LinuxBoken

En bok om hur man använder GNU/Linux

Marcus Rejås <marcus@rejas.se>

Version 0.1
17 januari 2003

Copyright

Till dig, kära läsare.

Ändringar

Version 0.0, 3 februari 1999

Började skriva på detta dokument.

Version 0.1, 17 januari 2003

Första publika versionen av detta dokument. Dokumentet skall vara läsbart men lång ifrån färdigt.

ToDo

“ToDo” lista

Index Ett index måste finnas för att boken skall gå att slå i. Kommer att implementeras till nästa version. *Prioritet Hög*

Bibliografi En riktig bibliografi skall skapas. I slutet av varje kapitel skall stå var man kan läsa mer om man är intresserad. *Prioritet Medel*

Mer enheteliga exempel Alla promptar skall se ut som $ eller #. Alla exempel skall behandla att exempel-filträd som är gemongående. *Prioritet Låg*

0.1 Förord till 0.1

Hej alla intresserade!

Det ni nu skall läsa är vad som brukar kallas “work in progress”. Det var från början inte alls tänkt att bli en så stor bok som det blev. Nu har jag skrivit lite under varje rubrik. På så sätt har jag själv bildat mig en uppfattning om hur boken kommer att bli då den blir klar, om den blir klar. Det mesta är skrivet ur huvudet varför det inte finns någon riktig källförteckning.

Mitt mål är nu att själv läsa boken. Fundera på om jag skall göra några strukturella ändringar. Sedan skall jag kapitel för kapitel, avsnitt för avsnitt, läsa den litteratur jag har tillgänglig och skriva mer heltäckande om varje ämne.

Nästa version, 0.2, kommer att ha vissa kapitel helt omskrivna. Blir det någon version 1.0 så kommer förmodligen hela boken att vara omskriven.

Det kan finnas “buggar” i typsättningen. Förutom att den är ful på vissa ställen. Jag prioriterar typsättningen lågt i förhållande till inehållet i detta skede.

Så snälla, läs denna bok, döm mig inte för hårt och skicka mig förslag inför senare versioner.

Tack!

Marcus Rejås, Gävle 1999-04-20

Innehåll

Figurer

Tabeller

Del I

Introduktion

Kapitel 1

Introduktion till Linux

Linux är ett operativsystem på frammarsh. Varje dag upptäcker nya användare dess fördelar. Linux är idag (mars -99) ännu inte lika enkelt att administrera som Windows 9x för en ovan hemma-användare. Men efter att ha läst lite litteratur i ämnet kan de flesta lära sig tillräckligt för att kunna utnyttja sitt system. Idag pågår på flera håll utveckling av Linuxsystem som skall vara lika enkla (t.o.m. enklare) som Windows att installera och underhålla.

Ett Linuxsystem är redan idag lika enkelt att använda för en ovan användare om de inte behöver utsättas för det som beskrivs under systemadministrationsdelen i denna bok. Framför allt om de inte ger sig in på att göra de mer avancerade saker som beskrivs i denna bok. Dessa saker går oftast inte ens att göra i system som till exempel Windows.

Linuxsamhället kallas ofta den inbitna grupp av Linuxanvändare som ständigt växer. De har själva skrivit sitt operativsystem och delar med sig av det till alla som vill ha. Tyvärr så kan det, ibland, verka som om denna grupp består av ett gäng omogna tonåringar. Detta beror till största del på att det är dessa som hörs i vissa nyhetsgrupper och på realtidschattar runt om på Internet. Men de som mest bidragit till Linux utveckling är proffs, studenter och personal vid högskolor och universitet runt om i världen. Även stora kommersiella företag har bidragit till utvecklingen av Linux. Även om de som till största delen har jobbat med Linux har gjort det på sin fritid.

1.1 Historia

För att beskriva Linux historia får vi ta och börja lite kort med UNIX historia. Detta för att Linux skrivits för att vara ett UNIXsystem och för att man skall få lite känsla för vad det handlar om.

Skall man skriva om UNIX får man i sin tur backa till något som kallades *Multics*. Multics var ett operativsystem som togs fram av Massachusetts Institute of Technology (MIT), AT&T Bell Labs och General Electric Co. (GE). Operativsystemet skulle köras på en GE 635 som var en mycket avancerad stordator. Tyvärr så blev projektet för stort och kostsamt. Bell Labs drog sig till slut ur projektet. Två stycken anställ-

da vid Bell Labs, **Ken Thompson** och **Dennis Ritchie** hade kommit i kontakt med Multics via en preliminär version som fanns på Bell Labs. Thompsom hade skrivit ett spel som kördes på GE 635:an och Multics. Nu hade dessa två tillgång till en minidator på Bell Labs, en PDP-7. De ville köra spelet som för övrigt hette *Space Travel* på PDP-7:an Att göra om spelet så det kunde köras på den maskinen innebar en hel del arbete. Utvecklingen skedde på en annan GE 635:a med ett mycket enkelt batchoperativsystem (GECOS). Efter fick de flytta programmet till den nya maskinen med pappersremsor. Detta var så tidsödande och tråkigt att Thompson skrev ett enkelt filsystem och en del program för PDP-7:an. Detta var grunden till det vi idag kallar UNIX. Namnet UNIX fick det något år senare (1970). Det har aldrig sagts officiellt vad UNIX står för eller betyder.

Trots att systemet var väldigt primitivt så uppskattades det av Ritchie och några andra kollegor. Till slut lyckades de få en mer avancerad maskin är den PDP-7 de använt. För att få denna maksin (en PDP-11) fick de lova att lägga till möjligheter att behandla text i systemet. UNIX var nu ett officiellt projekt hos Bell Labs.

1973 skrevs UNIX om helt och hållet. Nu i högnivåspråket C.

1.2 Hur det började

Jamfört med UNIX som vi nyss behandlat så är Linux ett nytt operativsystem. Det var så sent som 1991 den finske studenten Linus Torvalds påbörjade utvecklingen av sitt eget operativsystem. Linux har skrivits med UNIX som en förebild. Eftersom Linux har en förebild som är relativt utvecklad har en hel del barnsjukdommar som drabbade UNIX kunnat undvikas. I detta avsnitt beskrivs kort hur operativsystemet Linux kom till.

En student vi namn **Linus Torvalds** studerade på Helsingfors Universitet. Han hade tidigare provat en hel del operativsystem, men när han kom i kontakt med Unix för första gången insåg han att det var så en dator skulle fungera. Han ville naturligtvis ha ett likadant system hemma. Men ett Unix system kostade långt över 30000 kronor och var då inte ens komplett. Inte att tänka på för en student med andra ord. Han hade hört talas om en Unix-klon som hette Minix. Denna kostade cirka 1000 Mark och gick på en vanlig PC. Så Torvalds köpte en 386:a och beställde Minix. Han gillade dock aldrig Minix. Men eftersom det inte fanns några alternativ så körde han på det i flera månader. Han började utforska vad systemet kunde. Han postade i Juli -91 detta meddelande i nyhetsgruppen[1] `comp.os.minix`.

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: Gcc-1.40 and a posix-question
Message-ID: <1991Jul3.100050.9886@klaava.Helsinki.FI>
Date: 3 Jul 91 10:00:50 GMT

Hello netlanders,

Due to a project I'm working on (in minix), I'm interested in the posix
standard definition. Could somebody please point me to a (preferably)
```

[1] nyhetsgrupper (usenet) behandlas mer i avsnitt 15.5 på sidan 161.

```
machine-readable format of the latest posix rules? Ftp-sites would be
nice.
```

Och idag vet alla vad det var för projekt han jobbade på. Nämligen det vi idag känner som Linux. Lite senare postade han detta meddelande.

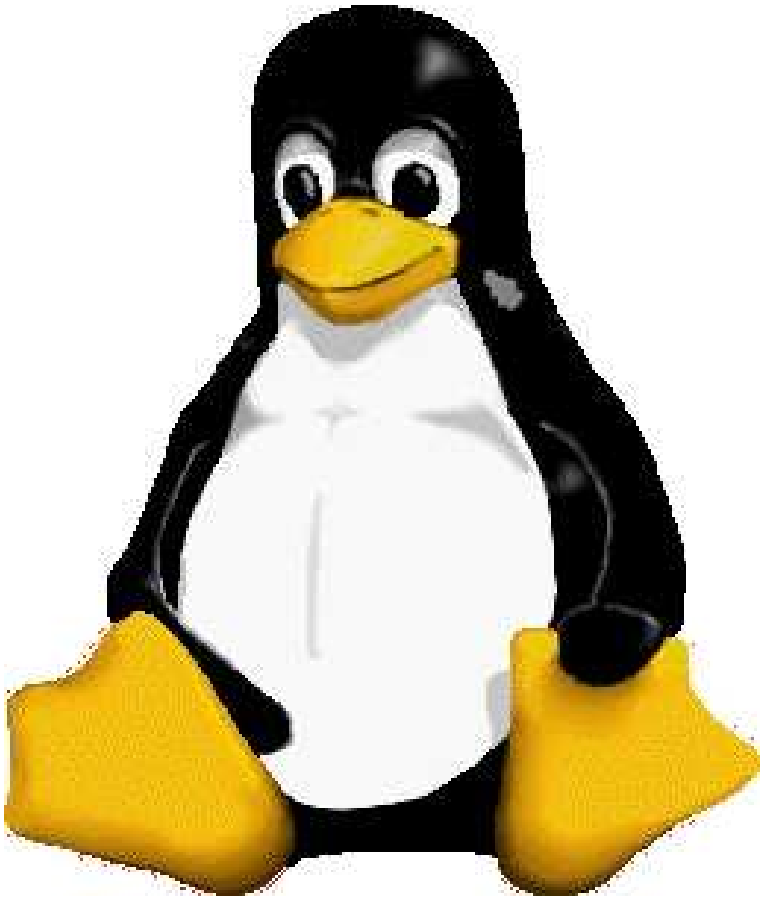
```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki


Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones.  This has been brewing
since april, and is starting to get ready.  I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
This implies that I'll get something practical within a few months, and
I'd like to know what features most people would want.  Any suggestions
are welcome, but I won't promise I'll implement them :-)

              Linus (torvalds@kruuna.helsinki.fi)

PS.  Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's all I have :-(.
```

Han hade alltså nu början till vad vi idag kallar Linux. Satsen "won't be big and professional" tydde väl på att han inte hade en aning om vad det var han höll på att starta. Eftersom Torvalds var irriterad på att han inte kunnat få det system han vill från början eftersom det var för dyrt bestämde han att Linux skulle finnas fritt tillgängligt till alla. Den första copyright han skrev innebar att inga pengar fick tas ut då Linux kopierades. Denna copyright ändrades han sedan till att använda GNU General Public License[2]. Denna går kortfattat ut på att man får ta betalt för att distribuera programvaran men källkoden måste hållas fri.

Namnet Linux kommer naturligtvis från sin upphovsmans förnamn, men är inte påhittat av honom själv. Torvalds kallade sitt operativsystem för *Freax*. Freax skulle vara en akronym för Free, Freak och X. Namnet Linux kom **Ari Lemmke** på. Lemmke var administratör på ftp.funet.fi. Då han skulle skapa katalogen för Linux kallade han den just för Linux eftersom Torvalds inte hade sagt vad den skulle heta. Namnet Linux fastnade nästan direkt och blev sedan det officiella namnet.

Linux har numera även en officiell logo. Den vann en tävling och godkändes av Linus Torvalds. Logon föreställer en pingvin och är ritad av **Larry Eving**. Han använde programmet Gimp när han gjorde den. Gimp är ett fritt grafikprogram som följer med de flesta distributioner av Linux. Figur 1.1 visar Linux logo. Pingvinen har naturligtvis ett namn. Han heter Tux.

[2] GNU och deras publika licens behandlas i kap 48



Figur 1.1: Linux officiella logo. Pingvinen Tux.

Pingvinen skapades av **Larry Eving**. *Han använde programmet GIMP som är ett mycket bra och avancerat ritprogram. GIMP, som är en förkortning för GNU Image Manupilation Program, är ett GNU program helt i klass med gcc och emacs.*

1.3 Utvecklingen

Hur Linux utvecklas idag är egentligen fantastiskt. Massor av programmerare världen över programmerar för operativsystemet. Vissa skriver på kärnkoden och andra skriver drivrutiner. Allt övervakas fortfarande av Linus Torvalds, som själv fortfarande skriver kod. Linus själv avgör då en ny version är klar att släppas och vad av det som skickas till honom som skall vara med i den. Vad som kanske är mest fantastiskt är att alla programmerare jobbar helt gratis och delar med sig till alla av det de gjort.

1.4 Distributioner

Linux är egentligen bara en kärna. Det vill säga det innersta i ett operativsystem. Bara denna kärna kan man dock inte göra så mycket med. För att kunna utföra något nyttigt behöver du också en samling program till att utföra dina arbeten. Ett program skrivet för ett operativsystem kan inte utan vidare köras på ett annat. De program du använder i Linux skall alltså vara Linuxprogram. Dessa program är ofta, precis som Linux, gratis. Vissa organisationer och kommersiella företag har specialiserat sig på att sätta samman Linuxkärnan och en mängd nyttiga (och onyttiga) program i ett paket. Detta paket levereras ofta på en eller flera CD-ROM tillsammans med en tryckt manual. Med följer också ett installationsprogram som enkelt hjälper dig att installera operativsystemet och programmen på en Dator. En sådan samling program brukar kallas en Linuxdistribution, eller bara distribution, om det är klart att man talar om Linux. Exempel på stora Linuxdistributioner är Debian, RedHat, Slackware, S.u.s.e och Caldera Open Linux men det finns många flera. Ett bra sätt att främja utvecklingen av Linux är att köpa en paketerad distribution. Då går pengar tillbaka till olika projekt.

Som nybörjare är det bäst att du skaffar dig en av de allra största eftersom dessa är lättast att installera och underhålla. Har du någon “Linuxguru” i din bekantskapskrets bör du skaffa samma distribution som han/hon använder så kan du lättare få hjälp. Detta är ett bra tips eftersom det skiljer en del mellan distributionerna hur man gör vissa systemadministrativa saker. Fråga alltså din guru vilken distribution som han/hon rekomenderar.

1.5 Namnet Linux

När man i dagligt tal säger Linux så menar man i regel inte bara Linux. Man menar även alla de program som ingår i systemet. Då de flesta program i en distribution kommer från GNU-projektet bör man istället för att kalla en distribution för Linuxdistribution kalla den en GNU/Linux distribution. Detta är fallet med Debian GNU/Linux. Detta tycker jag skulle bli mer vanligt för att ge GNU-folket (med **Richard Stallman** i spetsen) mer ära för sitt jobb.

1.6 Målet med Linux

Målet med Linux har ändrats sedan det började. Thorvalds första utrop tydde på att det var ett litet hobbyprojekt som inte skulle bli något stort. Senare blev det ett mycket bra nätverksoperativsystem. Linux har haft enormt stora framgångar som web och mailservrar. Det är nog inom dessa områden son Linux idag är störst.

Målet idag är bland annat skrivborden. Linux skall kunna användas i stället för exempelvis Windows som är det vanligaste idag. Ännu har Linux en bit att gå för att komma dit men det händer nya saker hela tiden. Jag tror och hoppas att Linux kommer att bli stort även på detta område. Jag använder sedan länge bara Linux och UNIX.

Ett annat område där Linux vinner mark är så kallade inbäddade system. Ett inbäddat system kan man säga är ett system där själva datorn inte syns för användaren. Det finns massor av exempel, som mikrovågsugnar, hissar, smarta lådor av olika slag, mm.

1.7 Mer läsning

Detta avsnitt kommer i så gott som alla kapitel. Här rekommenderar jag andra böcker som jag tycket är bra och ur vilka viss information i kapitlet är hämtat. Vill du få tag i böckerna så presentaras de en gång till med information om förlag eller annat i litteraturförteckningen i slutet på boken.

[Thomas och Farrow, 1989] Är en bok om systemadministration. Ur denna bok är den korta historien om UNIX i huvudsak hämtad.

Kapitel 2

Linuxsystemet

I detta kapitel beskrivs kort Linux uppbyggnad och vad som är speciellt för Linux. Kapitlet är lite kort i denna version av boken. Det kommer att byggas ut i senare versioner.

2.1 Inledning

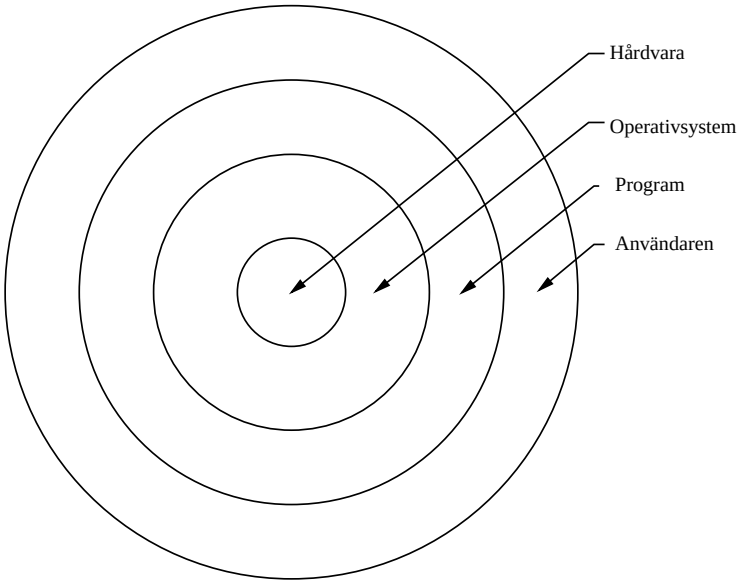
Linuxsystemet liknar till stor del UNIX. Unix i sin tur är inte en ensam produkt utan ett samlingsnamn för flera olika produkter. Linux skulle kunna vara Unix men är det inte. För den oinvigde kan det vara svårt att avgöra om man sitter vid en Linux eller UNIX maskin. Därför kommer jag ibland att skriva Linux och ibland UNIX. Vad som man också bör tänka på är att Linux bara är själva operativsystemet. Tack vare GNU-projektet och andra fria projekt, är många av de program man använder i Linux är exakt desamma som de man använder i UNIX. Detta eftersom dessa program är fria och sprids till flera plattformar. En Linuxdistribution består till så stor del av GNU-program att man gör rätt i att kalla den för GNU/Linux som är fallet med till exempel *Debians* Linuxdistribution som kallas just *Debian GNU/Linux*. Någon har föreslagit att Linux skall ändra på stavningen till “Lignus” för att få med GNU i namnet. Detta kommer dock aldrig att ske, och det är nog de flesta tacksamma för.

2.2 Vad är ett operativsystem

Att förklara detta på ett avsnitt i en bok om Linux är kanske att ta sig vatten över huvudet. Är du intresserad bör du köpa dig en bok i ämnet.

Det kan vara enkelt att börja med att tala om vad ett operativsystem inte är. Ett operativsystem är inte:

- Programmen som man ofta använder, kompilatorer editorer mm.



Figur 2.1: En modell av Operativsystemets uppgift

Operativsystemet har som uppgift att göra det möjligt (så enkelt som möjligt) för programmen att använda hårdvaran. Programmen gör det sedan möjligt för oss som användare att indirekt använda hårdvaran.

- Kommandoskalet och kommandoprogrammen (som `ls`) som används för att ge kommandon till operativsystemet hör inte till operativsystemet.

Operativsystemets uppgift är att göra det möjligt för användarprogrammen att kunna använda hårdvaran. Detta innebär att dölja hårdvaran och göra den tillgänglig för programmen via enkla funktioner och rutiner. Vidare skall operativsystemet dirigera och styra allt arbete så att hårdvaran och mjukvaran utnyttjas på ett så effektivt sätt som möjligt. Detta är faktiskt allt vad operativsystemet skall göra. Men om vi köper ett operativsystem så vill vi ju att det även skall kunna tala med oss. Men det är det faktiskt program som gör även om de följer med operativsystemet. En klassisk bild av detta visas i figur 2.1.

2.3 Versionsnummer

Detta med att Linux bara är en kärna gör det lite svårt att hänga med i den lilla djungel av versionsnummer som nämns i samband med Linux. Linux (alltså kärnan) har ett versionsnummer[1] (i skrivande stund är 2.2.7 den senaste). Medan alla distributioner har andra egna versionsnummer (Exempelvis har senaste RedHat nummer 6.0 och Debian 2.1 då detta skrivs).

[1] Kärnans versionsnummer beskrivs i avsnitt 2.4 på sid 39

2.4 Kärnan

Linux är egentligen just bara en kärna. Denna kärna har en hel del egenskaper. Linux är ett äkta 32 bitars operativsystem. Faktiskt det första för Intel baserade persondatorer.

Den första versionen som släpptes var version 0.1. Och den släpptes i maj 1991. Den hade nästan inget hårdvarustöd och använde Minix filsystem. Eftersom Torvalds utvecklade Linux på en Minixdator använde han det operativsystemets filsystem till att börja med. Man behövde alltså Minix på datorn för att kunna köra Linux. Men detta räckte för att programmerare skulle få upp ögonen för det nya operativsystemet.

En av det största händelserna i Linux utveckling kom först tre år senare. I Mars 1994 släpptes Linux 1.0. Nu hade systemet växt på sig ordentligt. Hårdvarustödet var mycket bättre. Nu kunde man använda vissa SCSI kort, CD-rom enheter och diskettenheter. Linux var fortfarande limiterad till att köras på PC-kompatibla datorer. Den största nyheten var nätverksstöd. Linux kunde köra TCP/IP över ethernet. Seriella förbindelser via kabel eller modem stöddes också, både PPP och SLIP. Linux hade också fått ett eget filsystem som var bättre än Minix filsystem. Filsystemet kallades Extended.

Version 2.0 släpptes i juni 1996. Den hade flera förbättringar. Prestandan hade förbättras och fler nätverksprotokoll stöddes. Nu kunde man också på allvar hävda att Linux blivit plattformsoberoende då portningar gjort till flera andra plattformar. Bland annat en äkta 64 bitars version till Alpha.

Då kärnan kontinuerligt utvecklas kommer det nya versioner hela tiden. För att kunna hålla reda på allt detta har man inför ett system. Varje versionsnummer består av tre siffror åtskilda av punkter. De två första talar om vilken version av Linux det är frågan om. Den tredje talar om vilken revision (patchlevel) det är. Eftersom det hela tiden sker utveckling finns det alltid en utvecklingskärna. Denna har alltid en udda siffra som andra siffra i versionsnummret. Alltså var version 2.1.100 en utvecklingskärna som inte kan anses som stabil. Då utvecklarna i serien 2.1 anser att kärnan är stabil byter den namn till 2.2.0 och får en jämn andra siffra. Nästa utvecklingsserie kommer således att få versionsnummer 2.3.x. Vill man vara säker på att man har den säkraste, stabilaste kärnan man kan hitta skall man använda den senaste kärnan i den stabila serien. Då detta skrivs är 2.2.7 den senaste stabila kärnan.

2.5 Moduler

En stor fördel med Linuxkärnan är att den kan ladda i och ur moduler under drift. Detta gör att man kan ändra kärnans funktion utan att behöva ta ner systemet. En modul är en del av kärnan. Den har samma privilegierade läge som kärnan. Det finns alltså inget som en modul inte kan göra. Moduler används för att implementera exempelvis drivrutiner, nätverksprotokoll eller filsystem eftersom dessa kräver den tillgång till hårdvaran som bara kärnan har.

Fördelarna med modulerna är många. Inte minst utvecklarna uppskattar möjligheten att kunna ladda i och ur sina alster på ett fungerande system. Hade de gjort modifieringarna i kärnan så hade de fått kompilera om kela kärnan och boota om maskinen för varje testkörning.

Samma sak kan naturligvis utnyttjas då drivrutinen (eller vad det nu är) skall distribueras. Man behöver bara distribuera modulen som användarna kan ladda i och ur utan att behöva kompilera om sina Linuxkärnor.

Att distribuera drivrutiner som moduler är också ett måste för vissa kommersiella företag som vill skriva drivrutiner till Linux. Om dessa företag skulle vara tvungna att lägga in sina drivrutiner i kärnan så skulle de, p.g.a. GNU GPL[2], vara tvungna att släppa källkoden fri under GPL.

Ytterligare fördel med moduler är att kärnans storlek kan begränsas. Drivrutiner man bara använder ibland kan man ha som moduler och ladda i när man behöver dem och ladda ur då men inte behöver dem.

2.6 Allt är filer

Precis som Unix så är er grundläggande filosofi i Linux att allt är filer. All hårdvara representeras av en fil. Vi har tidigare sett att hårddisken till exempel representeras av en fil (t.ex `/dev/hda` om det är en IDE-disk). De flesta filer i `/dev` katalogen representerar direkt eller indirekt hårdvara.

I vissa böcker har jag sätt något i stil med att “allt är filer utom kataloger”. Detta är dock inte riktigt sant eftersom även kataloger är filer. En katalog är en fil som innehåller namnet på katalogen och de namn som finns i katalogen.

De mest intressanta filerna när man pratar kärna och hårdvara finns i katalogen `/dev` men även katalogen `proc` är intressant. Katalogen `/dev` innehåller filer som representerar hårdvara. Den hårdvaran kan till exempel vara hårddisken som nämns ovan. Andra kända filer i `/dev` är `stdin` som representerar ditt tangentbord, `stdout` som representerar skärmen. Man kan till exempel skicka text till `/dev/stdout` och det visas på skärmen.

Katalogen `proc` är också intressant. Filerna i denna representerar olika saker i ditt operativsystem. Till exempel `/proc/kcore` representerar till internminne. Detta behandlas mer längre fram i boken.

2.7 Mer läsning

[Silberschatz och Galvin, 1998] är en mycket bra bok om operativsystem. Har faktiskt ett helt kapitel om Linux. Är du intresserad av hur ett operativsystem fungerar kan jag verkligen rekommendera denna bok.

[2] Se kapitel 48

Del II

Komma igång

Kapitel 3

Första stegen

Nu har vi talat en del om vad Linux är och hur det fungerar. Jag antar att du är väldigt nyfiken på vad detta system kan göra för dig. I denna del av boken går vi igenom de grundläggande stegen i hur man nyttjar ett Linux-system. Det kan kännas som att det som står här är tråkigt och onödigt, men det är det naturligvis inte (då skulle jag inte ha slösat tid på att skriva det).

Denna del förutsätter att du har tillgång till ett Linuxsystem (en UNIXarbetsstation går naturligtvis lika bra). Har du tänkt att installera Linux på din egen dator så blev du raskt systemadministratör på en gång. Det är inte så krångligt som det låter och dessutom väldigt kul och lärorikt. Hur du installarerar Linux och hur du skapar dig en användare beskrivs med början i kapitel 19 på sidan177.

Det första man gör när man skall använda ett Linuxsystem är att logga in. Alla som skall använda systemet måste därför ha en användarprofil som systemet känner till. Det finns i Linux en superanvändare som kallas för root. Man kan logga in som denna användare i ett system men det skall man inte göra i annat syfte än att skapa sig en ”vanlig” användare.

Har du nu installerat ditt eget system så glömde du väl inta att skapa dig en användare. Alla som använder systemet skall ha en användare (även systemadministratören). Som superanvändaren root skall man i princip *aldrig* köra. Tro mig, jag vet hur det kan gå. Om du bara kan logga in som root skall du innan du läser och testar det som står i detta kapitel läsa (och göra det som står i) “Lägga till användare” på sid 223 Som root kan du göra allt i systemet. Du hindras aldrig av några rättigheter och annat. Detta kan för den ovane (som kommer från Windows) kännas bekvämt. Ovana tenderar att se användandet av en vanlig användare som en begränsning snarare än ett skydd. Att använda en användare skyddar dig inte bara mot att du själv förstör ditt system, utan skyddar även mot virus och annat otyg som kan utnyttja det faktum att du exekverar dina aplikationer som root.

> **Varning!**
> *Kör* **aldrig** *som root om du inte måste!*

Alla exempel i detta kapitel utgår från att du använder skalet `bash`. Sitter du vid en

Linuxmaskin så är det förmodligen `bash` du kör. Använder du ett annat skal kan det i vissa fall bete sig något annorlunda. Du kan lätt hänga med i exemplen i alla fall. Da kan ta reda på vilket skal du kör med kommandot **`echo $SHELL`**. Vill du starta ett `bash`-skal så ger du kommandot **`bash`** så startar det. Detta förutsätter att `bash` finns installerat på maskinen.

Kapitel 4

Logga in och kör!

Okej nu har vi gruvat oss länge nog — Kavla upp ärmarna så kör vi!

4.1 Logga in

Det första du måste göra nu är att logga in. Det är ditt sätt att tala om för systemet vem du är. Detta är nödvändigt eftersom Linux säkerhet bygger på att olika användare får göra olika saker. Du talar alltså om ditt användarnamn och verifierar det med ett lösenord. Antingen är din skärm svart med en loginprompt. Eller så är X (det grafiska gränssnittet) startat och du har en något färggladare bild. I båda fallen ser du ordet Login: eller Namn: Där skriver du ditt användarnamn och trycker på Enter. Lösenordet skriver du där det står Password eller Lösernord följt av Enter, enkelt vad? När du skriver lösenordet visas inga tecken på skärmen, detta är inget fel utan skall vara så. Det är, bland annat, för att ingen skall se över din axel vad du skriver. Hur den grafiska inloggningsrutan ser ut varierar. Man kan nämligen använda flera olika program för detta. Dessutom kan de konfigureras för att se ut på alla möjliga vis med bakgrundsfärger och bilder. Exempel på hu den grafiska inloggningen kan se ut visas i figur 4.1

Okej, en inloggning kan se ut så här:

```
Debian GNU/Linux 2.2 keron tty1

keron login: marcus
Password:
keron:~$
```

Jag skriver mitt användarnamn (marcus) efter login: och mitt lösenord (som varken ni eller jag kan se) efter Password:. I det här fallet hade jag rätt lösenord och loggades in i systemet.

När man har fått fram prompten, som kan se ut som ovan, kan man börja skriva sina kommandon. För att då veta vad klockan är kan man ge kommandot `date`

Figur 4.1: Inloggning med `xdm`

`xdm` är ett inloggningssystem som tar dig direkt in i X. Det har även andra fördelar som behandlas längre fram i denna bok. Observera att du byter fält med Enter *och inte* tab *som i vissa andra system.*

