  192 ?        S    Mar02   0:00 /usr/sbin/sysl
   root        57  0.0  0.1  1276  152 ?        S    Mar02   0:00 /usr/sbin/klog
   root        59  0.0  0.0  1332   60 ?        S    Mar02   0:00 /usr/sbin/inet
   root        61  0.0  0.2  1540  312 ?        S    Mar02   0:04 /usr/local/sbi
   root        63  0.0  0.0  1796   72 ?        S    Mar02   0:00 /usr/sbin/rpc.
   root        65  0.0  0.0  1812   68 ?        S    Mar02   0:00 /usr/sbin/rpc.
   root        67  0.0  0.2  1172  260 ?        S    Mar02   0:00 /usr/sbin/cron
   root        77  0.0  0.2  1048  316 ?        S    Mar02   0:00 /usr/sbin/apmd
   root        79  0.0  0.1  1100  152 ?        S    Mar02   0:01 gpm
   root        94  0.0  0.2  1396  280 ?        S    Mar02   0:00 /usr/sbin/auto
   chris      106  0.0  0.5  1820  680 tty1     S    Mar02   0:08 -bash
   root       108  0.0  0.0  1048    0 tty3     SW   Mar02   0:00 [agetty]
   root       109  0.0  0.0  1048    0 tty4     SW   Mar02   0:00 [agetty]
   root       110  0.0  0.0  1048    0 tty5     SW   Mar02   0:00 [agetty]
   root       111  0.0  0.0  1048    0 tty6     SW   Mar02   0:00 [agetty]
[utmatning avklippt]
```

Här visas en hel mängd information. Här tillkommer upplysningar om vilken användare, som startat en process, hur stor andel av systemresurserna, en process använder (kolumnerna %CPU, %MEM, VSZ och RSS), samt vilket datum, processen har startats. Uppenbarligen är detta många upplysningar, som kan komma till nytta för en systemadministratör. Det visar en sak till: texten passerar skärmkanten, så att man inte ser allt. Det kan man lösa med flaggan `-w`.

Det är inte så vackert, men det fullgör sitt arbete. Vi har nu fått ut fullständiga listor över alla processer. Manualsidan för `ps` går på djupet med detta. Emellertid har vi här visat de mest omtyckta alternativen, som man oftast har anledning att ta i bruk.

11.4 `kill`.

Då och då händer det, att något program missköter sig, så att man måste tillrättavisa det. Det program, som brukas vid sådan maktutövning, heter `kill`(1), och det kan brukas vid hantering av processer på flera olika sätt. Det uppenbaraste sättet är bruket av `kill` till att döda någon process. Det behöver man göra, när ett program har skenat och slukar mängder av systemresurser, eller också när man bara är trött på, att det är igång och kör.

För att döda en process behöver man känna till dess PID eller dess namn. För att få tag i dess PID kan man använda kommandot `ps`, såsom det visats i föregående avsnitt. För att döda process nummer 4747, skulle man kunna kommendera följande:

```
$ kill 4747
```

Lägg märke till, att man måste vara processens ägare för att få döda den. Detta är av säkerhetsskäl. Finge man döda processer, som andra startat, så skulle man kunna begå allehanda illdåd. Självfallet får **root** döda vilka som helst processer i systemet.

Det finns en annan avart av kommandot `kill`, som heter `killall`(1). Detta program gör precis det, som namnet säger: det dödar samtliga processer med ett visst namn. Om man skulle vilja döda alla `vim`-processer, som är igång, så skulle man kunna kommendera:

```
$ killall vim
```

Vilka som helst och samtliga `vim`-processer skulle dö. Om **root** gjorde detta, skulle alla användares `vim`-processer dö. Detta för tanken till ett intressant sätt att sparka ut samtliga (sig själv inberäknat) ur systemet:

```
# killall bash
```

Ibland räcker det inte med ett vanligt `kill`. Vissa processer vill inte dö med `kill`. Man får ta till en kraftigare form. Om det förhatliga PID 4747 inte hade svarat på vår `kill`-begäran, så skulle vi ha fått ta till följande:

```
$ kill -9 4747
```

Detta skall nästan säkert döda process 4747. Man kan åstadkomma samma sak med `killall`. Detta sänder en annan signal till processen. Ett vanligt `kill` sänder signalen `SIGTERM` (terminate) till processen. Kommandot `kill -9` sänder däremot signalen `SIGKILL` (kill) till processen. Det finns en hel lista över sådana signaler, som står till förfogande. Vi kan få se en sådan förteckning genom att kommendera följande:

```
$ kill -l
  1) SIGHUP      2) SIGINT     3) SIGQUIT    4) SIGILL
  5) SIGTRAP     6) SIGABRT    7) SIGBUS     8) SIGFPE
  9) SIGKILL    10) SIGUSR1   11) SIGSEGV   12) SIGUSR2
 13) SIGPIPE    14) SIGALRM   15) SIGTERM   17) SIGCHLD
 18) SIGCONT    19) SIGSTOP   20) SIGTSTP   21) SIGTTIN
 22) SIGTTOU    23) SIGURG    24) SIGXCPU   25) SIGXFSZ
 26) SIGVTALRM 27) SIGPROF    28) SIGWINCH 29) SIGIO
 30) SIGPWR
```

Till `kill` måste man använda en siffra, medan namnet minus det inledande »`SIG`« kan användas till `killall`. Här följer ytterligare ett exempel:

```
$ killall -KILL vim
```

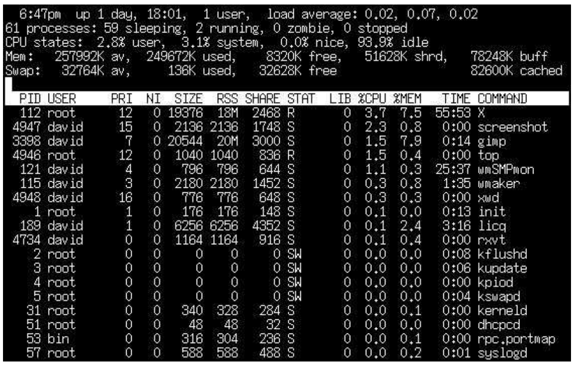
Slutligen kan man använda `kill` till att starta om en process. Om man sänder en `SIGHUP`, så läser de flesta processer in sina inställningsfiler på nytt. Detta är till särskilt god hjälp, när man skall få systemprocesser att läsa in sina konfigureringsfiler efter det, att man har redigerat de senare.

11.5 top.

Slutligen finns ett kommando, som man kan använda till att visa efter hand förnyade upplysningar om de processer, som är igång i systemet. Detta kommando heter `top`(1) och det startas på följande vis:

```
$ top
```

Detta visar en fullskärm med information om processer, som är igång i systemet, liksom en del allmänna upplysningar om systemet. Däri ingår genomsnittlig belastning, antal processer, CPU-tillstånd och information om ledigt minne samt enskildheter om processerna såsom PID, användare, prioritet, CPU- och minnesanvändning, körtid och programnamn.



```
  6:47pm  up 1 day, 18:01,  1 user,  load average: 0.02, 0.07, 0.02
61 processes: 59 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  2.8% user,  3.1% system,  0.0% nice, 93.9% idle
Mem:   257992K av,  249672K used,    8320K free,   51628K shrd,   78248K buff
Swap:   32764K av,     136K used,   32628K free                   82600K cached
```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	LIB	%CPU	%MEM	TIME	COMMAND
112	root	12	0	19376	18M	2468	R	0	3.7	7.5	55:53	X
4947	david	15	0	2136	2136	1748	S	0	2.3	0.8	0:00	screenshot
3398	david	7	0	20544	20M	3000	S	0	1.5	7.9	0:14	gimp
4946	root	12	0	1040	1040	836	R	0	1.5	0.4	0:00	top
121	david	4	0	796	796	644	S	0	1.1	0.3	25:37	wmSMPmon
115	david	3	0	2180	2180	1452	S	0	0.3	0.8	1:35	wmaker
4948	david	16	0	776	776	648	S	0	0.3	0.3	0:00	xwd
1	root	1	0	176	176	148	S	0	0.1	0.0	0:13	init
189	david	1	0	6256	6256	4352	S	0	0.1	2.4	3:16	licq
4734	david	0	0	1164	1164	916	S	0	0.1	0.4	0:00	rxvt
2	root	0	0	0	0	0	SW	0	0.0	0.0	0:08	kflushd
3	root	0	0	0	0	0	SW	0	0.0	0.0	0:06	kupdate
4	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	kpiod
5	root	0	0	0	0	0	SW	0	0.0	0.0	0:04	kswapd
31	root	0	0	340	328	284	S	0	0.0	0.1	0:00	kerneld
51	root	0	0	48	48	32	S	0	0.0	0.0	0:00	dhcpcd
53	bin	0	0	316	304	236	S	0	0.0	0.1	0:00	rpc.portmap
57	root	0	0	588	588	488	S	0	0.0	0.2	0:01	syslogd

Figur 11.1: Kommandot `top` visar information om processer.

Det kallas `top` därför, att de CPU-intensivaste programmen visas överst. En intressant anmärkning är, att `top` står överst på de flesta inaktiva (och en del aktiva) system p.g.a. dess CPU-användning. Emellertid är `top` mycket användbart, när man skall avgöra, vilket program det är, som missköter sig och behöver dödas.

11.6 Sammanfattning.

Detta kapitel har diskuterat, vad en process är, och hur man kan styra den. Häri ingår läggande av processer i bakgrunden och förgrunden såväl som bruk av `ps`, `top`, och `kill` för att hålla dem rättade i ledet. Man bör nu kunna avgöra vilka processer, som är igång i systemet, och hur man skall bli av med dem, om de upphör med att uppföra sig ordentligt.

12

Grundläggande systemadministration.

På vilken som helst dator, man har **root**-rättigheter till, är man *systemadministratör*. Det kan vara ens egen skrivbordsdator med en eller två användare, eller det kan vara en stor server med flera hundra användare. Oavsett vilketdera fallet är, så måste man veta, hur man skall hantera användare och hur man skall kunna stänga av systemet på ett säkert sätt. Bäggedera kan tyckas vara enkla, men de har några egenheter, man skall vänja sig vid. Därtill måste man bekanta sig en smula med det tänkande, som ligger bakom lösenordssystemet.

12.1 Användare och grupper.

12.1.1 Medföljande skript (kommandofiler).

Det enklaste sättet att hantera användare och grupper är att använda de medföljande skripten och programmen. Slackware innehåller programmen `adduser`, `userdel` (8), `chfn` (1), `chsh` (1) och `passwd` (1) till att hantera användare. Vidare ingår `groupadd` (8), `groupdel` (8) och `groupmod` (8) till att hantera grupper. Med undantag av `chfn`, `chsh` och `passwd` så kan dessa program endast köras av **root** och ligger följaktligen i katalogen `/usr/sbin`. `chfn`, `chsh` och `passwd` kan köras av vilken användare som helst och ligger i katalogen `/usr/bin`.

Man lägger till användare med programmet `adduser`. Vi börjar här med att gå igenom hela arbetsgången, så att vi visar de frågor, som ställs, och en kortfattad beskrivning av deras innebörd. Förvalt svar står inom klammer [ ] och detta svar kan väljas på nästan alla frågor, såvida man inte vill ändra på någonting.

```
# adduser
Login name for new user (8 characters or less) []: jellyd
```

Detta är det namn, användaren anger vid inloggning. Det måste bestå av åtta tecken eller färre, eftersom alla inloggningstillbehör förväntar sig, att det skall ha denna längd. I allmänhet bör man använda uteslutande gemena (små bokstäver), såvida man inte gärna vill utsätta sig för obehaget att knappa in stora bokstäver i situationer, när detta kan vara obekvämt.

```
User id for jellyd [ defaults to next available]:
```

Användar-ID (UID) är ett nummer, som är det verkliga sättet att bestämma ägarskap i Linux. Var användare har ett unikt nummer, och användarnas UID börjar med 1000 i Slackware.[1] Man kan välja ett UID för den nye användaren, eller också kan man låta programmet `adduser` tilldela användaren första lediga nummer.[2]

```
Initial group for jellyd [users]:
```

Alla användare placeras såsom förval i gruppen **users**.[3] Man kan lägga den nye användaren i någon annan grupp, men det anbefalles icke.

```
jellyd's shell [/bin/bash]
```

`bash` är förvalt kommandoskal för Slackware Linux och passar bra för de flesta. Om den nye användaren har ett förflutet i något annat Unix-system, så kan det hända, att han är van vid något annat skal. I så fall kan man ge honom ett annat skal ifrån början, eller också kan han ändra det själv senare med kommandot `chsh`.

```
jellyd's account expiry date (YYYY-MM-DD) []:
```

[1] I andra linuxdistributioner kan det hända, att man börjar användarnummer på 500 i stället för 1000, och FreeBSD börjar på UID 1001. Om man då flyttar filer och kataloger mellan systemen, kan det bli krångel med ägarskap. Ö.a.

[2] Om användaren ifråga skall ha tillgång till samma kataloger och filer, när en annan linuxdistribution köres, kan det vara förnuftigt att låta användaren få samma UID i de olika distributionerna. Annars behöver man inte ändra på detta. Ö.a.

[3] I vissa andra linuxdistributioner placeras nya användare såsom förval i en egen grupp med samma namn som användaren. Ö.a.

Ett användarkonto kan upprättas så, att det löper tillända på ett i förväg bestämt datum. Såsom förval finns inget slutdatum. Detta kan ändras, om så önskas. Denna möjlighet kan vara nyttig för en internetleverantör, som vill låta kontot upphöra ett visst datum, om nästa års avgift inte betalats in.

```
OK, I'm about to make a new account. Here's
what you entered so far:

New login name: jellyd
New UID: [Next available]
Initial group: users
Additional groups: [none]
Home directory: /home/jellyd
Shell: /bin/bash
Expiry date: [no expiration]

This is it... if you want to bail out, hit Control-C.
Otherwise, press ENTER to go ahead and make the account.
```

Nu får man se alla upplysningar, som matats in om det nya kontot, och så får man möjlighet att avbryta. Om man har skrivit in något felaktigt, så måste man använda `Ctrl-C` och börja från början igen. I annat fall kan man nu slå `Retur`, och så upprättas kontot.

```
Making new account...

Changing the user information for jellyd
Enter the new value, or press return for the default
        Full Name []: Jeremy Smith
        Room Number []: 130
        Work Phone []:
        Home Phone []:
        Other:
```

Alla dessa upplysningar är frivilliga. Man behöver inte mata in något av detta, om man inte vill, och användaren kan ändra på det när som helst med kommandot `chfn`. Det kan dock vara lämpligt att åtminstone skriva in fullständigt namn och telefonnummer, uti fall att man skulle behöva sätta sig i förbindelse med personen ifråga.

```
   Changing password for jellyd
Enter the new password (minimum of 5, maximum of 127 characters)
```

```
Please use a combination of upper and lower case letters and numbers.
New password:
Re-enter new password:
Password changed.
Done...
```

Nu skall man skriva in ett lösenord för den nye användaren. Om den nye användaren inte är kroppsligen tillstädes, så är det i allmänhet lämpligt att ge honom något enkelt lösenord och säga till vederbörande att ändra det själv till något säkrare.

Anmärkning: om val av nytt lösenord. Att ha ett säkert lösenord är första försvarslinjen emot intrång. Man vill inte gärna ha ett lösenord, någon annan lätt kan gissa sig till, eftersom det då blir lättare att bryta sig in i systemet. Ett säkert lösenord vore som bäst en på måfå alstrad teckenföljd med såväl versaler (stora bokstäver) som gemena (små bokstäver), siffror och andra sorters tecken. Man skall dock därvid komma ihåg, att ett tabulatortecken nog inte är ett klokt val, eftersom ett sådant kan gestaltas på olika sätt i olika datorer, man kan tänkas logga in ifrån.

I allmänhet gäller det bara att använda sig av sunda förnuftet: man bör inte välja ett lösenord, som betecknar någons födelsedag, ett vanligt talesätt, något föremål på skrivbordet eller något som helst, som är förknippat med en själv. Lösenordet »secure1« är t.ex. också dåligt.

Det är inte alls svårt att ta bort användare. Man kör bara `userdel` med namnet på det konto, som skall avlägsnas, såsom argument. Därvid måste man först förvissa sig om, att användaren ifråga inte är inloggad, samt att inga processer är igång, vilka startats av denne användare. Märk även väl, att en borttagen användare är och förblir borttagen.

```
# userdel jellyd
```

Om vi kör detta kommando, blir vi av med denne »jellyd«, som vi är rätt trötta på nu. Skönt. Detta avlägsnar användaren ifrån filerna `/etc/passwd` och `/etc/group`, men det tar inte bort användarens hemkatalog. Om vi hade velat ta bort användarens hemkatalog dessutom, skulle vi ha kommenderat:

```
# userdel -r jellyd
```

Hur man tillfälligt stänger av ett konto kan man läsa om i avsnittet »Att ändra lösenord« på s. 135, eftersom detta inbegriper ändring av användarens lösenord. Ändring av kontouppgifter skildras i avsnitten »Ändra lösenord« på s. 135 och »Ändra användarupplysningar« på s. 136.

De program, som lägger till och tar bort grupper, är mycket enkla. Programmet `groupadd` lägger bara till en ny rad i filen `/etc/group` med ett unikt grupp-ID, medan `groupdel` tar bort angiven grupp. Man får själv gå in och redigera filen `/etc/group`, om man vill lägga till bestämda, befintliga användare i någon viss grupp:

Man kan skapa en ny grupp med kommandot

```
# groupadd cvs
```

Man kan ta bort en grupp med kommandot

```
# groupdel cvs
```

12.1.2 För hand.

Det är självfallet möjligt att lägga till, ändra och ta bort användare eller grupper för hand. Efter en genomgång av detta förfarande tycker man nog, det är bekvämare att använda skripten.

Först skall vi lägga till en ny användare såväl i filen `/etc/passwd`(5) som i filen `/etc/shadow`(5) och i `/etc/group`. Filen `passwd` innehåller en del upplysningar om användarna, men (märkvärdigt nog) inte deras lösenord. Filen `passwd` måste vara läsbar för vem som helst, men man vill inte, att alla skall kunna läsa de krypterade lösenorden, eftersom detta skulle ge tilltänkta intränglingar en god början. Av den anledningen förvaras de krypterade lösenorden i filen `shadow`, vilken endast kan läsas av **root**, och i filen `passwd` är lösenorden ersatta med ett `x`. Filen `group` innehåller en förteckning över samtliga grupper och vilka användare, som ingår i dem.

Nu skall vi undersöka filen `/etc/passwd` och se, hur vi kan lägga till någon. En typisk post i `passwd` kan se ut så här:

```
chris:x:1000:100:Chris Lumens,Room 2,,:/home/chris:/bin/bash
```

Var rad är en post, som gäller en person, och fälten i denna post är avgränsade medelst kolon. Fälten utgöres av inloggningsnamn, krypterat lösenord (`x` för alla i ett Slackware-system, eftersom vi använder *shadow password suite*), ett användar-ID, ett grupp-ID, frivilliga upplysningar för programmet `finger` åtskilda av kommatecken samt slutligen hemkatalog och skal. När man skall lägga till en ny användare, skriver man in en ny rad (lika med post) i denna fil utan att utelämna något fält eller något obligatoriskt fältinnehåll.

Man skall därvid förvissa sig om, att lösenordet är ett `x`, att användar-ID är unikt, att användaren tillhör grupp 100 (gruppen »**users**« i Slackware) och att han har ett giltigt inloggningsskal.

Därefter måste vi lägga till en post i filen `/etc/shadow`, som innehåller lösenorden. En typisk post ser ut så här:

```
chris:$1$w9bsw/N9$UWLr2bRER6YyBS.CAEp7R.:11055:0:99999:7:::
```

Återigen är var linje en post för en person, och fälten avgränsas medelst kolon. Fälten utgöres av inloggningsnamn, krypterat lösenord, antal dagar efter *The Epoch*[4] (1 januari 1970) som lösenordet senast ändrats, antal dagar innan lösenordet får ändras, antal dagar varefter lösenordet måste ändras, antal dagar före lösenordets utgångsdatum, som användaren skall förvarnas, antal dagar efter utgången som kontot blir overksamt, antal dagar efter »Epoken«, som kontot skall vara overksamt, och slutligen ett reserverat fält.

Som synes gäller det mesta här kontots utgångsdatum. Om man inte använder någon information om sista datum för kontot, så behöver man bara fylla i ett fåtal fält med särskilda värden. I annat fall får man utföra några beräkningar och fatta några beslut, innan man kan fylla i dessa fält. För vår exempelanvändare kan vi fylla i något på måfå hopplockat skräp i lösenordsfältet. För ögonblicket spelar det ingen roll, vilket lösenordet verkligen är, eftersom vi skall ändra det snart. Det enda tecken, vi inte får skriva i lösenordsfältet, är kolon. Vi kan också lämna fältet för »dagar sedan lösenordet senast ändrats« tomt. Vi fyller i 0, 99999 och 7, såsom det står i exemplet, och så lämnar vi övriga fält tomma.

De, som ser vårt krypterade lösenord här ovan och tror, att de har hittat ett sätt att bryta sig in i vårt system, kan ju alltid försöka. Kan ni knäcka det lösenordet, så har ni lösenordet till ett testsystem bakom en brandväg. Det har man nytta av `:)`

Eftersom alla såsom förinställning är medlemmar i gruppen **users**, så behöver man inte lägga till nya användare i denna grupp. Skulle man vilja skapa en ny grupp eller lägga till den nye användaren i några andra grupper, så måste man ändra i filen `/etc/group`. Här är en typisk post:

```
cvs::102:chris,logan,david,root
```

Fälten utgöres av gruppnamn, grupplösenord, grupp-ID och gruppmedlemmar. Att skapa en ny grupp är bara en fråga om att lägga till en ny rad

[4] *The Epoch*: Tideräkningens början för Unix och unixliknande system. Ö.a.

med ett unikt grupp-ID och räkna upp alla de användare, man vill ha med i gruppen. Vilka som helst användare, som tillhör denna nya grupp och är inloggade, måste logga ut och logga in igen, för att dessa ändringar skall träda i kraft.

Nå, då skall vi gå tillbaka och använda kommandot `passwd` till att skapa ett nytt lösenord för användaren. Därefter skall vi använda `mkdir` till att skapa den nye användarens hemkatalog på det ställe, som angivits i filen `/etc/passwd`.

Om man har installerat `sendmail` (8) i systemet och verkligen använder epost, så skall man skapa en fil i katalogen `/var/spool/mail` med lämpliga rättigheter och tillhörighet till denne nye användare. Här följer ett exempel:

```
# touch /var/spool/mail/jellyd
# chown jellyd.users /var/spool/mail/jellyd
# chmod 660 /var/spool/mail/jellyd
```

Dessa kommandon skapar en fil för inkommande post till den nye användaren »**jellyd**« och förser denna fil med rätt ägarförhållande och tillstånd.

Att ta bort en användare gäller bara att bli kvitt allt detta, vi nyss skapat. Man tar bort användarens post i `/etc/passwd` och alla förekomster av denne användare i `/etc/group`, så att man tar bort hans inloggningsnamn ifrån alla grupper. Vidare utplånar man hans fil för inkommande epost, om en sådan finnes, och slutligen hemkatalogen, om det behövs.

Borttagande av grupper är bara en fråga om att ta bort raden för denna grupp ifrån filen `/etc/group`.

12.1.3 Ändra lösenord.

Programmet `passwd` byter ut lösenord genom att ändra i `/etc/shadow`. Denna fil innehåller alla systemets lösenord i krypterad form. När man vill ändra sitt lösenord, kommenderar man

```
$ passwd
Changing password for chris
Old password:
Enter the new password (minumum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
```

Som synes, blir man uppmanad att skriva in sitt gamla lösenord. Det syns inte på skärmen, medan man skriver in det, liksom fallet är, när man loggar in. Därefter uppmanas man att skriva in det nya lösenordet. Programmet `passwd` utför flera kontroller på det nya lösenordet och klagar, om det inte får godkänt. Man kan bortse ifrån varningarna, om man vill. Slutligen uppmanas man mata in det nya lösenordet en gång till som bekräftelse.

Om man är **root**, så kan man även ändra andra användares lösenord:

```
$ passwd ted
```

Sedan får man gå igenom samma förfarande som tidigare förutom det, att man inte behöver skriva in det gamla lösenordet (detta är en av de många fördelarna med att vara **root**...).

Om man har några bråkmakare i systemet, så kan man även tillfälligt stänga deras konton. Senare kan man öppna dem igen. Både att stänga ett konto och att öppna det igen är åtgärder, som kan utföras med `passwd`. Man stänger ett konto genom att såsom **root** kommendera

```
$ passwd -l david
```

Detta ändrar **david**s lösenord till någonting, som aldrig kan överensstämma med det krypterade lösenordet. Senare kan man återställa lösenordet igen med följande kommando:

```
$ passwd -u david
```

Därmed blir **david**s konto återställt till det normala tillståndet. Att tillfälligt stänga ett konto kan vara lämpligt, om användaren inte följer de regler, man har uppställt för åtkomst till systemet, eller om han har exporterat en stor mängd `xeyes` (1) till överanvändarens skrivbord i fönstersystemet X.[5]

12.1.4 Att ändra användarinformation.

Det finns två slags upplysningar om kontot, som en viss användare när som helst själv kan ändra för sitt eget konto: skalet och upplysningarna för `finger`. Slackware Linux använder `chsh` och `chfn` till att ändra dessa värden.

Användarna får välja mellan de skal, som är förtecknade i `/etc/shells`. De flesta klarar sig bra med `bash`. En del andra är kanske redan hemtama i något skal, som finns i unixsystemet på deras skola eller arbetsplats, så att de vill använda det, de redan känner till. Man ömsar skal med kommandot `chsh`:

[5] Programmet `xeyes` visar på skärmen ett par ögon, som följer muspekaren.

```
$ chsh
Password:
Changing the login shell for chris
Enter the new value, or press return for the default
     Login Shell [/bin/bash]:
```

När man har matat in sitt lösenord, skriver man in den fullständiga sökvägen till det nya skalet. Man skall därvid först förvissa sig om, att kommandoskalet ifråga står förtecknat i filen `/etc/shells` (5). Överanvändaren **root** kan även ömsa skal på en vanlig användare genom att köra `chsh` med ett användarnamn som argument.

Information för `finger` är frivilliga upplysningar såsom fullständigt namn, telefonnummer och rumsnummer. Dessa kan ändras med hjälp av `chfn`, och detta följer samma tillvägagångssätt som under skapandet av ett konto. Som vanligt får **root** ändra fingerinformation för vem som helst.

12.2 Stänga av på rätt sätt.

Det är mycket viktigt, att man stänger av systemet på rätt sätt. Att helt enkelt slå av strömmen kan medföra svåra skador på filsystemet. Medan systemet är igång, är filer i bruk, även om man inte gör någonting med systemet. Man skall komma ihåg, att många processer ständigt är igång i bakgrunden. Dessa processer styr systemet och håller många filer öppna. När man slår ifrån strömbrytaren, så blir inte dessa filer stängda på rätt sätt och därför skadade. Beroende på vilka filer, som skadats, så kan systemet ta varaktig skada. I vilket fall som helst, så måste man gå igenom en långrandig filsystemskontroll, nästa gång man startar systemet.

Således, när man skall starta om eller stänga av datorn, är det viktigt att göra detta på rätt sätt. Det finns flera olika sätt att åstadkomma detta; man kan välja det, som verkar trevligast. De flesta sätt att slå av datorn kan även användas till att starta om den.

Det första tillvägagångssättet är bruk av programmet `shutdown` (8), och detta är förmodligen det mest omtyckta. `shutdown` kan användas till att omstarta eller stänga av systemet vid en given tidpunkt och det kan visa ett meddelande för alla systemets användare, vilket talar om för dem, att systemet skall gå ner.

Man använder `shutdown` till att stänga av datorn så här:

```
# shutdown -h now
```

I detta fall sänder vi inget särskilt meddelande till användarna; de får se `shutdown`:s standardmeddelande. »`now`« anger tidpunkten för `shutdown`, och växeln »`-h`« betyder, att systemet skall gå ner till *halt*. Detta är inget trevligt sätt att behandla ett fleranvändarsystem, men det fungerar fint på hemdatorn. Ett bättre sätt på ett fleranvändarsystem vore att ge alla en liten varning i förväg:

```
# shutdown -h +60
```

Detta tar ner systemet om en timme (60 minuter), vilket torde vara tillräckligt i ett normalt fleranvändarsystem. Riktigt viktiga system schemalägger nertiden långt i förväg och anslår meddelanden om detta i `/etc/motd` (5).

Man kan starta om med samma kommando, men då ersätter man »`-h`« med »`-r`«:

```
# shutdown -r now
```

Man kan ange tiden på samma sätt för växeln `-r` som för `-h`. Det finns många andra saker, man kan göra med `shutdown` för att styra nersläckning eller omstart av maskinen. Se manualsidan.

Ett annat sätt att stänga av eller starta om datorn är att använda kommandona `halt` (8) och `reboot` (8). Såsom namnen antyder, så kommer `halt` att omedelbart stanna operativsystemet, medan `reboot` omstartar systemet. `reboot` är helt enkelt en symbolisk länk till `halt`. De anropas på följande vis:

```
# halt
# reboot
```

Ett sätt att starta om eller stänga av systemet på lägre nivå är att tala direkt till `init`. Alla de andra metoderna är helt enkelt bekväma sätt att tala till `init`, men man kan direkt meddela det, vad det skall göra, genom att använda `telinit` (8). Kommandot `telinit` talar om för `init` vilken *runlevel* (körnivå), det skall gå in i, vilket utlöser körning av ett särskilt skript. Detta skript dödar eller avföder processer, som endera behövs eller inte får förekomma vid körnivån ifråga. Detta fungerar till att starta om och stänga av, eftersom bägge är särskilda körnivåer.