```
keron:~/work/linuxboken$ date
Tue Mar 16 20:19:39 CET 1999
keron:~/work/linuxboken$
```

4.1.1 Logga ut

Skulle du nu av någon anledning vilja logga ut innan vi kommit till den delen i boken kan du göra det genom att ge kommandot

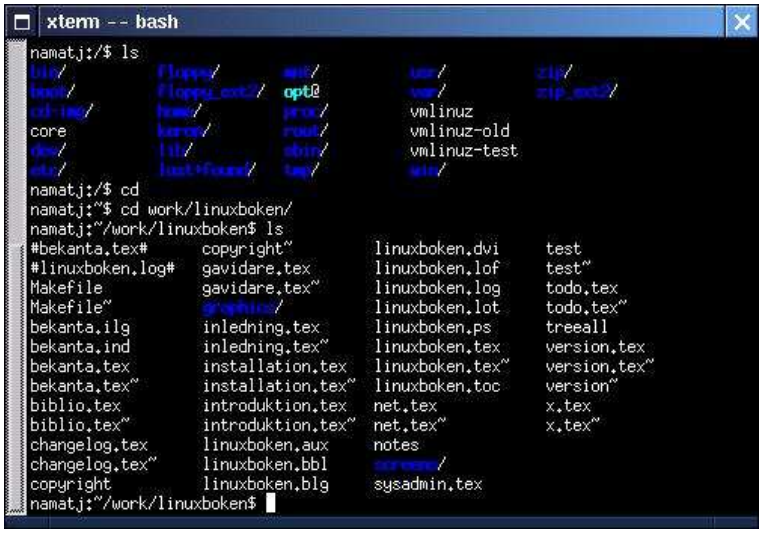
```
exit
```

. Som vanligt trycker du Enter efter kommandot.

4.2 Kommandon i X, terminalfönster

Nu när vi loggat in tänkte jag att vi skulle börja med några enkla kommandon vid kommandoprompten.

Om du när du loggar in kommer direkt in i ett grafiskt gränssnitt har du kanske inte direkt någon prompt att ge kommandon vid. Du måste då starta ett terminalfönster. Det vanligaste terminalemuleringsprogrammet är `xterm`. Detta eller ett annat terminalfönster brukar starta automatiskt så X startas Om detta inte sker kan du starta



Figur 4.2: `xterm`-fönster i X

Om du alltid kör i X behöver du starta ett terminalemuleringsprogram för att kunna ge kommandon till systemet. Kommandofönstret fungerar lika som om du satt vid en skärm utan grafik. Gör dig hemmastadd i detta program. Du kommer att använda det mycket . . .

det från någon av de menyer som finns. Hittar du inga menyer kan du prova att klicka med de olika musknapparna på rotfönstret (rotfönstret är det fönster som är som bakgrund i fönstersystemet, kallas skrivbord i Windows). Du kan prova att klicka med alla musknapparna i rotfönstret. Oftast för man olika menyer för de olika musknapparna. Terminalfönstret heter ofta något med `xterm` `shell` eller `command`. Ett trevligt terminalfönster som jag själv använder ibland heter **`rxvt`** och finns kanske installerat på din maskin. Leta bland menyerna så hittar du säker ett terminalprogram. Har du en mus med bara två knappar så trycker du in båda två samtidigt för att simulera en tredje knapp[1]. I terminalfönster ger du sedan alla kommandon som du vill köra. Du startar program, hanterar filer med mera från ditt terminalfönster. Är du van att använda Windows kan det upplevas som lite krångligt att jobba med en kommandorad. Men efter att du jobbat i Linux ett tag kommer du nog att tycka att det är väldigt jobbigt att jobba helt i grafiska gränssnitt. Figur 4.2 visar ett xtermfönster.

4.2.1 Logga ut ur X

Nu när du loggat in i X och börjat se dig omkring kanske du vill logga ut igen. Hur man gör varierar mellan olika fönsterhanterare. Titta i de menyer du hittar så skall det finnas ett alternativ som heter ”exit”, ”logout” eller något liknande.

[1] Fungerar inte detta hos dig så kan du läsa i kapitlet om X i systemadministrationsdelen hur man får det att fungera så.

DOS-Kommando	Linux-kommando	kommentar
cd	cd	Fungerar lika, obs “cd ..”
dir	ls	Ganska lika
copy	cp	Ganska lika
move	mv	
ren	mv	
attrib	chmod	Skiljer
del	rm	lika
deltree	rm -r	ganska lika
type	cat, less, more	Bättre
print	lpr	
help	man	Inte helt olikt
md	mkdir	Ganska lika

Tabell 4.1: DOS-kommandon och deras motsvarighet i Linux

Här listas kommandon som du kanske känner från DOS. De kommer att behandlas mer ingående längre fram i denna bok.

4.3 Grundläggande kommandon

Här presenteras några av de mest grundläggande kommandona[2]. Du som jobbat mycket i MS-DOS eller i Windows dosprompt kommer att känna igen dessa kommandon. Men tro inte att du är tillbaka i MS-DOSbara för det. I Linux finns massor med saker att göra som inte går i MS-DOS.

Låt oss gämföra några MS-DOS-kommanon och deras motsvarigheter i Linux.

4.3.1 MS-DOS-kommandon och deras motsvarighet

Om du är van att jobba i MS-DOS och vill komma igång snabbt med Linux rekommenderar jag att du läser dokumentet DOS-Win-to-Linux-HOWTO. Se vidare om detta i Källförteckningen. I tabell 4.1 presenteras en kort lista över de vanligaste DOS-kommandona och deras Linuxmotsvarighet. Idén till denna kommer från just DOS-Win-to-Linux-HOWTO, tack Guido. För dig som inte är van vid DOS eller har så brått kommer jag att mer ingående behandla kommandona längre fram. Så du kan lugnt skumma förbi denna tabell.

Vi kan testa redan nu att ge kommandot

```
ls
```

vid prompten. Det gör man genom att skriva ”ls” och trycka på enter.

[2] Vissa gör skillnad på kommandon och program. Det tycker jag är onödigt och blandar orden om var andra hela tiden.

4.4 Be systemet om hjälp

När du sitter vid ett Linuxsystem och undrar över något kan du få hjälp av systemet. Det finns flera olika sätt på vilka systemet kan hjälpa dig. De hjölpsystem som finns är också en mycket bra källa till kunskap.

4.4.1 Manualsidorna – Kommandot `man`

I de flesta (alla) Linux och UNIXsystem finns det något som heter manualsidorna (en. manpages). Dessa beskriver olika kommandon eller systemanrop. Vill du ha hjälp om ett kommando kan du anropa dess manualsida med kommandot **`man`** och kommandot som argument[3]. Vill du till exempel ha fram manualsidan för kommandot **`ls`** (list) ger du kommandot:

```
man ls
```

Manualsidorna kan tyckas svåra att förstå till en början. Men ju mer man läser dem desto lättare blir de att förstå. Alla är strukturerade på ungefär samma sätt. Har du tillgång till ett Linux eller UNIXsystem kan du titta på manualsidan för `ls` nu:

[3] Ett argument är något som skickas till programmet för att ge data till det. Ett argument är inte sällan ett fi lnamn. Argumenten åtskiljs från kommandot med mellanslag

```
keron:~$ man ls

LS(1)                                                       LS(1)

NAME
       ls, dir, vdir - list contents of directories

SYNOPSIS
       ls  [-abcdfgiklmnpqrstuxABCFGLNQRSUX1] [-w cols] [-T cols]
       [-I pattern] [--all]  [--escape]  [--directory]  [--inode]
       [--kilobytes]  [--numeric-uid-gid]  [--no-group]  [--hide-
       control-chars] [--reverse] [--size] [--width=cols] [--tab-
       size=cols]  [--almost-all] [--ignore-backups] [--classify]
       [--file-type] [--full-time] [--ignore=pattern] [--derefer-
       ence]     [--literal]     [--quote-name]     [--recursive]
       [--sort={none,time,size,extension}]   [--format={long,ver-
       bose,commas,across,vertical,single-column}]
       [--time={atime,access,use,ctime,status}] [--help]  [--ver-
       sion]  [--color[={yes,no,tty}]]  [--colour[={yes,no,tty}]]
       [name...]

DESCRIPTION
       This documentation is no longer being maintained  and  may
       be inaccurate or incomplete.  The Texinfo documentation is
       now the authoritative source.

       This manual page documents the GNU version of ls.  dir and
       vdir are versions of ls with different default output for-
       mats.  These programs list each given  file  or  directory
       name.   Directory contents are sorted alphabetically.  For
       ls, files are by default listed in columns, sorted  verti-
       cally,  if  the  standard  output is a terminal; otherwise
       they are listed one per  line.   For  dir,  files  are  by
       default  listed  in columns, sorted vertically.  For vdir,
       files are by default listed in long format.

   OPTIONS
       -a, --all
              List all files in directories, including all  files
              that start with '.'.

       -b, --escape
              Quote  nongraphic  characters  in  file names using
```

Vad kommandot man egentligen gör är att först formatera manualsidan med ett formateringssystem. Sedan skickas utdatat till ett program som gör det lätt för dig att läsa manualsidan, till exempel **`less`**, **`more`** eller **`xman`**. Det vanligaste i moderna Linuxdistributioner är **`less`**. Hur du använder **`less`** beskrivs mer ingående på sidan 106. Men du behöver inte läsa det nu. Du kan scrolla upp och med i manualsidan med piltangenterna. Står det `--More--` längst med på sidan så använder du **`more`**. Det står mer om `more` på sidan 105. Använd Enter för att gå nedår i filen och b för att backa.

Vill du ha hjälp om hur man fungerar ger du naturligtvis kommandot

```
keron:~$ man man
```

Mer om man

Manualsidorna är uppdelade i (minst) åtta olika avsnitt. Detta behöver man oftast inte tänka på. Men ibland finns ett kommando eller anrop i flera delar av manualen, då

måste man specificera vilket man vill ha. De åtta olika delarna är

1. Användarkommandon
2. Systemanrop
3. Subrutiner
4. Enhetsfiler
5. Filformat (för filer oftast i /etc)
6. Spel
7. Makropaket och Bibliotek
8. Kommandon för systemadministration

För att specificera i vilken del en manualsida finns använder du följande syntax:

```
man <del> kommando
```

[4].

exempelvis så listar:

```
$ man crontab
```

Manualsidan för crontab i del 1 av manualsidorna. Denna sida brukar skrivas som crontab(1). kommandot

```
$ man 5 crontab
```

visar istället manualen för filformatatet för crontabfiler. Denna manualsida kallas således crontab(5).

Om man inte anger vilken del av malualen en sida ligger i och en sida med samma namn finns i flera delar av manualen får man fram den sida som ligger först, det vill säga i den delen som har lägst nummer. I varje del finns det en manualsida som heter *intro*. Vill du veta vad som står i en viss del av manualen (det kan variera från plats till plats) kan du alltså ge kommandot **`man <del> intro`**.

Vet du inte säker vad kommandot du vill ha hjälp om kan du anropa programmet man med flaggan `-k`. K står för *Keyword* och tillåter således sökning med hjälp av nyckelord. Om jag vill ha hjälp om ett kommando som spelar upp en MPEG fil kan jag ge kommandot **`man -k mpeg`**. Observera att denna sökning kan ta lång tid. Studera följande exempel:

[4] Vissa versioner av man använder syntaxen **`man -S <del> kommando`** eller **`man -s <del> kommando`**

```
keron:~$ man -k mpeg
mpg123 (1)           - play audio MPEG 1.0/2.0 file (layers 1, 2 and 3)
mtv (1)              - MpegTV MPEG-1 Video player with audio/sync for X Window
(END)  <- Här trycker jag på q för att avluta och komma tillbaka till promten.
```

Här ser jag att `mtv` verkar vara ett bra program för att spela upp filen. Se mer information om detta i avsnittet 7.4 på sidan 108.

4.4.2 `xman`

Tycker man om att jobba i X (eller inte gillar textbaserade program) så finns det ett (eller det finns flera men jag tittar på ett) program för att läsa manualsidor i X. Programmet har det fantasifulla namnet **`xman`** och du kan starta det från ett terminalfönster eller via någon meny i din fönsterhanterare.

Då **`xman`** startar ser du bara ett litet fönster med tre knappar. Välj "*Manual Page*" för att komma till manualsidorna. För att visa manualsidan för ett program kan du nu göra på några olika vis. Antingen använder du Sökverktyget `Search` under *Options*-menyn. Där kan du söka både på programnamn och på söksträngar (man och man -k, ovan). Du kan också välja en del av manualsidorna och se en listning på alla program som finns i den genom att välja den delen under *Sections*-menyn. För att sedan titta på en manualsida klickar du bara på den. Figur 4.3 visar manualsidan för `ls` i just `xman`.

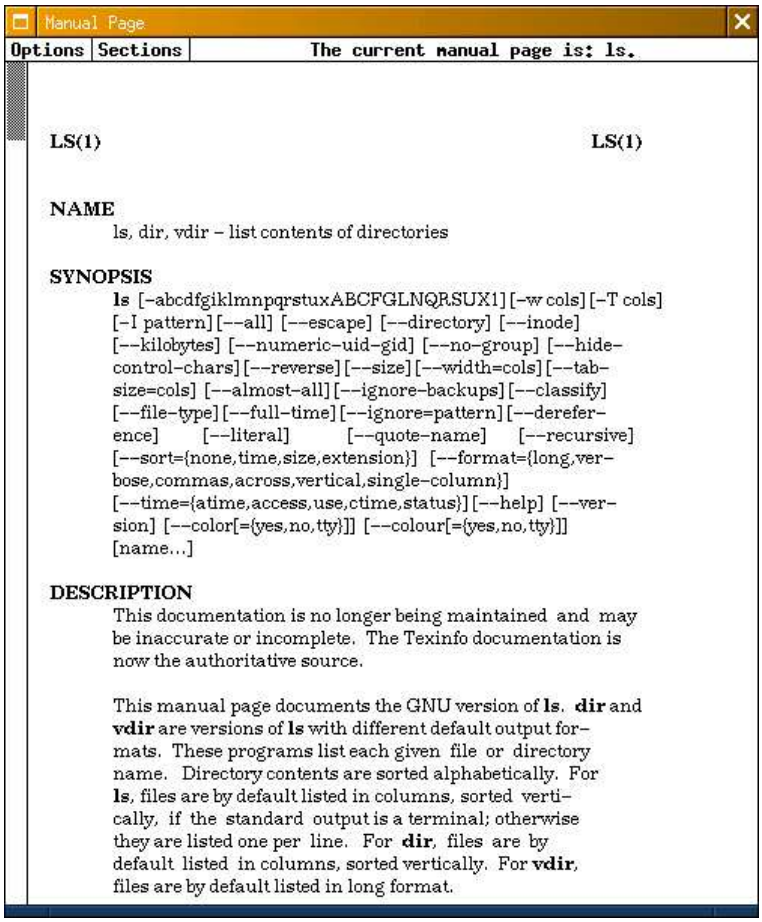
4.4.3 GNUs infosystem – `info`

Den uppmärksamme läsaren (som verkligen lästa manualsidan) såg att manualsidan för `ls` var gammal och en hänvisning till någonting som kallas Texinfo. Texinfo, eller bara info som det kallas är ett nyare sätt att publicera manualer. Vissa gillar det, andra gör det inte. Fördelan är att info är ett mer modernt system med en rad fördelar. En stor nackdel är dock att manualsidorna har funnits med så länge och är lite av en standard. Jobbigt att ha två system.

`Info` tillhandahåller betydligt fler finesser än vad `man` gör. Till exempel kan man ha hyperlänkar mellan olika sidor. Info-sidorna kan man läsa med kommandot **`info`** eller använda Emacs. Emacs är egentligen en editor, men kan användas till det mesta. Bild 4.4 visar infosidan för ls i ett xtermfönster.

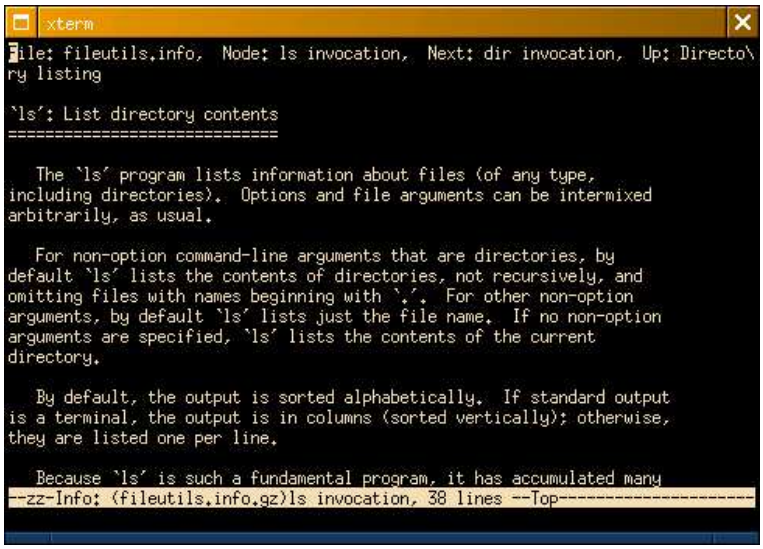
Det är smidigare och enklare att använda Info i Emacs. Man kommer åt Info genom att i Emacs hjälpmeny välja Manuals->Browse Manuals with Info eller genom att trycka Ctrl+h och sedan i. Bild 4.5 visar Infosidan för ls i GNU Emacs.

Hur info fungerar tänker jag inte gå in på här. I varje fall inte i denna version av denna bok. Kanske jag gör det senare. För att lära dig Info startar du info med kommandot `info info`. Då startar en guide som enkelt lär dig använda programmet.



Figur 4.3: xman

`xman` är ett program för att titta på manualsidor i X. Här visas manualsidan för `ls`.

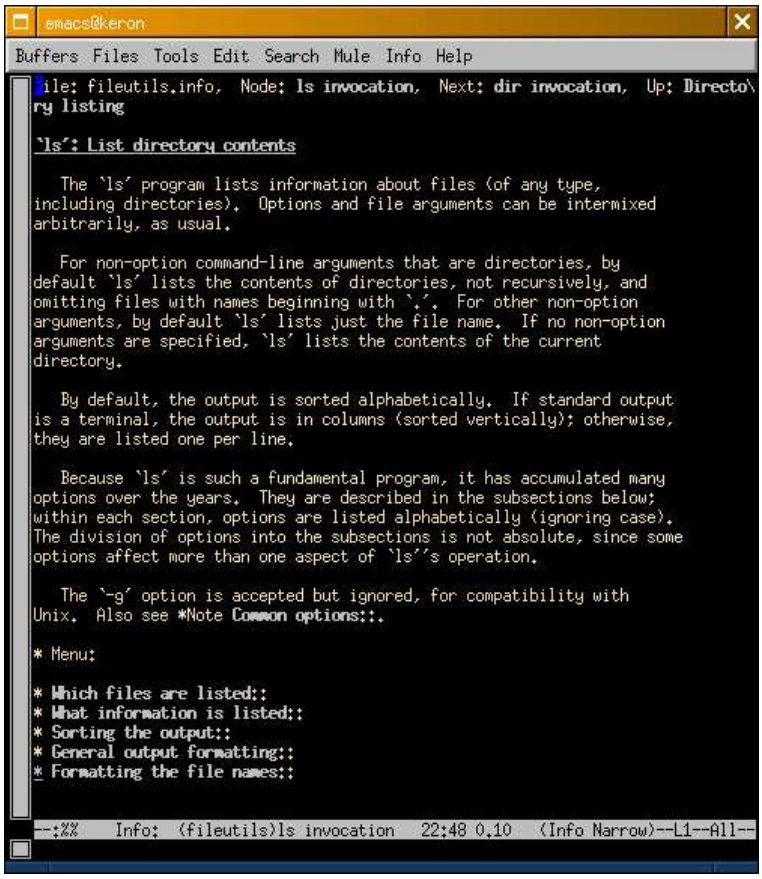


Figur 4.4: GNUs Infosystem, Texinfo

Infosidan för ls visas i GNUs Infosystem Texinfo. Man kan orientera sig med musen eller med hjälp av kommandon som man ger på kommandoraden längs ned i programmet.

4.4.4 Fråga programmen direkt

Flera Linuxprogram har inprogrammerad hjälp. Du når den oftast genom att ange flaggan -h eller --help till programmet. Exempel:



Figur 4.5: GNUs Infosystem, Texinfo i Emacs

Infosidan för ls visas i GNUs Infosystem Texinfo i Emacs. Man kan orientera sig med musen eller med hjälp av kommandon som man ger på kommandoraden längs ned i programmet.

```
keron:~$ ls -h
ls: invalid option -- h
Try ‘ls --help’ for more information.
keron:~$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuSUX nor --sort.

  -a, --all                  do not hide entries starting with .
  -A, --almost-all           do not list implied . and ..
  -b, --escape               print octal escapes for nongraphic characters
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         sort by change time; with -l: show ctime

  [Här har jag klippt bort lite för att spara lite utrymme. Kolla i ditt
   system om du är nyfiken på vad som försvunnit...]

  -X                         sort alphabetically by entry extension
  -1                         list one file per line
      --help                 display this help and exit
      --version              output version information and exit

By default, color is not used to distinguish types of files.  That is
equivalent to using --color=none.  Using the --color option without the
optional WHEN argument is equivalent to using --color=always.  With
--color=auto, color codes are output only if standard output is connected
to a terminal (tty).

Report bugs to fileutils-bugs@gnu.ai.mit.edu
keron:~$
```

I exemplet ovan ser vi att **`ls`** ger hjälp om man startar det med flaggan[5] **`--help`** (Du behöver inte läsa hjälpen nu, vi skall behandla `ls` senare).

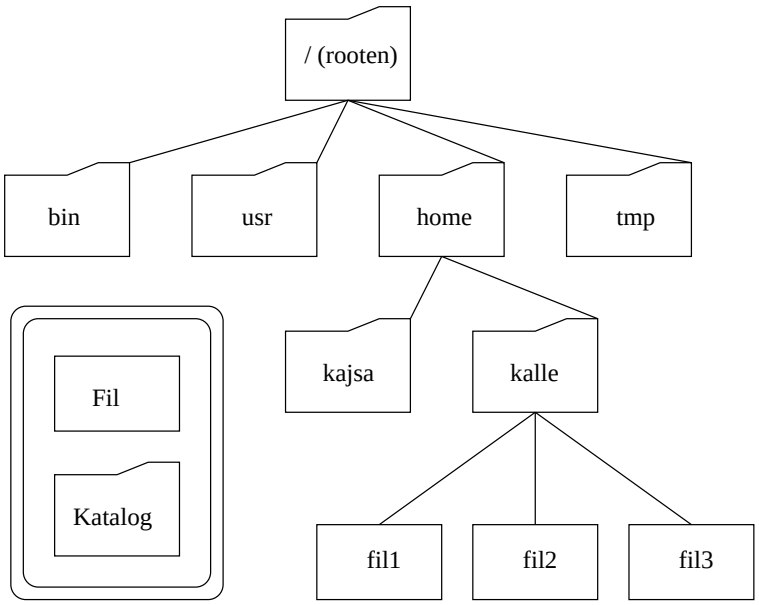
4.5 Filträdet

Filträdet i Linux är uppbyggt som ett träd. Trädet består ev en mängd filer. De två viktigaste filtyperna nu är *vanliga filer* och *katalogfiler*. En katalogfil innehåller i sin tur andra filer och blir på så sätt förgreningar i trädet. Trädet är hierarkiskt med roten som bas. Roten är således en katalog. Studera filsystemet i Figur 4.6.

4.5.1 Filnamn

En fil i Linux kan innehålla hur många tecken som helst (åtminstone teoretiskt, vissa program inför dock begränsningar) men man bör undvika fler än 255 tecken i ett filnamn. Ett filnamn kan dessutom bestå av vilka tecken som helst. Men man skall undvika följande tecken eftersom de ofta har andra betydelser: `(blankslag) | < > & ~ # ? ^ @ $ ’ ‘ ! * { } [ ] ;` , vidare skall man undvika de svenska tecknen å, ä och ö eftersom de inte fungerar överallt. Filnamnen i Linux måste inte (som i MS-DOS) ha ett suffix. Men det är ofta bra att använda det ändå för att hålla reda på vad för typ av fil det är fråga om. Exempel på suffix som används av konvention.

[5] en flagga är något som skickas till programet för att styra dess funktion. Oftast är de på formen ett streck och en bokstav eller två streck och ett ord. En flagga kallas ibland även för *växel* eller *option*.



Figur 4.6: Exempel på filträd

Exempel på ett filträd med rotkatalogen överst.

.c Källkod till ett C-program

.cpp Källkod till ett C++ program

.ps PostScript fil

.dvi Device Independent file (Fil för utskrift eller granskning på skärmen)

.gz Fil komprimerad med programmet `gzip`

.tar Filer sammanslagna med programmet `tar`

.tar.gz eller .tgz Filer sammanslagna med `tar` och sedan komprimerade med `gzip`

Om man har en fil som man inte vet vad den är av för typ så finns det ett program för det också. `File` är ett program som är till för just det. Studera följande lilla exempel:

```
$ file 13227.pdf
13227.pdf: PDF document, version 1.2
$ mv 13237.pdf Anonym_fil
$ file Anonym_fil
Anonym_fil: PDF document, version 1.2
$
```

I exemplet så ser vi att man får reda på vad det är för fil även om filsuffixet har försvunnit. `File` känner igen en mängd olika filformat. Så vet inte du vilken sorts fil det är frågan om så finns det en stor chans att `file` gör det.

Man kan ange filnamn på två olika sätt, *absolut* eller *relativt*. Ett absolut filnamn börjar alltid från roten. Ett absolut filnamn kan se ut så här:

```
/home/kalle/fil1
```

Det är det absoluta namnet på filen `fil1` som ligger i katalogen `/home/kalle`. Oavsett vilken katalog jag har som aktuell katalog så kommer ett absolut filnamn alltid att namnge samma fil.

Om jag har `/home/kalle` som aktuell katalog så kan jag istället ange filnamnet relativt. Då blir filnamnet `fil1` Om jag anger det filnamnet och har `/home` som aktuell katalog menar jag en helt annan fil, nämligen `/home/fil1` (som inte finns).

4.5.2 Vandra runt i katalogstrukturen

För att vandra runt i detta filträd används kommandon som är väldigt snarlika de du kanske är van från MS-DOS.

`pwd` **"Print Working Directory"**

Den katalog som för tillfälligt är aktiv. Det vill säga den katalog du "står i". Visas med kommandot `pwd`. Om du är vilse i katalogstrukturen ger du alltså kommandot `pwd`.

```
keron:~$ pwd
/home/marcus
keron:~$
```

Vandra runt i katalogstrukturen, `cd`

Med kommandot **`cd`** byter du aktiv katalog. Syntaxen är väldigt enkel. **`cd kalle`** byter aktuell katalog till `kalle`. Observera att `kalle` i detta fall måste vara in underkatalog till den katalog som är aktiv för att det skall fungera. Vi angav ju `kalle` som ett relativt filnamn(Börjar inte på `/`).

```
keron:~$ pwd
/home/marcus
keron:~$ cd work/
keron:~/work$ pwd
/home/marcus/work
keron:~/work$ cd ..
keron:~$ pwd
/home/marcus
keron:~$
```

I exemplet ser du tecknet ~ (tilde). Det representerar i de flesta skal din hemmakatalog. Skriver du bara **`cd`** utan argument hamnar du direkt i din hemmakatalog. Detta kan vara lämpligt om du är vilse i strukturen. Hemmakatalogen brukar på de flesta system vara `/home/användarnamn` men kan vara något helt annat. Tilde-tecknet (~) kan

även användas för att ange andras hemmakataloger. Vill du till exempel titta i Kalles hemmakatalog[6] anger du kommandot:

```
ls ~kalle
```

`~kalle` motsvarar alltså Kalles hemmakatalog.

I varje katalog finns det två stycken speciella filer. De heter ‘.’ och ‘..’ De är också kataloger. Den enkla punkten representerar den nuvarande katalogen och de dubbla punkterna representerar den katalog som är förälder[7] till den katalog som är aktuell. Nu kan vi lära oss hur man egentligen anger en fil i den aktuella katalogen relativt. Filen `fil1` i katalogen `kalle` namnges `./fil1` om `kalle` är aktuell katalog. De flesta program utgår från att bara ett filnamn avser en fil i aktuell katalog. Det är därför oftast inte nödvändigt att ange filnamnet med `./` i början. Men lägg det på minnet att det egentligen skall vara så. Den dubbla punkten används till exempel med kommandot **`cd`**. Till exempel gör kommandot **`cd ..`** att vi flyttar upp oss ett steg i hierarkin och **`cd .`** gör att vi flyttar oss till samma katalog som vi var i. Studera följande exempel:

```
keron:/$ tree                 <- tree visar filstrukturen som ett träd
.
|-- bin
|-- home
|   |-- kajsa
|   `-- kalle
|       |-- fil1
|       |-- fil2
|       `-- fil3
|-- tmp
`-- usr

6 directories, 3 files
keron:/$ cd home/             <- Byt katalog till /home
keron:/home$ pwd              <- Var är jag?
/home                         <- Jo! i /home
keron:/home$ cd kalle         <- Gå till kalle
keron:/home/kalle$ ls         <- Vad finns i katalogen kalle
fil1  fil2  fil3
keron:/home/kalle$ cd ..      <- Gå upp ett steg
keron:/home$ ls               <- Vad finns här? Bland annat kalle, så klart!
kajsa/  kalle/
keron:/home$ cd kalle         <- Tillbaka till kalle
keron:/home/kalle$ cd ../../  <- Gå till "katalogen ovanför katalogen ovanför"
keron:/$ ls                   <- Vad finns här?
bin/   home/  tmp/   usr/
keron:/$ cd usr               <- Gå ner i usr
keron:/usr$ cd ../home/kalle/ <- Gå till home/kalle i katalogen ovanför
keron:/home/kalle$ ls         <- Vad finns här?
fil1  fil2  fil3
keron:/home/kalle$ cd /       <- Gå till /
keron:/$
```

Ovanstående exempel använder kommandon som du kanske inte känner till ännu. Men hav tålamod. Det kommer längre ned i detta kapitel . . .

Man kan även ge kommandot **`cd -`** då “ångrar” man det senaste **`cd`**-kommandot, se följande lilla exempel.

[6] Enligt datalagen måste du ha tillstånd av Kalle för att göra det.
[7] i roten är “.” och “..” samma sak.

```
keron:~$ cd /                              <- Gå till roten '/'
keron:/$ cd usr/local/                     <- Byt aktuell katalog
keron:/usr/local$ pwd                      <- Var är jag?
/usr/local
keron:/usr/local$ cd -                     <- Ångra senaste cd-kommandot
keron:/$ pwd                               <- Var är jag nu?
/
keron:/$ cd usr/local/bin/                 <- Byt aktuell katalog
keron:/usr/local/bin$ pwd                  <- Var är jag nu?
/usr/local/bin
keron:/usr/local/bin$ cd ../../../var/log/ <- Byt aktuell katalog
keron:/var/log$ pwd                        <- Var är jag nu?
/var/log
keron:/var/log$ cd -                       <- Ångra senaste cd-kommandot
keron:/usr/local/bin$ pwd                  <- Var är jag nu?
/usr/local/bin
keron:/usr/local/bin$ cd                   <- Gör hemmakatalogen till aktuell
keron:~$
```

4.5.3 Lista innehållet i en katalog, `ls`

Kommandot **`ls`** används för att lista innehållet i en katalog. Om du anger kommandot helt utan argument listas filerna och katalogerna i aktuell katalog. Du kan även lista filerna i en annan katalog genom att ange denna katalog som argument till **`ls`**.

```
keron:~/screens$ ls
cd          date        login       prompt      pwd         rootprompt
keron:~/screens$ ls ../graphics/
filesys.eps      filtrad.fig.bak  unnamed.fig
filtrad.fig      unnamed.eps      unnamed.fig.bak
keron:~/screens$
```

Ett liknande kommando är **`tree`** som listar, inte bara innehållet i aktuell katalog, utan även alla kataloger under denna. Resultatet presenteras som ett grafiskt träd. Passa dig för att köra detta från roten eftersom det då ritar upp hela ditt filsystem. Det kan ta lite tid. Om du startar ett sådant kommando som tar lång tid kan du oftast avbryta det genom att trycka Ctrl+c. **`Tree`** finns inte alltid på alla installationer.

4.6 Skapa nya kataloger, `mkdir`

För att skapa en ny katalog använder du kommandot **`mkdir`**. **`mkdir`** skapar en ny katalog med det namn du anger som argument till kommandot.

Kommandot tar bara några flaggor. Den viktigaste är **`-p`** som skapar de underkataloger som behövs.

`mkdir` kan bara skapa kataloger i en nivå i taget. Det går alltså inte att skapa katalogen 'katalog/katalog1' om inte 'katalog' redan finns (utan att anvönda -p). Det är dock möjligt att skapa flera kataloger genom att ange dem som argument till programmet. Det går att skapa katalogerna som nämndes ovan om man ger kommandot **`mkdir katalog katalog/katalog1`**. Detta visas med några exempel:

```
$ mkdir kalle                     <- Skapa katalogen kalle
$ mkdir knatte tjatte fnatte      <- Skapa tre kataloger i ett svep
$ ls                              <- Titta så att det stämmer
fnatte/  kalle/   knatte/  tjatte/
$ mkdir kalle/kajsa               <- Skapa en underkatalog till kalle
$ mkdir alexander/lukas           <- Prova att kapa två kataloger i ett svep
mkdir: cannot make directory ‘alexander/lukas’: No such file or directory
$ mkdir alexander alexander/lukas <- Prova detta vis i stället
$ tree                            <- Visa strukturen med ’tree’
.
|-- alexander
|   ‘-- lukas
|-- fnatte
|-- kalle
|   ‘-- kajsa
|-- knatte
‘-- tjatte

7 directories, 0 files
$
```

4.7 Filnamnsmönster

Kommandot **ls** som vi kort behandlade ovan kan inte bara lista alla filer i en katalog, utan kan även lista bara de filer som skickas som argument till det. Detta kanske verkar onödigt vid en första anblick, men är mycket användbart. **ls** behandlas mer ingående i avsnitt 4.11 på sidan 76. Nu skall vi använda det för att beskriva hur man använder filnamnsmönster. Att lära sig dessa kan spara mycket arbete.

Först studerar vi hur ls fungerar i nästa exempel:

```
keron:~$ ls            <- Visa filerna i katalogen
Brev     Brev3    abcdefg  cba      fil11    fil2     fil5     fil8
Brev13   aKalleg  brev10   fil1     fil12    fil3     fil6     fil9
Brev2    abc      brev13   fil10    fil13    fil4     fil7
keron:~$ ls fil1 fil2  <- Visa filerna fil1 och fil2
fil1  fil2
keron:~$ ls cba        <- Visa filen cba
cba
keron:~$
```

Antag nu att vi vill visa filerna alla filer i exemplet ovan som börjar på “fil”. Då måste jag ju skriva **ls** och sedan en lista på alla filer. Detta verkar ju jobbigt och naturligtvis behöver man inte göra så. Man använder sig av något som kallas *jokertecken* (eng. wildcards). De vanligaste jokertecknen är * och ?. En * representerar ett godtyckligt antal (inklusive noll) godtyckliga tecken. Ett ? representerar exakt ett godtyckligt tecken. Istället för filnamnsmönstret skickar skalet en lista med alla filer som passar. Detta kallas för filnamnssubstitution. Programmet, i det här fallet **ls** vet inte att vi använt ett filnamnsmönster, det ser bara en lista med filer. Att lista alla filer ovan som börjar på fil kan man således göra med kommandot **ls fil***. Observera att **ls fil?** inte listar alla. Inte heller **ls fil??**. Ett exempel kan vara på sin plats:

```
keron:~$ ls       <- Visa filer
Brev     Brev3    abcdefg  cba      fil11    fil2     fil5     fil8
Brev13   aKalleg  brev10   fil1     fil12    fil3     fil6     fil9
Brev2    abc      brev13   fil10    fil13    fil4     fil7
keron:~$ ls fil*  <- Visa alla filer som börjar på fil
fil1   fil11  fil13  fil3   fil5   fil7   fil9
fil10  fil12  fil2   fil4   fil6   fil8
keron:~$ ls fil?  <- Visa alla filer som börjar på fil och har fyra tecken
fil1  fil2  fil3  fil4  fil5  fil6  fil7  fil8  fil9
keron:~$ ls fil?? <- Visa alla filer som börjar på fil och har fem tecken
fil10  fil11  fil12  fil13
keron:~$
```

Förmodligen förstår du nu hur dessa två fungerar. Men att öva på det skadar inte. Nämnas bör att bara en * representerar alla filer. Du som är van DOS vet att där representerar *.* alla filer. Det beror på att punkten i DOS har en speciell betydelse. I Unix betyder *.* “alla filer som innehåller en punkt”.

Tips!
I UNIX betyder inte ****.**** *alla filer i en katalog. Det betyder alla filer i en katalog som innehåller en punkt (.).*

Vi kör mer exempel:

```
keron:~$ ls       <- Visa filer
Brev     Brev3    abcdefg  cba      fil11    fil2     fil5     fil8
Brev13   aKalleg  brev10   fil1     fil12    fil3     fil6     fil9
Brev2    abc      brev13   fil10    fil13    fil4     fil7
keron:~$ ls *b*   <- Visa alla filer som innehåller bokstaven 'b'
abc      abcdefg  brev10   brev13   cba
keron:~$ ls f*2   <- Visa alla filer som börjar på 'f' och slutar på '2'
fil12  fil2
keron:~$ ls f??2  <- Som ovan fast filen måste ha exakt 4 tecken
fil2
keron:~$ ls *     <- Visa alla filer
Brev     Brev3    abcdefg  cba      fil11    fil2     fil5     fil8
Brev13   aKalleg  brev10   fil1     fil12    fil3     fil6     fil9
Brev2    abc      brev13   fil10    fil13    fil4     fil7
keron:~$ ls *.*   <- Visa alla filer som innehåller en punkt
ls: *.*: No such file or directory
keron:~$ ls *B*   <- Visa alla filer som innehåller bokstaven 'B'
Brev    Brev13  Brev2   Brev3
keron:~$ ls *B*3  <- Alla filer som innehåller 'B' och sklutar på '3'
Brev13  Brev3
keron:~$ ls *B*3* <- Alla filer som innehåller 'B' och '3'
Brev13  Brev3
keron:~$ ls *B*1* <- Alla filer som innehåller 'B' och '1'
Brev13
keron:~$
```

Observera att Linux inte behandlar stora och små bokstäver lika. Filerna `kalle` och `Kalle` är således två helt skilda filer. DOS användare får också vänja sig vid att punkten inte är ett speciellt tecken utan behandlas som vilket som helst.

Okej. Anta nu att vi vill lista alla filer som börjar på ’B’eller ’b’. Hur kan vi göra det? Jo då använder vi en tredje typ av joker tecken, nämligen hakparenteserna. [och]. [Bb] matchar mot ett tecken som antingen är ’B’ eller ’b’. Ytterligare ett exempel kan vara på sin plats.