```
# telinit 0
```

Runlevel 0 är körnivån för *halt.* Om vi säger till `init` att gå in i körnivå 0, så dödas alla processer, filsystemen avmonteras och maskinen stannas. Detta är ett fullt godtagbart sätt att ta ner systemet. På många bärbara stänger det även av själva maskinen.[6]

```
# telinit 6
```

Runlevel 6 är körnivån för omstart. Alla processer dödas, filsystemen avmonteras och maskinen startas om. Detta är ett fullt godtagbart sätt att starta om systemet.

Det finns ett sista sätt att starta om systemet. Alla andra metoder fordrar, att man är **root**. Emellertid är det möjligt att starta om maskinen, även om man inte är **root**, förutsatt att man kommer åt tangentbordet fysiskt. Genom trefingerhälsningen (`Ctrl` + `Alt` + `Del`) kan man få datorn att startas om direkt. Vad den då egentligen gör är, att den kör ett program, som heter `/usr/sbin/ctrlaltdel` (8). Så om detta program uppvisar underliga rättigheter eller inte alls är förhanden, så hjälper det inte att trycka på alla dessa tangenter.

12.3 Sammanfattning.

Detta kapitel har diskuterat tillvägagångssätt för att lägga till och ta bort användare och grupper. Man bör nu kunna utföra dessa uppgifter med skript eller för hand, om det behovet skulle uppstå. Därtill bör man känna till, vad de olika momenten i tilläggandet av en ny användare går ut på, hur man väljer lösenord och hur man ändrar användarinformation. Slutligen bör man nu veta, hur man skall stänga av datorn på rätt sätt och varför detta är viktigt. Detta är alltsammans viktiga moment i systemadministrationen, oavsett om man administrerar sin egen hemdator eller en stor nätverksserver.

[6] Liksom på maskiner med ATX-nätaggregat, om man har aktiverat APM. Ö.a.

13

Grundläggande nätverkskommandon.

Ett datornät består av flera sammankopplade datorer. Nätet kan vara enkelt, såsom ett fåtal förbundna datorer i hemmet eller på kontoret; eller också kan det vara något så invecklat som ett stort universitetsnät eller hela det världsomspännande Internet. När ens egen dator ingår i ett nät, så har man tillgång till dessa system endera direkt eller igenom olika tjänster såsom epost eller WWW.

Det finns allehanda nätverksprogram, man kan använda. En del är behändiga, när man skall ta pulsen på systemet för att kontrollera, huruvida allting fungerar som det skall. Andra (såsom epostläsare och vävläsare) är nyttiga till att få något arbete uträttat och till att upprätthålla kontakt med andra människor.

13.1 `ping`.

Kommandot `ping` (8) sänder ett paket med en s.k. ICMP ECHO_REQUEST till angiven värddator. Om värddatorn svarar, får man ett ICMP-paket tillbaka. Låter detta märkvärdigt? Nåja, man kan »pinga« en IP-adress, för att kontrollera, om en maskin är vid liv. Får man inget svar, så vet man, att någonting är i olag. Här följer ett exempel på samtal mellan två linuxanvändare:

Användare A: Loki är nere igen.

Användare B: Är du säker på det?

Användare A: Ja, jag har försökt att pinga den, men jag fick inget gensvar.

Det är i dylika fall, som `ping` är ett mycket användbart vardagskommando. Det är ett snabbt medel att ta till för att kontrollera, om någon viss maskin är igång och ansluten till nätet. Grundsyntax är:

```
$ ping <ip-adress eller värddatornamn>
```

Det finns självfallet åtskilliga ytterligare möjligheter, som kan anges med flaggor (växlar) till kommandot. Ytterligare upplysningar om detta kan inhämtas på manualsidan för `ping`(1).

13.2 `finger`.

`finger`(1) hämtar fram upplysningar om angiven användare. Man ger `finger` ett användarnamn eller en epostadress, så försöker det ta kontakt med lämplig server och skaffa fram användarnamn, kontor, telefonnummer och andra upplysningar. Här är ett exempel:

```
$ finger johnc@idsoftwre.com
```

`finger` kan lämna upplysningar om användarnamn, uppgifter om eposten, telefonnummer och några filer, som kallas för »*dot plan*« och »*dot project*«. Självfallet växlar det från fingerserver till annan vilken information, som lämnas ut. Den fingerserver, som medföljer Slackware, returnerar såsom förinställning följande upplysningar:

- Användarnamn
- Rumsnummer
- Hemtelefonnummer
- Telefonnummer på arbetsplatsen
- Huruvida användaren är inloggad
- Huruvida användaren har fått post
- Innehåll i filen `.plan` i användarens hemkatalog
- Innehåll i filen `.project` i användarens hemkatalog

De första fyra posterna kan sättas med kommandot `chfn`. Det lagrar värdena i filen `/etc/passwd`. Upplysningarna i filerna `.plan` och `.project` ändrar man genom redigering i sin älsklingseditor. Dessa filer måste finnas i användarens hemkatalog samt heta `.plan` och `.project`.

Många fingrerar sitt eget konto från en fjärrdator för att snabbt få reda på, om de har fått post. Eller också kan man kika på en viss användares plan eller pågående projekt. John Carmack på *id Software* förnyar t.ex. regelbundet sin planfil för att användarna skall få reda på, vad han arbetar med för tillfället.

Såsom fallet är med många kommandon, så finns många växlar/flaggor till `finger`. Upplysningar om dessa finns på manualsidan.

13.3 telnet.

Någon har en gång yttrat, att `telnet` (1) var det finurligaste, han någonsin sett på en dator. Möjligheten att logga in ifrån en fjärrdator och utföra saker på en annan dator är en vattendelare mellan Unix eller unixliknande operativsystem å ena sidan och övriga operativsystem å den andra.

`telnet` tillåter en att logga in på en dator precis som hade man suttit vid en av dess terminaler. När väl användarnamn och lösenord kontrollerats och godkänts, så får man en skalprompt. Därifrån kan man göra allt, som är möjligt vid en textkonsol. Skriva brev, läsa nysgrupper, flytta omkring filer o.s.v. Om man kör X och telnettar till en maskin ifrån ett xtermfönster, så kan man köra X-program på fjärrdatorn, vilka man visar på sin egen maskin. Se avsnittet om att exportera skärmvisning på s. 78 i kapitel 6 om detta.

Man loggar in på en fjärrdator medelst `telnet` med följande syntax:

```
$ telnet <värddatornamn>
```

13.3.1 Viktigt!

Viktig anmärkning: `telnet` krypterar inte den information, det sänder. Allting sändes såsom ren text, även lösenord. Det är icke tillrådligt att använda `telnet` över Internet. I stället bör man överväga att använda *Secure Shell*. Det krypterar överföringen och är kostnadsfritt tillgängligt. Se därom på sidan `http://www.ssh.org/`.

13.4 FTP-klienter.

FTP står för *File Transfer Protocol*. Det möjliggör överföring av filer emellan två datorer. Det finns en FTP-server och en FTP-klient. I detta avsnitt skall vi diskutera klienten.

»Klienten« är den användare, som startar klientprogramvaran för FTP på sin maskin och gör en begäran om överföring. »Servern« är den dator, som besvarar en sådan FTP-begäran och låter användaren logga in. Man kan såväl ladda upp filer till servern som ladda ner filer därifrån. FTP-klienten kan inte ta emot FTP-anslutningar, den kan endast ansluta till servrar.

13.4.1 `ftp`.

När man skall ansluta till en FTP-server, kör man kommandot `ftp`(1) och anger därvid även värddatorn:

```
$ ftp <värddator>
```

Om värddatorn kör en FTP-server, så kommer den att fråga efter användarnamn och lösenord. Man kan logga in som »anonymous«. Ställen på nätet, som håller med anonym FTP, är t.ex några omtyckta programvaruarkiv. Vill man hämta Slackware Linux genom FTP, använder man anonym FTP.

När man väl är ansluten, får man prompten `ftp>`. Det finns särskilda FTP-kommandon, men de liknar vanliga kommandon. Följande tabell visar några grundläggande kommandon samt vad, de utför.

Kommando	Syfte
`ls`	förteckna filer
`cd <katalognamn>`	växla katalog
`get <filnamn>`	ladda ner (hämta) fil
`put <filnamn>`	ladda upp (skicka) fil
`hash`	nerladdningsvisare (#) av/på
`prompt`	interaktiv nerladdning av/på
`mget <mask>`	ladda ner fil(er) med jokertecken
`mput <mask>`	ladda upp fil(er) med jokertecken
`quit`	logga ut ifrån FTP-servern

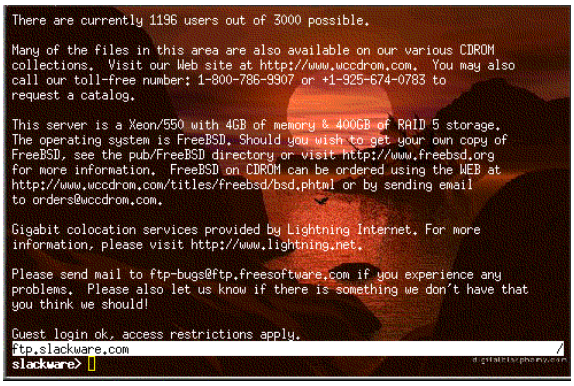
FTP-programmet är rätt lättanvänt men saknar användargränssnitt av det slag, många numera är vana vid. Manualsidan beskriver några av kommandoradsalternativen för `ftp`(1).

13.4.2 `ncftp`.

`ncftp` (uttalas »nick-F-T-P«) är ett alternativ till den vanliga `ftp`-klienten, som medföljer Slackware. Det är fortfarande ett textbaserat program men det erbjuder många fördelar framför `ftp`, däribland följande:

- Tabulatorkomplettering
- Bokmärkesfil
- Passiv och ickepassivt FTP-överföringssätt
- Friare användning av jokertecken
- Kommandohistorik

Såsom förinställning försöker `ncftp` logga in anonymt på angiven server. Man kan tvinga `ncftp` att visa inloggningsprompt genom bruk av växeln `-u`. När man väl har loggat in, kan man använda samma kommandon som i `ftp`, men man lägger märke till det trevligare gränssnittet, som påminner en smula om `bash` i sitt sätt att fungera.[1]



Figur 13.1: Exempel på, hur NcFTP kan ta sig ut.

13.5 Epost.

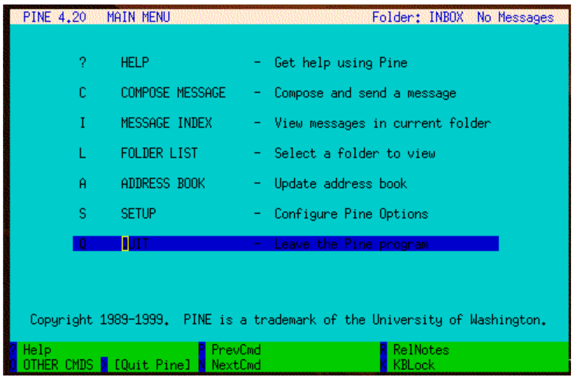
Elektronisk post är en av de mest omtyckta användningarna av Internet. Enligt uppgift skickades år 1998 mer elektronisk post än vanlig post. Eposten är förvisso både vanlig och nyttig.

[1] Om man installerar GNOME, får man även Gnome-FTP (`gftp`), som är ett grafiskt gränssnitt emot FTP. Ö.a.

Med Slackware levererar vi den vanliga epostservern och flera epostklienter. Alla de klienter, som här skildras, är textbaserade. Många windowsanvändare tycker kanhända inte om detta, men man upptäcker snart, att det är mycket bekvämt med en textbaserad epostklient, i synnerhet när man läser sin post ifrån en annan dator.

13.5.1 `pine`.

`pine` (1) är inte `elm`.[2] Så sägs det åtminstone. Washingtons universitet har skapat sitt program för nyhetsgrupper och epost på Internet utifrån ett behov av en lättanvänd epostläsare för sina studenter. `pine` är en av de mest omtyckta epostklienterna numera och finns tillgänglig för nästan varenda unixsystem och t.o.m. för Windows.



Figur 13.2: Huvudmenyn i Pine.

När man startar `pine`, får man längst ner se en kommandomeny i form av en rad tangentkombinationer för kommandon. `pine` är förvisso ett sammansatt program, så vi skall inte skildra alla dess egenskaper här.

Om man slår `i`, får man se, vad som finns i inboxen. Meddelanden förtecknas med datum, författare och ämne. Man sätter markörraden på det meddelande, man vill läsa, och trycker på `Retur`. Trycker man på `r`, får man skriva ett svar på meddelandet. När man har skrivit svaret, slår man `Ctrl` + `X`, så skickas det iväg. Man kan därefter trycka på `i` igen, så kommer man tillbaka till meddelandelistan.

Vill man ta bort ett meddelande, skall man trycka på `d`. Då markeras det meddelande, markörraden står på, för borttagande. `pine` utplånar sådana meddelanden, när man stänger programmet. `pine` medger även lagring av post i mappar. Man får se en lista över mapparna, om man trycker på `l`. I

[2]En tall är ju heller ingen alm.

meddelandelistan kan man trycka på `s`, om man vill spara ett meddelande i en annan mapp. `pine` frågar efter vilken mapp, meddelandet skall sparas i.

`pine` erbjuder åtskilliga finesser; man skall helt bestämt ta en titt på manualsidan. Den innehåller de färskaste upplysningarna om programmet.

13.5.2 `elm`.

`elm`(1) är ytterligare en textbaserad epostklient. Om än den inte är lika användarvänlig som `pine`, så har den åtminstone varit med längre.

Figur 13.3: Elms huvudvy.

Såsom förinställning kommer man in i sin inbox. Där förtecknas meddelanden med nummer, datum, avsändare och ämne. Man kan använda piltangenterna till att markera önskat meddelande. Sedan trycker man på `Retur`, så får man läsa meddelandet.

När man skall skriva ett nytt meddelande, trycker man på `m` i huvudvyn. Tangenten `d` flaggar ett meddelande för radering (*deletion*). Samtliga dessa tangenter visas längst ner på skärmen tillsammans med en prompt.

Manualsidan går närmare in på `elm`, så man bör rådfråga denna, innan man börjar använda `elm`.

13.5.3 `mailx`.

`mailx`(1) är en kommandoradsbaserad epostklient. Den är synnerligen primitiv och erbjuder nästan ingenting, som liknar ett användargränssnitt. Emellertid är `mailx` behändig vid sådana tillfällen, då man i en hast vill skicka ett epostmeddelande, när man vill använda skript för att göra massutskick o.s.v.

Det grundläggande kommandot är:

```
$ mailx -s <ämne> <adressatens adress>
```

Här följer ett exempel på, hur man kan sända en fil med C-källkod till någon annan:

```
$ cat randfunc.c | mailx -s "Här är funktionen" asdf@exempel.net
```

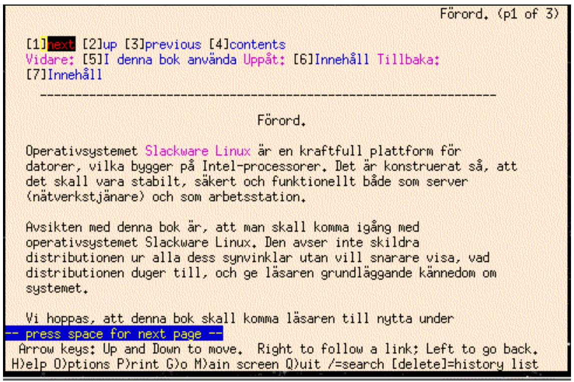
Manualsidan förklarar mer om vad, `mailx` kan göra, så man vill förmodligen läsa den, innan man använder `mailx`.

13.6 lynx.

`lynx` är en textbaserad WWW-läsare. Den erbjuder ett snabbt sätt att slå upp någonting på Internet. Ibland är grafiken bara i vägen, om man redan vet, vad man är ute efter.

Man startar `lynx` genom att skriva `lynx` på kommandoraden:

```
$ lynx
```



Figur 13.4: Lynx med förordet till denna bok.

Man kan vid start välja vilken sida, `lynx` skall visa:

```
$ lynx http://www.slackware.com
```

`lynx` visar några kommandotangenter och deras tillordningar längst ner på skärmen. Med pilknapp upp och pilknapp ner kan man röra sig omkring i ett dokument, tangenten `Retur` följer markerad länk och vänsterpil leder tillbaka till föregående sida. Tangenten `g` tar fram *Go*-prompten, där man kan ange en URL till en sida, som man vill gå till.

Det finns många andra kommandon i `lynx`. Man kan endera rådfråga manualsidan eller i `lynx` slå an tangenten `h` för att få se den inbyggda hjälpen.

13.7 wget.

`wget` (1) är ett kommandoradstillbehör, som laddar ner filer från angiven URL. Det kan användas, när man vill ladda ner ett helt WWW-ställe för att läsa det i nerkopplat läge, eller om man behöver en säkrare nerladdning av filer ifrån HTTP- eller FTP-servrar, än man kan få genom t.ex. Netscape. Kommandots grundläggande syntax är:

```
$ wget <url>
```

Man kan även ange växlar. T.ex. kommer följande kommando att ladda ner hela Slackwares vävställe med alla filer och kataloger:

```
$ wget --recursive http://www.slackware.com
```

`wget` skapar då en katalog vid namn `www.slackware.com` och lagrar filerna där med samma katalogstruktur som på det WWW-ställe, de hämtas ifrån.

`wget` kan även ladda ner filer ifrån FTP-ställen; man anger då en FTP-URL i stället för en HTTP-URL.

Kommandot `wget` har många fler växlar, vilka gör det trevligt att använda i skript (t.ex. för spegling av WWW-ställen o.s.v.). Manualsidan torde rådfrågas, om man behöver ytterligare upplysningar.

13.8 traceroute.

I Slackware ingår kommandot `traceroute` ifrån 4.4BSD. Det är ett användbart diagnosredskap för nätverket. `traceroute` visar var värddator, som ett paket passerar på väg till sitt mål. Man kan med följande kommando se, hur många »hopp« man befinner sig ifrån Slackwares WWW-ställe:

```
$ traceroute www.slackware.com
```

Var värddator kommer att visas tillsammans med svarstid. Ett exempel:

```
$ traceroute www.slackware.com
traceroute to www.slackware.com (204.216.27.13), 30 hops max, 40 byte packets
1  zuul.tdn (192.168.1.1)  0.409 ms  1.032 ms  0.303 ms
2  207.171.227.254 (207.171.227.254)  18.218 ms  32.873 ms  32.433 ms
3  border-sf-2-0-4.sirius.com (205.134.230.254) 15.662 ms 15.731 ms 16.142 ms
4  pb-nap.crl.net (198.32.128.20)  20.741 ms  23.672 ms  21.378 ms
5  E0-CRL-SFO-03-E0X0.US.CRL.NET (165.113.55.3) 22.293 ms 21.532 ms 21.29 ms
6  T1-CDROM-00-EX.US.CRL.NET (165.113.118.2)  24.544 ms  42.955 ms 58.443 ms
7  www.slackware.com (204.216.27.13)  38.115 ms  53.033 ms  48.328 ms
```

`traceroute` påminner om `ping` såtillvida, att det begagnar ICMP-paket. Det finns ett flertal växlar, som man kan ange tillsammans med `traceroute`. Det förinställda högsta antalet »hopp« är 30, men man kan ändra detta med flaggan »`-m`«. Andra växlar förklaras mer ingående på manualsidan.

13.9 Att tala med andra.

13.9.1 talk

`talk` tillåter två användare att tjatta (*chat*) med varandra. Det delar skärmen i två halvor med en vågrät linje. Med följande kommando begär man att få tjatta med en annan användare:

```
$ talk <användarnamn> [ttynamn]
```

Figur 13.5: Två användare tjattar genom talk.

Om man anger ett användarnamn enbart, förmodas tjattbegäran vara lokal, så att endast lokala användare efterfrågas. Man behöver ange `ttynamn`, om man vill kontakta en viss användare på en särskild terminal (ifall vederbörande är inloggad på mer än en). De upplysningar, man kan behöva om `talk`, finns på manualsidan för `w` (1).

`talk` kan även ringa upp användare på fjärrdatorer. Som användarnamn anger man då epostadressen. `talk` försöker då kontakta användaren med angiven adress på fjärrdatorn.

`talk` är en smula begränsat. Det betjänar endast två användare och i halvduplex.

13.9.2 `ytalk`.

`ytalk` (1) är en bakåtkompatibel ersättning för `talk`. Det medföljer Slackware såsom kommandot `ytalk`. Syntaxen är likartad men uppvisar några skillnader:

```
$ ytalk <användarnamn>[#ttynamn]
```

Figur 13.6: Tre användare tjattar medelst ytalk.

Användarnamnet och terminalen anges på samma sätt som med `talk` förutom det, att man måste skriva ihop dem med brädgårdstecknet (#).

`ytalk` erbjuder flera fördelar:

- Det kan betjäna fler än två användare.
- En meny med valmöjligheter kan visas när som helst, när man trycker på `Esc`.
- Man kan tillfälligt gå ut i skalet under körning av `ytalk`.
- Plus mer...

Om man administrerar en server, vill man förmodligen säkerställa, att `ntalk`-porten är satt till verksam i `/etc/inetd.con`. `ytalk` behöver detta för att fungera ordentligt.

13.10 Sammanfattning.

Man bör nu känna till grundläggande nätverksdiagnoskommandon. Med deras hjälp kan man avgöra, huruvida det föreligger något problem med en dator eller nätverket mellan det egna systemet och någon fjärrdator. Vidare bör man nu känna till en del om flera olika epostläsare, WWW-läsare, FTP-klienter och kommunikationsprogram.

14

Att arkivera filer.

I Slackware Linux finns flera program, som kan användas till att komprimera och arkivera filer. Dessa program kommer särskilt väl till pass, när man vill göra säkerhetskopieringar eller sända filer emellan maskiner i ett nätverk. Där ingår program för hantering av unixarkiv liksom för hantering av windowsarkiv.

14.1 gzip.

`gzip` (2) är GNU:s filkomprimeringsprogram. Det komprimerar enstaka filer var för sig. Kommandot användes på följande sätt:

```
$ gzip <infil>
```

Den resulterande filen kommer att heta `infil.gz` och är vanligen mindre än `infil`. Lägg märke till, att `infil.gz` ersätter `infil`. Detta betyder, att `infil` inte längre finns kvar, om än det finns en gzippad kopia av denna fil. Vanliga textfiler komprimeras väl, medan jpeg-bilder, mp3-musikfiler och andra liknande filer inte går så bra att komprimera, eftersom de redan är komprimerade. Det grundläggande användningssättet går balansgång emellan kompressionsgrad och komprimeringstid. Största möjliga kompression ernås med följande kommando:

```
$ gzip -9 <infil>
```

Nu tar det längre tid att komprimera filen, men den resulterande filen blir så liten, som `gzip` över huvud taget förmår göra den. Lägre värden på flaggan ger snabbare komprimering, men filen förminskas inte lika mycket.

Expanderandet av gzippade filer kan genomföras med endera av två kommandon, vilka egentligen är ett och samma program. `gzip` expanderar vilken som helst fil med igenkänd filnamnsändelse för komprimerade filer. En sådan filnamnsändelse är en av följande: `.gz`, `-gz`, `.z`, `-z`, `.Z` eller `-Z`. I första hand använder man kommandot `gunzip` (1) till att expandera en fil:

```
$ gunzip infil.gz
```

Detta ger en expanderad version av `infil` i aktuell katalog, och filnamnsändelsen `.gz` är avlägsnad ifrån filnamnet.

Ett annat sätt att expandera en gzippad fil är att använda kommandot `gzip` med flaggan `-d` på filen:

```
$ gzip -d infil.gz
```

Detta får samma verkan som föregående bruk av `gunzip`. Anledningen därtill är enkel: `gunzip` är bara en symbolisk länk till `/bin/gunzip`:

```
$ cd /usr/bin
$ ls -l gunzip
lrwxrwxrwx   1 root  root  9 Feb  2 09:45 gunzip -> /bin/gzip
```

Att köra `gunzip` innebär således i själva verket endast att köra `gzip` under ett annat namn. Programmet kan självt avgöra, hur det har anropats, och vidtaga motsvarande, lämpliga åtgärder. I detta fall vet `gzip`, att man har kallat på det såsom `gunzip` och kommer följaktligen att expandera filen. Således kan man använda `gzip` eller `gunzip` till att expandera vilken som helst gzippad fil.

14.2 `bzip2`.

`bzip2` är ett alternativt komprimeringsprogram, som medföljer Slackware Linux. Det använder en annan komprimeringsalgoritm än `gzip`, vilken ger några fördelar och några nackdelar. Den främsta fördelen för `bzip2` är filstorleken. `bzip2` komprimerar nästan alltid bättre än `gzip` I en del fall kan detta leda till dramatiskt förminskade filer. Detta kan vara till stor fördel för folk med långsamma modemförbindelser.

Nackdelen med `bzip2` är den, att det i allmänhet är mer CPU-intensivt än `gzip`. Detta betyder, att det tar längre tid att bzippa en fil och att detta belastar CPU mer, än om man skulle gzippa samma fil. När man överväger vilket komprimeringsprogram, man skall använda, så måste man väga snabbhet emot kompression och avgöra vilketdera, som är viktigast.

`bzip2` brukas på liknande sätt som `gzip`, så här skall inte spillas mycket tid på att diskutera detta. Man bzippar en fil på följande vis:

```
$ bzip2 infil
```

Den resulterande filen blir vanligen mindre än infilen och kallas `infil.bz2`. Liksom i fallet `gzip`, så kommer infilen inte längre att finnas till, eftersom `bzip2` ersätter infilen med en komprimerad kopia.

Man kan även använda en siffra till växel för att ange kompressionsgraden, liksom med `gzip`. Följande exempel visar, hur man skall uppnå maximal kompression genom `bzip2` och avsevärd CPU-insats:

```
$ bzip2 -9 infil
```

Det finns två kommandon till att expandera filer med `.bz2`-ändelsen, liksom i fallet med `gzip`; man kan använda `bzip2` eller `bunzip2`(1) till att expandera bzippade filer. När man använder `bzip2`, kräver detta en växel:

```
$ bzip2 -d infil.bz2
```

Detta expanderar den bzippade filen och ersätter den med en expanderad kopia. Denna resulterande fil saknar även ändelsen `.bz2`. På liknande sätt kan man använda `bunzip2` till att expandera filen:

```
$ bunzip2 infil.bz2
```

Man får samma uppförande i bägge fallen, tack vare en symbolisk länk. Om vi kontrollerar `/bin/bunzip2`, så visar det sig vara en symlänk till `/bin/bzip2`. Det utnyttjar samma knep, som `gzip`. Man skall upptäcka, att detta är ett älsklingsknep för linuxprogrammerare, att man kan kalla på ett och samma program men med olika namn för att få olika uppförande.

```
$ cd /bin
$ ls -l bunzip2
lrwxrwxrwx   1 root  root      5 Feb  2 09:45 /bunzip2 -> bzip2
```

14.3 tar.

`tar`(1) är *GNU tape archiver*. Den tar flera filer eller kataloger och skapar av dem en enda stor fil. Detta gör det möjligt att komprimera ett helt katalogträd, vilket är omöjligt med bruk av `gzip` eller `bzip2` enbart. `tar` har många växlar, vilka förklaras på dess manualsida. Detta avsnitt skall endast behandla några av de vanligaste användningarna.

Det vanligaste sättet att använda `tar` är att expandera och packa upp ett paket, som man har laddat ner ifrån nätet. De flesta paketfiler har ändelsen

`.tar.gz`. De är allmänt kända såsom »*tarballs*«. Detta betyder, att flera filer har arkiverats med `tar` till en fil, som sedan komprimerats med `gzip`. Man kan även få se ändelsen `.tar.Z`. Det betyder samma sak, men denna ändelse ser man vanligen på äldre unixsystem.

Eller också får man se en fil med ändelsen `.tar.bz2` någonstans. Källkoden till linuxkärnan distribueras på så vis, eftersom det blir mindre filer att ladda ner. Man gissar sig till, att detta är flera filer, som arkiverats med `tar` och därefter komprimerats med `bzip2`.

Man kan få ut alla filer ur ett sådant arkiv genom att använda `tar` och några växlar. Om man skall packa upp en komprimerad tarboll, behöver man använda `-z`-flaggan, som anger, att filen först skall expanderas med `gunzip`. Det vanligaste sättet att packa upp en sådan »tjärboll« är detta:

```
$ tar -xvzf hejaz.tar.gz
```

Det var en hel del växlar. Vad betyder de? Flaggan `-x` betyder *extrahera*. Detta är viktigt, ty det talar om för `tar` precis vilken åtgärd, det skall vidtaga med den inmatade filen. I detta fall skall vi dela upp den i alla de filer, den skapats ur. Flaggan `-v` innebär, att det skall ske *verbose* (ordrikt). Detta gör, att en lista över filerna rullar upp efter hand som de packas upp. Man kan mycket väl låta bli detta val, om än det vore en smula tråkigt. Eller också skulle man kunna välja `-vv` för att `tar` skall vara *very verbose* och skriva ut ännu mer information om varenda uppackad fil. Flaggan `-z` anger, att filen före uppackning skall expanderas med `gzip`. Slutligen anger växeln `-f`, att nästkommande textsträng är namnet på den fil, som skall behandlas.

Det finns flera olika sätt att skriva detta kommando. På äldre system utan GNU `tar` kan man få se det skrivet så här:

```
$ gzip -dc hejaz.tar.gz | tar -xvf -
```

Detta zippar upp filen och skicka utmatningen till tar. Eftersom `gzip` skriver sin utmatning till *standard out* om man befaller det, så skriver detta kommando den expanderade filen till *standard out*. Rörledningen sänder det vidare till `tar` att packas upp. Bindestrecket till sist betyder, att `tar` skall arbeta med *standard input*. Det packar upp den dataström, som kommer ifrån `gzip`, och skriver utmatningen till disk.

Ett annat sätt att skriva den första kommandoraden är att utelämna bindestrecket före flaggorna, så här:

```
$ tar xvzf hejaz.tar.gz
```

Man kan också stöta på ett bzippat arkiv. Den version av `tar`, som medföljer Slackware Linux, kan hantera dem på samma sätt som gzippade arkiv. I stället för `-z` skall man då använda `-y`:

```
$ tar -xvyf foo.tar.bz2
```

Det är viktigt att lägga märke till, att `tar` packar upp filer i förhandenvarande katalog. Om vi skulle ha ett arkiv i `/tmp`, som vi ville expandera i hemkatalogen, så finns två möjligheter. För det första skulle man kunna flytta filen till hemkatalogen och sedan köra den igenom `tar`. Eller också skulle man kunna ange sökvägen till arkivfilen på kommandoraden:

```
$ tar -xvzf /tmp/bar.tar.gz
```

Om då hemkatalogen är aktuell katalog, så packas arkivet upp i hemkatalogen, medan den ursprungliga komprimerade filen finns kvar i `/tmp`.

På andra plats bland vanligt bruk av `tar` kommer skapandet av arkivfiler; det behövs bara några andra växlar till det.

För att skapa ett komprimerat `tar`-arkiv av alla filer i aktuell katalog (inberäknat underkataloger med filer), så skulle man använda `tar` som så:

```
$ tar -cvzf arkiv.tar.gz .
```

På denna kommandorad användes flaggan `-c` som direktiv till `tar` att skapa (*create*) ett arkiv, medan `-z` får arkivet att komprimeras med `gzip`. `archive.tar.gz` är den fil, man vill skapa. Man kan kalla den vad man vill och om man anger en fullständig sökväg, så hamnar arkivet i angiven katalog. Här är ett exempel på det:

```
$ tar -cvzf /tmp/arkiv.tar.gz .
```

Arkivet hamnar då i `/tmp`. Man kan även räkna upp alla filer och kataloger, som skall ingå i arkivet, direkt efter kommandot. I detta fall använder vi punkten `.` till att beteckna aktuell katalog, som skall arkiveras. Denna punkt skulle kunna ersättas med en lista över olika filer, eller vad man nu vill arkivera.

14.4 `zip`.

Slutligen finns två tillbehör, som kan användas på zipfiler. Sådana är mycket vanliga i windowsvärlden, så att linuxprogram måste kunna hantera dem. Komprimeringsprogrammet kallas `zip`(1), och motsvarande expanderingsprogram kallas `unzip`(1).

Att komprimera en fil är lätt:

```
$ zip foo *
```

Detta skapar filen `foo.zip`, som innehåller alla filer i aktuell katalog. `zip` lägger av sig själv till ändelsen `.zip`, så man behöver inte ange det i filnamnet. Man kan även göra en rekursiv arkivering genom katalogen och dess underkataloger:

```
$ zip -r foo *
```

Det är lika lätt att packa upp och expandera:

```
$ unzip foo
```

Detta packar upp och expanderar filerna i arkivfilen `foo.zip`, inberäknat alla kataloger i arkivet.

Ziptillbehören har flera avancerade växlar för att skapa självuppackande arkiv, utelämna filer, styra filstorleken, skriva ut vad som händer och mycket mer. Se manualsidorna för `zip` och `unzip`.

14.5 Sammanfattning.

Detta kapitel har behandlat de program, som användes till att komprimera och expandera arkivfiler. Man bör nu veta, vad en arkivfil är, hur man skapar en sådan med `tar` och hur man väljer komprimeringsprogram; hur man expanderar ett arkiv och hur man hanterar windowsarkiv. Vid nästan alla upp- och nerladdningar kommer man i kontakt med arkiv, så att detta är viktiga kunskaper och färdigheter.

15

vi.

vi (1) är den gängse textredigeraren i Unix, och det är väsentligt för systemadministratörer att bemästra den. Flera versioner (eller efterlikningar) av vi finns tillgängliga, däribland vi, elvis, vile och vim. Åtminstone någon utav dessa är tillgänglig i så gott som vilken som helst unixversion liksom i Linux. Samtliga dessa versioner innehåller samma grunduppsättning av egenskaper och kommandon, så att det bör vara lätt att lära sig en ny avart av vi, när man redan kan en av dem.

vi är försedd med en mängd kraftfulla egenskaper, däribland syntaxkänslig markering av nyckelord, programkodsformatering, kraftfull sökning och ersättning, makron m.m. Dessa egenskaper tilltalar särskilt programmerare, dem som utvecklar för WWW m.fl. Systemadministratörer brukar uppskatta den automatisering och integrering med kommandoskalet, som är möjlig.

I Slackware Linux är det elvis, som är standardversion av vi. Även andra versioner — däribland vim och gvim — finns tillgängliga, om man har installerat motsvarande paket. gvim är en vi-version för fönstersystemet X. Den är försedd med verktygslister, löstagbara menyer och dialogrutor.

15.1 Starta vi.

vi kan startas från kommandoraden på flera olika sätt. Det enklaste är:

```
$ vi
```

Detta startar vi och öppnar en tom buffert. Först ser man en nästan tom skärm. Editorn är nu i »kommandoläge« (»command mode«) och väntar på, att man skall göra någonting med den. Arbetslägen i vi behandlas på s. 160 i avsnittet vid namn *Arbetslägen*. När man skall lämna vi skriver man

```
:q
```

Förutsatt att inga ändringar har gjorts i filen, så kommer `vi` att stänga den. Om ändringar har gjorts, så kommer `vi` att varna för detta och tala om, hur man skall bära sig åt för att stänga utan att spara.

Man kan även starta `vi` och samtidigt öppna en befintlig fil. Vi kan t.ex. öppna filen `/etc/resolv.conf` på följande vis:

```
$ vi /etc/resolv.conf
```

Slutligen kan `vi` startas på en bestämd rad i en fil. Till exempel kan man starta `vi` och öppna rad 47 i filen `/usr/src/linux/init/main.c` på följande vis:

```
$ vi +47 /usr/src/linux/init/main.c
```

`vi` visar begärd fil och ställer inskrivningspunkten på angiven rad. Ifall man anger ett alltför högt radnummer, som skulle befinna sig någonstans efter filens slut, så ställer `vi` markören någonstans på sista raden. Detta är särskilt behändigt för programmerare, eftersom de kan hoppa direkt till det ställe i filen, där ett fel har tillstött, utan att behöva söka efter detta.

15.2 Arbetslägen.

`vi` arbetar i olika lägen, som användes till att lösa skiftande uppgifter. När man först startar `vi`, kommer man in i kommandoläge. Där har man tillgång till olika kommandon för att hantera text, flytta omkring i filen, spara, lämna och växla arbetsläge. Redigering av text sker i inmatningsläge. Man kan snabbt växla mellan lägen medelst ett antal tangentkombinationer, vilka förklaras nedan.

15.2.1 Kommandoläge.

Ifrån början kommer man in i kommandoläge. Från detta läge kan man inte direkt mata in text eller redigera befintligt innehåll. Dock kan man hantera texten, söka, lämna programmet, spara, ladda nya filer m.m. Detta avsnitt avses endast vara en introduktion till kommandoläget. Se beskrivningar av tangentkommandon i avsnittet om `vi`-tangenter på s. 165.

Kommandoläge är således det, man kan använda för att röra sig omkring i filen. På en del system kan man använda piltangenterna. På andra system kan man behöva använda de traditionella tangenterna »`hjkl`«. Här är en enkel förteckning över dessa tangenter, vilka brukas till att förflytta sig omkring i filen:

Tangent	**Förflyttning**
`h`	vänster ett tecken
`j`	ner ett tecken
`k`	upp ett tecken
`l`	höger ett tecken

Man trycker på en av de uppräknade tangenterna enbart. Såsom vi senare skall se, så kan dessa tangenter kombineras med siffror för att ge effektivare förflyttningar.

Många av de kommandon, man utnyttjar i kommandoläge, börjar med kolon. Man avslutar t.ex. programmet med `:q`, såsom tidigare nämnts. Detta kolon anger endast, att det är fråga om ett kommando, medan det är »`q`«, som befaller `vi` att stänga. Andra kommandon kan vara en siffra följd av en bokstav. Dessa kommandon föregås inte av kolon och brukas vanligen till att hantera text.

Exempelvis kan man ta bort en rad ifrån en fil genom att slå `dd`. Detta raderar hela den rad, markören står på. Kommandot `4dd` skulle befalla `vi` att avlägsna den rad, markören står på, samt ytterligare tre rader efter. I allmänhet anger siffror hur många gånger, `vi` skall utföra kommandot.

Man kan kombinera en siffra med någon av rörelsetangenterna för att förflytta markören flera tecken åt gången. Till exempel skulle `10k` förflytta markören tio rader uppåt.

Kommandoläge kan även brukas till att klippa och klistra, infoga text samt läsa in andra filer i förhandenvarande buffert. Kopiering av text åstadkommes medelst tangenten `y` (för *yank*). Man kopierar aktuell rad med `yy`, och detta kommando kan förses med en siffra före, så att det då gäller flera rader. Därefter flyttar man dit, där kopian skall infogas, och slår `p` för *paste*. Texten klistras då in på raden efter den, markören står på.

Urklipp sker genom bruk av `dd`, och `p` kan även användas till att klistra tillbaka urklippt text. Inläsning av text ifrån en annan fil är en enkel procedur. Man slår `:r` åtföljt av mellanslag och namnet på den fil, som skall infogas. Filens innehåll klistras in i aktuell buffert på raden efter den, markören står på. Mer förfinade `vi`-kopior kan till och med komplettera filnamn på liknande sätt som skalet.

Slutligen skall vi behandla sökning. Kommandoläget medger enkel sökning såväl som komplicerade sök-och-ersätt-kommandon, vilka utnyttjar en kraftfull version av regelbundna uttryck. En uttömmande diskussion om regelbundna uttryck (reguljära mönster) ligger bortom synranden för detta kapitel, så detta avsnitt skall endast behandla några enkla sätt att söka.

En enkel sökning genomföres, när man slår an tangenten `/` åtföljd av den text, som sökes. `vi` söker framåt ifrån markören till filens slut och stannar, när det hittar en träff. Lägg märke till, att även icke fullt överensstämmande träffar får `vi` att stanna upp. Till exempel kan sökning efter »de« få `vi` att stanna på »dem«, »detta« o.s.v. Detta beror på, att alla dessa ord innehåller »de«, om än bara i början.

Efter att `vi` hittat första träffen, kan man fortsätta med att söka efter nästa träff genom att slå an tangenten `/` åtföljd av `Retur`. Man kan även söka baklänges igenom filen, om man ersätter snedstrecket med frågetecknet `?`. Sökningen bakåt efter »de« skulle t.ex. kunna åstadkommas med `?de`.

15.2.2 Inmatningsläge.

Infogande och ersättande av text sker i inmatningsläge. Såsom vi tidigare sagt, så kan man välja inmatningsläge genom att slå an tangenten `i`, när man befinner sig i kommandoläge. Därefter matas allt, man slår på tangenterna, in såsom text i aktuell buffert. När man slår an tangenten `Escape`, kommer man tillbaka till kommandoläge.

Ersättning av text kan åstadkommas på flera sätt. I kommandoläget slår man `r` och kan sedan ersätta ett tecken under markören. Man slår in det nya tecknet, så ersätter det tecknet under markören. Därefter blir man omgående återförpassad till kommandoläge. Om man slår an `R`, får man därefter ersätta så många tecken, man önskar. Först när man slår an tangenten `Escape`, kommer man ut ur detta läge och tillbaka till kommandoläge.

Det finns ytterligare ett sätt att växla mellan infogning och ersättning. Om man slår an tangenten `insert` i kommandoläge, så kommer man till inmatningsläge. När man väl är i inmatningsläge, fungerar tangenten `insert` såsom vippomkopplare mellan infogning och ersättning. Trycker man en gång, får man därefter ersätta text. Trycker man en gång till, får man åter infoga text.

15.3 Öppna filer.

`vi` låter oss öppna filer från kommandoläge såväl som att ange en fil, som skall öppnas, på kommandoraden. Vi kan t.ex. öppna `/etc/lilo.conf`:

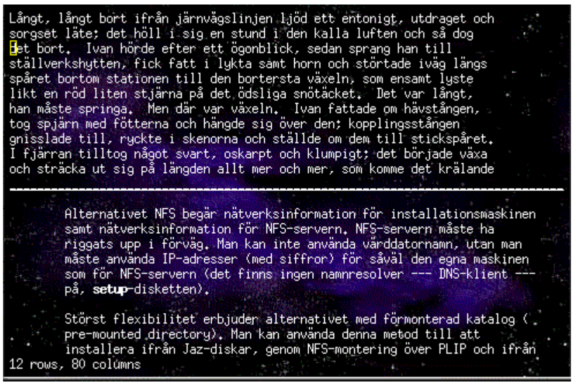
```
:e /etc/lilo.conf
```

Om man har gjort ändringar i aktuell buffert utan att spara, så klagar `vi`. Man får likväl öppna filen utan att spara aktuell buffert, om man slår `:e!` åtföljt av mellanslag och filnamnet. I allmänhet kan `vi`:s varningar undertryckas, genom att kommandot åtföljes av utropstecken.

Om man vill öppna aktuell fil på nytt, kan man göra det genom att slå `e!`. Detta är användbart, när man på något vis har rört till innehållet i filen och vill öppna den på nytt.

En del `vi`-kopior (t.ex. `vim`) medger flera öppna buffertar på en gång. Till exempel skulle vi kunna öppna filen `09-vi.sgml` i vår hemkatalog, medan en annan fil är öppen, med kommandot