```
keron:~$ ls        <- Vilka filer finns det
176-167          Alexander        Kajsa            Knatte
176-671          Fnatte           Kalle            Oppfinnar-Jocke
176-761          Joakim           Karlsson         Tjatte
keron:$ ls *[tT]*  <- Lista alla med stort eller litet 't' i filnamnet
Fnatte  Knatte  Tjatte
keron:~$ ls *[aA]* <- Lista alla med 'a' eller 'A'
Alexander        Kajsa            Knatte
Fnatte           Kalle            Oppfinnar-Jocke
Joakim           Karlsson         Tjatte
keron:~$ ls *[tf]* <- Lista alla med 't' eller 'f' i filnamnet
Fnatte           Knatte           Oppfinnar-Jocke  Tjatte
keron:~$ ls [KO]*  <- Alla som börjar med 'K' eller 'O'
Kajsa            Karlsson         Oppfinnar-Jocke
Kalle            Knatte
keron:~$ ls *[e]   <- Alla som slutar på 'e'
Fnatte           Knatte           Tjatte
Kalle            Oppfinnar-Jocke
keron:~$
```

Nu kanske man vill lista alla filer som börjar på en stor bokstav. Det blir lite jobbigt att skriva hela alfabetet versalt inom hakparenteser. Det mesta som kräver mycket skrivande går att lösa på enklare sätt. Även detta. Inom hakparenteserna kan man även ange intervall. Till exempel så passar [A-Z] mot det versala alfabetet. Nu kan man undra varför jag inte tog med ÅÖÄ i exemplet ovan. Det var naturligtvis ingen slump. För det första bör man ej använda ÅÄÖ i filnamn. För det andra så fungerar intervallen så att de representeras av intervall av nummer i ASCII-tabellen. I ASCII-tabellen kommer inte ÅÄÖ efter Z (de kommer inte ens i rätt ordning). Hur du får med de svenska tecknen för du snart reda på. Men först ett exempel som visar enkla intervaller:

```
keron:~$ ls        <- Vilka filer finns
176-167          Alexander        Kajsa            Knatte
176-671          Fnatte           Kalle            Oppfinnar-Jocke
176-761          Joakim           Karlsson         Tjatte
keron:~$ ls [A-J]* <- Lista alla som börjar på A till J
Alexander  Fnatte     Joakim
keron:~$ ls [A-Z]* <- Lista alla som börjar på A-Z
Alexander        Kajsa            Knatte
Fnatte           Kalle            Oppfinnar-Jocke
Joakim           Karlsson         Tjatte
keron:~$ ls *[A-J]* <- Alla som i namnet har en bokstav i intervallet A-J
Alexander        Fnatte           Joakim           Oppfinnar-Jocke
keron:~$ ls *[a-z]* <- Alla som i namnet har en liten bokstav a-z
Alexander        Kajsa            Knatte
Fnatte           Kalle            Oppfinnar-Jocke
Joakim           Karlsson         Tjatte
keron:~$ ls *[0-9]* <- Alla som i namnet har en siffra
176-167  176-671  176-761
keron:~$
```

Nu vet vi hur man matchar enstaka tecken och hur man matchar intervaller. Då vill vi ju kunna kombinera dessa. Enkelt! Följande matchar alla versala bokstäver **`[A-ZÅÄÖ]`** och följande matchar alla bokstäver, stora som små, **`[A-Za-zÅÄÖåäö]`**. Det sista kan naturligtvis också skrivas **`[A-ZÅÄÖa-zåäö]`**. Börjar du förstå? Vi tar ett exempel till

```
keron:~$ ls            <- Vilka filer finns
176-167          Alexander        Kajsa            Knatte
176-671          Fnatte           Kalle            Oppfinnar-Jocke
176-761          Joakim           Karlsson         Tjatte
keron:~$ ls *[a-zK]*   <- De som i namnet har en bokstav mellan a-z eller ett K
Alexander        Kajsa            Knatte
Fnatte           Kalle            Oppfinnar-Jocke
Joakim           Karlsson         Tjatte
keron:~$ ls [a-zK]*   <- De som börjar på en bokstav mellan a-z eller ett K
Kajsa     Kalle     Karlsson  Knatte
keron:~$ ls *[a-zK]   <- De som slutar på en bokstav mellan a-z eller ett K
Alexander        Kajsa            Knatte
Fnatte           Kalle            Oppfinnar-Jocke
Joakim           Karlsson         Tjatte
keron:~$ ls *[A-ZÅÄÖ] <- De som slutar på en stor bokstav
ls: *[A-ZÅÄÖ]: No such file or directory
keron:~$
```

Antag nu att vi vill lista att filer som inte börjar på ’z’. Då skulle vi ju enligt ovan behöva räkna upp alla tecken utom ’z’. Vore det inte smidigare att helt enkelt säga “alla tecken utom z”? Jovisst! det gör mam med tecknet ’^’[8]. Till exempel så listar **`ls [^a-zåäö]*`** alla filer som inte börjar på en liten bokstav.

Den uppmärksamme kanske börjar ana ett problem. Tänk om jag vill lista alla filer som börjar på ’^’ eller ’a’. Eller alla filer som innehåller ett ’a’, ’b’ eller ett ’-’. Naturligtvis så går det att lösa. Det gäller bara att lista så att tecknet ’-’ kommer först och att tecknet ’^’ inte kommar först. Problemem ovan går alltså att lösa på följande vis, **`ls [a^]*`** och **`ls [-ab]`**. Ett sista problem. Hur kan man specifiera tecknet ’]’ inom klamrar utan att parentesen avbryts? Jo genom att sätta ’]’ omedelbart efter ’[’ så tolkas det som en del av vad som innesluts av parentesen. Exemplelvis så matchar **`[]a-zåäö]`** alla gemena bokstäver och tecknet ’]’.

Dessa filnamnsmönster kan sedan kombineras i all oändlighet. Jag rekommenderar att du övar på detta tills du känner dig 100% säker på hur det fungerar. Det kan spara dig mycket arbete framöver.

4.8 Ytterligare information om filer

I föregående avsnitt visades lite snabbt att `ls` kan visa ytterligare information om filerna än bara namnet. Med kommandot `ls -l` fick man ju en massa information. Vi skall nu studera vad den informationen är för något.

4.8.1 Accessrättigheter och `chmod`

I avsnitt 28.9 behandlas mer avancerade accessrättigheter.

Linux har betydligt mer avancerade accessrättigheter än vad till exempel DOS har. Detta har att göra med att Linux är ett fleranvändarsystem. Du kan se vilka accessrätigheter en fil har med kommandot `ls -l` (long list). Varje fil presenteras i en rad som ser ut så här:

[8] tecknet ’!’ gör samma sak och känns kanske mer naturligt för C-programmerare.

```
-rw-rw-r--   1 marcus   marcus    1359472 Apr  6 15:02 linuxboken.ps
 ^^^^^^^^^
  Access
```

När det gäller accessrättigheter så är det de 10 första tecknen (`-rw-rw-r-`) som är intressant. Dessa kan delas in i fyra grupper. Den första gruppen består av bara ett tecken (“-” i exemplet). Detta tecken talar om vilken typ av fil det är. Tecknet kan vara:

- **“-”** Filen är en vanlig fil.
- **“d”** Filen är en katalog.
- **“l”** Filen är en länk.
- **“c”, “b”, “p”** Filen är en specialfil.

De andra nio tecknen är uppdelade enligt följande:

```
-rwxrwxrwx
 |||--------- Ägarens accessrättingheter
    |||------ Gruppens accessrättigheter
       |||--- Andras accessrättigheter
```

Varje person, eller grupps, rättigheter består alltså av tre tecken. Vad dessa tre tecken betyder visas i tabell 4.2

Bokstav	Rättighet
-	Ingen rättighet
r	läsrättighet (Read)
w	skrivrättighet (Write)
x	Exekveringsrättighet (Execute)

Tabell 4.2: Accessrättigheter och deras betydelser

Accessrättigheterna och deras betydelser. Observera att positionen på tecknen har betydelse. Observera att ordet rättighet är väldigt missvisande. Du har inte rätt att läsa andras filer även om de har satt läsrättigheter på dem. Det är bara möjligt för dig att göra det, förutsatt att du har fått lov till det.

Nu kanske du, med all rätt, undrar vem ägaren och gruppen till en fil är. Alla personer som har tillgång till systemet har ett användarnamn (alla program som körs har också en ägare). Användarnamnet är det namn du använder för att logga in på systemet. Du kan även se det genom att ange kommandot `whoami` eller till och med `who am i`. Det kan verka onödigt att ha ett kommando för att ta reda på vem man är. Hur tankspridd får man vara. Men det är nyttigt i flera fall. Det är möjligt att växla användare

medan man jobbar. Man kan ju också jobba på flera maskiner på samma gång. Jag lovar, man glömmer fort vem man är. Ett annat scenario är att du kommer fram till en terminal där någon har glömt att logga ur. Genom att ge kommandot `whoami` får du reda på vem som slarvat!

```
keron:~$ whoami
marcus
keron:~$ who am i
keron!marcus   ttyp2    Sep 16 14:46 (:0.0)
keron:~$
```

Vidare så tillhör alla användare minst en grupp. På små system är det vanligt att man tillhör en grupp som har samma namn som användarnamnet. På stora system skulle detta generera så många grupper att det sällan är praktiskt. Du kan få reda på vilken eller vilka grupper du tillhör med kommandot `groups`.

```
keron:~/work/linuxboken$ groups
marcus disk http
keron:~/work/linuxboken$
```

Hur vet man då vem som äger en fil och vilken grupp den tillhör. Jo det har du förmodligen redan listat ut, vi tittar på en utskrift från `ls -l` igen:

```
-rw-r--r--   1 marcus   http         1413 Aug 30 16:33 index.shtml
 ^^^^^^^^^^     ^^^^^^   ^^^^^     ^^^^^^^ ^^^^^^^^^^^^ ^^^^^^^^^^^
  Access       Ägare    Grupp     Storlek    Datum      Filnamn
```

Filen `index.shtml` tillhör tydligen användaren *marcus* och gruppen *http*. Vi studerar nu rättigheterna:

- Ägaren (marcus) får läsa och skriva till filen (`rw-`)
- Gruppen (http) får läsa och skriva till filen (`rw-`)
- Alla andra får bara läsa filen (`r--`)

Märk skillnade mellan ägaren och gruppen. I gruppen kan det ingå flera användare. I fallet ovan får alla som är medlemmar i gruppen http skriva till filen (Filen är en websida och alla i gruppen http jobbar med webutveckling och kan ändra den).

En sak som kan verka lite konstig är följande, antag att en fil har följande konstiga rättigheter

```
----rw-rw-   1 marcus   marcus          0 Sep 16 17:31 konstig_fil
```

I fallet ovan får inte ägaren läsa eller skriva till filen. Men däremot får gruppen och alla andra göra det. Vad händer då om ägaren försöker läsa innehållet i filen. Vi provar så får vi se:

```
keron:~$ ls -l konstig_fil
----rw-rw-   1 marcus   marcus          0 Sep 16 17:31 konstig_fil
keron:~$ whoami
marcus
keron:~$ cat konstig_fil
cat: konstig_fil: Permission denied
keron:~$ su sigge
Password:
keron:/home/marcus$ whoami
sigge
keron:/home/marcus$ echo "Far jag skriva till filen?" > konstig_fil
keron:/home/marcus$ cat konstig_fil
Far jag skriva till filen?
keron:/home/marcus$ exit
exit
keron:~$ whoami
marcus
keron:~$ cat konstig_fil
cat: konstig_fil: Permission denied
keron:~$
```

Kanske lite överraskande så får inte ägaren varken skriva till eller läsa filen, fast alla får det. Dina rättigheter bedöms efter den första grupp du passar in i. Först kollas om du är ägaren, är du det får du de rättigheter som ägaren har. Är du inte ägaren så provas om du ingår i gruppen, gör du det får du gruppens rättigheter till filen. Tillhör du inte heller gruppen får du de rättigheter som alla andra har. Detta kan kanske klassas som överkurs, men kan vara bra att känna till.

Vilken grupp tillhör filer jag skapar? Eftersom du kan tillhöra flera grupper kan du välja vilken du för tillfället har som primär grupp. Det gör du med kommandot `newgrp`. Om du till exempel skall jobba med filer som du delar med någon annan så vill du ju byta till den grupp som du ock din kollega har gemensam. Se följande exempel

```
$ groups          <- Vilka grupper tillhör jag, den första är den primära
marcus disk http
$ touch kalle     <- Skapar en fil 'kalle'
$ ls -l
total 0
-rw-r--r--   1 marcus   marcus          0 Sep 17 10:07 kalle
$ newgrp http     <- Byter primär grupp till 'http'
$ groups          <- Kollar att den nu står först
http marcus disk
$ touch kajsa     <- Skapar filen 'kajsa'
$ newgrp disk     <- Byter grupp till 'disk'
$ touch joakim    <- Skapar filen 'joakim'
$ ls -l           <- En lång lista visa filernas grupptillhörighet
total 0
-rw-r--r--   1 marcus   disk            0 Sep 17 10:08 joakim
-rw-r--r--   1 marcus   http            0 Sep 17 10:08 kajsa
-rw-r--r--   1 marcus   marcus          0 Sep 17 10:07 kalle
$ groups          <- Vilken grupp var den primära nu igen?
disk marcus http
$ newgrp marcus   <- Jag vill att 'marcus' skall vara den primära gruppen
$ groups
marcus disk http
$
```

Men om jag skapar en fil och kommer på att jag har fel grupp som primär, kan jag byta grupp på den filen i efterhand då? Javisst, med kommandot `chgrp`[9]. Kommandot anropas på följande sätt:

[9] `chgrp` behandlas ytterligare i avsnitt 28.9.2

```
chgrp <gruppnamn> <filnamn>
```

där gruppnamn är en grupp som du tillhör och filnamn är ett filnamn eller en lista av filnamn. Eftersom det kan vara en lista av filnamn kan man med fördel använda ett filnamnsmönster här.

Andra accessrättigheterna `chmod`

Du kanske vill att alla skall kunna läsa en fil i din hemmakatalog, eller vill att de inte skall kunna göra det. Då måste du kunna ändra accessrättigheterna till filen. Det är bara ägaren som kan ändra rättigheterna till filen. Detta gör man med kommandot `chmod`.

Man kan ändra accessrättigheterna på några olika vis. Antingen men en kombination av bokstäver eller med en kombination av siffror. Vi tittar på bokstäverna först.

Tabell 4.3 visar vilka tecken vi kan använda.

Bokstav	Betydelse
u	Ägaren (User?)
g	Gruppen
o	Andra (Others)
a	Alla = ugo
+	Lägg till rättighet
-	Ta bort rättighet
=	Sätt rättighet till
r	läsrättighet (Read)
w	skrivrättighet (Write)
x	exekveringrättighet (Execute)

Tabell 4.3: Bokstavsförkortningar till chmod

Dessa förkortningar används då man sätter rättigheter på filer med `chmod`*. Man kan även använda siffror.*

Alla dessa bokstäver kan sedan kombineras på ett speciellt sätt. Först specifieras vem det gäller (ugoa), sedan på vilket sätt rättigheten skall ändras (+-=) och slutligen vilken eller vilka rättigheter det gäller. Enklast att förklara detta i en tabell till (Tabell 4.4).

Nu vet du hur man sätter rättigheter med bokstäver. Det är dock ofta lättare att sätta rättigheterna med siffror. Antag att du vill sätta rättigheten `-rwxr-x--x`. Skall man göra det med bokstäver får man ange komamndot **`chmod u=rwx,g=rx,o=x filnamn`**. Som du nu börjar förstå så finns det genvägar för nästan allt som innebär mycket skrivning. I detta fall kan man använda en sifferkombination för att sätta en fils accessrättigheter.

Du kan ge vilka rättigheter du vill genom att använda en siffer kombination för att repsentera ägarens, gruppens och andras rättigheter i ett svep. Varje rättighet har ett värde.

Kommando	Betydelse	Resultat
chmod ugo=r filnamn	Alla får bara läsrättigheter	`-r--r--r--`
chmod ug+w filnamn	Ägaren och gruppen får även skriva	`-rw-rw-r--`
chmod a+x filnamn	Alla får även exekvera	`-rwxrwxr-x`
chmod o-x filnamn	Andra får inte exekvera	`-rwxrwxr--`
chmod ugo+rwx filnamn	Alla får göra allt	`-rwxrwxrwx`
chmod a+rwx filnamn	Samma som ovan	`-rwxrwxrwx`
chmod g-rwx filnamn	Gruppen får inte göra nått	`-rwx---rwx`

Tabell 4.4: Bokstavsförkortningar till chmod

Här visas hur man sätter rättigheter med `chmod` *och bokstavskombinationen. Det är samma fil som ändras uppifrån och ned i tabellen. Man kan även använda siffror.*

För att få till exempel ägarens rättigheter lägger man ihop värdena av de rättigheter han, eller hon skall ha. Vad varje rättighet är värd ser du i tabell 4.5.

Rättighet	Siffervärde
läsrättighet (r)	4
skrivrättighet (w)	2
exekveringsrättighet (x)	1
Ingen rättighet	0

Tabell 4.5: Bokstavsförkortningar till chmod

Accessrättigheter och deras siffer värde. För att få rättigheterna för en grupp lägger man ihop värdena av de rättigheter som önskas.

Vill man ge till exempel ägaren läs- och skrivrättigheter så motsvaras det av siffran 6 (4+2). Då man tilldelar en fil rättigheter skriver man tre siffror istället för bokstavskombinationerna i tabell 4.4. Se exempel i tabell 4.6

Prova dig gärna fram med både bokstäver och siffror. Jag tycker att siffrorna är smidigast, men det finns tillfällen då bostäverna är bättre. I vilket fall som heslt så skall man känna till båda.

Nu tror du kanske att du kan allt om accessrättigheter, men jag skall ta ner dig på jorden. Det blir svårare i avsnitt 28.9.

Accessrättigheter för kataloger

Nu har vi tittat på accessrättigheter för filer. Men kataloger har ju också accessrättigheter. De fungerar inte riktigt på samma sätt. Man ändrar dem på precis de sätt som du nyss lärt dig för flier men rättigheterna har nu lite annorlunda betydelse. Studera tabell 4.7

Kommando	Accessrättighet
chmod 111 filnamn	`---x--x--x`
chmod 222 filnamn	`--w--w--w-`
chmod 644 filnamn	`-rw-r--r--`
chmod 755 filnamn	`-rwxr-xr-x`
chmod 711 filnamn	`-rwx--x--x`
chmod 600 filnamn	`-rw-------`

Tabell 4.6: Exempel på sifferrepresentation av accessrättigheter

Accessrättigheter och deras siffer värde. För att få rättigheterna för en grupp lägger man ihop värdena av de rättigheter som önskas.

Bokstav	Rättighet
-	Ingen rättighet
r	läsrättighet (Read), Kan lista innehållet
w	skrivrättighet, kan lägga till filer i kstalogen
x	Rätt att göra katalogen till aktiv (Execute)

Tabell 4.7: Accessrättigheter och deras betydelser för kataloger

Accessrättigheterna och deras betydelser för kataloger. Observera att ordet rättighet är väldigt missvisande. Du har inte rätt att läsa andras filer även om de har satt läsrättigheter på dem. Det är bara möjligt för dig att göra det, förutsatt att du har fått lov till det.

Vill du ge andra tillgång till någon av dina kataloger börjar du med att ge dem Exekverinsrättigheter. Nu har de mäjlighet att läsa de filer i katalogen som da har satt läsrättigheter på. Men de kan inte lista innehållet i katalogen utan måste få filnamnen av dig, eller lista ut dem. Det är lite lagom säkert tycker jag. Observer att du måste ha exekveringsrättigheter på alla kataloger i sökvägen till den katalog du vill dela med dig av. Vill du dessutom att andra skall kunna lista innehållet i en katalog måste du även sätta läsrättigheter på den även här gäller att alla kataloger i sökvägen måste vara ha läs rättigheter. Du kan även ge andra skrivrättigheter till en katalog. Med då du i så fall även måste ge dem skrivrättigheter i din egen hemmakatalog så är det inte att rekomendera. Behåll du skrivrättigheterna för dig själv.

Nu tror du kanske att du kan allt om accessrättigheter, men jag skall ta ner dig på jorden. Det blir svårare i avsnitt 28.9.

4.8.2 Ta bort filer, `rm` och `rmdir`

För att ta bort filer från systemet använder man kommandot `rm` och `rmdir`. `rm` tar bort vanliga filer och `rmdir` tar bort kataloger (som måste vara tomma). Observera att det i Linux inte finns något “undelete” det som tas bort försvinner!

`rmdir` är det enklare av de två kommandona så vi börjar med det. `rmdir` tar bort den eller de kataloger som ges som argument till kommandot. Katalogerna måste vara tomma för att `rmdir` skall fungera. Anropa helt enkelt `rmdir` med de kataloger som skall tas bort som argument till kommandot. Kommandot kan naturligtvis anropas med ett filnamnsmönster som argument. Studera följande lilla exempel

```
$ ls                              <- Kolla vilka filer som finns
alexander/  fnatte/     kalle/      knatte/     tjatte/
$ rmdir fnatte                    <- Ta bort katalogen fnatte
$ ls                              <- Kolla att den försvann
alexander/  kalle/      knatte/     tjatte/
$ rmdir kalle                     <- Ta bort kalle
rmdir: kalle: Directory not empty <- Oops kalle var inte tom
$ rmdir kalle/kajsa/              <- Ta bort det som fanns i kalle
$ ls                              <- Kalle är fortfarande kvar
alexander/  kalle/      knatte/     tjatte/
$ rmdir kalle                     <- Ta bort kalle
$ ls                              <- Kolla vad som finns kvar
alexander/  knatte/     tjatte/
$
```

Nu kan vi ta bort kataloger. Men man måste ju även kunna ta bort filer. Vi skall också se hur man kan ta bort kataloger och dess innehåll i ett svep.

Kommandot `rm` är ett avancerat och mycket farligt kommando. Använd alltid `rm` med största försiktighet! Kom ihåg att jag har varnat dig!

För att ta bort en fil anger man den som kommando till `rm`. Man kan också ange en lista med filer eller ett filnamnsmönster som argument. `rm` tar också en mängd flaggor. I tabell 4.8 listas de viktigaste.

Antag nu att man har en fil som heter `-rf`. Hur skall man ta bort den? `rm -rf` fungerar ju inte. Jo man kan göra på två sätt. Antingen anger man ett absolut filnamn till filen. Eller så kör man `rm` med följande flaggor och argument: **`rm -- -rf`** Flaggan

Flagga	Ändrar beteende enligt
`-i` eller `--interactive`	Detta är ett av de viktigaste argumenten till rm, genom att använda det frågar programmet innan några filer tas bort. Använd det ofta!
`-f` eller `--force`	Också en viktig flagga. Med den tas filer bort hänsynslöst. Du får ingen fråga om en fil skall tas bort. Smidigt, smidigt men farligt, farligt
`-r`, `-R` eller `--recursive`	Tar bort filer och kataloger rekursivt. Du kan till exempel ta bort hela filträd. Är det många filer kan det användas tillsammans med `-f` för att du skall slippa alla frågor.
`--help`	Visa inbyggd hjälp
`--version`	Visa versionsinformation

Tabell 4.8: Flaggor till programmet `rm`

Flaggor till programmet `rm` *vill du veta alla får du läsa den inbyggda hjälpen eller manualsidan.*

-- betyder att det inte kommer några fler flaggor efter den.

4.8.3 Kopiera, flytta och byt namn på filer, `cp` och `mv`

Två kommandom som påminner om varandra är **`cp`** och **`mv`**. Dessa två används för att kopiera respektive flytta filer. Båda tar två eller fler argument. Vi börjar med att titta på **`cp`**.

Om du vill skapa en kopia av filen `fil` som skall heta `kopia_av_fil` och ligga i samma katalog så ger du kommandot **`cp fil kopia_av_fil`**. Alltså *kopiera från till*. Det sista argumentet kan naturligtvis vara ett filnamn i en helt annan katalog. När **`cp`** anropas med filnamn som argument så måste argumenten vara exakt två till antalet. Annars kan inte kommandot lista ut vad det skall göra.

Om det sista kommandot är en katalog då kan antalet argument vara hur många som helst. Observera dock att det är bara det sista som får vara en katalog. **`cp`** tar då alla filer som den får som argument (kan vara bara en) och skapar kopior av dem i den katalog som ges som sista argument.

Nu vet vi hur **`cp`** fungerar. Men om bara sista argumentet får vara en katalog hur skall man då kunna kopiera en katalog till en annan? Det verkar ju jobbigt att behöva flytta fil för fil. Visst **`cp`** klarar även detta. Jag lurades alltså lite innan då jag skrev att bara det sista argumentet får vara en katalog. Man kan anropa **`cp`** med argumentret **`-r`** då kopieras även kataloger rekursivt. Allt hamnar dock precis som tidigare i den katalog som ges som sista argument.

`mv` fungerar på i stort sätt samma sätt med den skillnaden att filerna flyttas. Det vill

Flagga	Ändrar beteende enligt
`-i` eller `--interactive`	Detta är ett av de viktigaste argumenten till cp, genom att använda det frågar programmet innan några filer skrivs över. Använd det ofta!
`-d` eller `--no-dereference`	Detta är ett avancerat alternativ. Då du kopierar en symbolisk länk[10] kopieras vanligtvis en kopia av filen, Med -d kopieras länken som den är.
`-R` eller `--recursive`	
`-a` eller `--archive`	
`-p` eller `--preserve`	
`-f` eller `--force`	
eller	
eller	
`--help`	Visa inbyggd hjälp
`--version`	Visa versionsinformation

Tabell 4.9: Flaggor till programmet `cp`

Detta är bara en del av de flaggor som finns till `cp` *vill du veta alla får du läsa den inbyggda hjälpen eller manualsidan.*

Flagga	Ändrar beteende enligt
`-i` eller `--interactive`	Detta är ett av de viktigaste argumenten till cp, genom att använda det frågar programmet innan några filer skrivs över. Använd det ofta!
`-r`	flyttar kataloger rekursivt.
`-u` eller `--update`	Flyttar inte filer som redan finns på destinationen och har samma eller nyare modifikationsdatum.
`-v` eller `--verbose`	Skriver ut namnet på varje fil innan den flyttas
`--help`	Visa inbyggd hjälp
`--version`	Visa versionsinformation

Tabell 4.10: Flaggor till programmet `mv`

Detta är bara en del av de flaggor som finns till `mv` *vill du veta alla får du läsa den inbyggda hjälpen eller manualsidan.*

säga ursprungsfilen tas bort så fort kopian är gjord. Vissa skillnader finns dock.

4.9 Skapa en fil, `touch`

Kommandot `touch` skapar en tom fil. Detta kan vara smidigt då man laborerar med filers rättighter och liknande. Men inte skriver man ett kommando för bara detta utan `touch` huvudsakliga användningsområde är att uppdatera en fils tidsstämpel. Något som kanske verkar onödigt men är väldigt användbart vid till exempel programkompilering med programmet `make`. Men detta behöver du inte bekymmra dig om ännu.

För att skapa en ny tom fil ger du bara kommandot

```
touch <filnamn>
```

`touch` kan, som sagt, också uppdatera en fils tidsstämpel. Om filen ovan redan fanns så uppdaterades tidssämpeln till den tid då du körde kommandot. Man kan också sätta en valfri tid. Detta görs på formen

```
touch -t MMDDhhmm[[CC]YY][.ss]
```

Där MM=Månad (två siffror), DD=Datum, hh=timme, mm=minut, CC=Århundrade, ss=sekunder. Ovanstående är taget direkt ur manualsidan för `touch`. Titta på den nu. Var betyder nu åvanstående. Jo det som står inom hakparenteser är frivilligt. Programmet behöver inte dessa men kan ta dem. Om de inte anges antas att aktuellt datum gäller. Studera följande exempel:

```
$ touch ny_fil                   <- Skapa en ny fil
$ ls -l --full-time              <- Kolla dess tidsstämpel
total 0
-rw-r--r--   1 marcus   marcus          0 Sat Sep 18 06:46:25 1999 ny_fil
$ touch ny_fil                   <- Uppdatera tidsstämpeln
$ ls -l --full-time              <- Kolla att den ändrades
total 0
-rw-r--r--   1 marcus   marcus          0 Sat Sep 18 06:46:57 1999 ny_fil
$ touch -t 12312359 ny_fil       <- Ändra den till annan tid än nu
$ ls -l --full-time              <- Kolla att det blev som vi ville
total 0
-rw-r--r--   1 marcus   marcus          0 Fri Dec 31 23:59:00 1999 ny_fil
$ touch -t 12312359.59 ny_fil    <- Sista sekunden dett millenium
$ ls -l --full-time
total 0
-rw-r--r--   1 marcus   marcus          0 Fri Dec 31 23:59:59 1999 ny_fil
$ touch -t 123123592000.59 ny_fil <- Sista sekunden år 2000
$ ls -l --full-time
total 0
-rw-r--r--   1 marcus   marcus          0 Sun Dec 31 23:59:59 2000 ny_fil
keron:~/tmp/katalog$
```

Jag tänker inte behandla touch mer än så i denna bok. Läs dess manualsida (eller infosida) om du vill lära dig allt. Nu vet du i alla fall det viktigaste.

4.10 Länkar, kommandot `ln`

Den intresserade och nyfikne läsaren har säkert upptäckt att det inte nämnts ett pip om siffran som står före ägaren i utskriften från `ls -l`.

```
-rw-r--r--   1 marcus   http         1413 Aug 30 16:33 index.shtml
           ^^^
```

Den siffran har någonting att göra med vad detta avsnitt handlar om, nämligen länkar.

I Unix kan man utbyttja något som kallar länkar. En länk kan man använda då man vill att en fil skall finnas på två ställen. Ett sätt är ju att kopiera filen, med det har många nackdelar. För det första så tar det ju dubbelt så mycket plats om samma fil skall finnas på två ställen. För det andra, vad händer om den ena av filerna uppdateras, och den andra förblir orörd? Då kan ju filerna, som skulle vara lika, helt plötsligt blivit olika. Vi slänger oss in i hur länkar fungerar.

Det finns två olika typer av länkar, *hårda* och *symboliska*.

4.10.1 Hårda Länkar

En hård länk gör att man ger en fil två stycken namn. Namnen kan ligga i olika kataloger men inte på olika filsystem. Båda filnamnen avser samma fil men har ingen bindning till varandra. Båda är likvärdiga. Filen tar upp samma plats på disken oavsett hur många länkar den har till sig. Tas det ena bort finns det andra och filen kvar. Studera följande exempel:

```
keron:~/links$
keron:~/links$ echo Trallallalla > fil
keron:~/links$ cat fil
Trallallalla                        <-- Filen fil innehåller Trallallalla
keron:~/links$ ln fil link          <-- Skapar en hård länk
keron:~/links$ ls -l                <-- ls -l ger att det nu finns två filer.
total 2
-rw-r--r--   2 marcus   marcus         13 Apr 24 13:06 fil
-rw-r--r--   2 marcus   marcus         13 Apr 24 13:06 link
keron:~/links$ cat link
Trallallalla                        <-- link inehåller samma sak som fil
keron:~/links$ echo Trallalej >> fil <-- Ändrar fil
keron:~/links$ cat link             <-- Kontrollerar att ändringen även finns
Trallallalla                        <-- i link
Trallalej
keron:~/links$ rm fil               <-- Tar bort fil
rm: remove 'fil'? y
keron:~/links$ cat link             <-- Kontrollerar att link finns kvar
Trallallalla
Trallalej
keron:~/links$
```

4.10.2 Symboliska länkar

Hårda länkar har sina begränsningar. För det första så måste filnamnen ligga på samma fysiska partition. Vidare kan de inte användas på kataloger. Båda dessa begränsningar

kan man komma förbi med hjälp av symboliska länkar. Symboliska länkar inför dock nya begränsningar. Dessa behandlas här.

Skillanden mellan en hård och symbolisk länk är att en hård länk är två namn på samma fil. Det är absolut ingen skillnad på dem. I fallet med en symbolisk länk så är filen oförändrad och länken är en egen fil som pekar på en annan fil. Man kan använda dem på samma sätt.

Om filen som en symbolisk länk pekar på tas bort så finns länken kvar och pekar på en fil som inte finns. Länken är då oanvändbär.

Man bör tänka sig för då man skapar symboliska länkar. Eftersom länken bara innehåller en sökväg till en annan fil bör man vara smart så att den fungerar i så många fall som möjligt. Ibland kan det vara bra att använda absoluta filnamn, en annan gång inte. Tänk på att ett filsystem kan monteras på olika ställen. En länk till en fil innom samma katalog skall bör till exempel inte innehålla ett absolut filnamn. Då fungerar ju inte länken om katalogen byter namn. XXXXXXX Fyll på med mer, eller flytta till sysadmindelen, är detta intressant i ett så tidigt skede? XXXXXXX

Vad betydde nu siffran i utskriften från `ls -l`?

```
-rw-r--r--   1 marcus   http         1413 Aug 30 16:33 index.shtml
           ^^^
```

Jo, siffran representerar det antal namn en fil har. Alltså hur många hårda länkar det finns till filen. Filen i exemplet har bara ett enda namn. Det finns alltså ingen annan hård länk till den. Är det en katalog motsvarar siffran antalet filer i katalogen.

4.11 Mer om `ls`

Kommandot **`ls`** Kan naturligtvis göra mycket mer än bara visa filerna i en katalog. **`ls`** har varit med sedan Unix barndom och har blivit mer och mer avancerat.

`ls` kan ta en mängd flaggor som ändrar dess beteende. I tabell 4.11 presenteras de viktigaste.

Ytterligare information får du i infobladet om ls eller i direkthjälpen. Det enklaste sättet att få en känsla för hur det fungerar är som bekant att prova. **`ls`** är ju dessutom ett helt ofarligt kommando. Så testa på.

Flagga	Ändrar beteende enligt
`-a` eller `--all`	Visar även filer som börjar med ’.’ det gör inte ls som default
`-A` eller `--almost-all`	Visar filer som börjar med punkt, dock inte ’.’ och ’..’
`-d` eller `--directory`	Visa namnet på kataloger i stället för att lista innehållet i dem. Annars listar ls innehållet i ett led nedåt.
`-l`	Visa långt format, ger mer information
`-m`	Skriver en kommaseparerad rad istället för kolumner. Då raden är full skrivs på nästa
`-p`	lägger till ett tecken som visar filtypen, smidigt
`-r` eller `--reverse`	Sorterar i omvänd ordning
`-s` eller `--size`	Visa filstorleken vid varje fil
`-R` eller `--recursive`	Listar alla underkataloger rekursivt
`-S`	Sortera efter storlek
`-t`	Sortera efter när filen senast ändrades, tillsammans med l (-lt) visas denna tid.
`-u`	Sortera efter när filen senast öppnades, tillsammans med l (-lu) visas denna tid.
`-U`	Sortera inte alls, visa filerna i den ordning de ligger.
`--color`	Visar färger, om displayen klarar det.
`--help`	Visa inbyggd hjälp
`--version`	Visa versionsinformation

Tabell 4.11: Flaggor till programmet `ls`

Detta är bara en del av de flaggor som finns till `ls` *vill du veta alla får du läsa den inbyggda hjälpen eller manualsidan.*

Överkurs!
Att använda **ls** *med färger kan vara ett mycket bra sätt att lätt se vad dina filer är för något. För att detta skall fungera krävs att du har en terminal eller terminalprogram som klarar att visa färger. Jag rekommenderar terminalemulerningsprogrammet* **rxvt** *som både är litet, snabbt och bra. Det smartaste sättet att få* **ls** *att visa färger är att skapa ett* **alias** *för* **ls** *med kommandot* **alias ls="ls --color"** *(Mer om alias, i bash kan du läsa i avsnitt 7.16.3 på sidan 129). Efter att du kört kommandot så visar* **ls** *snygga färger. Nu finns det dock ändå mer man kan göra med detta. Man kan nämligen själv bestämma vilken färg det skall vara på filerna och även vilka filer som skall färgas. Man kan dels ange att filer av typen* exekverbara *skall färgas. Man kan även säga att alla filer som matchar ett visst filnamnsmönster skall färgas. Allt detta gör man genom att sätta miljövariabeln* **LS_COLORS** *till ett bra värde. Värdet skall ha formen* **"xx=YY;YY:xx=YY;YY..."** *där xx är en kombination av två bokstäver eller ett filnamnsmönster. Vilka alternativ som finns för xx och YY står i manualsidan för* **ls** *så jag skriver dem inte här. Ett trevligt värde på variabeln kan vara* **"or=31;1:*core=31;1"**. *som gör att brustna länkar och filer med namnet* ***core** *färgas röda. Sätt variabeln med detta kommando (i* **bash***)*

```
$ export LS_COLORS="or=31;1:*core=31;1"
```

Det var allt vi tar upp om `ls` för den här gången. Du kommer förmodligen inte att använda alla flaggorna (jag gör det inte). Lär dig de som du tycker är användbara och lär dig dem väl så att du har dem till hands då de behövs.

4.12 Köra program

En fråga man ibland får ifrån nybörjare är “Hur kör jag ett program i Linux?”. Man kan starta program på flera olika sätt. Använder du *fönstersystemet X* kan du starta de flesta program från menyer i din fönsterhanterare. Men det kanske enklaste är att starta programmen från ett terminalfönster. Då har du dessutom enkelt möjligheten att skicka *argument* och *flaggor* till programmet. Ett argument skickar ytterligare information till programmet. Ett argument är ofta ett filnamn. En flagga[11] däremot ändrar programmets beteende utan att tillföra information. En flagga skrivs ofta som ett streck och ett tecken eller som två streck och ett ord.

Att starta ett program från kommandoprompten är det enda alternativet om du kör utan X. För att köra ett program skriver man helt enkelt programmets namn vid promten. Till exempel, **keron:~$ emacs** för att starta editorn Emacs. Tänk på att Linux är känsligt för stora och små bokstäver. Nästan alla programnamn består av uteslutande små tecken (gemener) För att detta skall fungera måste programmet emacs ligga i en katalog som finns i miljövariabeln PATH. Ligger inte programmet i en sådan katalog måste du tala om för skalet i vilken katalog programmet ligger. För att köra programmet “kalle” i aktuell katalog (som inte är med i PATH) måste du således ge kommandotr:

[11] Kallas även växel

```
keron:~$ ./kalle
```

Punkten betyder ju "*aktuell katalog*".

4.12.1 Köra program i bakgrunden

Ganska ofta vill man starta ett program, göra något annat och sedan återvända till programmet. Linux klarar ju *multitasking* så det kan köra flera program på samma gång. Du kan starta ett prgram, till exempel editorn `pico` i ett terminalfönster. Vill du sedan göra något annat så kan du stoppa programmet med tangentkombinationen Ctrl+Z. Du kommer då tillbaka till prompten och kan ge andra kommandon. Editorn har nu *stoppats* och körs inte. Vill du komma tillbaka till editorn kan du ge kommandot **`fg`**. Nu tas editorn fram i förgrunden igen och fortsätter sin exekvering där den stoppades. I fallet ovan stoppades editorn då vi tryckte Ctrl+Z, det är ju ingen mening att ha en editor gående i bakgrunden. Antag att det istället för en editor var ett kommando som tar lång tid, till exempel en säkerhetskopiering. Då vill man kunna använda prompten fast programmet körs i bakgrunden. Då kan man göra så att då man stoppat programmet ger kommandot **`bg`** så startas programmet fast i bakgrunden. Man kan då jobba på som vanligt fast maskinen gör flera job samtidigt. Ett annat sätt att köra program i bakgrunden är att starta dem med ett &-tecken efter, då startas de direkt i bakgrunden och du för tillbaka prompten på en gång. Detta är synnerligen lämpligt då man startar program i X.

OK det var ju inte så svårt, men hur hanterar jag flera olika program som körs i bakgrunden? Då ett program eller, process som det kallas, körs i bakgrunden kallas den för ett job. Skillnaden egentligen mellan ett jobb och en process är att ett jobb kan innehålla flera processer.

4.13 Miljövariabeln PATH

För att systemet skall kunna hitta de program du vill köra måste de ligga i en katalog som listas i miljövariabeln `PATH`. Du kan studera din `PATH` med följande kommando:

```
keron:~$ echo $PATH
```

En sak som förbryllar många, framför allt DOS-användare är att aktuell katalog inte ligger med i `PATH`. Detta är en säkerhetsåtgärd. Antag att du har aktuell katalog i din PATH. Du har för tillfället Kalle-Hackers hemmakatalog som aktuell katalog. Du vill se vad han har i katalogen och ger kommandot **`ls`**. Nu kanske Kalle-Hacker har listat ut att någon kommer att göra just detta. Han har då skrivit ett program som heter `ls` och lagt i sin hemmakatalog. Helt utan att du vet om det kan du nu, med dina rättigheter köra de kommandon Kalle-Hacker vill ha körda. Om Kalle-Hacker har skrivit till exempel `rm -rf $HOME` i skriptet så är det din hemmakatalog som raderas och inte Kalle-Hackers. Kalle-Hacker har naturligtvis också sett till att det sista programmet gör är att köra det riktiga `ls`. På så sätt kanske du inte ens märker

att du utträttat job åt Kalle-Hacker. Du kan ju fundera på vad som skulle kunna hända om användaren root kör Kalle-Hackers program.

Du bör naturligtvis på samma sätt se till att du inte har någon katalog som någon annan än systemadministratören (eller någon annan du litar på) administrerar i din $PATH. Eftersom du då kan köra de program som läggs upp i denna katalog utan att tänka på det.

Tips!
Vill du att det skall fungera som i MS-DOS och dess vänner så kan du lägga till "." i PATH det gör du med kommandot.

```
export PATH=$PATH:.
```

Detta lägger till "." sist i din PATH

Varning!
Om du vill ha "." i din PATH (vilket jag inte rekomenderar) så skall den ligga **SIST** *(Som i exemplet ovan). Användaren root skall* ***INTE*** *ha aktuell katalog i PATH.*

4.14 Tabkompementering

Tabkomplementering är en underbar sak i skalen man använder i Linux. Det fungerar lite olika i olika skal men det som beskrivs här gäller för **bash**. Det är troligast att du har detta skal om du kör Linux.

De flesta datamänniskor[12] är slöa. De skriver inte gärna fler tecken än vad som behövs. Därav tabkomplementering. Tabkomplementeringen är också väldigt bra för att få en bekräftelse på att man skrivit rätt. Använd tabkomplementeringen ofta! Hur fungerar det då?

Om jag står i katalogen `/home` och vill göra `/home/kalle` till aktiv katalog räcker det med att jag skriver tillräckligt många bokstäver i namnet kalle för att skalet skall förstå vilken katalog jag menar. I detta fall srkiver jag **`cd ka`** och trycker på Tab. Nu fyller skalet på så att kommandot blir **`cd kalle`**. Hade det funnits en katalog `/home/kajsa` också så hade skalet inte vetat vilken katalog jag menar etersom både `kalle` och `kajsa` börjar på ka. Då piper[13] det till. Då vet du att du inte angivit tillräckligt många tecken. Om du utan att skriva något trycker på Tab igen så visas en lista på de alternativ som finns kvar. I detta fall måste jag ange även l för att specifiera katalogen `kalle` så jag skriver ett l och trycker på Tab. Stdera följande exempel på detta:

```
keron:/home$ cd ka     <- Här trycker jag <tab> 2 gånger
kajsa  kalle
keron:/home$ cd kalle/ <- Här skriver jag <l> och trycker på <tab>
keron:/home/kalle$
```

Tabkomplementering fungerar i flera sammanhang. Till exempel då du skall skriva

[12] Programmerare i synnerhet
[13] Om du inte har "pipet på" så händer inte något. I vissa fall blinkar fönstret till.

ett kommando. Skiver du b Tab Tab får du en lista över alla kommandon som börjar på b. Mycket smidigt. Blir listan av alternativ väldigt lång får man en varning om detta. Nedan vill jag veta alla kommandon på t:

```
keron:~$ t   <-  Här trycker jag <tab> 2 gånger
tac                     tfmtodit                tkmc
tail                    tftopl                  tkmib
tailf                   tftp                    tknewsbiff
talk                    tgatoppm                tkpasswd
tangle                  tgz                     tksysv
taper                   theme-selector-capplet  tload
tar                     then                    todo

 [ Här har jag jag klippt bort för att spara papper ]

texi2html               timetool                twm
texi2pdf                timidity                type
texindex                tin                     typeset
text2gif                titel                   tzselect
text2sf                 tixindex
textmode                tixwish4.1.8.0
keron:~$ t   <-  Här kan jag nu fylla på med vad som passar
```

Tag för vana att använda tabkomplementering så ofta du kan. Om de sista bokstäverna i ett kommando skrivs dit via tab så vet du att hela kommandot är rätt inskrivet. Eller åtminstone utan skrivfel. Antag att du skall lista innehållet i `/usr/share/emacs/20.3/lisp/`. Risken är stor att du någonstans på vägen skriver fel. Eller att katalogen heter något annat. Detta eleminerar du genom att låta tabkomplenemteringen skriva de sista tecknen (du kan trycka på Tab flera gånger för att få fram ett så pass långt filnamn) vet du att filen, eller katalogen, finns.

4.15 Klipp och klistra

Att klippa ut text i Linux och sedan klistra in det på ett annat ställe ar enkelt. Markera texten genom att markera den med musens vänstra knapp. Du kan sedan om du vill ändra storlek på din markering med den högra knappen. Då din markering är klar klickar du där du vill klistra in texten med mittenknappen[14] på musen. Klart! Detta förutsätter att du kör i X. Även i konsolen kan du klippa och klistra om du kör `gpm` *General Purpose Mouse* då markerar du med vänster musknapp, och klistrar in genom att klicka med höger musknapp där du vill att texten skall klistras in.

Övningsuppgifter

Uppgift 4.1 Filnamnsmönster. I avsnittet om filnamnsmönster så användes komamndot **`ls`** för att demonstrera hur substitutionen går till. Vill man se vad som sker i klartext kan man använda commandot **`echo`** istället. Använd detta och ta reda på vad som händer då ingen fil matchar filnamnsmönstret.

Uppgift 4.2 Skapa en katalog i din hemmakatalog med kommandot **`mkdir`** gör sedan

[14] I Unix utgår man från att musen har tre knappar. Har du bara två och kör i X kan du prova att trycka på båda knapparna samtidigt för att simulera den tredje knappen. Mer om detta står i kapitel 6

den till aktuell katalog med kommandot **`cd`** Skapa sedan följande filer med kommandot **`touch <filnamn>`**:

- Kalle
- Kajsa
- Alexander
- Knatte
- Tjatte
- Fnatte
- Oppfinnar-Jocke
- 176-671
- 176-167
- 176-761
- Karlsson
- Joakim

Du känner säker igen namnen från någon Kalle Anka tidning. De numeriska namnen är Björnbusar, som ju inte har annat än fångnummer som namn. Använd nu kommandot **`ls`** för att lista:

1. Alla namn
2. Alla Knattar, Knatte, Tjatte och Fnatte
3. Alla utom Björnbusarna
4. Bara Björnbusarna
5. Paret Kalle och Kajsa
6. Alla med ett k (versalt eller gement) i namnet

Kapitel 5

Editorer

För att kunna utforska och laborera med ett Linuxsystem måste man lära sig en editor. En editor är ett program med vilket man kan läsa och editera rena textfiler. I de flesta Linuxsystem har du en hel del olika editorer att välja mellan. Jag rekommenderar att du kollar vilka du har tillgång till, väljer en och lär dig den väl.

5.1 Vi

`vi` är en editor som jag verkligen rekommenderar att du lär dig. I alla fall om du kommer att jobba på flera olika system. `vi` finns nämligen installerad på i princip alla Linux och Unixsystem du kan komma i kontakt med. Den kan dock kännas mer än lovligt bakvänd att jobba i om man sedan tidigare är var editorer som exempelvis Microsoft's Notepad.

Skall man vara petig så är risken inte särskilt stor att `vi` verkligen finns installerad på ditt Linuxsystem. Kommandot **`vi`** startar antingen en vi-klon som heter `evlis` eller så startas en förbättrad variant av `vi` som heter `vim`. Du kan ge kommandot **`vi --version`**[1] för att få reda på vilken du kör. Kör du `elvis` eller verkligen `vi` så gäller allt som står i detta kapitel. Kör du `vim` så gäller också allt som står i detta kapitel med vissa skillnader som beskrivs i nästa kapitel.

`vi` står för *visual*. `vi` är en fullskärmseditor, det vill säga den använder hela det fönster, eller den bildskärm den körs i/på. `vi` är en textbaserad applikation och är avsedd att köras på en terminal eller i ett terminalfönster i X. Både `elvis` och `vim` har X gränssnitt som också kan användas. Dessa ger programmen fler funktioner, eller rättare sagt, gör dem mer lättillgängliga för dem som inte kan motsvarande kommandon..

`vi` är en mycket avancerad editor. Jag kommer bara att ta upp det nödvändigaste kommandona här. Vill du bli en avancerad `vi`-användare måste du läsa mer dokumentation och öva dig en hel del. Man lär sig inte en editor genom att läsa en text, utan man lär sig den genom att använda den. All läsa om en funktion och försöka lägga den på min-

[1] fungerar inte det startar du vi och anger kommandot **`:version`**

net är i princip omöjligt. Däremot om man jobbar i editorn och tänker “jag vill göra si och så, går det?” och sedan hittar svaret på sin fråga så är det lätt att lära sig. Nu kör vi igång med `vi`!

`vi` startas med kommandot **`vi`** och filen som skall redigeras som argument:

```
keron:~$ vi filnamn
```

Följande visas:

```
~
~
~
~
~
~
~
~
filnamn [NEW FILE]
```

Tecknet ~ (tilde) används av `vi` för att visa tomma rader. Sista raden där filnamnet och [New File] visas kallas för statusraden. Hade du öppnat en befintlig fil hade det sett ut så här:

```
%
%
%
\chapter{Editorer}

För att kunna utforska och laborera med ett Linuxsystem måste man lära
sig en editor. En editor är ett program med vilket man kan läsa och
editera rena textfiler. I de flesta Linuxsystem har du en hel del
olika editorer att välja mellan. Jag rekommenderar att du kollar vilka
du har tillgång till, väljer en och lär dig den väl.

\section{vi}
Vi är en editor som jag verkligen rekommenderar att du lär dig. I
alla fal om du kommer att jobba på flera olika system. Vi finns
nämligen installerad på i princip alla Linux och Unixsystem du kan
komma i kontakt med. Den kan dock kännas mer än lovligt bakvänd att
jobba i om man sedan tidigare är var editorer som exempelvis
Microsoft’s Notepad.

Vi står för {\em visual}. Vi är en fullskärmseditor, det vill säga den
använder hela det fönster, eller den bildskärm den körs i/på. Vi är en
textbaserad applikation och är avsedd att köras på en terminal eller i
ett terminalfönster i X.
```

`vi` är en gammal editor. Den fanns i ett tidigt skede i UNIX-världen. Då var det inte självklart med varken datormöss eller avancerade tangentbord. Därför använder `vi` av bara det allra mest grundläggande. Det vill säga bokstavstangenterna, esc och Enter. Detta är även idag en fördel. Det gör att du kan använda `vi` på vilken terminal som helst.

För att detta skall fungera jobbar man alltid i olika lägen[2]. Det finns två olika lägen. *kommandoläget* och *editeringsläget*. I kommandoläget ger du kommandon till programmet och i editeringsläget skriver du text i den fil som du har öppen. Då programmet startas befinner sig programmet i kommandoläget. Allt du nu skriver kommer att

[2] Det jag kallar lägen kallas ofta tillstånd eller moder

av programmet tolkas som kommandon. För att komma till editeringsläget kan du till exempel ge kommandot **i**. Nu fungerar tangentbordet som en skrivmaskin, alla tecken du skriver blir text i den fil du skriver. För att komma tillbaka till kommandoläget trycker du alltid på Esc. Detta kan du göra även om du redan är i kommandoläget. Du kan alltså trycka på Esc om du är osäker på i vilket läge programmet är så vet du att du är i kommandoläget.

Kommandoläget

I kommandoläget (som du kommer i efter att tryckt Esc) tolkas alla tangenter som kommandon. Till exempel så raderar kommandot **x** det tecken markören står över. Varje kommando utförs direkt då det ges. Man trycker alltså inte på Enter för att det skall utföras.

Att det bara finns ett kommandoläge och ett editeringsläge är dock en sanning med modifikation. Det finns faktiskt ett läga till. I kommandoläget består alla kommandon av ett bestämt antal tecken och utförs direkt. Vissa kommandon består dock av relativt många tecken eller kan ges argument. För att man praktiskt skall kunna ange dessa kommandon måste ett tredje läge införas. Det läget kallas kommandoradsläget. För att komma till det ges kommandot **:**. Ett vanligt kommandoradskommando är kommandot **:q!** som betyder “avsluta utan att spara ändringar”. Man kan bara komma till kommandoradsläget från kommandoläget. Kommandoradskommandot utförs efter ett tryck på Enter. Du kan alltså ändra kommandot du håller på att ge innan du trycker på Enter och kommandot utförs. Då kommandot har utförts hamnar man automatiskt i kommandoläget igen. I tabellerna i detta kapitel så ser man att ett kommando är ett kommandoradskommando på att det börjar med ett : (kolon).

Inskrivningsläget

Till inskrivningsläget kommer man vanligvis på fyra olika vis. Med kommandot **i** (Insert) växlar `vi` till inskrivningsäget och markören placeras där den stod innan. **o** (Open line) skapar en ny rad under den markören står på. **O** skapar en rad där markören står och flyttar ner den aktuell raden ett steg. Markören placeras i båda fallen först i den nya raden. **a** (Append) är lika som insert men placerar markören efter tecknet där den står. **a** och **i** kan verka väldigt lika, men används på olika sätt. Vill du skriva ett ord före ett annat (insert) så ställer du markören på ordets första bokstav och ger kommandot **i**. Vill du skriva efter ett ord (append) ställer du markören på ordets sista bokstav och ger kommandot **a**.

Raderingskommandot

I **vi** heter raderingskommandot **d**. Det är kraftfullare än de raderingsfunktioner man är van vid från enklare editorer. För att radera en hel rad ger man kommandot **dd**. Ett specialfall är då du bara vill radera ett tecken då anväder du kommandot **x** som raderar tecknet under markören. De flesta komamndon kan upprepas ett antal gånger. Då ger

vi-Kommando	Utför	kommentar
r	Byter tecknet under markören	Tänk på Replace
cw	Ändrar ordet från där markören står och till ordets slut	Tänk på *Change Word*. Detta kammando placerar dig automatiskt i inskrivningsläget.
c$	Ändrar raden från där markören står och till radens slut	Tänk på *Change Word*. Detta kammando placerar dig automatiskt i inskrivningsläget.

Tabell 5.1: Kommandon i editorn `vi` för att förflytta sig i buffern.

vi-Kommando	Utför	kommentar
h, l, j, k	Flyttar markören	Prova, det är mer logiskt än det verkar. Piltangenterna fungerar oftast också men inte alltid så det är bra att lära sig dessa.
x G	Flyttar markören till rad *x*	Om inget tal ges före G så flyttas markören till sista raden.

Tabell 5.2: Kommandon i editorn `vi` för att förflytta sig i buffern.

man kommandot ***x*** **kommando** så körs kommandot *x* gånger. Exempelvis så raderar **5 x** fem tecken från markören och framåt.

Ersättningskommandon

Antag att du stavat ett ord fel. En av bokstäverna skall bytas ut mot en annan. Till detta finns kommandot **r** som i *Replace*. Ställ markören över den bokstav som skall ändras och ge kommandot **r** följt av vilken bokstav det skall vara. Ibland är det inte bara en bokstav som har blivit fel utan kanske hela ordet måste ändras till detta finns kommandot **cw** som i *Change Word* **cw** raderar ordet från där markören står och byter läge till inskrivningsläget. Vill du fortsätta i kommandoläget får du trycka på Esc för att komma tillbaka.

Kommandot **c** (som ovan användes med **w**) Kan även användas med andra objekt. Till exempel ändrar **w$** hela raden från där markören står.

Klipp och klistra kommandon

Det senast raderade objektet kan enkelt klippas in igen med kommandot **p** som i *Put*.

Ångra kommandon

Ibland vill man göra något ogort (tänk avd lätt det är i datorvärlden). I `vi` finns det två kommandon för att göra detta. Nämligen **u**, **U**. Det första ångrar senast gjorda ändringen. Det andra återställer en hel rad och det sista ångrar senaste ångringskommandot. Observera att du bara kan ångra i ett steg[3] steg. Ger du kommandot **u** två gånger så ångrar du först senaste ändringen och sedan ångara du ångringen. Din ändring införs alltså igen.

[3] Detta är en av skillnadera mellan `vi` och `vim`

vi-Kommando	Utför	kommentar
u	Ångrar sista kommandot	
U	Återställ hela den rad du står på till det den var föra ändrningarna.	

Tabell 5.3: Kommandon i editorn `vi` för att förflytta sig i buffern.

vi-Kommando	Utför	kommentar
:w	Sparar aktuell fil	
:wq	Sparar aktuell fil och avslutar	
:q	Avslutar utan att spara filen. Om filen är ändrad fungerar inte detta.	
:q!	Avsluta utan att spara filen även om den är ändrad	
i	Byter till inskrivningsläge, markören står kvar där den är.	Tänk på Insert
a	Byter till inskrivningsläge flyttar fram markören ett steg	Tänk på Append

Tabell 5.4: Kommandon i editorn `vi`

I `vi` finns det kommandon för att söka i text. Dessa är lika som i `more` och `less`, eller rättare sagt de är lika som `vi`. För att söka efter en text ger du kommandot **`/`** följt av texten du söker. Då du trycker enter hamnar du på den första förekomsten av det du söker från där markören stod och nedåt i texten. Vill du söka på nytt på samma söktext gen du kommandot **`n`** som i *next* eller *nästa*.

Man kan också göra mer avancerade sökningar. Man kan söka på så kallade reguljära uttryck. Hur sådana kan se ut behandlas i avsnitt 7.5 på sidan 7.5.

5.2 xelvis

`elvis` har också en fasad för X. Du startar den med `xelvis`.

5.3 vim

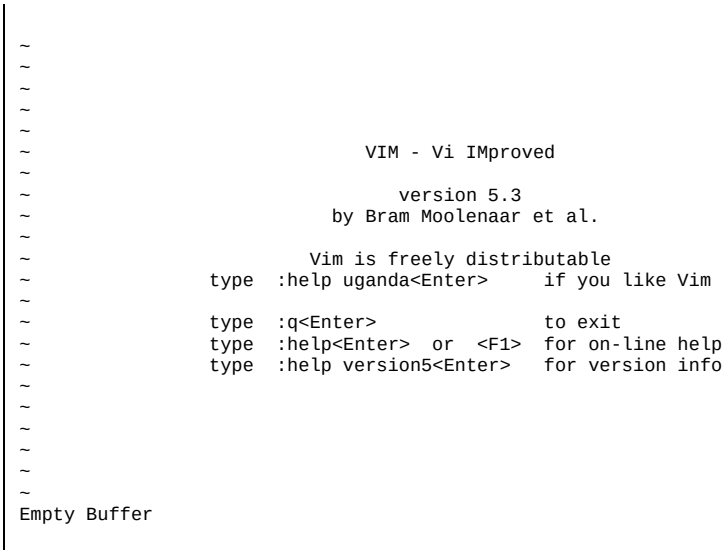
Vim står för vi improved och är en förbättring av vi. Den fungerar på samma sätt men har en hel del förbättringar. I flera linuxdistributioner är det faktiskt vim som du startar när du tror att du startar vi. Om du startar vi utan att öppna en fil så ser du vilken du kör:

vi-Kommando	Utför	kommentar
x dd	Radera *x* rader från och med aktuell rad och nedåt.	Bara **`dd`** raderar aktuell rad och är således lika som **`1 dd`**.
x	raderar tecknet under markören	
dw	Raderar från där markören står till ordets slut	Tänk på Delete Word
d$	Raderar från där markören står till radens slut	

Tabell 5.5: Raderingskommandon i editorn `vi`

vi-Kommando	Utför	kommentar
/	Sök textsträng	Bra att kunna, lika i `more` och **`less`**
n	sök nästa	samma sak här som ovan

Tabell 5.6: Sökkommandon i editorn `vi`



Allt som beskrevs i föregående kapitel fungerar i både vi och vim. Vill du veta vad som skiljer dem åt kan du starta vim och ge kommandot `:help vi_diff`.

Undo i flera steg

Multi level undo. |undo| 'u' goes backward in time, 'ctrl-R' goes forward again. Set option 'undolevels' to the number of changes to be remembered (default 1000). Set 'undolevels' to 0 for a vi-compatible one level undo. Set it to -1 for no undo at all.

GUI `gvim`

Graphical User Interface (GUI) |gui| Included support for GUI: menu's, mouse, scrollbars, etc. You can define your own menus. Better support for CTRL/SHIFT/ALT keys in combination with special keys and mouse. Currently for X11 (with Motif and Athena interfaces), Win32 (Windows 95 and Windows NT), BeOS, Amiga and Macintosh.

Syntax highlighting

Syntax highlighting. |:syntax| Vim can highlight keywords, patterns and other things. This is defined by a number of ":syntax"commands, and can be made to highlight most languages and file types. A number of files are included for highlighting the most common languages, like C, C++, Java, Pascal, Makefiles, shell scripts, etc. The

colors used for highlighting can be defined for ordinary terminals, color terminals and the GUI with the ":highlight"command.

Block operators

Block operators. |visual-block| With Visual a rectangular block of text can be selected. Start Visual with CTRL-V. The block can be deleted (’d’), yanked (’y’) or its case can be changed (’ ’, ’u’ and ’U’). A deleted or yanked block can be put into the text with the ’p’ and ’P’ commands.

Mouse support

Mouse support |mouse-using| The mouse is supported in the GUI version, in an xterm for Unix, for MS-DOS, and Win32. It can be used to position the cursor, select the visual area, paste a register, etc.

5.4 pico

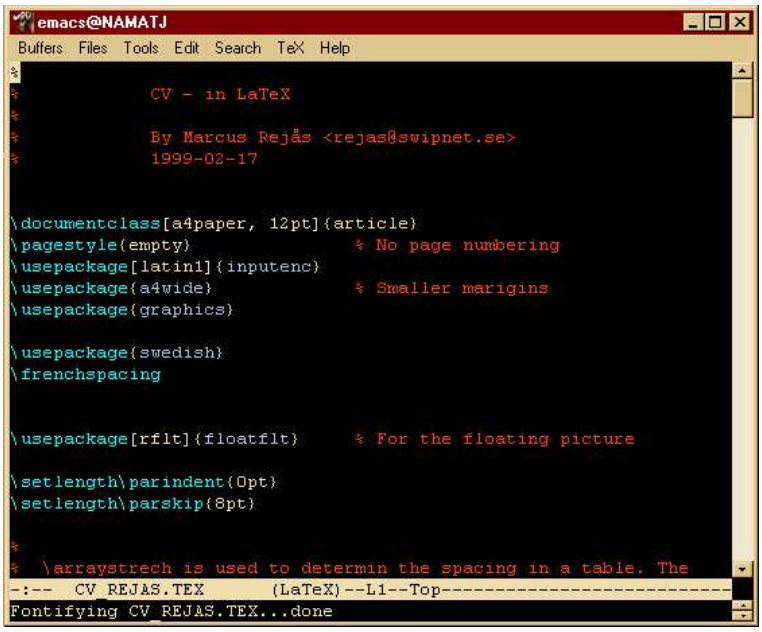
Om du har pico installerad på ditt system och du vill komma enkelt undan skall du lära dig den. Den påminner om editorer från DOS-världen och du lär dig att editera enkla filer i den väldigt snabbt.

Pico är också en textbaserad editor, det vill säga den är gjord för att köra på en text-terminal eller i ett termilanfönster.

5.5 emacs

`Emacs` är en mycket speciell editor. Den kan nämligen nästan allt (utom koka kaffe och massera trötta programmerare på axlarna). Du kan editera filer i den förstås, men du kan även läsa och skriva epost och surfa på internet med den. Jag rekommenderar dock att du lär dig någon av de mindre editorerna utöver emacs eftersom den är onödigt stor att dra igång för att ändra ett tecken i en initieringsfil.

Här tänker jag behandla den textbaserade varianten av `emacs`. Det gör jag inte för att den används så ofta nu för tiden utan snarare för att den kan vara lite knivigare att lära sig. Om du startar `emacs` i X så får du tillgång till menyer som kan underlätta ditt arbete. Att lära sig den textbaserade varianten är i alla fall inte bortkastat. För visst är menyer lätta att lära sig men inget gör snabbare än ett väl invant tangentbordskommando. Dessutom slipper du flytta händerna från tangentbordet (extra bra om du, som jag, ibland har tangentbordet i knät). Vill du starta den textbaserade varianten av emacs i X (fast det går lika bra med den med gui) ger du vanligtvis kommandot **`emacs-nox`**. Som i Emacs No X.



Figur 5.1: GNU Emacs i Windows

Flera av GNU-projektets program finns portade till andra platformar. Så om du till exempel på jobbet är bunden till Windows så kan du ände köra flera av dina favoritprogram. Bilden visar hur portningen av Emacs till Windows ser ut.

5.6 xemacs

Xemacs är inte, som man kanske kan tro, en X-fasad till emacs. Nej, xemacs är ett helt annat program. Xemacs tillför en del finesser som inte emacs har men jag måste tillåta mig att vara partisk och påstå att emacs är bättre ...

5.7 Många fler

Ja det finns en uppsjö av editorer. Att fråga någon om vilken som är bäst ger som regel en hel hög med olika svar. Det är huvudsaken att du väljer en editor och lär dig hantera grunderna i den väl.

En editor som du bör känna till men som jag inte behandlar i denna bok ar `ed`. `Ed` är precis som `vi` en standard editor som man kan anta finns i alla system. Det är, till skillnad från `vi`, inte en fullskärmseditor utan är en kommandoradseditor. Det vill säga du jobbar från en kommandorad. Anldningen till att den inte behandlas här är att den inte används särskillt mycket och att jag själv inte kan använda den.

Övningsuppgifter

Uppgift 5.1 Emacs har en "tutorial" du startar den genom att i Emacs trycka Ctrl + H T. Starta den och gör den.

Uppgift 5.2 Vim har en liknande tutorial. Den heter `/usr/doc/vim-5.1/tutor/tutor` (för version 5.1) kopiera den till din hemmakatalog och öppna den i `vim`. Gör det som står i den. Hittar du inte filen så starta `vim` utan argument och ge kommandot **`:help tutor`** så får du reda på var filen ligger.

Kapitel 6

Fönstersystemet X

Detta kapitel handlar om hur man använder fönstersystemet X eller bara X som det kallas. I bland ser man att namnet Xwindows[1] används för att hänvisa till denna mjukvara. Detta tyder på att man aldrig har tittat på manualsidan för X som har följande text i sin inledning:

```
The X Consortium requests that the following names be used
when referring to this software:

                              X
                      X Window System
                        X Version 11
                X Window System, Version 11
                             X11

X Window System is a trademark of X Consortium, Inc.
```

Personligen så anvnder jag namnet X då det är klart vad man menar. Då det kan vara oklart använder jag namnet fönstersystemet X som är en översättning av det andra av X konsortiets alternativ. Använder du något annat namn än de som nämns här så kommer du att uppfattas som okunnig av vana UNIXanvändare i till exempel nyhetsgrupper.

Har man installerar X finns beskrivet i kapitel 20.1 på sidan 189.

6.1 Vad är X

X är ett plattformsoberoende grafiskt gränssnitt. X finns till de flesta plattformar och operativsystem. Du kan till exempel köra en X-server på en maskin med Windows 95 som operativsystem.

6.2 Historia

X är inte ett program utan en specifikation för hur X skall fungera. Den implementation som vanligtvis används i Linux heter XFree86 och är helt fri och gratis.

X började utvecklas 1984 vid MIT (Massachusetts Institute of Technology). Målsättningen var att skapa ett plattformsoberoende grafiskt gränssnitt. Inblandade företag var Digital och IBM.

[1] Detta är enda gången jag skriver detta fula ord.

Första versionen kom 1986 och kallades X10.4 som är en förkortning för X Window System version 10, release 4.

1998 bildades X-konsortiet. X-konsortiet består av alla större datorleverantörer. Medlemmarna styr på inget sätt utvecklingen utan får bar information om hur utvecklingen fortlöper.

X har idag nått version X11.6 Implementationen XFree86 har nått version 3.3.

6.3 Uppbyggnad

Eftersom X skapades för UNIX är det konstruerat med nätverk i åtanke. Men det kan användas lika bra på fristående maskiner. X är platformsobroende och finns till de flesta platformar. Vi börjar med lite grundläggande begrepp.

Arbetsplats (eng. Display) En arbetsplats, eller arbetsstation, är den maskin vid vilken användaren sitter. En arbetsplats har minst en bildskärm och exakt en X-server. Arbetsplatsen har också ett tangentbord och en mus.

Bildkärm (eng. screen) Bildsärmen är den monitor som användaren tittar på. En avancerad arbetsplats kan ha flera skärmar även om det är ovanligt.

X-server Varje arbetsplats kör en X-server. Det är X-servern som ritar allt som skall visas på bildskärmen. X-servern kontrollerar också tangentbordet och musen.

X-klienter Ett X-klient är det program som användaren exekverar. X-klienten körs antingen på samma maskin som X-servern eller på en annan (ofta kraftigare) maskin. En X-server kan serva flera X-klienter.

Fönsterhanteraren Fönsterhanteraren (som också själv är en X-klient) styr alla X-klienters fönster på bildskärmen.

Nu vet vi villka komponeneter ett X system består av. Men hur fungerar de tillsammans? Det skall jag försöka förklara här.

Kom ihåg att en server är mjukvara och inte hårdvara. Många personer är vana att kalla den stora maskinen i nätverket (t.ex. en filserver) för servern och de små maskinerna för klienter. Detta är ofta riktigt eftersom den stora maskinen servar de mindre klienterna. Det finns dock inget som hindrar att man kör en server på en klient som servar något på den stora datorn. Precis detta är fallet då man kör X.

X-servern körs alltid lokalt vid den arbetsplats användaren sitter. Detta för att den kräver mycket kontakt med den aktuella hårdvaran. X-servern servar X-klienterna med de tjänster de vill ha. Till dessa hör att rita ut deras grafik så att användaren ser dem. Tjänsterna innebär också att skicka information från arbetsplatsen till X-klienten. Om du till exempel trycker på en tangent eller klickar med musen så skickas den infomationen, av X-servern, till X-klienten som tar hand om ditt indata.

X-klienten är det program du vill köra. T.ex. xterm, netscape eller gimp. Programmet utnyttjar de tjänster X-servern erbjuder dem för att bli synliga för dig. X-klienterna kan köras på din maskin lokalt eller på en avlägsen maskin.

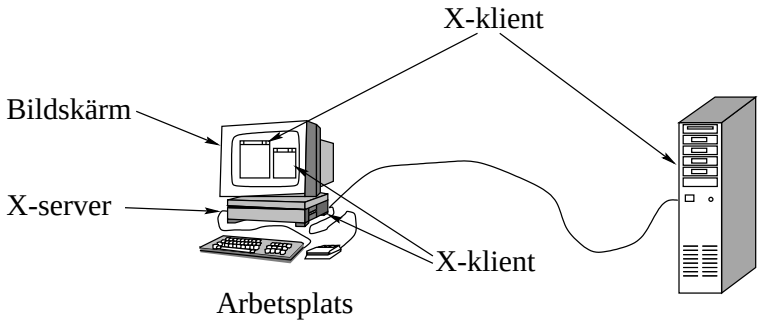
Den tredje delen, fösnterhanteraren är egentligen också en X-klient. Men eftersom den är lite speciell så behandlar vi den för sig. Till skilland från andra X-klienter så har fönsterhanteraren inget eget fönster (möjligen rotfönstret) utan intresserar sig istället för alla andra fönster på skärmen. Den ser också till att du får tillgång till menyer och ikoner för att starta olika program (X-klienter!) Vi tar ett exempel; X-klienten netscape utnuttjar X-serverns tjänster för att visa dig programmet netscape. Alla knappar och menyer i programmet vill netscape att du får se. X-servern ser till att det sker. X-servern ser också till att netscape får reda på om, och vilken, knapp du eventuellt klickar på. Men du vill förmodligen ha en ram runt netscape. Du vill kunna växla mellan netscape och andra program som du kör. Du vill kunna flytta på netscape över skärmen, du vill kunna förminska/förstora netscape. Vilken del skall ta hand om detta. Jo det gör fönsterhanteraren! Fösterhanteraren är en X-klient som sköter det ovanstående. Det finns flera olika fönsterhanterare att välja mellan. De ser olika ut och fungerar lite olika. Vilken man föredrar är en smaksak. Vissa är dock väldigt krävande och kan inte köras på klena maskiner. I figur6.1 visas en bild på ett enkelt nätverk med två datorer där den ena är en X-arbetsstation.

6.4 Starta X Window System

Det finns två sätt att starta X. Antingen är X-servern redan i gång på din maskin då du loggar in. Du möts då av en ruta som den i figur4.1. Eller så loggar du in i så kallat konsolläge.

6.4.1 xinit, startx

Om du loggar i i konsolläge så kan du, förutsatt att X finns installerat, starta X med kommandot **`xinit`**. `Xinit` är ett program som startar en X-server på din maskin och startar några program så att du kan börja jobba. `Xinit` tar några argument för att veta vad det skall starta. Dessa kommer att behandlas i senare versioner av denna bok. I de flesta Linuxsystem



Figur 6.1: X – Uppbyggnad

I denna figur ser vi hur komponenterna i X samarbetar. X-klienten till vänster på skärmen exekveras på de stora datorn till höger. Men den visas på arbetsplatsen på skärmen vid arbetsplatsen av X-servern. Den andra X-klienten exekveras lokalt på arbetsplatsen. Även den använder X-servern för att visa sin grafik och för att få sina indata från tangentbord och mus. Observera även att maskinen till höger inte bara är en filserver. Den ena X-klienten exekveras på den maskinen.

så finns ett skript som bruka heta `startx` som gör alla nödvändiga inställningar. Vill du starta egna program skall du ändra i filen `.xinitrc` i din hemmakatalog, `xinit` läser denna vid start av X. Alltså för att starta X från konsolen ger du kommandot:

```
$ startx
```

6.5 xdm

xdm är ett alternativ till den vanliga inloggningen med *login*. Då xdm används startas X direkt och visar en grafisk inloggningsruta. Desktopsystemet GNOME och KDE har egna varianter till xdm som är mer avancerade med till exempel möjligheter att stänga av datorn, välja användare genom att klicka med muspekaren mm.

Under inloggningen så läses filen `.xsession` varför du kan lägga till program som skall startas och andra inställningar i den.

Här skall jag djupare förklara vilka filer som läses in vid uppstart av X. Fast bara ur användarsynpunkt. De andra skall behandlas i sysadmin delen.

6.6 Allmänna flaggor till X-klienter

De flesta X-klienter kan ta standardiserade flaggor. De vanligaste presenteras i detta avsnitt. Mer information om dessa finns i manualsidan för X, man X.

6.6.1 Vilken skärm, `-display`

6.6.2 Storlek ock placering, `-geometry`

Med flaggan **`-geometry`** kan man på ett enkelt sätt styra var x-klienten skall öppnas och hur stort fönstret som öppnas skall vara. Efter flaggan **`-geometry`** anger man ett argument som beskriver hur fönstret skall vara. Syntaxen är *breddx-höjd+x+y* där bredd och höjd är fönstrets storlek i antingen pixlar eller tecken beroende på vilken typ av applikation det

är. Är det en textbaserad applikation (till exempel ett terminalemuleringsprogram så är det tecken, annars är det pixlar) X och y är antalet pixlar mellan skärmens vänstra (x) och övre (y) kant. Till exempel så öppnar **`xclock -geometry 100x100+0+0`** en `xclock` med storleken 100 x 100 pixlar i skärmens övre vänstra hörn. Om man vill referera från skärmens högra och nedre kant byter man ut plustecknen mot minustecken. Till exempel så startar **`xclock -geometry 100x100-0-0`** en klocka i skärmens nedre högra hörn. Hur detta kan kombineras visar jag med följande små exempel.

- 100x100+0+0 Bredd 100, Höjd 100, Övre vänstra hörnet.
- 50x100+10-0 Bredd 50, Höjd 100, 10 pixlar från vänster skärmkant, mot skärmens nedre kant.
- 100+100 Bredd 100, Höjd 100, Standard placering.
- +50+50 50 pixlar åt höger och 50 pixlar ned från skärmens övre vänstra hörn, standardstorlek på fönstret.

6.6.3 Färger, `-bg` och `-fg`

De flesta program kan ta emot flaggor som specifierar vilka färger saker och ting skall ha. De två vanligaste är -fg och -bg som sätter för- och bakgrundsfärgen. Till exempel så startar **`xterm -fg green -bg blue`** ett xtermfönster med de specifierade färgerna. Det finns flera färger man kan specifiera på detta sätt. Läs manualsidan för X för mer information.

Vilka färger finns

Vilka färger som finns tillgängiga på ditt system kan du se i textfilen `rgb.txt` den kan ligga på lite olika ställen men vanligast är att den ligger i katalogen `/usr/X11R6/lib/X11`. Har du din X installation i en annan katalog än `/usr/x11` så ligger den förmodligen i `/lib/X11` under den katalog du har ditt X installerat.

I filen `rgb.txt` listas alla färgen som X känner igen. De listas med ett värde och ett namn. Du kan titta i den efter vilka färger som finns definierade i ditt system.

Man kan även ange en färg med dess RGB-värde. Det gör man enligt följande syntax.

```
rgb:tal/tal/tal
```

där talen är antingen antingen 4, 8, 12 eller 16 bitar. Talen uttrycker i hexadecimala tal. Alltså 0-f, 00-ff, 000-fff, 0000-ffff.

Värderna anger hur stora delar av färgerna Röd, grön och blå som en färg består av. Man kan se det som att man blandar till färgen man vill ha med hjälp av tre färgburkar. Tar man ingen färg alls ur burkarna får man färgen svart (rgb:0/0/0). Tar man all färg ur alla burkarna får man vit (rgb:fff/fff/fff). Observera att rgb:f/f/f och rbg:ffff/ffff/ffff är samma färg (vit). Alla representerar all färg ur alla burkar. Bara det att det är uttryckt i olika färgdjup. Ju fler antal bitar man har i de olika värdena desto fler färger kan man blanda till. Sedan är det upp till grafikhårdvaran hur många som kan visas. För att blanda till gul, som är en blandning av röd och grön anger man rgb:f/f/0. Vill man ha en ljusare gul (närmare vit) så får man slaska i lite blå färg också. En ljus gul kan alltså vara rgb:f/f/6. Här följer en lista på några vanliga färger och deras rgb-värden:

grundfärger: svart rgb:0/0/0 röd rgb:ffff/0/0 grön rgb:0/ffff/0 blå rgb:0/0/ffff gul rgb:ffff/ffff/0 magenta rgb:ffff/0/ffff cyan rgb:0/ffff/ffff vit rgb:ffff/ffff/ffff

andra färger grå ljusblå

Man kan till exempel starta ett xterm-fönster med blå bakgrund kan man ange kommandot **`xtern -bg rgb:0/0/f`**.

6.7 Resursdatabasen

Om man inte vill starta sina program med färgerna med mera som argument till dem så kan man lagra denna information i något som kallas resursdatabasen. Denna skapas när X startar utifrån de defaultvärden som finns. Men man kan även lägga till egna. Det gör man vanligtvis genom att uppdatera sin `.Xdefaults` fil som ligger i hemmakatalogen. Man måste också se till att kommandot **`xrdb /.Xdefaults`** körs då X startas. Det gör det vanligtvis utan att du som användare behöver tänka på det, men inte alltid. Om du ändrar filen kan du naturligtvis köra kommandot från kommandoprompten för att uppdateringen skall genomförras.

6.8 Användbara program i X

6.8.1 xmessage

`Xmessage` är ett litet program som används för att skicka smo meddelanden vanligtvis till dig själv. Det kan användas för att skicka meddelanden till andra också men då måste du ha tillgång till deras display.

6.8.2 xwininfo

`Xwininfo` är ett program med vilket du kan få information om de X-klienter du kör.

6.8.3 xsetroot

Ibland är det trevligt att ha en snygg bild som bakgrund. Eller kanske bara en mörk skön färg. Detta kan du åstadkomma med programmet `xsetroot`.

xloadimage

Xloadimage är ett program som används för att visa bilder. Det kan visa bilder i ett X-fönster eller i rotfönstret. Xloadimage kan anropas som `xview` eller `xsetbg` då det direkt visar en bild i ett fönster eller i rotfönstret.

6.8.4 xwd

Om man vill visa någon (helst någon som tror att Linuxär ett kommandobaserat system som ser ut ungefär som MS-DOS) vilken trevlig mijö du jobbar i kan de vara smidigt att ta en skärmdump. Det gör man med programmet `xwd` (X Window Dump). Programmet kan dumpa hela skärmen eller bara ett fönster. Fönstret kan dumpas med, eller utan, dekorationer.

6.8.5 xset

Med `xset` kan man ange sina inställningar för hur X skall uppföra sig. Till exempel om skärmsläckare skall används. Om terminalen skall låta eller ej (skönt att stänga av om man ofta använder tabkomplementering). Här beskrivs några av de vanligaste sätten att anropa `xset`. XXXXXXXXXXXXXXXX

6.8.6 Alternativ skärmsläckare, xscreensaver

I X finns en inbuggd skärmsläckare. Den är väldigt enkel och kanske inte så kul att använda. Då kan man i stället använda `xscreensaver` som innehåller en mängd olika skärmsläckare.

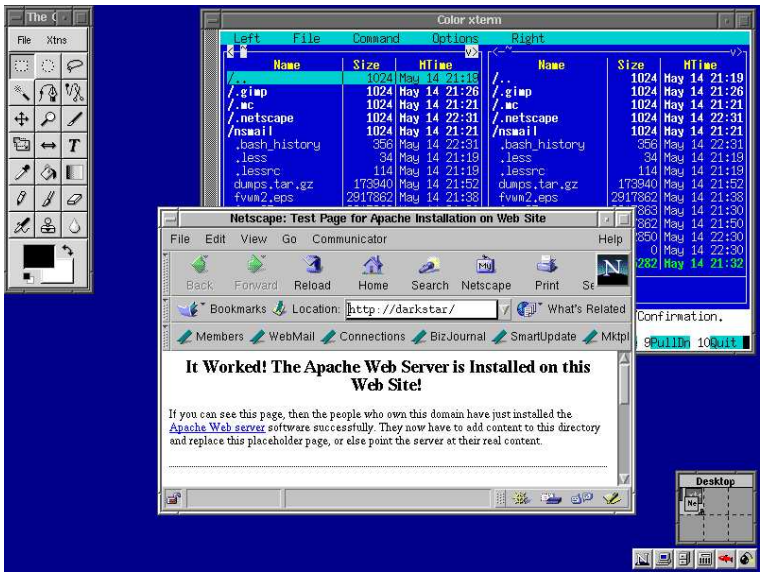
6.8.7 xclock

`xclock` visar en analog eller digital klocka på skärmen. Det finns en mängd andra klockor som integrerar sig fint med olika fönsterhanterare, men detta är urklockan i X.

6.8.8

6.9 Fonter i X

Vi har tidigare sett hur xset används i samband med fonter. I denna sektion kommer vi att djupare behandla fonter.



Figur 6.2: Fvwm2 – Free Virtual Window Manager version 2

Fvwm2 är en mycket praktisk fönsterhanterare. Den är smidig och ändå kraftfull.

6.9.1 `xlsfonts`

6.9.2 TrueType fonter

6.10 Fönsterhanterare

En fönsterhanterare är ett program som gör X-servern trevligare att arbeta vid. Utan fönsterhanteraren kan X-servern visa programfönstren och bilder. Du kan dock inte flytta fönstren. Figur 6.2 – 6.6 visar olika fösterhanterare. Vilken man väljer beror helt på tycke och smak.

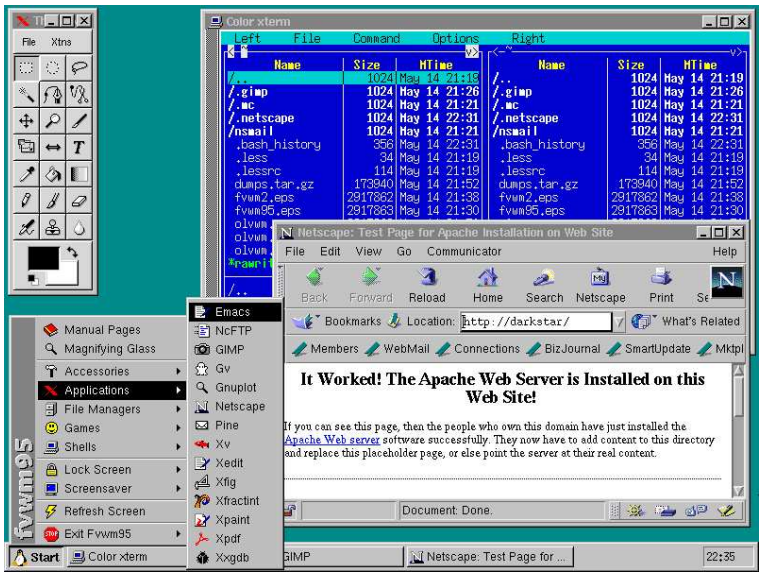
6.11 Desktopsystem

Ett desktopsystem är mer än en fönsterhanterare. Idag fi nns det två stora fönstersystem till Linux[2] nämligen GNOME och KDE. Vad dessa system gör för dig är små fi nesser som till exempel att kunna lägga fi ler på skrivbordet, klippa och klistra mellan program, mm, mm. XXXXXXXXXXX Fyll på med mer XXXXXXXXXX

6.11.1 GNOME

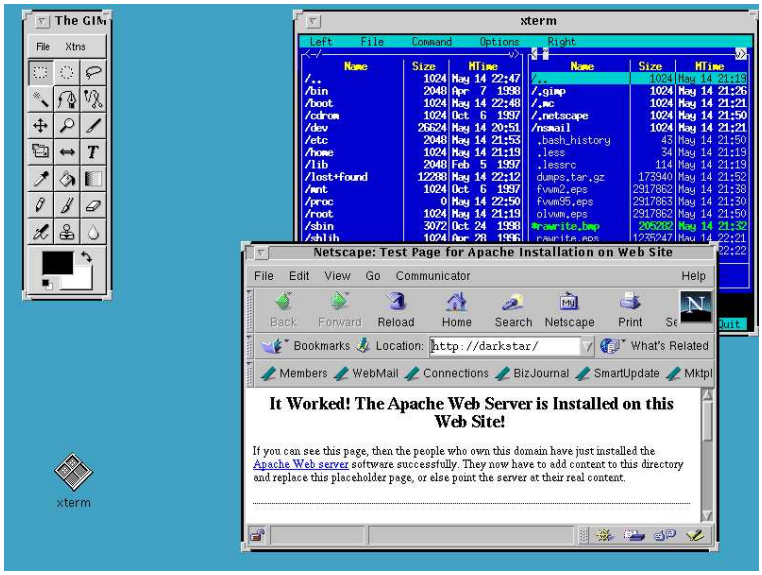
GNOME är en förkorting för *GNU Network Object Model Environment* vad nu det kan betyda. GNOME tillhandahåller själva desktopsystemet. Till det kan du använda vilken fönsterhanterare som helst. Du får ut mest av GNOME om du använder en GNOME-kompatibel fönsterhanterare. Idag är det bara Enlightenment som är det.

[2] Det fi nns fler, men de som behandlas här är fria och kan laddas hem av alla.



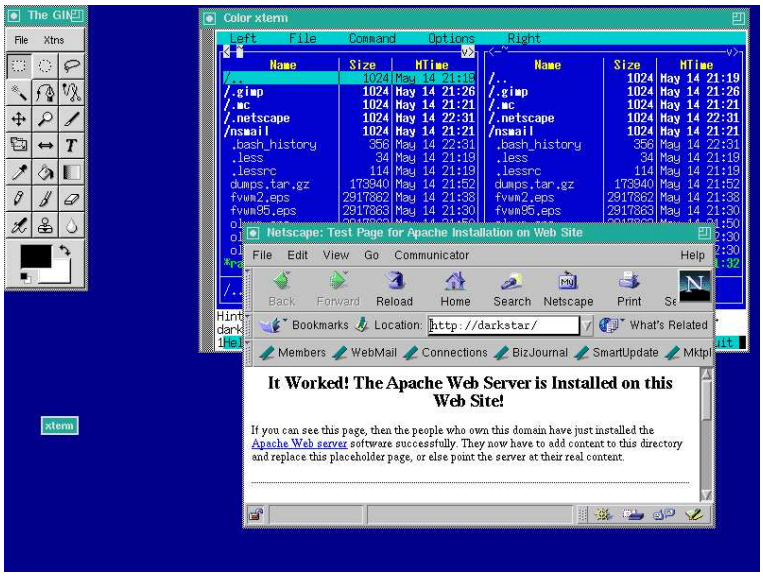
Figur 6.3: Fvwm95

Fvwm95 är egentligen Fvwm2 som konfigurerats att så lång det är möjligt likna och kännas som Windows95.



Figur 6.4: olvwm – Open Look Virtual Window Manager

olw(v)wm finns i två varianter. Med och utan "v" i namnet. Det som skiljer är funktionen med virtuella skrivbord



Figur 6.5: twm -Tab Window Manager

twm är en av de enklaste fönstarhanterarna.

6.11.2 K Desktop Environment, KDE

KDE är det idag största Desktopsystemet för Linux. KDE är en förkortning för *K Desktop Environment* Bokstaven K står inte för någonting utan fi nns bara där. Förkortnngar är lite heligt i Unix-världen ofta verkar det som att betydelsen av en förkortning väljs så att forkortningen skall låta bra. Figur 6.6 visar ett exempel på hur KDE kan se ut.

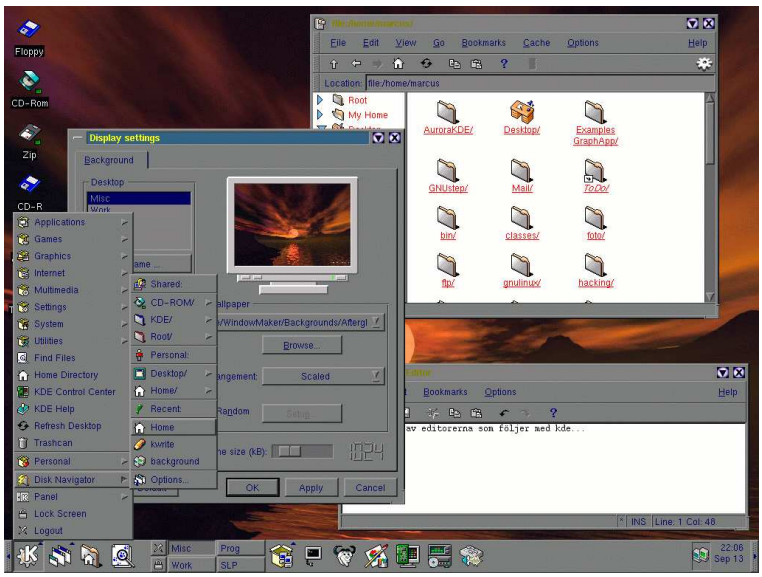
6.11.3 Common Desktop Environment, CDE

CDE är en kommersiell fönsterhanterare. Jag rekommenderar inte att du köper den till Linux men den är vanlig i fera varianter av UNIX, så det kan vara bra att känna till den.

Övningsuppgifter

Uppgift 6.1 Skicka ett meddelande till dig själv med programmet **`xmessage`**.

Uppgift 6.2 Skicka ett meddelande till dig själv med programmet **`xmessage`**.



Figur 6.6: KDE – K Desktop Envinonment

KDE är inte bara en fönsterhanterare utan ett helt desktop system. Det innebär att den tillhandahåller en hel del finesser som inte en vanlig fönsterhanterare gör

Del III

Gå vidare

Kapitel 7

Gå vidare

I detta kapitel beskrivs små saker som kan hjälpa dig i ditt dagliga arbete i Linux. Det här är saker som jag inte lyckats passa in i andra kapitel.

7.1 Textfiler

Ofta vill man bara läsa en textfil. Då kan det kännas onödigt att starta upp en texteditor bara för detta. Det finns flera verktyg för detta i ett Linuxsystem. Dessutom kan man ofta vilja ha innehållet i en textfil skrivet på standard utmatningsenheten för att kunna "pipa" den vidare in i ett annat kommando. Detta är mycket vanligt i skalprogrammering.

7.1.1 `cat`

För att titta på en fil kan man använda `cat`.

Detta är egentligen inte vad `cat` är avsett för att göra men det fungerar utmärkt eftersom det skriver ut en fil på standard utmatningsenheten. Det `cat` gör är att helt utan att formatera filen skriver ut den på skärmen.

> **Tips!**
> *Om du för* ***`cat`*** *på en binärfil kan det hända att din terminal bara visar konstiga tecken. För att återställa detta ger do kommandot* ***`reset`***. *Du får förmodligen skriva det utan att se det men det fungerar i alla fall.*

7.1.2 `more`

`More` (och `less` nedan) är så kallade *pagers*. Dessa används för att läsa data från standard utmatningsenheten (kan vara utdata från ett program eller en textfil). Det dessa program tillför är att du kan gå fram och tillbaka i datamängden och även söka efter text i den.

`more` more är ett mer sofistikerat kommando för att granska textfiler. Det låter dig studera en skärmsida i taget. Då du läst en sida trycker du på mellanslagstangenten för att visa nästa sida.

`More` har kommandon som liknar de som editorn `vi` har. För att söka efter ett reguljärt uttryck[1] så är kommandot **`/<reguttryck>`**. För att upprepa sökningen ger man kommandot **`n`** som i *Nästa*. Du orienterar dig i filen med mellanslag som flyttar fram en skärmsida. Enter flyttar fram en rad i taget och B flyttar dig bakåt i filen. Observera att `more` automatiskt avslutas då slutet på filen nås. Läs även om `less` här nedan som på många sätt är ett mer avancerat program för att granska textfiler. Manualsidan ger som vanligt också mer information.

[1] Läs om reguljära uttryck i avsnitt 7.5 på sidan 7.5

7.1.3 less

```
$ whatis less
less (1)             - opposite of more
```

less är ett ännu bättre program än more då det gäller att läsa textfiler. I stället för att trycka på mellanslagstangenten för att studera en sida kan du i less vandra runt i filen med piltangenterna. Detta är mer intuitivt och upplevs som enklare av de flesta. Att less är bättre än more stämmer väl in på ett klassiskt Unix-uttryck "less is more". Även less har en mängd avancerade funktioner. Vill du ta större steg än en rad i taget så kan du ta större (default 8 rader) med piltangenterna höger och vänster. Du kan söka efter ett reguljärt uttryck på samma sätt som med more, det vill säga **/<reg uttr>** söker nedåt i filen och **?/<reg uttr>** söker uppåt i filen. Kommandot **n** upprepar senaste sökningen och **N** upprepar sökningen fast åt motsatt håll i filen. FÖr ytterligare information hänvisar jag som vanligt till manualsidan.

7.1.4 tail

tail används för att studera slutet på en fil. En del filer är väldigt stora och du kanske bara vill se vad som hänt i slutet på dem. Detta är ofta fallet med exempelvis logg-filer. Vill du i realtid se vad som händer i en fil, till exempel systemloggen om du är administratör, kan du starta tail med flaggan **-f** i ett terminalfönster. Slutet på filen kommer då alltid att visas i det fönstret även om mer data läggs till i filen.

7.1.5 head

head fungerar precis lika som tail fast på början av en fil. Head kan du inte använda med flaggan **-f**, eftersom den tittar i börjag av filen. Där kan data inte läggas till.

7.2 Slå ihop filer mm. cat

cat nämndes i föregående avsnitt. cat är dock egentligen en förkortning för *CATinate* vilket antyder att programmet är till för att slå ihop filer. cat tar de filer som ges som argument och skriver ut dem på standard utmatningsenheten. Outputen från programmet kan dock med fördel styras om till till en fil.

Exempel. Kommandot:

```
namatj:~$ cat fil1 fil2 fil3 > fil4
```

Slår ihop filerna fil1, fil2, fil3 till fil4. I fallet ovan had vi lika gärna kunnat skriva

```
namatj:~$ cat fil* > fil4
```

förutsatt att det bara är filerna ovan som matchar uttrycket fil*. Vidare måste man tänka på att filnamnsmönstret listar filerna i den ordning man vill att de skall slås ihop. Det är klokt att först killa det med till exempel ls.