```
:split ~/09-vi.sgml
```



Figur 15.1: Visning av två filer samtidigt i `vi`.

Den nya filen visas på skärmens övre halva, medan den gamla filen visas på den nedre. Det finns en mängd kommandon, som hanterar delad skärm, och flera av dessa kommandon börjar likna någonting i EMACS. Det bästa stället att leta efter sådana kommandon är manualsidan för respektive `vi`-kopia. Lägg märke till det, att många kopior saknar stöd för delad skärm, så att man kanske inte alls kan använda det.

15.4 Spara filer.

Det finns många sätt att spara filer i `vi`. Om man vill spara aktuell buffert till filen `randomness`, så slår man

```
:w randomness
```

När man väl har sparat en fil en gång, så behöver man bara slå `:w`. Alla ändringar skrives då till filen. När man har sparat filen, kommer man tillbaka till kommandoläge. Om man vill spara filen och lämna `vi`, (en mycket vanlig åtgärd), så slår man `wq`. Det befaller `vi` att spara filen och återvända till skalet.

Ibland händer det, att man vill spara en fil, som är märkt *read-only*. Det kan man göra, om man lägger till ett utropstecken efter kommandot, som så:

```
:w!
:wq!
```

Emellertid inträffar det likväl, att man inte kan spara en sådan fil (t.ex., om man försöker redigera en fil, som någon annan äger). När detta händer, säger `vi`, att det inte kan spara filen. Om man då verkligen vill redigera filen, får man komma tillbaka och redigera den som **`root`**.

15.5 Lämna `vi`.

Ett sätt att lämna `vi` är genom kommandot `:wq`, vilket sparar den öppna bufferten, innan programmet stänges. Man kan även stänga utan att spara, om man kommenderar `:q` eller `:q!`. Det senare kommandot användes, när man har gjort ändringar i filen, inte vill spara ändringarna.

Ibland kan det hända, att maskinen störtar eller att `vi` störtar. Emellertid vidtar då såväl `elvis` som `vim` mått och steg för att i möjligaste mån förminska skadan på öppna buffertar. Bägge dessa textredigerare sparar öppna buffertar i en tillfällig fil då och då. Denna tillfälliga fil heter oftast något, som liknar namnet på den öppna filen, men är försedd med en punkt i början av namnet. Detta gör filen till en dold fil.

Denna temporära fil avlägsnas så ofta editorn stänges på normalt vis. Detta betyder, att den tillfälliga kopian finns kvar, om någonting störtar, men inte annars. När man sedan vill redigera filen igen, uppmanas man välja mellan att använda säkerhetskopian eller bortse från den. `elvis` sänder även ett epostmeddelande (ifrån Graceland, märkvärdigt nog :), vilket talar om, att det finns en säkerhetskopia.

15.6 Ställa in `vi`.

Den `vi`-kopia, man helst använder, kan ställas in på många sätt.

En mångfald av kommandon kan matas in, medan man kör `vi` i kommandoläge, för att konfigurera `vi` så, som man vill ha det. Beroende på vilken editor, man använder, så kan man aktivera egenskaper, som underlättar programmering (såsom syntaxkänslig färgmärkning av nyckelord, automatiska indrag m.m.), rigga upp makron för att automatisera uppgifter, möjliggöra ersättning av fraser i texten m.m.

Nästan samtliga dessa kommandon kan skrivas in i en konfigureringsfil i användarens hemkatalog. `elvis` förväntar sig en sådan fil vid namn `.exrc`, medan `vim` läser sina inställningar ur `.vimrc`. Flertalet inställningskommandon, som kan matas in i kommandoläget, kan även placeras i inställningsfilen. Detta inbegriper inställningar, ersättningsfraser, makron m.m.

Dessa valmöjligheter och skillnaderna mellan textredigerarna utgör ett omfattande tema. För att få ytterligare upplysningar om dem, bör man läsa manualsidan eller WWW-stället för den avart av `vi`, man föredrar. En del editorer (såsom `vim`) har omfattande inbyggd hjälp, som man kan nå med kommandot `:help` eller liknande. Man kan även läsa i förlaget O'Reillys bok *Learning the vi Editor* av Lamb och Robbins.

Många vanliga program i Linux brukar vara förinställda till att ladda textfiler i `vi`. Exempelvis startas `vi` såsom förval, när man redigerar sina `crontab`-filer (se avsnittet om `cron`). Om man inte tycker om `vi` och hellre vill använda en annan textredigerare, så skall man sätta miljövariabeln `VISUAL` till namnet på den editor, man föredrar. Upplysningar om, hur man kan sätta miljövariabler, står i avsnittet om miljövariabler på s. 101 i kapitel 8. Om man vill förvissa sig om, att älsklingseditorn blir förval var gång, man loggar in, så skall man lägga till inställningen för variabeln `VISUAL` i sin `.bash_profile` eller `.bashrc` i hemkatalogen.

15.7 Tangentkombinationer i `vi`.

Åtgärd	**Tangent**	**Åtgärd**	**Tangent**
vänster, ner, upp, höger	`h`, `j`, `k`, `l`	utplåna rad	`dd`
till radens slut	`$`	utplåna fem rader	`5dd`
till radens början	`^`	ersätta tecken	`r`
till filens slut	`G`	utplåna tecken	`x`
till filens början	`:1`	utplåna tio tecken	`10x`
till rad 47	`:47`	ångra senaste åtgärd	`u`
läsa in utmatning ifrån ls	`:r !ls`	förena rad med nästa	`J`
söka efter »asdf«	`/asdf`	upprepa sökning framåt	`/`
söka bakåt efter »asdf«	`?asdf`	upprepa sökning bakåt	`?`
lämna	`:q`	lämna utan att spara	`:q!`
spara och lämna	`:wq`	spara utan att lämna	`:w`
ladda om öppen fil	`:e!`	öppna fil `hejaz`	`:e hejaz`
spara buffert till fil `asdf`	`:w asdf`	läsa in fil `asdf`	`:r asdf`

15.8 Sammanfattning.

Man bör nu ha bekantat sig med `vi` — standardeditorn i Unix. `vi` är ett tämligen sammansatt program med mängder av kommandon och inställningsmöjligheter. Man bör dock kunna öppna en fil, förflytta sig omkring i en fil, redigera en fil och lämna `vi`. Mer än så behöver man inte lära sig för vardagsuppgifter. Efter hand som man känner behov av mer makt över detta redskap, så kan man ta till `vi`:s väl utbyggda hjälp för att lära sig mer om programmet.

16

Pakethantering i Slackware.

Ett programvarupaket är en bunt program, som hör nära ihop med varandra och är färdiga att installeras och köras. När man laddar ner ett källkodsarkiv, så måste man konfigurera, kompilera och installera för hand. I ett programvarupaket är detta redan gjort. Det återstår endast att installera paketet. En annan behändig sida av att installera färdiga programpaket är, att de är mycket lätta att ta bort eller uppgradera, om man så skulle önska. Slackware innehåller program för alla pakethanteringsbehov. Man kan mycket lätt installera, ta bort, uppgradera (förnya), skapa och undersöka programpaket.

16.1 Översikt över paketformat.

Innan man lär sig använda tillbehören, bör man bekanta sig med programpaketens format i Slackware. Ett slackwarepaket är helt enkelt en arkivfil, som skapats med arkiveringsprogrammet `tar` och komprimerats med `gzip`. Ett sådant paket är uppbyggt på så vis, att det skall kunna packas upp i rotfilsystemet.

Här är ett påhittat, tänkt program och ett exempel på paketering:

```
./
usr/
usr/bin/
usr/bin/makehejaz
usr/doc/
usr/doc/makehejaz-1.0/
usr/doc/makehejaz-1.0/COPYING
usr/doc/makehejaz-1.0/README
usr/man/
usr/man/man1
usr/man/man1/makehejaz.1.gz
install/
install/
install/doinst.sh
```

Pakethanteringssystemet kommer att packa upp denna fil i rotkatalogen för att installera det. Det kommer i paketdatabasen att skapas en post med detta pakets innehåll, så att det senare kan uppgraderas eller avlägsnas.

Man skall lägga märke till underkatalogen `install/`. Detta är en särskild katalog, som kan innehålla ett skript (en kommandofil) vid namn `doinst.sh`, vilket skall utföras efter installationen. Om pakethanteringssystemet upptäcker denna fil, så kommer det att exekvera skriptet, när paketet är installerat.

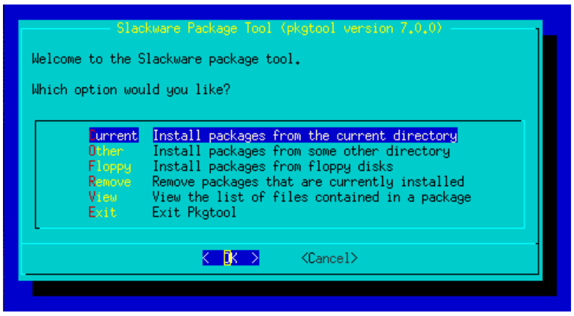
Även andra skript kan inbäddas i paketet, men de diskuteras mer ingående i avsnittet om `makepkg` på s. 171.

16.2 Tillbehör för pakethantering.

Det finns fyra viktiga tillbehör för pakethantering. Dessa installerar, avlägsnar och förnyar programpaket.

16.2.1 `pkgtool`.

`pkgtool` (8) är ett menystyrt program, som medger installation och avinstallation av programpaket. Dess meny ser ut som i figuren.

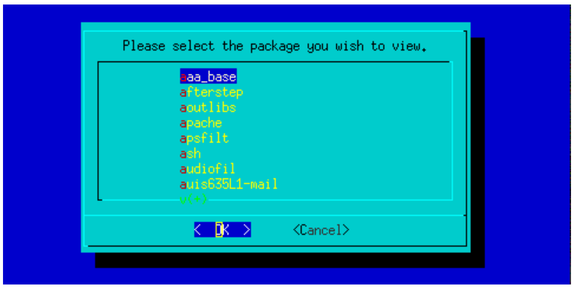


Figur 16.1: Huvudmenyn i `pkgtool`.

Man kan även granska en förteckning över innehållet i installerade paket, vilken då kan se ut som i nästa figur.

Om man vill ta bort paket, så väljer man alternativet *remove* och så får man en krysslista över samtliga installerade paket. Man kryssar med mellanslagstangenten för dem, man vill ta bort, och så tar `pkgtool` bort dem.

En del användare föredrar detta tillbehör framför kommandoraden. Emellertid skall man lägga märke till, att kommandoradsverktygen erbjuder många fler valmöjligheter. Möjligheten att uppgradera paket finns därtill endast, när man använder kommandoradsverktygen.



Figur 16.2: Granskningsläge för `pkgtool`.

16.2.2 `installpkg`.

`installpkg` (8) hanterar installation av nya paket i systemet. Följande syntax gäller:

```
# [ROOT=<path>] installpkg [alternativ] <paketnamn>
```

Kommandot `installpkg` har tre möjliga växlar; endast en av dem kan användas åt gången.

Flagga	**Verkan**
`-m`	Kör `makepkg` på innehållet i aktuell katalog.
`-warn`	Visar vad som skulle hända vid installation men installerar ingenting. Försiktighetshetsmått på system i drift.
`-r`	Installerar rekursivt alla paket i aktuell katalog och neråt. Jokertecken kan användas till sökmask vid rekursiv installation.

Om man har satt miljövariabeln för `ROOT` före körningen av `installpkg`, kommer denna sökväg till rotkatalogen att användas vid installation. Detta kan vara katalogen `/mnt` eller något annat än `/`.

Databasen över installerade paket förvaras i `/var/log/packages`. En »post« i denna databas är faktiskt bara en ren textfil, en för vart paket. Om paketet ifråga innehåller ett skript, som skall utföras efter installation, så skrives detta skript till `/var/log/scripts/<paketnamn>`.

Man kan ange flera programpaket eller använda jokertecken för paketnamnet. Tänk på, att `installpkg` inte talar om, ifall det skriver över ett redan installerat paket. Det installerar helt enkelt det nya paketet ovanpå det gamla. Om man vill försäkra sig om, att gamla filer ifrån en tidigare version avlägsnas på säkert sätt, så skall man använda kommandot `upgradepkg`.

16.2.3 removepkg.

removepkg (8) hanterar avlägsnandet av installerade programpaket. Följande syntax gäller:

```
# [ROOT=<path>] removepkg [option] <package name>
```

removepkg har fyra möjliga växlar. Endast en åt gången kan användas.

Flagga	Verkan
-copy	Paketet kopieras till katalogen för bevarade paket. Detta skapar ett träd av det ursprungliga paketet utan att ta bort det.
-keep	Sparar tillfälliga filer, som skapats under borttagandet. Endast användbart för avlusningsändamål.
-preserve	Paketet avlägsnas men kopieras ändå till katalogen för bevarade paket.
-warn	Visar vad som skulle hända, om man avlägsnade paketet.

Om man sätter miljövariabeln ROOT innan man kör removepkg, så kommer denna sökväg att användas som rotkatalog. Detta är till nytta, när man riggar upp nya hårddiskar till rotkatalog. De monteras vanligen under /mnt eller något annat än /.

removepkg granskar andra installerade paket och avlägsnar endast filer, som är unika för angivet paket. Det genomsöker även efterinstallationsskript efter angivet paket och tar bort symboliska länkar, det skapat.

Under avinstallationsprocessen visas en tillståndsrapport. Efter borttagandet flyttas paketets post i databasen till /var/log/removed_packages och efterinstallationsskriptet flyttas till /var/log/removed_scripts.

16.2.4 upgradepkg.

upgradepkg (8) förnyar ett redan installerat Slackwarepaket med en nyare version. Följande syntax gäller:

```
# [ROOT=<path>] upgradepkg <package name>...
```

eller

```
# [ROOT=<path>] upgradepkg <gammalt paket>%<nytt paket>
```

upgradepkg installerar först det nya paketet och tar sedan bort det gamla, för att gamla filer inte längre skall finnas kvar i systemet. Om det förnyade paketet har ett annat namn, så använder man ovan angivna syntax med procenttecken mellan namnen för det gamla och det nya paketet.

Om man har satt miljövariabeln ROOT, innan man kör upgradepkg, så kommer denna sökväg att gälla för rotkatalogen. Detta är till nytta, när man riggar upp nya hårddiskar för rotkatalogen. De monteras vanligen under /mnt eller något annat än /.

16.2.5 `rpm2tgz/rpm2targz`.

Red Hat Package Manager är ett omtyckt paketeringssystem numera. Många programdistributörer erbjuder sina produkter i RPM-format. Eftersom detta inte är vårt eget inhemska format, så anbefaller vi inte folk att förlita sig på sådana paket. Emellertid förekommer program, som endast är tillgängliga såsom RPM-filer (även källkod).

Vi medlevererar därför ett program, som omvandlar RPM-paket till vårt eget `.tgz`-format. Detta gör det möjligt att utvinna innehållet ur ett paket (kanske med `explodepkg`) till en tillfällig katalog och där undersöka detta innehåll.

Programmet `rpm2tgz` skapar ett slackwarepaket med ändelsen `.tgz`, medan `rpm2targz` skapar ett arkiv med ändelsen `.tar.gz`.

16.3 Skapa paket.

Att skapa programpaket för Slackware kan vara både lätt och svårt. Det finns inget bestämt tillvägagångssätt för att skapa ett paket. Det fordras endast, att paketet skall vara en gzippad tarfil, och om den innehåller ett efterinstallationsskript, så skall detta heta `/install/doinst.sh`.

Om man är intresserad av att skapa paket till sitt eget system eller till ett nätverk, man har hand om, så bör man ta en titt på de olika *build*-skripten i Slackwares källkodskatalogträd. Det finns åtskilliga sätt att skapa paket.

16.3.1 `explodepkg`.

`explodepkg` (8) gör detsamma, som `installpkg` gör för att packa upp paketet, men installerar ingenting och registrerar ingenting i paketdatabasen. Det bara packar upp arkivet i aktuell katalog.

Om man tittar på Slackwares källkodskatalogträd, så får man se, hur vi använder detta kommando till »ram«-paket. Dessa paket innehåller en stomme av vad det slutliga paketet skall bli. De innehåller alla nödvändiga filnamn (med nollängd), befogenheter och ägarskap. Ett *build*-skript cattar paketinnehållet från källkodskatalogen till paketbyggnadskatalogen.

16.3.2 `makepkg`.

`makepkg` (8) packar ihop innehållet i aktuell katalog till ett giltigt slackwarepaket. Det genomsöker katalogträdet efter symboliska länkar och skapar ett block i installationsskriptet för att dessa länkar skall kunna återskapas, när paketet installeras. Det varnar även för nollängdsfiler i paketets katalogträd.

Detta kommando kör man normalt efter det, att man har skapat paketets katalogträd.

16.4 Skapa taggar och taggfiler (för `setup`).

Slackwares `setup`-program hanterar installation av programvarupaket i systemet. Det finns filer, som talar om för `setup`, vilka paket som måste installeras, vilka som är valfria och vilka som skall väljas såsom förinställning av `setup`.

En taggfil (märkfil) finns i första programvaruseriekatalogen och kallas `tagfile`. Det innehåller en förteckning över filerna i en viss diskettserie (programvaruserie) och deras ställning (*status*). Deras ställning kan vara någon av följande:

Flagga	**Verkan**
`ADD`	Paketet är erforderligt, för att systemet skall fungera riktigt.
`SKP`	Paketet överhoppas automatiskt (*skip*).
`REC`	Paketet är icke erforderligt men anbefalles (*recommended*).
`OPT`	Paketet är valfritt (*optional*).

Formatet är enkelt: `<paketnamn>: <ställning>`

Ett paket per rad. Den ursprungliga taggfilen till var programvaruserie förvaras såsom `tagfile.org`. Om man ställer till det med sin egen, så kan man återställa den till ursprungligt utförande.

Många administratörer föredrar att skriva sina egna taggfiler och börja med att välja »full« installation. Programmet `setup` läser då taggfilerna och utför installationen i överensstämmelse med deras innehåll. Om man använder `REC` eller `OPT`, visas en dialogruta med frågan om huruvida användaren vill ha ett visst paket eller ej. Därför anbefalles det, att man håller sig till `ADD` och `SKP`, när man skriver taggfiler för automatiserad installation.

Se bara till, så att taggfilerna sparas på samma ställe som originalen. Eller också kan man ange en egen sökväg till taggfilerna, om man har skräddarsydda taggfiler.

16.5 Sammanfattning.

Man bör nu vara förtrogen med programvarupaketets grundtanke och hur sådana paket användes i Slackware. Man bör även vara hemtam i de olika pakethanteringsverktygen och kunna använda dem. De viktigaste delarna av detta kapitel behandlar installation, avlägsnande och förnyande av programpaket. Detta är de vanligaste användningarna av pakethanteringstillbehören.

Emellertid bör man även ha fått någon insikt i, hur man skapar och undersöker paket.

17

ZipSlack och BigSlack.

17.1 Vad är ZipSlack och BigSlack?

»*Zipslack*« är en särskild version av Slackware Linux. Det är en färdig installation av Slackware, vilken är klar att köras ifrån en Windows- eller DOS-partition. Det är en grundläggande installation, och man får inte allt, som annars ingår i Slackware. Vill man ha en motsvarighet till ZipSlack, som innehåller hela Slackware, så skall man prova »*BigSlack*«.

Namnet ZipSlack härrör ur den form, paketet distribueras i, nämligen en stor `.ZIP`-fil. Windows- och DOS-användare är förmodligen förtrogna med sådana filer. De är komprimerade arkivfiler. ZipSlack-arkivet innehåller allt, som är nödvändigt, för att man skall kunna få igång Slackware.

Det är viktigt att lägga märke till, att ZipSlack och BigSlack skiljer sig betydligt ifrån en vanlig installation. Om än de fungerar på samma sätt och innehåller samma program, så är deras målgrupper och funktion olika. Fördelarna och nackdelarna med ZipSlack och BigSlack uppräknas nedan.

Slutligen bör man alltid läsa dokumentationen i förhandenvarande ZipSlack- eller BigSlack-kataloger. Den innehåller de senaste upplysningarna beträffande installation, systemstart och produkternas bruk i allmänhet.

17.1.1 Fördelar.

- Fordrar ingen ompartitionering av hårddisken.

- Bra sätt att lära sig Slackware Linux utan att ta sig igenom den vanliga installationsprocessen.

17.1.2 Nackdelar.

- Använder DOS' filsystem, som är långsammare än Linux' eget filsystem.
- Fungerar inte under Windows NT.

17.2 Anskaffning av ZipSlack och BigSlack.

Det är lätt att skaffa sig ZipSlack eller BigSlack. Om man har köpt den officiella CD-satsen med Slackware Linux, så har man redan både ZipSlack och BigSlack. Man letar upp den CD-skiva, som innehåller den, man vill ha, och stoppar denna skiva i CD-ROM-läsaren. Det är vanligen tredje eller fjärde skivan, men man bör därvid hysa mer tilltro till etiketten än till denna dokumentation.

Om man vill ladda ner ZipSlack eller BigSlack, så bör man först besöka vår sida »*Get Slack*« för att få de senaste upplysningarna om nerladdning:

`http://www.slackware.com/getslack/`

ZipSlack och BigSlack ingår i var utgåva av Slackware. Man letar upp den utgåva, man vill ha, och så går man till motsvarande katalog på FTP-stället. Katalogen med senaste utgåvan finns under

`ftp://ftp.slackware.com/pub/slackware/slackware/`

Där finnes ZipSlack i underkatalogen `/zipslack` och BigSlack i underkatalogen `/bigslack`. ZipSlack erbjudes såsom en enda stor `.ZIP`-fil eller uppdelad på flera mindre `.ZIP`-filer i lämplig storlek för disketter. De senare finns i katalogen `/zipslack/split`. BigSlack finns endast uppdelad på flera filer.

Förutom `.ZIP`-filerna skall man även ladda ner dokumentationsfiler och avbilder av startdisketter, som kan finnas i katalogen.

17.3 Installation.

När man väl har laddat ner de nödvändiga delarna, så skall man packa upp `.ZIP`-filen eller `.ZIP`-filerna. Man måste därvid använda en 32-bitars uppzippare. Storleken och filnamnen är för mycket för en 16-bitars uppzippare. Exempel på 32-bitars uppzippare är WinZip och PKZip för Windows.

Både ZipSlack och BigSlack är avsedda att packas upp direkt i rotkatalogen på någon enhet (såsom `C:` eller `D:`). Där skapas då en katalog vid namn `\LINUX`, vilken innehåller den egentliga installationen av Slackware. Man skall i denna katalog även finna de erforderliga filerna för att man skall kunna starta systemet.

När man har packat upp filerna, skall man ha en katalog vid namn `\LINUX` på vald enhet (hädanefter kallar vi den för `C:`).

17.4 Starta ZipSlack och BigSlack.

Det finns flera olika sätt att starta ZipSlack och BigSlack. Det vanligaste är att använda den medlevererade filen `LINUX.BAT` till att starta systemet ifrån DOS (eller ifrån DOS under Windows 9x). Denna fil måste redigeras för att passa befintligt system, innan den fungerar.

Först öppnar man filen `C:\LINUX\LINUX.BAT` i sin älsklingseditor. Längst uppe i filens början lägger man märke till en stor kommentar. Den förklarar, vad man behöver redigera i denna fil (och även vad man skall göra, om man startar ifrån en extern zipenhet). Det gör ingenting, om man inte begriper meningen med inställningen `root=`. Det finns flera exempel, så det är bara att ta ett av dem och prova på. Om det inte fungerar, kan man redigera filen igen, kommentera bort den nyss avkommenterade raden och välja en ny.

När man har avkommenterat den rad, man vill använda, genom att ta bort »`rem`« ifrån radens början, så sparar man filen och stänger editorn. Därefter går man ut i DOS-läge, om man kör Windows.

VARNING! Det går *icke* att använda ett DOS-fönster i Windows 9x!

Man kommenderar `C:\LINUX\LINUX.BAT` för att starta systemet. Om allt går vägen, så skall man snart få en inloggningsprompt.

Där loggar man in som **root** utan att ange något lösenord. Förmodligen vill man ge **root** ett lösenord och lägga till ett användarkonto för sig själv. Vad de sakerna beträffar, så kan man rådfråga motsvarande avsnitt i denna bok om allmänt handhavande av systemet.

Om det inte gick att starta med `LINUX.BAT`, så får man vända sig till den medföljande `C:\LINUX\README.1ST` för att få reda på andra sätt att starta.

17.5 Lägga till, ta bort och förnya program.

ZipSlack och BigSlack använder samma programpaket, som en vanlig slackwareinstallation. Det betyder, att man kan använda Slackwares vanliga pakethanteringsverktyg för att lägga till, ta bort och förnya programvara. Man kan t.o.m. lägga till paket direkt ifrån Slackwares CD-ROM. Se s. 168 i kapitel 16 om pakethanteringen.

17.6 Vanliga svårigheter.

Nedan uppräknas de vanligaste svårigheter, som tillstöter för användare av ZipSlack och BigSlack. Följande avsnitt går närmare in på dem.

17.6.1 Unable to open initial console.

Detta felmeddelande orsakas vanligen av, att man har angivit fel partition såsom rotpartition vid `root=` i filen `LINUX.BAT`. Man får då redigera `LINUX.BAT` igen och välja en annan partition för `root=`. Om man inte har någon som helst aning om, vad detta innebär, så får man helt enkelt prova sig fram, till man får en inställning, som fungerar.

Detta felmeddelande kan även bottna i, att man har packat upp `.ZIP`-filerna på ett annat ställe än enhetens rotkatalog. Arkiven måste packas upp direkt i rotkatalogen på en enhet och inte i någon underkatalog.

17.6.2 Kernel panic: VFS: Unable to mount root fs.

Detta betyder, att man har angivit fel enhet (*device*) såsom `root`-enhet. Om man inte vet, vilken det är, så får man prova allesamman, tills man hittar någon, som fungerar.

17.7 Att få hjälp.

Innan man ber om hjälp, så bör man läsa igenom dokumentationen i katalogen `C:\LINUX` för att se, om där finns något svar på frågan.

Om man har läst dokumentationen men likväl har problem, så kan nedan uppräknade hjälpmedel förhoppningsvis lösa svårigheterna.

17.7.1 ZipSlack FAQ.

En lista över vanliga frågor och deras svar finns både i filen `FAQ.TXT` i katalogen `C:\LINUX` och på WWW-stället

`http://www.slackware.com/faq/`

17.7.2 ZipSlack Discussion Forum.

Det finns ett diskussionsforum om ZipSlack på nätet, där man kan tala med andra användare av ZipSlack och BigSlack. Detta är ett strålande sätt att få direkt hjälp av andra användare.

`http://www.slackware.com/forum/`

17.7.3 Användarstöd genom epost.

Slackwares *Support Team* försöker hjälpa dem, som har problem med ZipSlack och BigSlack. Man skall därvid alltid formulera problemet klart och framlägga alla upplysningar, man tror kan vara nödvändiga för problemets lösande.

`support@slackware.com`

17.8 Sammanfattning.

Man bör nu förstå, vad ZipSlack och BigSlack är. För det fall att man har bestämt sig för att använda endera av dessa, så bör man nu även veta, hur man skall kunna installera, starta, felsöka och få hjälp. ZipSlack och BigSlack kan vara mycket bekväma, om man bara vill prova på Slackware men inte vill ta bort befintliga windowspartitioner.

Ordlista.

Arbetskatalog (working directory) Den katalog, vari ett program anser sig köra.

Bakgrunden (background) Man säger om en process, som är igång utan att ta emot eller styra inmatning på en terminal, att den köres i bakgrunden.

Bibliotek (library) En samling funktioner, som kan delas av olika program.

Darkstar Förinställt värddatornamn i Slackware; datorn kallas för »darkstar«, om man inte anger något annat namn.

En av Patrick Volkerdings utvecklingsdatorer, uppkallad efter sången »Dark Star« av the Grateful Dead.

Demon (daemon) Ett program, som är avsett att köras i bakgrunden och utan användarens ingripande utför en bestämd uppgift (vanligen tillhandahållande av en tjänst).

Device node En särskild sorts fil i filsystemet `/dev`, som inför operativsystemet företräder en viss maskinvarukomponent.

DNS *Domain Name Service*. Ett system, som översätter datorernas namn till IP-adresser med siffror.

Domännamn (domain name) En dators namn i DNS (värdnamnet hör också dit).

Dotted quad IP-adresserna är uppdelade på fyra tal (i intervallet 0-255 decimalt), som avdelas med punkter. Detta skrivsätt kallas ofta i engelskspråkiga texter för *dotted quad*.

Drivare (device driver) Ett stycke kod i linuxkärnan, som styr en bestämd maskinvaruenhet.

Dynamisk laddare (dynamic loader) När ett program kompileras i Linux, så använder det ofta små stycken av programkod (funktioner) ifrån externa programbibliotek. När sådana program skall köras, måste dessa bibliotek finnas tillgängliga och nödvändiga funktioner laddas i minnet. Detta är den dynamiska laddarens uppgift.

Epoch En epok är ett tidsavsnitt i historien; i Unix börjar »The Epoch« klockan 00:00:00 UTC den 1 januari 1970. Detta anses vara »tidernas morgon« i Unix och unixliknande operativsystem, och all övrig tid räknas efter denna tidpunkt.

Filsystem (filesystem) En avbild av lagrade data, där »datafiler« hålls lagrade i »kataloger«. Filsystemet är allmän form för återgivande av data, som lagrats på skivor (såväl fasta skivor som löstagbara).

Framebuffer En sorts grafikanordning; i Linux är det ofta fråga om en programvarumässig *framebuffer*, som tillhandahåller ett standardgränssnitt för programmen, medan den samtidigt döljer de särskilda maskinvarudrivarna för dem. Detta är ett abstraktionsskikt, som befriar programmen ifrån behovet av att tala olika språk med olika maskinvarudrivare.

FTP *File Transfer Protocol.* FTP är ett omtyckt sätt att flytta filer mellan datorer.

Fönsterhanterare (window manager) Ett X-program, som har till uppgift att tillhandahålla ett grafiskt gränssnitt, vilket går utöver X-fönstersystemets enkla rektangelritande. Fönsterhanterare brukar tillhandahålla titelbalkar för fönster, menyer o.s.v.

Fönstersystemet X (X Window System) Nätverksorienterat grafiskt gränssnitt, som kommer till användning i de flesta unixliknande operativsystem, Linux inberäknat.

Förgrunden (foreground) Ett program, som tar emot eller styr inmatning på en terminal, säges köra i förgrunden.

Gateway En dator som förmedlar data från ett nät till ett annat nät.

GID *Group Identifier.* GID är ett unikt nummer, som tillordnas en bestämd användargrupp.

Grupp (group) I Unix tillhör var användare en *grupp*, som kan innehålla många användare. Grupptillhörighet utnyttjas för att styra åtkomst och rättigheter mer allmänt än det vore möjligt med förekomsten av olika användarkonton enbart.

GUI *Graphical User Interface.* Ett mjukvarugränssnitt, som använder grafiska element såsom knappar, bläddderlister, fönster o.s.v. i stället för enbart textbaserad in- och utmatning.

GUI Toolkit En samling bibliotek, som ger programmeraren färdig kod till framställning av s.k. *widgets* såsom bläddderlister, kryssrutor o.s.v. och bygger upp det grafiska gränssnittet. En sådan GUI-verktygslåda, som ett program använder sig av, är ofta avgörande för, hur detta program ser ut och hur det betjänas.

Hemkatalog (home directory) En användares hemkatalog är den katalog, som användaren kommer direkt in i vid inloggningen. Användaren har fullständiga rättigheter och får regera mer eller mindre fritt i sin egen hemkatalog.

HOWTO Ett dokument, som beskriver, hur man gör ett eller annat, såsom att konfigurera en brandvägg eller hantera användare och grupper. Det finns en stor samling sådana dokument, som kan skaffas ifrån *Linux Documentation Project.*

HTTP *Hypertext Transfer Protocol.* HTTP är det protokoll, som Världsväven *World Wide Web* främst arbetar med.

ICMP *Internet Control Message Protocol.* Ett rudimentärt nätverksprotokoll, som huvudsakligen användes för `ping`.

Internt kommando (shell builtin) Ett kommando, som ingår i kommandoskalet, i motsats till ett kommando, som tillhandahålles av ett externt program. Exempelvis är kommandot `cd` inbyggt i `bash`.

Konto (account) Alla upplysningar om en användare, inberäknat användarnamn, lösenord, fingerinformation, UID och GID samt hemkatalog. Att skapa ett konto är detsamma som att lägga till och definiera en användare.

Kompilera (compile) Översätta källkod (människoläsbar programtext) till maskinbegriplig »binär« kod.

Källkod (source code) Den för människor mer eller mindre läsbara kod, som de flesta program är skrivna i. Källkoden måste kompileras (översättas) till »binär« kod, för att datorn skall kunna köra programmen.

Kärna (kernel) Operativsystemets hjärta. Kärnan är den del, som tillhandahåller grundläggande processtyrning och ett gränssnitt emot datorns maskinvara.

Kärnmodul (kernel module) Ett stycke programkod till kärnan, vanligen något slags drivare, som kan laddas in i minnet och tas bort därifrån igen för sig själv medan huvuddelen av kärnan hela tiden ligger kvar i minnet. Moduler är behändiga, när man skall uppgradera drivare eller prova nya inställningar för kärnan, eftersom de kan laddas och tas bort igen, utan att man behöver starta om.

LILO *LInux LOader.* LILO är den systemladdare, som fått vidsträcktast användning i Linux.

LOADLIN Ett program, som kör under MS DOS eller Windows och därifrån startar Linux. Det användes främst på datorer med flera operativsystem (däribland självfallet Linux och DOS/Windows).

Manualavsnitt (man section) Sidor i den gängse hjälpen för Unix (»`man`«) är ordnade i avsnitt, för att man enkelt skall kunna hänvisa till dem. Alla sidor om C-programmering ligger i avsnitt 3, systemadministration i avsnitt 5 o.s.v.

MBR *Master Boot Record.* Ett område på hårddisken, vilket är förbehållet upplysningar om systemstart. LILO eller andra systemladdare kan skrivas hit.

Miljövariabel (environment variable) En miljövariabel är en variabel, som är satt i en viss användares skal. Användaren eller program, denne användare kör i detta skal, kan då hänföra sig till denna variabel. Miljövariabel används vanligen till att lagra personliga inställningar och andra förvalda parametrar.

Monteringsställe (mount point) En tom katalog i ett filsystem, varunder ett annat filsystem kan »monteras« eller ympas på.

Motif En uppsättning verktyg och bibliotek för programmering av grafiska gränssnitt. Användes i många äldre X-program.

MOTD *Message of the Day.* Detta är ett meddelande, som visas för alla användare, när de loggar in. Texten lagras i `/etc/motd`. Sedan gammalt fungerar detta som en anslagstavla, där systemadministratören kan lämna meddelanden till samtliga användare.

Namntjänare (name server) En server (nätverkstjänare), som hanterar DNS (domännamnstjänsten). Namntjänare översätter värddatornamn till numeriska IP-adresser.

NFS *Network File System.* NFS gör det möjligt att montera filsystem på andra datorer som om de funnes i den egna datorn. NFS erbjuder på så vis transparent fildelning.

Nätverksgränssnitt (network interface) Kärnan tillhandahåller en virtuell representation av var nätverksenhet såsom nätverksgränssnitt, så att användare och program kan komma åt nätverksenheter.

Oktal (octal) Ett talsystem, som bygger på basen 8, är oktalt. Det har siffrorna 0-7. Talet 8 blir 010 oktalt, talet 10 blir 012 o.s.v.

Pager I fönstersystemet är en *pager* ett program, som låter användaren se och växla mellan flera olika virtuella »skrivbord«.

Partition En avgränsad del av en hårddisk. Filsystem finns i partitioner.

PPP *Point-to-Point Protocol.* PPP används huvudsakligen vid förbindelser genom modem och telefonledning till en ISP, *Internet Service Provider* (internetleverantör).

Programpaket (software package) Ett program och dess tillhörande filer, arkiverat och komprimerat till en enda fil tillsammans med nödvändiga skript och upplysningar, som hjälper till med installation, uppgradering och borttagande av programfilerna.

Programserie (software series) En samling samhörande programpaket i Slackware. Alla KDE-paket tillhör serien »KDE«, nätverkpaket serien »N« o.s.v.

Process Ett program, som är igång och kör.

Punktfil (dot file) Dold fil. I Linux börjar namnen på dolda filer med en punkt (*dot*).

Root disk Den skiva (vanligen en hårddisk), där rotkatalogen lagras. Då man installerar Slackware medelst start på disketter, använder man förutom startdisketten även en *root disk*, som då också är en diskett.

Rotkatalog (root directory) Rotkatalogen är högsta nivån i filsystemet och betecknas med det vanliga, framåtlutade snedstrecket `/`. Alla andra kataloger förgrenar sig i ett »fil- och katalogträd« under rotkatalogen.

Routingtabell (routing table) Den uppsättning upplysningar, som kärnan använder för att välja rätt väg, när data skickas omkring på nätverket. Routingtabellen innehåller uppgift om förvald *gateway* (se d.o.), vilket nätverksgränssnitt som är anslutet till vilket nätverk o.s.v.

Secure shell Ett sätt att logga in krypterat (och därmed säkert) på en fjärrdator. Flera säkra skal är tillgängliga; man behöver därvid både en klient och en server.

Shadow password suite Skugglösenordssviten gör det möjligt att dölja krypterade lösenord för användare, medan `/etc/passwd` för övrigt likväl kan läsas av alla. Detta bidrar till att motverka försök att med ren råstyrka knäcka lösenord.

Signal Unixprogram kan meddela sig med varandra medelst enkla »signaler«, som är numrerade och vanligen har särskilda betydelser. Kommandot `kill -l` visar en förteckning över tillgängliga signaler.

Skal (shell) Ett skal (kommandoskal, kommandotolk) ger användaren ett kommandoradsgränssnitt. När man ser en textprompt såsom dollartecknet `$` eller brädgårdstecknet `#`, då befinner man sig i ett skal.

Skrivbordsmiljö (desktop environment) Ett grafiskt användargränssnitt (GUI) som köres ovanpå fönstersystemet X och tillhandahåller sådant som integrerade tillämpningsprogram, enhetligt sammanhängande stil på program och beståndsdelar, fil- och fönsterhanteringsmöjligheter o.s.v. Ett steg längre än en enkel fönsterhanterare.

SLIP *Serial Line Interface Protocol.* SLIP är ett protokoll, som liknar PPP på så vis, att det användes till att förbinda två maskiner genom ett seriellt gränssnitt.

Standard Error (stderr) I Unix är detta den vanliga utmatningen av felmeddelanden. Programmen skickar felmeddelanden till *stderr*, så att de kan skiljas ifrån vanlig utmatning.

Standard Input (stdin) Detta är i Unix »ingången« för data till ett program. Data kan omdirigeras eller skickas i rörledning till *stdin* för ett program ifrån vilken källa som helst.

Standard Output (stdout) Detta är i Unix »utgången« för data från ett program. Vanlig textutmatning ifrån ett program skrives till *stdout*, som är åtskilt ifrån felmeddelandena, som skrivs till *stderr*, och kan omdirigeras eller skickas i rörledning till ett annat programs *stdin* eller till en fil.

Startdiskett (boot disk) En självstartande diskett, som innehåller ett operativsystem (i vårt fall en linuxkärna), varmed en dator kan startas.

Subnet Ett IP-adressområde, som är en del av ett större område. 192.168.1.0 är t.ex. ett *subnet* till 192.168.0.0 (där 0 är en mask, som betyder »odefinierad«); nämligen undernätet ».1«.

Superblock I Linux talar man om partitioner utifrån begreppet *block*. Ett block är 512 byte. Superblocket är de första 512 byten på en partition.

Supplemental disk I Slackware är detta en diskett, som användes under en installation, men som varken innehåller någon linuxkärna (som finns på en *boot disk*, startdiskett) eller ett rotfilsystem (som finns på en *root disk*), utan ytterligare filer, som kan behövas, såsom t.ex. nätverksmoduler eller stöd för PCMCIA.

Suspended process En process, som har blivit »fryst«, så att den är pausad tills den endera dödas eller återupptages.

Swap space Diskutrymme, som av kärnan användes till virtuellt minne. Det är långsammare än RAM (verkligt minne), men eftersom hårddiskutrymme är billigare än minne, så kan man använda swap i rikligare mått. Swap-utrymmer kan vara till nytta för kärnan, när den behöver hushålla med minnesutrymmet för data, som inte behöver användas så ofta, eller såsom reserv, när förrådet av riktigt, fysiskt RAM är uttömt.

Symbolisk länk (symbolic link) En särskild fil, som enbart pekar på var någonstans, en annan fil finns. Symboliska länkar kommer till användning, när man vill undvika att kopiera en fil i onödan, men likväl filen behövs på mer än ett ställe.

Tagfile En fil som används av Slackwares `setup`-program under installation. I en taggfil beskrives den uppsättning paket, som skall installeras.

Terminal Ett gränssnitt mellan människa och dator, vilket består av åtminstone en skärm (eller virtuell skärm) och något slags inmatningsanordning (vanligtvis åtminstone ett tangentbord).

Tjänst (service) Delande av information och/eller data mellan program och datorer från en enda »*server*« (tjänare) till flera »klienter«. HTTP, FTP, NFS o.s.v. är sådana nätverkstjänster (*services*).

UID *User Identifier.* Ett unikt nummer, som identifierar en bestämd användare för systemet. UID används av de flesta program i stället för användarnamn, eftersom det är lättare att hantera siffror än namn; användarnamn används i allmänhet endast då, när användaren måste se det.

VESA *Video Electronics Standards Association.* Benämningen »VESA« betecknar ofta en standard, som denna sammanslutning har fastställt. Nästan alla moderna grafikkort överensstämmer med någon VESA-norm.

Virtuell terminal (virtual terminal) Användning av programvara till att simulera flera terminaler, fastän man endast förfogar över en enda uppsättning in- och utmatningsenheter (tangentbord, bildskärm, mus). Särskilda tangentkombinationer växlar mellan de virtuella terminalerna på en enda fysisk terminal.

Wrapper program Ett program, vars enda uppgift är att köra andra program, men på något vis ändra deras uppförande genom att förändra deras omgivning eller filtrera deras utmatning.

X server Det program i X-fönstersystemet, som arbetar direkt emot den grafiska maskinvaran och hanterar den faktiska körningen av X-program.

GNU General Public License.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program. You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this

License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances. It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice. This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

12. NO WARRANTY

    BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL,