```
namatj:~$ ls fil*
```

cat kan även användas till andra saker. Eftersom cat inte gör något med filerna som ges som argument kan man använda cat till att till exempel skicka filer till olika specialfiler. Ett exempel:

```
namatj:~$ cat /usr/share/sndconfig/sample.au > /dev/audio
```

Spelar upp filen sample.au på systemets högtalare.

7.3 Hitta filer, find och locate

Ofta vill man veta var i systemet en viss fil finns. I Linux finns det två bra kommandon för att hitta en viss fil. De fungerar på lite olika vis.

7.3.1 find

Find är ett mycket flexibelt kommando. Som med de flesta kommandon i denna boks tidiga skede går jag bara igenom de vanligaste alternativen. Find söker rekursivt igenom ett filträd med en katalog du anger som rot efter filer som matchar de kriterier som du anger. Du kan även ange att ett kommando skall utföras på varje fil. Detta är väldigt smidigt för alla, men kanske framför allt för systemadministratören.

Find har följande syntax:

```
$ find <rot> <uttryck>
```

Det vanligaste är att man söker efter en fil som har ett viss namn. För att söka efter filer som heter core i din hemmakatalog är kommandot;

```
$ find ~ -name core
```

Hemmakatalogen (=~) är rot för sökningen. **-name** anger att vi skall söka efter filnamn. Man kan även söka efter filer som passar ett visst filnamnsmönster. Filnamnsmönstret måste då omges med citationstecken ("). Följande exempel söker efter alla filer i hela systemet som slutar på tilde (~[2]).

```
$ find / -name "*~"
```

En annan sak som kan vara smidigt att söka på är filernas tidsstämpel. Till detta finns det flera olika alternativ. För att söka alla filer som modifieras den senaste timmen ger man kommandot:

```
$ find / -mmin 60
```

Och för att hitta alla filer som modifierats den senaste veckan ger man kommandot:

```
$ find / -mtime 7
```

Det sista som jag kommer att ge exempel på är hur man kan hitta filer som ägs av en viss användare eller grupp. Detta görs smidigast med alternativen **-user** eller **-group** följt av användaren eller gruppens namn. Om du titt som tätt jobbar som root kanske du får filer i din hemmakatalog som tillhör root. Dessa hittar du med komamndot

```
$ find ~ -user root
```

Find har flera andra spännande alternativ. Detta avsnitt kommer att växa i framtida versioner har jag tänkt. Tills vidare för jag hänvisa till manualsidan.

7.3.2 locate

Locate är snabbare än find. Dock kräver det att din systemadministratör kört ett program som skapar en databas över alla filer på systemet[3]. Find söker sedan i den databasen efter den fil du letar. Eftersom du inte söker i realtid så kommer resultatet du får att vara så som systemet såg ut då databasen skapades. Det gäller alltså att din systemadministratör uppdaterar databaseb ofta. Vanligt brukar vara att den körs varje natt. Vissa systemadministratörer väljer att inte använda locate över huvud taget. Ok hur fungerar det nu då?

Det eklaste sättet är att bara anropa locate med ett filnamn som argument. Skriver du bara en textsträng så kommer locate att lista alla filnamn som innehåller strängen i filnamnet. Det går också att söka efter ett filnamnsmönster. Då måste mönstret omges av apostrofer (’) för att filnamnsmönstret inte skall substitueras av skalet.

Kommandot

```
$ locate kalle
```

Hittar alla filer vars filnamn innehåller ‘kalle’.

```
$ locate '*kalle*'
```

[2] Sökningar över hela filsystemet kan ta ganska lång tid . . .
[3] är du din egen administratör så kan du läsa om detta på sidan **??**

hittar också alla fi ler vars fi lnamn innehåller ‘kalle’.

Observera att

```
$ locate *kalle*
```

kommer att, av skalet, bytas ut mot en lista av alla fi lnamn, i aktuell katalog, som matchar fi lnamnsmönstret.

7.4 Hitta information om kommandon

Ibland funderar du på vad ett kommando gör. Ett gott råd. Prova inte utan att veta utan använd istället den information som systemet kan ge dig.

7.4.1 `whatis`

7.4.2 `apropos`

7.4.3 `which`

Du startar ju ett program genom att ange dess namn på kommandoraden. Men vilken fi l är det igentligen som du kör. Programmet `which` talar om det för dig. Anropa `which` med programmet du undrar över som argument så får du reda på var det ligger. `Which` kan med fördel användas tillsammans med kommandosubstitution som behandlas i avsnitt 7.16.9 på sidan 135 för att till exempel editera ett skript.

```
$ which emacs
/usr/bin/emacs
$
```

7.4.4 `whereis`

Kommandos `whereis` ger information liknande den du får med `which`. `Whereis` fungerar dock på ett helt annat sätt och ger dig lite mer information.

```
$ whereis emacs
emacs: /usr/bin/emacs /usr/lib/emacs /usr/share/emacs /usr/man/man1/emacs.1
$
```

7.5 Reguljära uttryck

Reguljära uttryck används för att matcha text i olika sammanhang. Förväxla inte dessa med filnamnsmönster som vi gick igenom tidigare i denna bok. Reguljära uttryck är betydligt mer kraftfulla och mycket bra att kunna.

De uttryck som behandlas här är så som de beskrivs i standarden *POSIX 1003.2 regular expresseions*. Det görs en uppdelning mellan utökade och grundläggande reguljära uttryck. Här behandlas bara de utökade. I slutet på detta stycke beskrivs vad som inte kan göras om man bara har tillgång till de grundläggande reguljära uttrycken. Till exempel editorn `ed`. Är man inrtresserad så kan man lära manualsidorna `regex(7)` och `ed(1)`.

Alla reguljära uttryck skall vi testa på följande text.

BAD MOON RISING
John C Fogerty

I see the bad moon rising,
I see trouble on the way
I see earthquakes and lightning,
I see bad times today

Don't go around tonight,
well it's bound to take your life
There's a bad moon on the rise

I hear hurricanes blowing,
I know the end is coming soon
I fear rivers overflowing,
I hear the voice of rage and ruin

Dont go around...

Hope you got your things together,
hope you are quite prepared to die
Looks like we're in for nasty weather,
one eye is taken for an eye

Dont go around...

Jag kommer att skriva ett reguljärt uttryck, sedan kommer jag att skriva ut raden där det finns en matchning. Själva matchningen är skriven i fetstil.

Alla tecken som har speciell betydelse listas i tabell 7.1

.	Matchar ett valfritt tecken
*	Tecknet eller yttrycken omedelbart före kan förekomma ingen eller godtyckligt antal gånger
+	Tecknet eller uttrycket omedelbart före kan förekomma en (1) eller flera gånger
?	Tecknet eller uttrycket omedelbart före kan förekomma noll (0) eller en (1) gånger
{i}	Uttrycket före matchas exakt i gånger ($0 \geq i \geq 255$).
{i,j}	Uttrycket före matchas minst i och max j gånger
{i,}	Uttrycket före matchas minst i gånger.
^	Om det står först så matchar efterkommende uttryck bara om det står först i en rad. På annat ställe så matchar det sig självt.
$	Om det står sist så matchas bara uttrycket om det står sist på en rad. Annars matchar det sig själv.
\	Gör att tecknet efter matchar sig självt även om det är ett specialtecken. Detta gäller inte tecknen {, }, (,), >, <, b, B, w, W, + och ? eftersom de har andra betydelser när de står före \.
[abc]	Mängd tecken. Matchar något av tecknen inom parentesen en (1) gång.
[a-z]	Intervall. Man kan även ange instervall.
[^a-z]	Matchar alla tecken UTOM de som ingår inom hakparentesen en (1) gång.

Tabell 7.1: Specialtecken vid reguljära uttryck

Vi börjar med att titt på tecknet '.' (Punkt) som matchar ett valfritt tecken.

`t..e` får följande matchningar;

*I see bad **time**s today*

*well it's bound to **take** your life*
*Hope you got your things **toge**ther,*
*one eye is **take**n for an eye*

Antag nu att vi vill hitta alla strängar som börjar på t och slutar på e oavsett hur många tecken som finns mellan dem. Då kan vi använda '*' (stjärnan) som gör att uttrycket framför matchas noll (0) eller flera gånger.

`t.*e` får följande matchningar;

*I see **the** bad moon rising,*
*I see **trouble on the** way*
*I see ear**thquake**s and lightning,*
*I see bad **time**s today*
*well i**t's bound to take your life***
*There's a bad moon on **the rise***
*I know **the e**nd is coming soon*
*I hear **the voice of rage** and ruin*
*Hope you got your **things togethe**r,*
*hope you are qui**te prepared to die***
*Looks like we're in for nas**ty weathe**r,*
*one eye is **taken for an eye***

Oj då. Redan på andra raden blev det lite konstigt. Där finns det ju flera träffar i vandra. I sådana fall matchas det längsta möjliga. Lägg också märke till att matchningen är helt radorienterad. Ingen matchning sker mellan raderna.

Nu kanske man vill hitta alla strängar som börjar på s och slutar på a och har exakt 6 tecken. Då kan man använda krullparenteserna, eller måsvingarna som jag kallar dem ({ och }) i dem kan man nämligen spefifiera exakt hur många gånger uttrycken före skall matchas. För att lösa frögeställningen ovan kan man alltså använda följande uttryck:

`s.{4}a` som får följande matchningar;

*I **see ea**rthquakes and lightning,*
*I **see ba**d time**s toda**y*
*There'**s a ba**d moon on the rise*

(Ovan hade naturligtvis **`s....a`** fungerat på samma sätt)

Man kan även använda måsvingarna på formen {i,j} då matchas uttrycket före minst i och max j gånger. Ett annat alternativ är {i,} som gör att uttrycket före matchas minst i gånger. Exempel på hur dessa fungerar lämnas som övning till läsaren.

Som du säkert märkt så görs ingen skillnad på mellanslag och andra tecken. Detta är ju lite synd om man vill hitta ord som följer ett mönster. Naturligvis så finns det bot på detta bara man är lite klurig. Till sin hjälp har man sekvenserna '\<' och '\>' som matchar början respektive slutet på ett ord. Okej, nu vill vi hitta alla ord som börjar

på a. Då får man först ställa sig frågan, vad är ett ord som börjar på ‘a’? –Jo det är ett “början på ord” direkt följt av ett a sedan kommer det ett valfritt antal bokstäver och slutligen kommer det ett ordslut. Observera att du inte kan använda punkten (.) för att matcha alla bokstäver utan man får använda hakparenteserna ([) och (]). Så här kan det gå till att hitta alla ord som börjar på a.

<a[A-Za-z]*\> som får följande matchningar;

*I see earthquakes **and** lightning,*
*Don’t go **around** tonight,*
*There’s **a** bad moon on the rise*
*I hear the voice of rage **and** ruin*
*Dont go **around**...*
*hope you **are** quite prepared to die*
*one eye is taken for **an** eye*
*Dont go **around**...*

Skall man göra det ovanstående många gånger kan det bli jobbigt att ange mängen av alla tecken som **[a-zA-Z]** eller något liknande. Därför finns det färdigdefinierade mängder att använda. Dessa kallas teckenklasser och visas i tabell 7.2 observera att dessa inte skall användas på svenska texter. Det är möjligt att det fungerar på din maskin om den är konfigurerad för att tala svenska[4], men räkna inte med att det fungerar överallt.

alnum	Alfa-Numeriska tecken [A-Za-Z0-9]
digit	Siffror [0-9]
punct	! ‘ # $ % & ‘ ( ) * + , - . / : ; < = > ? @ [/] ^ _ { | } ~
alpha	[A-Za-z]
graph	Matches any of the characters defined as a printable
character	except those defined to be part of the space character class.
space	Matches a tab, new line, vertical tab, form feed, carriage return, or space.
blank	
lower	[a-z]
upper	[A-Z].
cntrl	Alla som inte är med i; [:upper:], [:lower:], [:alpha:], [:digit:], [:punct:], [:graph:], [:print:], or [:xdigit:]
print	any printable character
xdigit	[0-9A-Fa-f]

Tabell 7.2: Teckenklasser

Man kan använda teckenklasserna på två olika sätt, [:upper:] matchar ett tecken som ingår i klassen upper. Man kan åckså använda formen [=+=] som matchar ett tecken som ingår i samma klass som ‘+’.

Personligen tycker jag att användandet av klasser gör att uttrycken blir svårare att läsa. Det är ofta enklare att räkna upp de tecken man avser inom hakparenteser. Men det är

[4] Läs om *locale* på sidan **??**

naturligtvis en smaksak.

Som jag sade tidigare så söks en rad i taget. Det finns även tecken som matchar början eller slutet på en rad. Dessa är ^ och $ som när de står först repektive sist i ett uttryck representerar just dessa. Antag att vi vill hitta alla rader som börjar på ett stort eller litet h. Vi kan då använda det reguljära uttrycket:

^[hH] som får följande matchningar;

***H**ope you got your things together,*
***h**ope you are quite prepared to die*

Observera att det bara är det första tecknet som matchar och inte hela raden. Man kan även matcha hela rader men detta är sällan nödvändigt. Se till exempel kapitlet om `grep`.

Vill man matcha hela raderna kan man naturligtvis använda följande uttryck

^[hH].*$ som får följande matchningar;

Hope you got your things together,
hope you are quite prepared to die

De reguljära uttryck vi behandlat här är de så kallade utöka reguljära uttrycken. Vissa program stöder bara de grundläggande reguljära uttrycken. I nästa stycke förklaras kort vad som skiljer.

7.5.1 grundläggande reguljära uttryck

Oftast fungerar ovanstående sätt att skriva reguljära uttryck. Men vissa program stöder bara de äldre grundläggande reguljära uttrycken.

Om man inte kan använda de utökade reguljära uttrycken så är det följande funktioner, beskrivna ovan, som inte fungerar:

- |, +, ? räknas som vanliga tecken och har ingen speciell funktion.
- Tecknen {, }, (,), matchar sig själva om de inte föregås av ett bakvänt snedstreck. Då fungerar de som ovan (snurrigt).
- ^, $ har bara funktion i början, resp slutet av ett uttryck, annars matchar de sig själva.

Önskas utterligare information så läs manualsidan regex(7).

7.6 Söka text i filer, grep

Nu när vi kan reguljära uttryck ligger det nära till hands att prova dem tillsammans med programmet grep. Detta är lång ifrån det enda användningsområdet för reguljära uttryck men ett nyttigt och bra sätt att öva dem på.

Grep (**G**rab **R**egular **E**xpression and **P**rint) fungerar så att det ger som utdata alla rader där ett reguljärt uttryck matchar. Syntaxen är, när den är som enklast, **grep <reg-uttr> <filnamn>** och returnerar alla rader i *filnamn* som innehåller strängar som matchar det reguljära uttrycket *reg-uttr*.

XXXXXXXXXXX

7.7 Schemalägga jobb cron och at

Om du har ett återkommende jobb som du till exempel vill skall utföras en gång i timmen kan du använda dig av något som heter cron. Vill du istället att ett jobb skall utföras en gång om exempelvis tre timmar kan du använda at

7.7.1 cron

cron är ett verktyg som används för att köra ett jobb med jämna mellanrum. Det kan vara ett kommando som du kör en gång varje vecka eller så och inte vill glömma. Om du gör det till ett “cron-jobb” behöver du inte tänka på det mera.

Crontabfilen är en fil som används för att schemalägga job. Allt som du brukar göra regelbundet kan du lägga i crontab-filen och sedan glömma bort. Är du systemadministratör finns det en hel del saker att lägga i crontabfilen.

Crond är en daemon som konstant kollar alla användares crontab-filer. Och då något skall göras så görs det.

Nedanstående exempel, som passar bra som just exempel, är faktiskt min crontab-fil. Observera hur jag lagt in en rad med hjälp för att kunna komma ihåg vilket fält som är vilket.

```
#
# Crontab for marcus, 970903
#

# min(0-59) hour(0-23) dayOfMounth(0-31) mounth(0-12) dayOfWeek(0-7)

0  5  *  *  *  $HOME/bin/daily.job
15 5  30 *  *  $HOME/bin/mounthly.job
30 5  *  *  6  $HOME/bin/weekly.job
```

För att slippa hålla reda på alla crontab-commandon så låter jag cron köra olika script som ligger i min hemkatalog. Där kan jag sedan peta in de jobb jag vill skall utföras.

Hur gör man då?

För det första så måste man skapa en crontab-fil. Detta gör man med kommandot **`crontab -e`**

7.7.2 at

`at` är ett verktyg som du kan använda för att utföra kommandon vid en annan tidpunkt. Det kan vara till exempel ett jobb som belastar systemet så till den grad att du gärna ser att det körs på natten. Till skillnad från `cron` så utförs ett “at-jobb” bara en gång.

`At` är även smidig för att skicka påminnelser till dig själv även om det kanske inte är det vanligaste användningsområdet.

7.8 Komprimera och packa upp filer

Ofta vill man att filer skall ta mindre plats. Det kan vara för att spara plats vid till exempel säkerhatskopiering. Det är också vanligt att man komprimererar filer för att spara bandbredd då filer skall flyttas mellan olika system.

7.8.1 `gzip` – `gunzip`

Det vanligaste sättet att komprimera filer är `gzip`. `Gzip` finns på alla system och är därför bra att använda. Man känner igen en “gzippad” fil på dess suffix. Suffixet är `.gz` eller i vissa fall `.tgz`.

För att hantera sådana filer använder man programmen `gzip` och `gunzip`. Båda programen är enkla att använda. För att komprimera en fil ger man helt enkelt commandot **`gzip filnamn`**. `Gzip` skapar då den komprimerade filen `filnamn.gz`. Observera att originalfilen tas bort efter komprimeringen. Man kan komprimera flera filer i taget genom att ge dem alla som argument till `gzip`. Vill man rekursivt komprimera hela filträd så kan man ange flaggan **`-r`** vilken gör att alla kataloger som anges som argument efter köringen bara kommer att innehålla komprimerade filer. Ävem underkataloger gås igenom rekursivt.

Att packa upp filer är lika enkelt. Då använder man programmet `gunzip`. Eller `gzip -d`. `Gunzip` är känsligt för vad filerna har för namn. Filerna måste sluta på något av `.gz, -gz, .z, -z, z` eller `.Z` för att packa upp dem. `Gunzip` känner också igen, och packar upp, `.tgz` och `.taz` som är förkortningar för `.tar.gz` och `.tar.Z`. Mer om dessa filtyper kan du läsa i avsnittet om `tar` nedan. För att packa upp en fil så skickar man den, eller de, bara som argument till `gunzip`. Flaggan **`-r`** kan användas på samma sätt som i samband med komprimering.

Om man inte vet vad en fil är för typ kan man använda kommandon `file` för att ta reda på det. Det är smidigt om du fått en, kanske, komprimerad fil av en kompis och filnamnet kanske har trunkerats på vägen förbi något konstigt program eller filsystem. `File` anropas med en fil som argument och svarar med vilken filtyp det är. `File` använder inte filnamnet för att avgöra vilken sorts fil det är.

7.8.2 bzip2 – bunzip2

bzip har en bättre komprimeringsalgoritm än gzip. Det fungerar användarmässigt på precis samma sätt. Dock skall man passa sig för att dela med sig av sina filer i "bzippat" format eftersom det inte finns installerat på alla system. Flaggan **-r** för att arbeta rekursivt finns inte i bzip2. I alla fall inte i den version jag har tillgånhg till nu (Version 0.9.0b).

7.8.3 tar – tar

De koprimeringsprogram vi tittat på hittils kan bara komprimera filer en och en. Ibland kan det ju vara praktiskt att komprimera flera filer till en enda stor komprimerad fil. Till detta kan man använda tar. Tar står för *Tape ARchiver* och är tänkt att användas för att kopiera filer till en tape. Programmet har dock flera andra användningsområden. Tar komprimerar inte filerna utan slår bara ihop dem till en enda stor fil. Det är vanligt att man först slår ihop flera filer till ett tar-arkiv och sedan komprimerar det med gzip eller bzip2. Tar är ett ganska avancerat program. I denna version av boken kommer jag bara att gå igenom det nödvändigaste, det som man använder ofta. De viktigaste flaggorna till tar är:

- c, Skapa nytt arkiv (Create)
- x, Packa upp ett arkiv (eXtract)
- t, Lista filerna i ett arkiv (lisT)

Någon av dessa (det finna flera också) måste vara med för att tar skall veta vad det skall göra. Andra flaggor som vi skall titta på är:

- v, Visa vad du gör (Verbose)
- f, Nästa argument är en fil, använd den istället för stdin eller stdout (file)
- z, Använd gzip för att komprimera eller dekomprimera

Det sista är smidigt då det är väldigt vanligt att distribuera programvaror och andra filer som .tar.gz. Använder du tar med flaggan **z** behöver du inte använda gzip utan behandlar filen som om det var ett vanligt tar-arkiv.

Tar använder filändelsen .tar. För att skapa ett tar-arkiv, (Kallas populärt för en *tarboll* på svenska). Använder man flaggorna c och f. och anger det filnamn mav vill att arkivet skall få. Kommandot

```
$ tar cf tarfil.tar *
```

Skapar en fil tarfil.tar i katalogen som innehåller alla filer i katalogen. När man sedan packar upp filen kommar alla filer att packas upp i den katalog som är aktuell eftersom ingen katalog information sparades i tarfilen. Vill man att det skall bildas

en ny kaalog då man packar upp arkivet måste man aven få in denna information i tar-arkivet. Med kommandot

```
$ cd ..
$ tar cf tarfil.tar katalog
```

kommer även katalogen `katalog` med i arkivet. Då det packas upp kommer katalogen filekatalog att bildas om den inte redan finns. Om man har ett arkiv som man inte känner till hur det är skapat så kan man titta på innehållet med flaggan **`t`**. Om filnamnen innehåller en katalog så kommer den att skapas vid uppackningen. Man kan naturligtvis skapa arkiv som innehåller flera kataloger och filer direkt i trädets rot.

Ofta då man skapat ett arkiv vill man komprimera det så att det skall ta mindre plats. Detta är så vanligt att `tar` kan göra det på en gång. Om man anger flaggan **`z`** så komprimerar, eller dekomprimerar tar filen med `gzip` på en gång. Du kan alltså hantera en `.tar.gz` eller `.tgz` fil precis som om det vore en vanlig `.tar` fil.

Slutligen skall vi titta på flaggan **`v`**. Om du ger flaggan **`v`** till programmet så visar det information om vilka filer som packats eller packats upp. Jag visar nu ett exempel på hur jag först skapar ett arkiv med två filer, sedan listar innehållet och slutligen packar upp dem. Anledningen till att jag bara tar två filer är att jag vill spara utrymme:

```
$ tar czvf tararkiv.tar.gz README katalog/fil
README
katalog/fil
$ tar tzvf tararkiv.tar.gz
-rw-rw-r-- marcus/marcus     0 1999-10-01 19:08 README
-rw-rw-r-- marcus/marcus     0 1999-10-01 19:08 katalog/fil
$ tar xzvf tararkiv.tar.gz
README
katalog/fil
$
```

Observera att den ena filen ligger i en underkatalog.

Det var allt jag tar om `tar` i den här versioen av boken (kanske aldrig kommer att bli mera heller). Vill du veta mer kan du xom vanlig läsa manualsidan.

7.8.4 `zip` – `unzip`

Ett annat program som kan slå ihop och komprimera filer är `zip`. `Zip` är kompatibelt med **Phil Katz** `PKZIP` som är mycket vanligt i PC-världen och bland annat används i `WInZip`[5] `Zip` har en mängd flaggor, är du intresserad av att lära dig alla får du läsa manualsidan. De vi gå igenom här är. XXXXXXXXXXXXX

7.8.5 `compress` – `uncompress`

`Compress` är ett gammalt format for komprimerade filer. Det anväds inte längre i någon större utsträckning. Du känner igen en fil på dess suffix som är `.Z`. Även det

[5] `WinZip` kan även packa upp fi ler som är tarade och gzippade.

vanligare programmet `gzip` kan packa upp dessa filer sp användandet av `compress` lär bli mindra och mindre. Nu vet du vad `compress` är. Vill du använda det hanvisar jag helt fräkt ännu en gång till manualsidan (vad gjorde man utan manualsidor?).1

7.9 Dela filer i delar – `split`

I bland kanske man vill flytta en fil mellan två system på diskett (förhoppningsvis är detta snart ett utdött förfarande) och upptäcker att filen är för stor. Det man då vill göra är att dela upp den i två (eller fler) mindre delar som var för sig får plats på en diskett.

Man kan definiera hur stora de nya filerna skall vara på två olika sätt. Antingen kan man definiera hur många rader varje fil skall bestå av. Detta är läpligt om man vill dela en textfil. Eller så kan man välja hur många byte varje fil skall vara. Detta är lämpligast om man skall dela en komprimerad fil för att den skall få plats på ett gäng disketter.

Nedan kommer ett antal exempel:

```
namatj:~$ split -l 1000 stor_text-fil
```

Ovanstående kommando delar filen i ett antal filer med 1000 rader i varje.

```
namatj:~$ split -b 1000 stor-fil
```

Ovanstående delar filen i delar om 1000 bytes.

Det kan vara jobbigt att ange storleken på delfilerna i bytes. Man kan ange vilken enhet man skall använda till exempel:

```
namatj:~$ split -b 95m stor-fil
```

Delar stor fil i delar om 95 MB. Istället för m så kan man ange b eller k. B anger filstorleken i antal 512 bytes och k anger storleken i kilobytes.

Då man sedan vill sätta ihop filerna igen använder man `cat` som beskrivs i avsnitt 7.1.1 på sidan 105. Split namnger filerna i ordning så att ett filnamnsmönster som `x*` listar filerna i rätt ordning.

7.10 Omdirigering

Följande gäller (som allt annat i denna bok) för skalet bash. I andra skal kan det skilja något. Men det brukar som regel se ut ungefär så här.

Som regel använder sig ett kommandoradsprogram av tre stycken filer (allt är ju filer i Linux) de filerna är `stdin`, `stdout` och `stderr`. Dessa filer ligger, som alla andra, specialfiler i katalogen `/dev`. Normalt så är `stdin` kopplad till tangentbordet och de

andra två till bildskärmen. Ett vanligt program som till exempel `sort`[6] tar sitt indata från `stdin` och presenterar utdata på `stdout`. Filerna fungerar så att allt som skrivs till `stdout` och `stderr` visas på skärmen. Allt som skrivs på tangentbordet kan läsas från filen `stdin`.

Det är dock inte alltid så smart att ha det så. Tänk till exempel att du vill sortera alla namn i en fil. Om `sort` måste få alla dessa namn från tangentbordet så måste du ju skriva av filen till programmet sort. Det behöver du ju naturligvis inte göra utan du omdirigerar indata till sort så att det läser från filen med namn istället.

`Sort` presenterar sedan resultatet (en sorterad lista av namn) på stdout (alltså bildskärmen). Det är ju inte så praktiskt om det är många namn. Smartare vore det ju om den sorterade listan hamnade i en annan fil. Detta åstadkommer du genom att omdirigera stdout till en annan fil. Hur detta går till skall vi snart behandla.

Inte alla program kräver något indata utan nöjer sig med utdata. Exempel på ett sådant är `date` som ju bara visar dagens datum om det anropas utan argument. Om du vill omdirigera utdata från ett program så använder du operatorn > studera följande exempel:

```
namatj:~/tmp$ date > datumfil
namatj:~/tmp$ cat datumfil
Tue Sep 14 19:39:14 CEST 1999
namatj:~/tmp$ date > datumfil
namatj:~/tmp$ cat datumfil
Tue Sep 14 19:39:23 CEST 1999
namatj:~/tmp$
```

Som du ser så skrivs filen över varje gång man använder omdirigeringsoperatorn >. Detta gör att omdirigering kan var en farlig operation. Man kan dock skydda sig mot detta genom att sätta omgivningsvariabeln “noclobber”. Studera följande exempel:

```
namatj:~/tmp$ set noclobber
namatj:~/tmp$ date > datumfil
namatj:~/tmp$ date > datumfil
bash: datumfil: Cannot clobber existing file
namatj:~/tmp$ cat datumfil
Tue Sep 14 19:48:16 CEST 1999
namatj:~/tmp$
```

I exemplet ovan ser man att man inte kan skriva över filen med den nya tiden.

Varning!
Blanda inte ihop tecknen | och > eftersom det senare skriver över filen.

Men tänk om man skulle vilja skapa en lista av tider. Den skall fyllas på var gång kommandot körs. Detta kan man göra med omdirigeringsoperanden “>>”. Studera följande exempel:

[6] sort kan också ta en fil som argument och in annan fil för utdata men det förstör resonemanget här...

```
namatj:~/tmp$ date >> datumfil
namatj:~/tmp$ date >> datumfil
namatj:~/tmp$ date >> datumfil
namatj:~/tmp$ date >> datumfil
namatj:~/tmp$ cat datumfil
Tue Sep 14 19:51:19 CEST 1999
Tue Sep 14 19:51:21 CEST 1999
Tue Sep 14 19:51:24 CEST 1999
Tue Sep 14 19:51:25 CEST 1999
namatj:~/tmp$
```

Tidigare så nämndes en tredje fil, stderr. Vad gör den för nytta då? –Jo ett program skiljer på utdata och felmeddelanden. Detta ser du ju inte normalt eftersom båda filerna är kopplade till bildskärmen. Anledningen är att om du dirigerar om utdatat så kanske du inte vill ha med felmeddelanden. Antag att du vill lista alla filer som heter börjar på A och B i en katalog till en fil. Studera följande

```
namatj:~/tmp$ ls A* B*
ls: B*: No such file or directory
Alexander  Anders  Anna  Annika
namatj:~/tmp$ ls A* B* > fil
ls: B*: No such file or directory
namatj:~/tmp$ cat fil
Alexander
Anders
Anna
Annika
namatj:~/tmp$
```

Vi ser att kommandot `ls` generar ett felmeddelande eftersom det inte finns någon fil som börjar på B. Eftersom felmeddelandet skrivs på stderr så kommer det inte med i filen utan skrivs, trots omdirigeringen ut på skärmen. Detta är ju trevligt. Men anta att vi vill ha felmeddelandet i en fil och utdatat i en annan. Vi vill alltså omdirigera både stdin och stderr till olika filer. Stdout kan vi redan så vi tittar på omdirigeringsoperanden "`2>`" som omdirigerar stderr. Följande exempel visar vad jag menar:

```
namatj:~/tmp$ ls A* B* > fil 2> error
namatj:~/tmp$ cat fil
Alexander
Anders
Anna
Annika
namatj:~/tmp$ cat error
ls: B*: No such file or directory
namatj:~/tmp$
```

Ovanstående används ofta då man inte vill att felmeddelandena skall synas alls då gör man så att man omdirigerar `stderr` till /dev/null. /dev/null är också en specialfil. Allt som skickas till denna försvinner spårlöst. Man kan se den som systemets papperskorg. /dev/null växer dock aldrig, blir aldrig full och det som skickas till den kommer aldrig tillbaka. Den är mer användbar än vad man från början kan tro...

```
namatj:~/tmp$ ls A* B*
ls: B*: No such file or directory
Alexander  Anders  Anna  Annika
namatj:~/tmp$ ls A* B* 2> /dev/null
Alexander  Anders  Anna  Annika
namatj:~/tmp$
```

Så långt är allt frid och fröjd. Men anta att jag vill att filen fil skall bli precis som det ser ut på skärmen. Hur fixar jag det? Det kan man fixa genom att använda operatorn “>&”. Den omdirigerar både sdterr och stdout till en fil. Vi tar ett exempel på det också:

```
namatj:~/tmp$ ls A* B*
ls: B*: No such file or directory
Alexander  Anders  Anna  Annika
namatj:~/tmp$ ls A* B* >& fil
namatj:~/tmp$ cat fil
ls: B*: No such file or directory
Alexander
Anders
Anna
Annika
namatj:~/tmp$
```

Okej, nu har vi omdirigerat utdata från ett program. Antag att vi vill omdirigera indata också. Det vill man ofta. Inledningsvis nämndes programmet `sort`. Antag att vi vill sortera ett antal namn i boskavsordning. sort läser som default från stdin och skriver på stdout. Sdudera följande:

```
namatj:~/tmp$ sort
Kalle
Kajsa
Knatte
Tjatte
Fnatte
Oppfinnar-Jocke
Alexander  <-- Här trycker jag <ctrl-D> för att avbryta inmatningen.
Alexander
Fnatte
Kajsa
Kalle
Knatte
Oppfinnar-Jocke
Tjatte
namatj:~/tmp$
```

Antag att vi har en fil met tusen namn. Då vill man ju inte mata in allt från tangentbordet utan man vill skicka hela filen till programmet. Det kan man göra med operatorn “<”. Ännu ett exempel:

```
namatj:~/tmp$ cat osort
Kalle
Kajsa
Knatte
Tjatte
Fnatte
Oppfinnar-Jocke
Alexander
namatj:~/tmp$ sort < osort
Alexander
Fnatte
Kajsa
Kalle
Knatte
Oppfinnar-Jocke
Tjatte
namatj:~/tmp$
```

Har man tusen namn så vill man ju inte att utdatat skall skrivas på skärmen utan man

vill ha det i en annan fil. Detta fixar man genom att kombinera det vi lärt oss ovan. Se ett sista exempel:

```
namatj:~/tmp$ sort < osort > sort
namatj:~/tmp$ cat sort
Alexander
Fnatte
Kajsa
Kalle
Knatte
Oppfinnar-Jocke
Tjatte
namatj:~/tmp$
```

Ovanstående exempel är väldigt bra då man skall förklara omdirigeringar. Programmet sort är dock så pass smart att det inet bara kan läsa från stdin och skriva på stdout. Följande kommando gör precis samma sak som i exemplet ovan:

```
namatj:~/tmp$ sort osort -o sort
```

Sort tar alltså en fil som argument och använder växeln -o för att specificera en fil för utdata (output).

7.11 Använda pipes (rör)

Anta att du vill lista innehålet i en katalog med mycket filer. Du skriver `ls -l` i katalogen. Tyvärr så syns inte alla filer i ett fönster[7]. Eftersom du vet hur man omdirigerar utskriften från ls till en fil så gör du det.

```
namatj:~$ ls -l > tmpfil
```

Nu kan du titta på filen med till exempel more eller less. Efter det så får du radera filen för att städa efter dig.

Detta går naturligvis att lösa på ett annat sätt. Nämligen genom att använda så kallade pipes, eller rör som de ibland kallas på svenska. En pipe representeras av tecknet "|" och tar utdata från det som står före "|" och använder det som indata till programmet efter. Exemplet ovan kan då istället skrivas i ett kommando, utan mellanlagring i någon fil som:

```
namatj:~$ ls -l | less
```

Detta förfaranda är mycket användbart i många situationer. Bara din fantasi sätter gränsen. Blanda dock inte ihop det med omdirigering (> och >>) som bara dirigerar om utdata till en fil. Misstaget

```
namatj:~$ ls -l > less
```

[7] Du kan trycka Shift + Pg Up för att scrolla uppåt både i ett xtermfönster och i konsollen.

skapar en fil `less` i aktuell katalog med innehållet av listningen. Är inte “noclbber” satt enligt ovan riskerar du dessutom att förlora informationen i filen less. En pipe kan ersättas av en mellanlagring och två stycken omdirigeringar.

7.12 Anpassa din miljö

I denna sektion behandlas hur du som användare kan få en miljö som är anpassad efter dina behov. Vi utgår från att ditt skal är `bash` som är standardskal i de flesta Linuxdistributioner. Har du ett annat skal så står det i skal-delen i denna bok hur du konfigurerar det. Har du ett helt annat skal kan du läsa manualsidan för det.

7.12.1 bash

Initieringsfiler
.bashrc
.bash_profile eller .profile

7.12.2 X

Vilka filer skall man andra i? Förklara när .xsession resp .xinitrc används.

7.13 Viruella konsoler

Linux stöder något som kallas *Virtuella konsoler*. Det innebär att du som användare kan logga in på samma maskin flera gånger på olika så kallade virtuella konsoler. En konsol är en arbetsplats som Är textbaserad och har direkt kontakt med datorn. Då du startar maskinen kommer du antingen till en konsol eller till en grafisk innloggnings-skärm (som också körs på en konsol). Du växlar mellan konsolerna med Alt+Fx där x är ett nummer mellan 1 och (oftast) 7. Då du befinner dig i X (som brukar vara konsol 7) så tas tangentkombinationen ovan hand om av X. Så för att byta konsol från den grafiska måste du ge tangentkombinationen Ctrl+Alt+Fx och i det fallet är x ett tal oftast mellan 1 och 6.

Man kan naturligvis logga in som olika användare i de olika konsolerna. Om man vill kan man vara inloggad som root i en av dem. Det är i och för sig inte att rekommendera eftersom det dels är lätt att glömma bort att logga ut och dels för att det blir lite för smidigt att byta till root. Att alltför lätt kunna byta till användaren root är en nackdel eftersom det inbjuder till att oftare köra kommandon som root utan att tänka efter tillräckligt. Det kan dock vara bra att alltid köra som root i en textkonsol eftersom det gör att det blir lite speciellt att köra som root.

De flesta distributioner är konfigurerade så att de har 6 konsoler plus den grafiska. Detta kan dock ändras om man skulle vilja ha flera.

7.14 Hantera processer och job

Mer avancerad processhantering beskrivs i kapitel 35.

I Linux har du möjligheten att även i konsol läge[8] köra flera program på samma gång och enkelt växla mellan dem. Det som skrivs i detta avsnitt gäller `bash`. Använder du ett annat skal hänvisar jag till manualsidan för detta.

&-tecken, fg, bg, jobs. mm.

screen kanske skall nämnas.

Övningsuppgifter

Uppgift 7.1 Titta på filen `/etc/passwd` med cat.

Uppgift 7.2 Titta **bara** på din rad i filen `/etc/passwd` (den börjar med ditt användarnamn).

Uppgift 7.3 Öppna två stycken terminalfönster. Skapa i det ena en fil `Kalle` med `touch`. Ordna nu så att du i det ena, med hjälp av omdirigering andrar filens innehåll. I det andra skall du i "realtid" kunna se hur filen ändras.

[8] Brukar kallas DOS-läge i Microsoft sammanhang, det vill säga en textskärm utan grafik.

Del IV

Skalprogrammering

I denna bok utgår alla exempel från att du använder `bash` som skal. Här tittar vi på de alternativ som finns och behandlar `bash` på djupet.

7.15 Inledning

Detta kapitel går in lite djupare på hur du använder kommandoskalet[9] i Linux. Hur man skriver så kallade skalprogram eller skript. Här behandlas de vanligaste strukturer som man använder vid programmering. Det vill säga *if-satser, for-satser case-srtukturer* mm. Och lite andra vanliga kommandon.

Som vi konstaterade i avsnitt 2.2 hör skalet inte till operativsystemet. Därför finns det flera olika skal att välja på till Linux. Skalet är det program som tolkar dina kommandon. Förser dig med en prompt vid vilken du kan kommunicera med systemet. Skalet förser dig också med en mängd finesser. Vissa skal har tab-komplettering, historielistor och andra finesser.

På UNIX system har man möjligheten att skriva mycket kraftfulla skript. Ett skript är en textfil som innehåller kommandon som kommandoskalet utför. Detta brukar kallas skalprogram men är egentligen inte ett program utan just ett skript. Skillnadem mellan vad som anses vara ett program och ett skript är att ett program är en kompilerad binärfil som direkt körs av operativsystemet. Ett skript är en textfil som tolkas av ett annat program som är gjort för att tolka denna typ av filer.

Skriptet kan skapas i vilken texteditor som helst. För att du skall kunna exekvera det måste du ändra rättigheterna på det. Det kan du gör med kommandot `chmod a+x`[10]. Du kan också starta en ny instans av till skal och ge den ditt skript som argument. Till exempel:

```
$ bash mitt_skript.sh
```

Du kan skriva ett skript för ett visst skal och sedan exekvera det från vilket annat skal som helst. Detta är möjligt om man i skriptets första rad specifierar vilket skal skriptet är avsett för. Den första raden, brukar ibland kallas för *magisk rad*, skall se ut så här:

```
#!/sökväg/till/tolk
```

Observera att denna rad gör ditt skript beroende av vilken dator du kör på. Sökvägen skall nämligen vara absolut och skalen kan ligga på olika ställen på olika maskiner. Detta dock inget större problem eftersom det alltid står på första raden. Sökvägen kan också peky ut andra saker än just skal, till exempel en perl-tolk om det är ett perl-skript.

Ser den första raden ut på ett annat sätt så kommer alla kommandon i skriptet att exekveras i det skal som för tillfället används.

Allt som kommer efter tecknet # (fyrkant?) fram till radens slut är kommantarer och

[9] Skalet kallas även Shell som det heter på engelska.
[10] Läs mer om `chmod` i avsnitt 13 på sid 68

kommer inte att utföras av skalet. Använd detta för att kommantera dina skript. Antag att du skriver ett skript och använder det en tid. Du kommer på att du vill ändra dess funktion på ett sätt. Har du inte skrivit någon kommentar som förklarar skriptet kan du få svårt att förstå vad du själv har gjort. Kommentera inte för mycket, inte för litet, kommentera lagom. Hur man kommenterar är inget jag kommar att behandla i denna bok. Alla programmerare har sin egen stil.

Vidare måste filrättigheterna vara sätta så att man kan exekvera filen. De som för exekvera filen måste alltså ha ett x i filrättigheterna. Detta fixar du med kommandot `chmod`[11]. Till exempel **`chmod a+x skriptfil`**. Vidare måste man också ha läsrättigheter på skriptet för att kunna köra det.

Ett annat sätt att starta ett skript är att skicka det som ett argument till ett skal. Du kan alltså ge kommandot **`bash skriptfil`** i det fallet behöver filen inte vara exekverbar. I detta fall har den magiska första raden ingen betydelse.

7.16 bash

Bash är standardskalet i de flesta Linux-installationer. Bash har många trevliga egenskaper och är mångas favorit-skal.

7.16.1 Historia

Bash är en förkortning för *Bourne Again SHell* och är som namnet antyder en vidareutveckling av *Bourne Shell* Bourne shell är egentligen bara ett smeknamn på skalet (efter sin skapare **Stephen Bourne**). Det heter egentligen *sh* som är en förkortning för *SHell*. Bourne shell är idag omodernt men används ofta i skript eftersom det bör finnas ett skal som ör kompatibelt med `sh` i alla installtioner. Men **`bash`** är inte bara ett förbättrat **`sh`** det har även funktioner lånade från Korn-shell och C-shell (och dess efterträdare TC-shell).

Bash har kraftfulla funktioner både för skalprogrammering och för att kommandoradsarbete. Vi börjar med att titta på vad `bash` kan göra som ditt kommandoskal.

7.16.2 History-funktionen

Bash har en historikfunktion dom är väldigt enkel att använda. Genom att använda piltangenterna kan du bläddra genom de kommandon du tidigare har angett. Ett tryck på ↑ ger dig det senaste angivna kommandot. Bara detta är naturligvis väldigt smidigt och användbart. Men det finns fler finesser.

Kommandot **`history`** visar en lista på de kommandon du gett systemet. Varje kommando har ett nummer bredvid sig. Vill du köra ett kommando med ett nummer, säg 15, ger du kommandot **`!15`**.

[11] `chmod behandlas i avsnitt 13 på sidan 68.`

Man kan söka igenom historielistan. Ctrl+r söker uppåt i listan från där du är. Har du inte listan framme (håller på att bläddra i den) då är du längst ned i listan. På samma sätt så söker Ctrl+s listan fast nedåt.

Som vanligt så finns det ännu mera information om detta i infosidan för `bash`. Har du tänkt att bli en avancerad `bash` användare så rekommenderas du att läsa hela infosidan och sedan öva dig.

7.16.3 Aliaser

I `bash` kan man enkelt sätta något som kallas aliaser. Antag att du varje dag vill ansluta med ssh[12] till en maskin som heter kalles.jva.ankeborg.dl. Kommandot för detta blir **`ssh kalle.jva.ankeborg.dl`**. Det är ju lite jobbigt att skriva detta kanske flera gånger om dagen. Då kan jag skapa en alias till detta kommando. Syntaxen för att göra detta är enkel **`alias namn="kommando"`**. I exemplet ovan kan man använde kommandot

```
alias ssh-kalle="ssh kalle.jva.ankeborg.dl"
```

Nu behöver jag bara ange kommandot **`ssh-kalle`** för att köra det långa kommandot. Ett alias gäller bara så länge man är i det skal man satte det i. Alias ärvs inte till nya subskal. Därför är det lämpligt att lägga sina alias i filen `.bashrc` mer om detta senare i detta kapitel.

Tips!
Lämpliga alias kan vara:

```
alias rm="rm -i"
alias mv="mv -i"
alias ls="ls -color=auto -p"
```

Varning!
*Att göra alias för **`rm -i`** och **`mv -i`** förtar lite av deras funktion. Eftersom man alltid får bekräfta att en fil försvinner så trycken man lätt y på ren rutin utan att tänka efter!*

7.16.4 Omgivningsvariabler

export, set, =, mm.

7.16.5 Omgivningsvariabler

Bash och många anfra skal har en del fördefinierade variabler. Dessa kan du använda i ditt arbete med skalet eller ändra dem för att ditt skal skall uppföra sig annorlunda.

[12] se sidan 162

Dessa gäller i bash.

```
‘PATH’
     A colon-separated list of directories in which the shell looks for
     commands.

‘HOME’
     The current user’s home directory.

‘CDPATH’
     A colon-separated list of directories used as a search path for
     the ‘cd’ command.

‘PS1’
     The primary prompt string.

‘PS2’
     The secondary prompt string.
```

7.16.6 Startfiler

```
Bash Startup Files
==================

   When and how Bash executes startup files.

     For Login shells (subject to the -noprofile option):

         On logging in:
            If ‘/etc/profile’ exists, then source it.

            If ‘~/.bash_profile’ exists, then source it,
               else if ‘~/.bash_login’ exists, then source it,
                  else if ‘~/.profile’ exists, then source it.

         On logging out:
            If ‘~/.bash_logout’ exists, source it.

     For non-login interactive shells (subject to the -norc and -rcfile options\
):
         On starting up:
            If ‘~/.bashrc’ exists, then source it.

     For non-interactive shells:
         On starting up:
            If the environment variable ‘ENV’ is non-null, expand the
            variable and source the file named by the value.  If Bash is
```

```
            not started in Posix mode, it looks for 'BASH_ENV' before
            'ENV'.

   So, typically, your '~/.bash_profile' contains the line
     'if [ -f '~/.bashrc' ]; then source '~/.bashrc'; fi'

after (or before) any login specific initializations.
```

7.16.7 Funktioner

I bash och flera andra skal kan man klumpa ihop kommandon till en funktion. Denna funktion kan man sedan anropa och få alla kommandon utförda. XXXXX

7.16.8 skript

Bash är ett bra kommandospråk för att skriva skalprogram. Som med alla andra skal så skriver du skalskriptet i en textfil. Den första *magiska* raden skall innehålla hela sökvägen till `bash`. För att kunna exekvera ditt skript för du inte glömma att göra det exekverbart (det vill säga se till att den som skall köra det har x satt i filrättigheten)[13].

> **Vanligt Fel!**
> *Glöm inte att göra filen exekverbar med till exempel kommandon* ***`chmod a+x skriptfil`***.
> *Glöm inte heller att ange både katalog och filnamn till ditt skript då du kör det. Står du i samma katalog som skriptet ger du kommandot* ***`./skriptfil`*** *för att starta det.*

7.16.9 Boolska yttryck

Ett boolskt uttryck är ett uttryck som kan vara antingen sant eller falskt. **bash** kan inte själv värdera sådana uttryck utan använder sig av programmet `test`. Programmet ges ett uttryck som argument och svarar genom sin exit-kod. Exit-koden 0 betyder att jämförelsen gick bra alltså sant. Alla andra värden (1) betyder att jämförelsen misslyckades. `Test` kan jämföra både strängar och tal. Vid jämförelse av strängar används tecknen = (lika med) och != (inte lika med). Lika med tecknet blir sant om strängarna det står mellan är lika, annars falskt. Inte lika med tecknet blir sant om strängarna är olika, och falskt om de är lika. Ett exempel kanske skulle vara på sin plats.

$$\begin{array}{lcll} Kalle & = & Kalle & \Rightarrow \; SANT \\ Kalle & != & Kalle & \Rightarrow \; FALSKT \\ Kalle & = & Kajsa & \Rightarrow \; FALSKT \\ Kalle & != & Kajsa & \Rightarrow \; SANT \end{array}$$

Det verkar väl inte så svårt. Dessvärre blir det lite svårare då det gäller tal. Att de

[13] läs om hur du ändrar fi lrättigheterna i anvsnitt 13 på sidan 13

matemaska tecknen används gör att `test` automatiskt antar att du menar strängar. Vill du jämföra tal få du använda något av nedanstående alternativ.

Operand	Testar	Kommentar
-eq	Om två tal är lika	Equals
-ne	Om två tal är olika	Not Equal to
-lt	Om tal1 är mindre än tal2	Less Than
-le	Om tal1 är mindre än eller lika med tal2	Less than or Equal to
-gt	Om tal1 är större än tal2	Greater Than
-ge	Om tal1 är större än eller lika med tal2	Greater than or Equal to

Tabell 7.3: Jamförelseoperander på tal i `test`

`Test` kan inte bara användas för att jämföra tal och strängar utan det kan även användas för att göra tester på filer. Dessa operander tar bara ett argument och det stär efter operanden. Till exempel så är **`-e fil`** sant om filen fil existerar. Tabell 7.4 visar vilka tester på filer som test kan utföra.

Operand	Testar	Kommentar
-b	Sant om bilen fi nns och är en blockadresserad specialfi l	b i fi lrättighetsraden
-c	Sant om fi len fi nns och är en teckenadresserad specialfi l	c i fi lrättighetsraden
-d	Sant om fi len fi nns och är en katalogfi l	d i fi lrättighetsraden
-e	Sant om fi len existerar	Exists, Existerar
-f	Sant om fi len fi nns och är en vanlig fi l	File, Fil
-g	Sant om fi len fi nns och är SGID	g som i Group ID
-k	Sant om fi len fi nns och har "sticky"biten satt	sticky = Kladdig knske
-L	Sant om fi len existerar och är en symbolisk länk	Länk
-n	Sant om fi len fi nns och har en läng större än 0	not Zero
-p	Sant om fi len fi nns och är en "named pipe"	Pipe
-r	Sant om fi len fi nns och är läsbar	
-s	Sant om fi len fi ns och inte är tom	
-S	Sant om fi len fi nns och är en socket	Socket
-t	Sant om fi len fi nns och är öppen på en terminal	Terminal
-u	Sant om fi len fi nns och är SGID	
-w	Sant om fi len fi nns och är skrivbar	Writable
-x	Sant om fi len fi nns och är exekverbar	Executable, Exekverbar
-O	Sant om fi len fi nns och och ägs av aktuell ägare	Ovner
-G	Sant om fi len fi nns ocg tillhör aktuell grupp	Group
-z	Sant om fi len fi nns och har längden 0	Zero

Tabell 7.4: Operander på filer i `test`

Förutom att göra tester på filer kan två filer jämföras. Till detta finns följande operander.

Operand	Testar	Kommentar
-nt	Sant om fi l1 är nyare än fi l2	Newer Than
-ot	Sant om fi l1 är äldre än fi l2	Older Than
-ef	Sant om fi l ett är samma fi l. Genom hård länk.	samE File kanske ...

Tabell 7.5: Jämförelser på filer med `test`

Slutligen kan man också testa och jämföra boolska uttryck. Till detta används följande tre operander.

Operand	Testar	Kommentar
! <test>	Sant om test är FALSK	Negering, Icke
-a	Sant om test1 OCH test2 är sanna	logiskt AND
-o	Sant om test1 ELLER test2 är sanna	logiskt OR

Tabell 7.6: Boolska (logiska) operander till `test`

Som tur är så behåver man inte skriva programnamnet `test` i sina skript utan man kan skriva på ett förenklat sätt. Studera följande:

```
test fil1 -ef fil2 ⇔ [ fil -ef fil2 ]
```

Observera att hakparenteserna måste omges av blanksteg och att det är en programsats och därför skall avsluta en rad eller separeras från nästa kommando med ett semikolon (;).

I skalskript kan du naturligtvis anända vilket skrivsätt du vill eftersom de är ekvivalenta men det andra är mycket smidigare och ger ett skript som är lättare att läsa. Jag använder bara det kortare skrivsättet i denna bok. Anledningen till att jag förklarade detta med `test` var att du skulle få lättare att förstå varför `bash` kan verka väldigt petigt med vart hakparenteserna står och förhoppningsvis slippa några av alla de skrivfel som brukar uppstå i början på grund av detta.

Nu kan du skriva boolska uttryck med `test`. Då kan vi ge oss in på lite skalprogrammering.

Selektioner

`Bash` kan verka lite kinkigt då det gäller hur man skall utforma seletioner. Detta beror på att villkoret som ställs egentilgen är en exekvering av programmet `test` och därför utgör en *sats*. En sats skall avslutas med ett returtecken eller ett semikolon (;). Precis som det gäller vid kommandoprompten. Den vanliga *if-satsen* ser ut så här i `bash`

```
if [ Boolskt yttryck ]
then
        Linux−kommandon
fi
```

Den fungerar som så att om det boolska yttrycket är sant så utförs kommandona som står mellan then och fi. Observera att hakparenteserna måste omges av blanktecken och att de utgör en sats och måste avslutas med ett returslag eller ett semikolon som i nedanstående lite kompaktare form.

```
if [ Boolskt yttryck ]; then
        Linux−kommandon
fi
```

Ibland vill man utföra ett val av flera. Då kan man använda kommandot elif. Elif (utläses else-if) fungerar så att om det första yttrycket inte är sant så testas det andra

uttrycket. Om det andra yttrycket är sant så utförs kommandona efter det. Det kan aldrig inträffa att båda satserna utförs. Utan det är antingen eller. Man kan ha flera elif-satser om man vill men har man många så kan **case** vara ett smartare alternativ.

```
if [ Boolskt yttryck ]; then
        Linux−kommandon
elif [ Boolskt yttryck ]; then
        Linux−kommandon
fi
```

Ett tredje fall är om man vill att “antingen ska det hända, annars det, eller det, om inget stämmer – gör det här” Det vi nu söker är satsen else. Den utförs om och bara om inget av de ovanstående boolska yttrycken är sanna. Else är så att säga en sista utväg. Eller den allmänna om ifsatserna utgör specialfallen. Ifsatsen kan vara bara en och då använder man naturligtvis inte elif-satsen. En sådan sats kallas if-then-else sats

```
if [ Boolskt yttryck ]; then     # Om villkoret i if−satsen är sant
        Linux−kommandon
elif [ Boolskt yttryck ]; then   # Om villkoret i elif −satsen är sant
        Linux−kommandon
else                             # Om inget av ovanstående är sant
        Linux−kommandon
fi
```

Vi har nu sätt hur man kan göra selektioner med hjälp av if-satser, elif-satser, if-elif-else och if-else satser. Men om man vill utföra bara ett alternativ av många kan man ta till case-satsen som är till för just detta. Den ser ut så här

```
case variabel in
        mönster)
                Kommandon
                ;;
        mönster)
                Kommandon
                ;;
        mönster)
                Kommandon
                ;;
        ∗)
                Kommandon
                ;;
esac
```

Case-satsen fungerar så att värdet i variabel jämförs med mönster. Det första mönstret jamförs först och då fortsätter det nedåt. Så fort ett mönster passar så utförs kommandona under detta. Om variabel inte pasar någonstans så utförs kommandona under mönstret *. Mänster kan byggas upp med jokertecknen * och ?. Man kan även använda hakparenteserna [och] och de fungerar på samma sätt som de gör vid filnamnsmönster.

Vanligt Fel!
Glöm inte att ange $ för att komma åt värdet i en variabel.

Man kan även använda | för att sybolisara logiskt eller. Mönstret **[aA]* | *[aA]** passar alltså alla strängar som börjar ELLER slutar på a (antingen versalt eller kursivt).

Studera följande exempel

```
#!/bin/bash

echo −n "Ange en av Kajsa, Kalle eller Joakim: "
read person              # Läs in i variabeln persom
```

```
case $person in
        [Jj]oakim)
                echo "Personen är rik!"
                ;;
        [Kk]ajsa |[ Kk]alle)
                echo "Personen är inte rik"
                ;;
        *)
                echo "Du angav inte ett korrekt alternativ"
                ;;
esac
```

Programmet ovan skriver ut ett meddelande (echo -n förhindrar radmatning) sedan läses ett värde in från stdin till variabeln person. sedan väljs ett av de tre alternativen i case-satsen. Skriv gärna in detta program och provkör det.

Iterationer

Ofta då man programmerar vill man att någonting skall upprepas ett antal olika gånger. Då behöver man satser för upprepning, eller *iteration*. De vanligaste som finns i de flesta programspråk är *while*, *until* och *for*.

Vi börjar med att titta på `while`. While-satsen används för att utföra något så länge som ett uttryck är sant. While-satsen ser ut så här:

```
while [ boolskt uttryck ]; do
        Kommandon
done
```

Aritmetik

Kommandosubstitution

Kommando substitution är ett smidigt verktyg att använda. Inte minst vid kommandoradsarbete. Det fungerar så att ett kommando inom "grava accenter" (‘)[14] substitueras mot sitt utdata. Vi tar ett exempel som använder kommandot **`which`**. **`which awk`** ger ju hela sökvägen till det program **`awk`** som skulle exekveras om vi körde det från komamndoraden. Anta att jag nu vill veta dels var **`awk`** ligger i för katalog **och** om det möjligvis är en symbolisk länk till ett annat program. Det skulle vi kunna göra genom att först köra **`which awk`** och sedan **`ls -l`** på den fil vi fick som svar för att se om det är en symbolisk länk. Nu är man ju slö och vill inte göra det med två kommandon utan jag vill ha svaret direkt. Jag kan inte använda en pipe eftersom **`ls`** inte läser från `stdin`. Men jag kan använda kommandosubstitution. Studera följande exempel:

```
$ which awk
/bin/awk
$ ls -l /bin/awk
lrwxrwxrwx   1 root     root            4 Apr 15 18:26 /bin/awk -> gawk
$ ls -l ‘which awk‘
lrwxrwxrwx   1 root     root            4 Apr 15 18:26 /bin/awk -> gawk
$
```

[14] Du får fram det genom att trycka på Shift + Knappen till höger om +

I exemplet ovan kollar jag först med **`which`** var **`awk`** ligger. Sedan studerar jag det närmare med **`ls -l`**. Det tredje kommandot gör samma sak som de två tidigare men snyggare och snabbare! Kommandosubstitution är mycket användbart.

7.16.10 Tips

Glöm inte att du kan ange "skalskript" direkt på kommandoprompten. Antag att du har en het katalog med textfiler. Du vill nu till varje filnamn lägga till ändelsen `.txt`. Detta skulla man kunna skriva ett skript som fixar. Men eftersom det är så pass enkelt och endast skall göras en gång kan man göra det direkt från prompten (var det någon som sade att grafiska filhanterare var kraftfulla?).