```
    INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR
    INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS
    OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED
    BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE
    WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY
    HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

  END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
  possible use to the public, the best way to achieve this is to make
  it free software which everyone can redistribute and change under
  these terms.

  To do so, attach the following notices to the program. It is safest
  to attach them to the start of each source file to most effectively
  convey the exclusion of warranty; and each file should have at
  least the "copyright" line and a pointer to where the full notice
  is found.

  <one line to give the program's name and a brief idea of what it
  does.> Copyright (C) <year> <name of author>

  This program is free software; you can redistribute it and/or
  modify it under the terms of the GNU General Public License as
  published by the Free Software Foundation; either version 2 of the
  License, or (at your option) any later version.

  This program is distributed in the hope that it will be useful, but
  WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
  USA

  Also add information on how to contact you by electronic and paper
  mail.

  If the program is interactive, make it output a short notice like
  this when it starts in an interactive mode:

   Gnomovision version 69, Copyright (C) year name of author
   Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type
   `show w'.  This is free software, and you are welcome to
   redistribute it under certain conditions; type `show c' for
   details.

  The hypothetical commands `show w' and `show c' should show the
  appropriate parts of the General Public License. Of course, the
  commands you use may be called something other than `show w' and
  `show c'; they could even be mouse-clicks or menu items--whatever
  suits your program.

  You should also get your employer (if you work as a programmer) or
  your school, if any, to sign a "copyright disclaimer" for the
  program, if necessary. Here is a sample; alter the names:
```

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
program 'Gnomovision' (which makes passes at compilers) written by
James Hacker.

<signature of Ty Coon>, 1 April 1989

 Ty Coon, President of Vice

 This General Public License does not permit incorporating your
 program into proprietary programs. If your program is a subroutine
 library, you may consider it more useful to permit linking
 proprietary applications with the library. If this is what you want
 to do, use the GNU Library General Public License instead of this
 License.
```

Figurer