```
$ ls
 textfil1     textfil2     textfil3     textfil4     textfil5
$ for  fil in *
> do
> mv $fil $fil.txt
> done
$ ls
 textfil1.txt     textfil2.txt     textfil3.txt     textfil4.txt     textfil5.txt
$
```

7.17 Mer läsning

[FSF, 199X] är GNU folkets bashmanual. Den är mycket bra och tar upp allt du vill veta om bash.

7.18 tcsh och csh

7.19 zsh

`Zsh` är ett mycket trevligt skal att arbeta i. Då det inte är så spritt bör man inte skriva skript i det om skriptet kommer att köras på flera olika maskiner. Jag kan dock rekommendera att du installerar det på din maskin (om det inte redan finns) och sedan provar att använda det som ditt standardskal. `Zsh` förenar många av de olika fördelarna andra skal har i ett och samma.

7.20 ksh

Övningsuppgifter

Uppgift 7.1 Ta med ett kommando reda på i vilken katalog editorn `vi` ligger och om det möjligtvis är en annan editor som startar[15] då du ger commandot **`vi`**.

[15] Kolla om `vi` är en länk till en annan fil

Uppgift 7.2 Skapa en katalog med ett antal filer. Byt sedan namn på den till `Xnamn` där X är ett unikt nummer för varje fil. Detta skall göras från kommandoprompten med så få kommandon som möjligt. Du får använda vilket skal du vill.

Del V

Dagligt arbete i Linux

Kapitel 8

Praktiska program

Nu när du börjar komma igång att arbeta i Linux och förmodligen upptäckt lite av dess kraftfullhet kanske du börjar fundera lite på hur du skall kunna utföra de arbetsuppgifter som du gjorde i ditt förra system i detta nya trevliga system. Du undrar säker också vad mer som gör detta system så uppskattat av sina anhängare.

Detta kapitel handlar lite om vilka applikationer som finns till Linux. Eftersom alla har olika intressen och jobbar med olika saker kommer detta kapitel inte att passa alla.

Vidare är det naturligtvis en omöjlighet att få en komplett lista av alla program. Det finns helt enkelt för många. Här listar jag upp sådana som vanligvis följer med din distribution. Men även andra som till exempel så kallade Office Paket. Jag har förmodligen missat en hel del bra program. Detta är program som jag själv har använt mycket under min tid med Linux.

Kapitel 9

Manipulera PostScriptfi ler

Ett arv från UNIX är att i princip alla Linux-program genererar utskrifter i formatet *PostScript*. Har man en skrivare som hanterar PostStript är detta väldigt smidigt. Men nu är det så att dessa skrivare ofta är väldigt dyra. Har man ingen sådan så får man använda programmet `ghostscript` som kan översätta postscript till ett format som din skrivare förstår. Är detta riktigt konfigurerat så märker man inte att denna omvändling sker. Då man tror att man skriver till en skrivare skriver man egentligen till ghostscript som i sin tur skriver till skrivaren. Hur man, som systemadministratör, konfigurerar skrivare behandlas i kapitel 31.

9.1 Skriva ut `lpr`, `dvips`

Ett dokument kan skrivas ut med kommandot **`lpr filnamn`**. Det enda kravet är att formatet på dokumentet är ett som skrivaren förstår. Om skrivaren är bra konfigurerad så skall man kunna skicka åtminstone PostScript filer och rena textfiler till skrivaren.

Om `lpr` startas utan filnamn som argument så väntar `lpr` att indata skall komma från standard inmatningsenheten. Detta gör det möjligt att använda `lpr` med pipes. Till exempel ger följande exempel en utskrift av filen `fil.txt`

```
cat fil.txt | lpr
```

I kommandot ovan är ju naturligtvis onödigt att använda en pipe eftersom man bara skriver ut en fil. Men det visar väl ändå lite hur det fungerar.

`Lpr` kan ta flera olika flaggor, här presenteras de vanligaste. De vanligaste visas här och presenteras i tabell 9.1.

För att skriva ut på en annan skrivare än den som är standard använder man flaggan `-P`. Exempelvis så gör kommandot

```
lpr -P pr100 linuxboken.ps
```

flagga	Utför
-P skrivare	Skriv ut på skrivaren "skrivare" istället för den skrivare som är standard. Vilken skrivare som är standard bestäms vanligen av miljövariabeln **PRINTER**.

Tabell 9.1: Flaggor till `lpr`

att dokumentet `linuxboken.ps` skrivs ut på skrivaren *pr100*. Anges inte flaggan `-P` så skickar `lpr` till den skrivare som anges i miljövariabeln **PRINTER**. Finns inte **PRINTER** används någon annan standard enhet vanligen `lp`.

För att skriva ut flera kopior av ett och samma dokument så kan man använda flaggan `-#`.

Här skall skrivas mer om hur man sätter en defaultskrivare, $PRINTER mm, mm.

9.2 Hantera skrivarköer

Då ett dokument skrivs ut skickas det till en skrivarkö som hanteras av en daemon som vanligtvis heter `lpd`. För att se vilka job som ligger i kön kan du ange kommandot **`lpq`**. Vill du ta bort ett job som du lagt till skrivarkön kan du ta bort det med `lprm`.

Finns mycket mer att säga om dessa kommandon, kanske imorgon . . .

9.3 Förhandsgranska

Ofta vill man först titta på det man vill skriva ut på skärmen. Det kanske till och med räcker med detta. Det är om inte annat bra för skogen. Eftersom en hel del papper kan sparas om man bara skriver ut en, eller ingen, gång. Eftersom det som du granskar är detsamma som skickas till skrivaren så kan du vara helt säker på att det du ser är det du får.

9.3.1 GhostView

9.3.2 gv

9.3.3 xdvi

En dvi fil (Device Independant) måste konverteras till PostScript innan den kan skrivas ut. Detta kan vara onödigt om man inte vill skriva ut den utan bara titta på den. Det kan man göra med programmet `xdvi`.

Här skall beskrivas hur xdvi startas och hur det används.

9.3.4 xpdf

9.3.5 Acrobat Reader

9.4 pstops

Det kan låta som ett fånigt kommando. `Pstops` konverterar, precis som namnet antyder, från PostScript till PostScript. Naturligtvis så gör det lite annat på vägen. Kommandot kan användas för att till exempel skriva ut två eller fyra ark per fysisk sida eller annat.

Här skall skrivas mer om pstops syntax. Samt även hänvisas till mina skript.

Kapitel 10

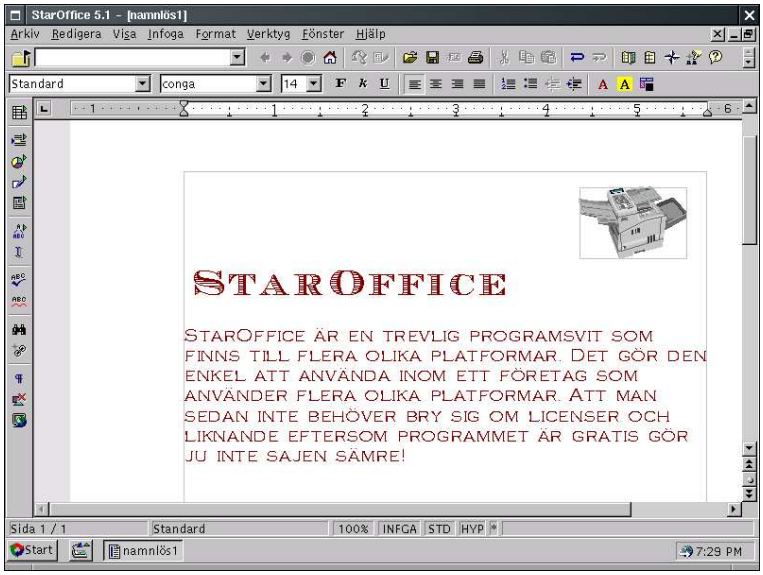
Offi cepaket

De som kommer från Windows till Linux brukar sakna Microsoft Office^{TM}. Detta beror till största delen på att den programsviten är så vanlig att alla förväntas kunna läsa de filer som produceras i den. Paradoxen är att filfomaten som används i denna svit inte är fria att använda. På så sätt blir det i princip omöjligt att producera 100% kompatibla program. Man kan ju drömma om att det någon gång i framtiden blir en standard för hur till exempel ett ordbehandlingsdokument skall sparas. På så sätt skulle alla ordbehandlingsprogram kunna vara kompatibla med varandra. Detsamma skulle gälla alla program. Detta gäller idag till exempel vissa grafikfilformat. Man kan alltså redigera en bildfil i vilket grafikprogram som helst. Eller titta på den oavsett vilken plattform man arbetar på.

Det finns fria alternativ till Microsofts programsvit. KOffice är en programsvit för KDE som är under utveckling. Den befinner sig ännu i alfastadiet men verkar lovande. Flera fria “Officepaket” finns, men ingen så lovande som KOffice.

Dett finns naturligtvis också kommersiella program till Linux. Det finns två stora Officepaket. StarOffice och Applixware. StarOffice är gratis för privatpersoner och kan laddas ner från internet. Applixware är en helt kommersiell produkt som du köper hos någon återförsäljare.

Sun har numera köpt StarOffice. Det är nu gratis inte bara för privat personer utan för alla som vill använda det. Detta är mycket trevligt eftersom man nu kan använda en och samma office-svit på både Microsofts operativsystem och på Linux. Figur 10.1 visar StarOffice med ordbehandling igång.



Figur 10.1: StarOffice från Sun

StarOffice är en komplett Office-svit från Sun. Den finns till flera olika platformar och i många olika språk.

Kapitel 11

Textbearbetning och typsättning

11.1 Typsättning med LaTeX

LaTeXär en uppsättning med macron till tysättningssystemet TeX. LaTeX skrevs av Leslie Lamport och TeX skrevs av D.E. Knuth.

Att typsätta med LaTeX är verkligen kul. Detta dokument är till exempel skapat med det. LaTeX är inget för dig som skriver saker som inbjudningskort och liknande (det kan användas till det också, men det är knappast praktiskt). LaTeX kommer mer till sin rätt på större dokument. LaTeX är perfekt för tekniska rapporter och böcker eftersom det hanterar matematiska yttryck på ett mycket bra sätt.

LaTeX är inget WYSIWYG verktyg utan du skriver texten i en texteditor, som t.ex. `emacs`. Du märker upp hur texten skall se ut med små textkommandon som brukar kallas “taggar” från det engelska “tags”.

I detta tidiga skeda av denna boks utveckling tänker jag inte kasta mig in på ett så stort område som LaTeX utan hänvisar till **Att skriva rapporter med LaTeX** av **Per Foreby** perf@efd.lth.se

11.1.1 LyX och KLyX

Kanske tycker du att det är för krångligt att skriva LaTeX kod direkt men vill ändå skriva proffsiga dokument. Då kan du använda LyX eller KLyX. Dessa båda program är fasader till LaTeX.

Kapitel 12

Grafik i Linux

Att bearbeta grafik är något som många gör. I ditt Linuxsystem finns redan flera bra program för detta.

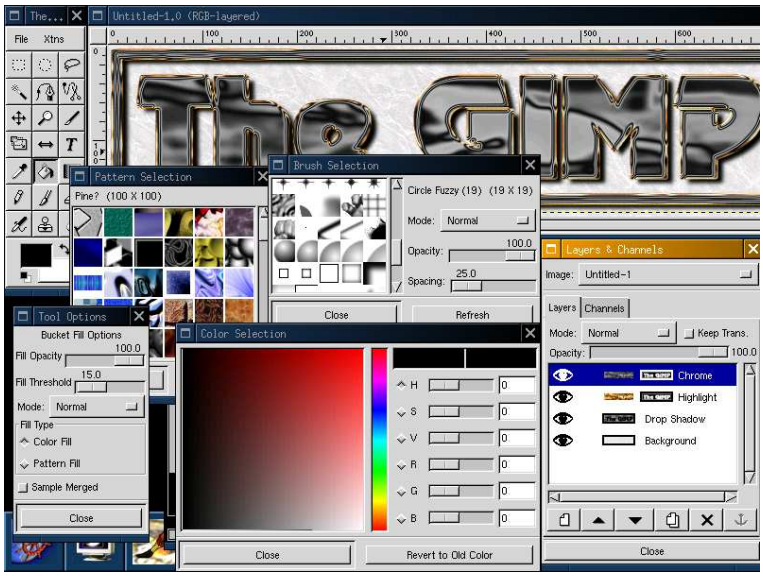
12.1 Gimp

Gimp är ett mycket kraftfullt grafikprogram. Det liknar Adobes Photoshop säger de som vet. Själv har jag inga erfarenheter av Adobes program. Gimp är en förkortning för *GNU Image Manipulation Program*. Gimp är ett program av mycket hög kvalitet. Bild12.1 visar en skärmdump med några av programmets verktyg öppna.

Gimp innehåller också ett skriptspråk kallat Script-Fu. Detta är användbart. Antag att du till exempel skall göra knappar till en hemsida. Knapparna skall se likadana ut men ha olika text. Då gör man ett skript som gör alla momenten och tar texten som ett "argument". En annan stor fördel om man, som jag, inte är det minsta konstnärligt lagd så kan man använda andras färdiga skript. Bilden som kan skymtas i skärmdumpen i Bild12.1 har jag gjort utan att ritat något. Allt är gjort av ett skript som följer med Gimp. Massor av skript finns naturligtvis att hämta från internet.

12.2 `xpaint`

`Xpaint` är ett verktyg liknande `gimp` men kommer inte ens i närheten av hur avancerat `gimp` är. Jag kan inte se någon anledning att använda `xpaint` förutom om din maskin är för klen för att använda `gimp` eller om du är rädd om ditt diskutrymme och inte är intresserad av bildbehandling.



Figur 12.1: The Gimp

Här visas en prottrig skärmdump med Gimp. Lägg märke till lagerhanteraren ner till höger. Den är mycket kraftfull och enkel att använda. Bilden som producerats är gjord med ett av många Script-Fu skript som följer med programmet.

12.3 xfig

Xfig är ett bra vektorbaserat program för att rita till exempel flödesdiagram. En nackdel är att det kräver hög upplösning på skärmen. 1024x768 är minimum[1].

Fyll på kort om hur man använder Xfig, hävisa till figurer i denna bok gjorda med xfig.

12.4 convert

Ofta är man i den situationen att man har en bild i ett grafikformat och vill konvertera den till ett annat. Det första man kanske gör då är att fundera över vilket program som klarar av att öppna filen i fråga och sedan spara om den i det format man önskar. Nu behöver man inte fundera mer. Med convert kan man på enklast möjliga vis konvertera en fil mellan olika format. Jag använder det till exempel för att konvertera bilder till .eps format för att infoga i LaTeX-dokument som till exempel denna bok. För att konvertera bilden bild.jpg till .eps så anger jag kommandot **convert bild.jpg filnamn.eps**. Filen konvereras då och sparas i filen som jag anger som andra argument. Den filen behöver naturligtvis inte finnas utan skapas vid konverteringen. Infomation om vilka filformat det är fråga om från programmet från filnamnsprefixen.

12.5 xv

Xv är ett kommersiellt program som är mycket spritt.

[1] jag använder det ibland i 800x600 på min laptop men då syns inte alla funktioner

Kapitel 13

Matematik och räknare

Linux är ett system som används av många i forskningssyfte. Då är det ett måste med bra matematikprogram. Det fins kommersiella sådana till Linux men de som jag presenterar här ar fria. Det här är bara en liten del av alla program som finns. Eftersom jag inte äe någon matematiker så är dessa program inte heller de mest avancerade.

13.1 dc

13.2 xcalk

`Xcalc` är en mycket enkel miniräknare som man kan använda till enkla beräkningar.

13.3 octave

Octave är ett program som liknar Matlab. Man kan skriva kommandofiler till det eller mata in kommandon direkt. Jag rekommenderar att man skriver inputfiler till det.

Förklara kort hur det fungerar, skaka fram dina Matlabkunskaper bara ...

13.4 gnuplot

Gnuplot används för att rita ut grafer och diagram över dina beräkningar.

Kapitel 14

Bearbeta textdokument från kommandoraden

I ett Linuxsystem finns flera så kallade texfilter som du kan använda för att på ett kraftfullt sätt bearbeta textfiler. Ta kopior på dina textfiler i början när du exprimenterar med dessa kommandon så att du inte av misstag förstör något av dina texdokument.

Flera av dessa program tar indata från standard input och skickar sitt utdata på standard output. Detta gör att de fungerar väldigt bra i samband med pipes. Därför kallas dessa program ofta för filter.

14.1 Bearbeta textfiler

Till Linux finns en hel del kraftfulla program för att bearbeta textfiler. Då menar jag naturligtvis inte editorer, som det också finns en hel del av, utan kommandoprogram som är både kraftfulla och smidiga.

14.1.1 `pr`

14.1.2 `sed`

14.1.3 tr

`Tr` är ett program som används för att byta ut eller ta bort tecken i en text. Texten läser det in från `stdin` och skriver resultatet på `stdout`. Detta gör att det lämpar sig alldeles utmärkt för omdirigering och pipning.

14.1.4 awk

`Awk` eller `gawk`, som följer med de flesta Linuxdistributioner är en interpretator för skript skrivna för just den. `Awk` är mycket bra på att bearbeta texter.

14.1.5 perl

Perl är ett programmeringsspråk som egentligen inte har något här att göra. Men Perl är så bra på just detta att det känns fel att utelämna det. Jag kommer dock inte att gå in på hur det fungerar här.

Kapitel 15

Använda nätverket

Linux är ett operativsystem som verkligen gör sig bra i en nätverksmiljö. Här presenteras lite av vad du kan göra med det.

15.1 Surfa på webben, WWW

Att surfa på World Wibe Web är något som alla gör idag. Webben är inte alls lika "hype" idag som den varit utam börjar mer och mer bli vardag för de flesta. För att surfa på webben måste man ha en webläsare. Det finns en uppsjö av dessa att hämta. Här nämner jag de mest använda.

15.1.1 `Lynx`

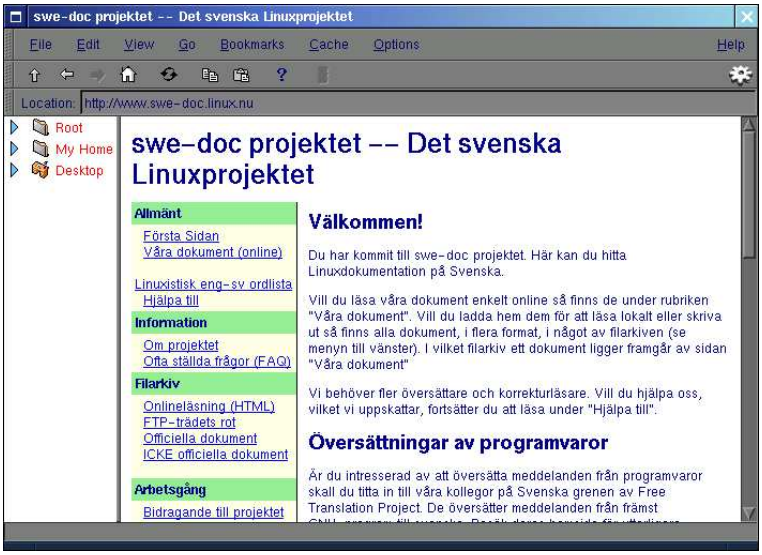
`Lynx` är en helt texbaserad webläsare. Det kan verka tråkigt att surfa i konsol-läge, men faktum är att det många gånger är smidigt. Till vardags använder de flesta nog en browser för X. Men antag att du inte får igång X av någon anledning som du inte vet, och vill leta information om detta på webben. Då är en texbaserad webläsare vad du behöver. Det finns även en mer avancerad textbaserad web-browser som heter `links`.

15.1.2 `Links`

`Links` är en mycket kapabel texbaserad web-browser.

15.1.3 `Netscape`

`Netscape` är den mest använda webläsaren. Den fungerar utmärk och stöder den senaste tekniken. Hur den fungerar går jag inte in på här utan hänvisar till dess hjälpfunktion.



Figur 15.1: `Kfm` som webbläsare

`Kfm` är inte bara KDEs filhanterare, den är också en fullvärdig webläsare.

15.1.4 `Kfm`

`Kfm` är egentligen filhanteraren till *K Desktop Environment*, *KDE* men den är även en fullvärdig webläsare. Skriv bara in en *url* i adressfältet. Figur 15.1 visar `kfm` som webbläsare.

15.1.5 Andra webläsare

Andra webläsare som kan vara bra att känna till som jag inte behandlar här är *Arena*, *Mosaic*, *HotJava* och *Opera*.

15.2 Hitta fi ler med Archie

Ett gammalt sätt att hitta filer som inte så ofta används idag är Archie.

15.2.1 `karchie`

15.2.2 `xarchie`

15.3 Flytta filer med ftp

15.3.1 `ftp`

15.3.2 `ncftp`

15.3.3 `mc`

15.3.4 `wx_ftp`

15.3.5 `kfm`

Filanteraren i KDE är inte bara filhanterare och webläsare. Den är dessutom en fullfjädrad ftp-klient.

15.4 E-post

Utan e-post stannar de flesta företag idag. Det mesta kan få krångla men inte e-posten.

Hur man konfigurerar e-post bör stå i systemadministrationsdelen. Här bör klienter tas upp.

Procmail, fetchmail och några mua's skall beskrivs här. Lite teori skall det också vara.

15.5 News

Internet news eller usenet som det också kallas är ett av de bästa sätten att skaffa sig informatoion då man kört fast. Har man problem med Linux så bör man läsa gruppen `se.dator.sys.unix` där bland annat Linux diskuteras. Kom dock ihåg att det är en avancerad grupp som inte bara behandlar Linux. Har du problem med Linux måste du ange att det är Linux det gäller och även vilken distribution och version det gäller.

15.5.1 slrn

15.5.2 mozilla

15.5.3 gnus

15.6 Chat, IRC

15.7 Fjärranslut till andra datorer

Har du flera datorer i ett nätverk eller konto på en dator någon annanstans så kan du enkelt logga in på andra datorer från Linux. Här behandlar jag de vanligaste sätten att göra det.

15.7.1 telnet

Telnet är ett protokoll sam använd for fjärranslutningar. Det är också ofta namnet på programvara som nyttjar protokollet. Telnet har en del begränsningar. Den största är att ingen kryptering sker av dataöverföringen. Det gör det möjligt för någon illasinnad att lyssna på trafiken från din dator för att till exempel snappa upp lösenord. På grund av detta är det i de flesta distributioner inte möjligt att logga in som root via telnet. Detta skydd är dock en konfigureringssak och går att ta bort om man känner sig vågad.

> **Varning!**
> *Telnet bör man aldrig använda. Se följande katitel om* `ssh` *i stället.*

15.7.2 ssh

Ett modernare protokoll än telnet är `ssh`. `Ssh` är en förkortning för *Secure SHell* och som namnet antyder är det säkrare än `telnet`. `Ssh` krypterar trafiken mellan maskinerna och gör det på så sätt svårera att avlyssna informationen som skickas.

XXXXXXXXXXXXXXX Besksriv ssh

15.7.3 `rlogin`

15.7.4 `rsh`

15.8 Terminalemulering

15.8.1 Minicom

15.8.2 Seyon

15.9 “Tala” med andra i nätverket

15.9.1 talk

Ett enkelt program som används för att tala över nätverket är programmet `talk`. Programmet används för att prata med en annan person. Du ser det du skriver i ett fönster och det din kompis skriver i ett annat. `Talk` arbetar helt i realtid och du ser tecknen din kompis skriver direkt. Visst finns det mer avancerade program, men `talk` är enkelt, fungerar och finns nästan överallt.

15.10 Skicka fax

Här beskrivs hur man använder en faxtjänst. Hur man sätter upp en sådan beskrivs i avsnitt 36.7.1 på sidan 258.

Om man skriver ett fax i ett ordbehandlingsprogram. Varför skriva ut faxet på papper för att sedan gå till faxen för att faxa det till någon annan när det går att faxa direkt? Faxar man direkt från datorn så behöver man ju dessutom inte någon fax för att kunna skicka fax.

Kapitel 16

Emulatorer

Ibland kanske du tycker att de Linuxprogram som finns inte riktigt räcker till för dina behov. Det finns då en möjlighet att du kan köra dina program i Linux ändå. Detta kan vara möjligt med hjälp av de emulatorer som finns. Jag rekomenderar dock att du i första hand försöker att hitta ett annat program som är porterat till Linux. Du får då ett program som med största säkerhet fungerar bättre än ett program som körs i en emulator.

Till Linux finns en del olika emulatorer. En emulator har till uppgift att emulera en annan miljö i vilken program kan exekveras. Det finns två typer av emulatorer, de som emulerar en hårdvarumiljö och de som emulerar ett operativsystem.

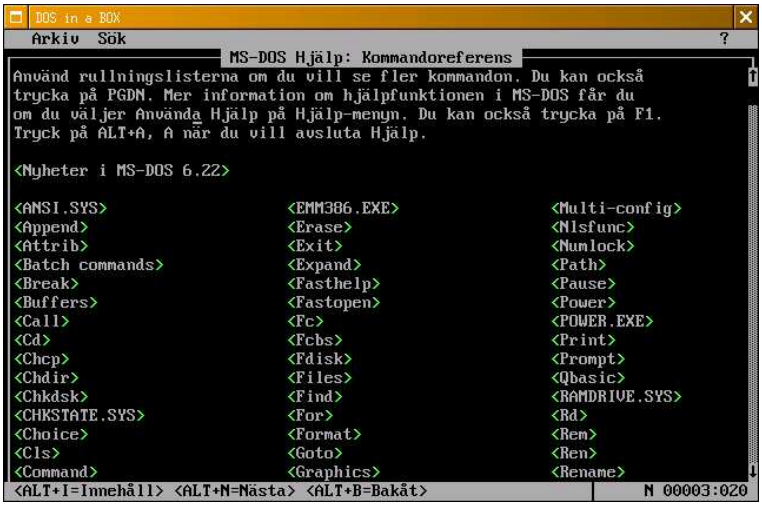
16.1 DOSemu

DOSemu är en emulator som tillåter dig att köra DOS program. Figur 16.1 visar DOS 6.22 i DOSemu.

16.2 Windowsemulatorn Wine

Om du inte kan slita dig från dina Windowsprogram finns det stor chans att de kommer att kunna köras i Linux med hjälp av Windowsemulatorn `Wine`. Wine är idag i ett tidigt stadium och bara en del Windowsprogram kan köras. Figur 16.2 visar en skärmdump på då Microsofts winfile (Filhanteraren) körs i Linux. Jag går inte in på hur man installerar och konfigurerar wine här eftersom programmet är på ett så tidigt stadium i utvecklingen. Risken är att du bara blir besviken för det program du vill köra inte fungerar. Jag hänvisar istället till hemsidan för projektet, `http://www.winehq.com`.

Om det Windowsprogram du vill köra inte finns i en variant för Linux tycker jag att du skall säga till återförsäljaren eller tillverkaren att du vill ha programmet till Linux. Detta hjälper förmodligen inte dig, men om alla frågar så måste tillverkaren till slut



Figur 16.1: DOSemu

Här körs DOS 6.22s utmärkta hjälp i DOSemu.

ge vika för sina kunders önskemål. Detta gäller alla företag. Men bara om tillräkligt många kunder gör sina röster hörda.

Wine emulerar en Windows miljö. Du startar alltså inte upp Windows utan kör Windowsbinärer direkt genom Wine.

16.3 VMware

En kommersiell emulator som blivit populär är `VMware`. VMware är en virtuell maskin på vilken du kan köra olika operativsystem. Prestanda skall nligt företaget bli nästan lika bra som om operativsystemet kördes direkt på maskinen. WMware kräver naturligtvis en hel del av maskinen, minst 128 megabyte internminne bör man ha.

Då man kör VMware startas en ny “bildskärm” i din miljö. På denna virtuella monitor kan du boota och köra ett annat operativsysytem.

En utförligare beskrivning kommer kanske i senare versioner av boken.

Figur 16.2: Microsofts winfile.exe genom wine

Här kan du se en skärmdump på hur det kan se ut då `wine` kör ett windowsprogram. Detta kan vara en lösning om det finns ett Windowsprogram du verkligen måste köra. Jag rekommenderar att du i första hand försöker hitta en Linuxvariant av programmet.

Del VI

Rekreation

Kapitel 17

Inledning

Lite kul skall man ju ha. Även om jag personligen inte är så mycket för dataspel så har de ju ändå bidragit en hel del till datorutvecklingen. Detta kapitel handlar kort om spel och andra sätt att ha roligt med datorn.

17.1 Ljud

Att skapa och spela upp ljud är något som många vill göra med sina datorer.

17.1.1 MPEG musik

MPEG, i synnerhet mpeg3 (mp3), är ju ett format som blir allt vanligare. Till Linux finns det flera olika program för att spela upp sådana filer.

xmms, är ett avanverat program för att bland annat spela upp mp3 musik.

mpg123 är ett textbaserat program för att spela upp mp3 filer. Det är mycket resurssnålt och går utan att klaga att köra på en 486:a.

play

17.2 Dataspel

Dataspel är ett område där inte Linux ännu ligger så långt framme. Detta beror i första hand på att spelutvecklarna är drivna av kommersiella intressen och vill man i dagsläget tjäna pengar på att utveckla programvara så skall man man rikta sig mot Windowsanvändarna. Detta är något som jag hoppas, och tror, kommer att ändras i framtiden. Ett problem med att sälja program till Linux användare är att de ofta är så bortskämda att de finner det onödigt att betala för programvara som de är vana att få gratis.

Övningsuppgifter

Uppgift 17.1 Inte kan man väl öva på att spela spel och leka, ta en paus och en kopp kaffe istället!

Del VII

Systemadminstration

Kapitel 18

inledning

Att läsa dett gör dig definitivt inte till någon systemadministratör. Är det det du är ute efter bör du köpa en bok i just det ämnet. Systemadministration är alldeles för stort för att täcka i denna bok. Detta kapitel bör läsas av alla intresserade eftersom det ger en något djupare inblick i hur saker och ting är uppbyggt. Detta kapitel lär dig också hu du på ett smidigt sätt kan underhålla ditt eget system.

Att vara systemadministratör är ett stort ansvar. Eftersom man har tillgång till användaren root:s lösenord så kan man göra allt med systemet. Man kan radera filer, skapa filer, ändra alla inställningar, mm, mm. En sak som man naturligtvis kan göra är att titta i alla användares hemmakataloger, läsa deras epost, skicka post i deras namn och så vidare.

Nu är det så att bara för att det är möjligt så är det inte tillåtet. Du skall ha användarens tillstånd för att till exempel läsa användarens epost. Eftersom det är möjligt för systemadministratören att läsa andra användares filer måste helt enkelt användaren lita på att det som de sparar i sina kataloger inte läses av andra.

På vissa företag har systemadministratören (och chefer mm) rätt att läsa användares filer. Detta skall i så fall användaren veta om. Samma sak gäller till exempel skrivbordslådor. Chefen skall inte rota i dina skrivbordslådor utan att du gett tillstånd om det. Har systemadministratören tillstånd att läsa andras mail bör denna ha tystnadsplikt.

En sak som jag tycker är viktig är att detta även gäller i hemmet, inom familjen. Om du som läser detta har en dator hemma så är du förmodligen dess administratör. Du har kanske skapat användare åt din familj och kanske även dina vänner. Vet dessa i så fall om att du kan läsa deras post? Vet de inte om det så skall du heller inte göra det. Det ligger i din heder som systemadministratör att inte göra det. Du läser väl inte andras vanliga post (den som kommer i brevlådan)?

Du skall också veta att det är väldigt lätt att förstöra sitt system som systemadministratör, medan det är (nåja, nästan) omöjligt som användare. Som root begränsas du inte på något sätt av systemet, utan du kan göra vad du vill. Detta ses av många nybörjare som en fördel. De ser att köra som vanlig användare som ett hinder snarare än

ett skydd. Kom ihåg detta, jag har sagt det förr och kommer att säga det igen. Kör **ALDRIG** som root om du inte måste. Användaren root har en hemmakatalog som vanligtvis ligger i roten av filsystemet. Den skall alltid vara (i princip) tom!

> **Varning!**
> *Kör **aldrig** som root*

Så har du nu inget Linuxsystem att tillgå så får du börja med att instalera ett. Därför följer närmast några kapitel om hur du installerar Linux och hur du får det att fungera. Sedan går vi in mer på det man kanske i första hand tänker på då man talar om systemadministration.

Kapitel 19

Installation

I detta kapitel behandlas vad man bör tänka på vid installation av ett nytt Linuxsystem. Tänker du bara använda Linux till ditt dagliga arbete och har någon annan som installerar åt dig kan du lungt hoppa över detta kapitel. Detsamma gäller naturligvis om du redan sitter vid ett fungerande Linuxsystem och inte vill veta hur det hamnade där.

Installationen varierar från distribution till distribution. Här har jag bara tagit upp det som generellt gäller för alla. Just din distribution kanske löser saker och ting på ett eget sätt men allt som står här måste göras på ett, aller annat sätt. Detta kapitel behandlar bara installation på PC eftersom det, idagsläget, är det enda författaren har tillgång till och behärskar.

Läs först igenom detta kapitel så att du vet vad som skall göras. Gå sedan tillbaka och gör vad som står i tur och ordning.

19.1 Hur får jag tag på Linux?

Eftersom Linux är fritt kan man få tag i det på en mängd olika ställen. Man kan ladda ner en Distribution från nätet om man har tillgång till en snabb uppkoppling. Det rör sig om hundratals megabyte så det är inta att tänka på om man har modem. Var man kan hitta Linux på Internet beskriver jag i kapitel 7.4.

Vill man ha distributionen på en CD, och har tillgång till en CD-brännare så rekommenderar jag att man laddar hem en så kallad iso-fil. En iso fil är en avbild av CD skivan som den skall se ut. Ditt CD-brännar program kan sedan bränna den direkt till en CD. På så sätt får du en CD som ser ut precis som tillverkaren vill ha den. Man får då bland annat en bootbar CD vilket kan vara lite krångligt att fixa om man skall göra det själv.

Då och då dyker det upp en Linuxdistribution på CD i nån av alla de datortidningar som finns. Ofta finns det då en kort beskrivning i tidningen hur man installerar från deras CD. Detta sätt är ett enkelt och billigt sätt att få tag i Linux om man inte har en

snabb koppling till Internet.

I bokhandeln säljs det böcker om Linux. Dessa brukar oftast innehålla en distribution. Då får du dessutom rikligt med dokumentation med på köpet. En varning dock att dessa böcker naturligtvis säljs under en längre tid än t.ex. datortidningar och därför ofta innehåller gamla versioner av Linux. Linux utvecklas i en rasande fart och det är klokt att man installerar det senaste som finns tillgängligt.

Har man en snabb uppkoppling till internet så behöver man dessutom ingen distribution på CD utan man kan installera direkt från ett nätverk. Nätverket kan vara ett internt nät eller internet. Detta är att föredra om du har tillgång till en sådan koppling. I och med att du inte har filerna på en skiva så har du alltid tillgång till de senaste filerna. En CD blir ju gammal nästan så fort du brännt, eller pressat, den.

Ingen av ovanstående metoder att skaffa sig Linux berättigar till support (möjligtvis med undantag för boken). Skall du använda Linux professionellt eller av annan anledning känner att du vill ha support skall du köpa en Linuxdistribution från tillverkaren. Detta är också det bästa sättet att ekomomistk stödja utvecklingen av detta underbara system. Det finns flera återförsäljare i Sverige (se kapitel 7.4). Då får du förutom distributionen även en tryckt manual för just den aktuella distributionen.

Jag anser dock att man som regel klarar sig utan support. Visst är Linux lite krångligt i början men det finns många som kan hjälpa dig. I kapitel 7.4 beskriver jag hur man enkelt kan be andra om hjälp. Man bör dock försöka att först lösa sina problem själv med all den dokumentation somm finns tilgänglig.

19.2 Sök mer information

I denna bok försöker jag att hålla mig till generella metoder som fungerar i alla system. Installationen av olika distributioner skiljer sig åt. Var noga med att läsa installationshandledningen[1] till just din distribution.

19.3 Vilken hårdvara krävs

En modern Linuxdistribution kan köras på de flesta maskiner. Hur höga prestanda maskinen måste ha beror på vad du vill göra. Ett abslolut minimum är en 486:a med 8 MB internminne och en hårddisk på ca 400 MB[2]. Men jag rekomenderar en Pentium dator med 32 MB internminne och 1 GB hårddisk. För att det skall gå någorlunda snabbt. Naturligtvis är det så att ju mer internminne och processorkraft desto snabbare går det. Linux kan dessutom med fördel köras på multiprocessorsystem. Har du en tillräckligt stor hårddisk (eller flera) kan du installera Linux parallellt med ditt nuvarande operativsystem. Du kan då välja operativsystem då du startar datorn.

[1] Har du ingen tryckt manual så fi nns informationen på CDn eller där du hämtar din distribution.

[2] Det talas ibland om betydligt lägre hårdvarukrav, men detta är det minsta för att få in en modern distribution. Jag jobbar själv med Linux i inbäddade system och vet att Linux går att installera på det mesta om man bara vill. Det är dock långt utanför denna boks räckvidd.

Linux stöder idag ännu inte all hårdvara. Detta beror på att ny hårdvara kontinuerligt kommer ut på marknaden. Eftersom drivrutiner och annan mjukvara som är bunden till en produkt som regel skrivs av frivilliga utan kantakt med företaget som producerat hårdvaran, tar det naturligtvis längre tid än att få fram drivrutiner till exempelvis Windows. Till Windows skriver tillverkaren själv drivrutiner medan de utvecklar hårdvaran. Saker som idag vållar vissa problem är de allra senaste grafikkorten och perferienheter. Linux stöder ännu (april -99) inte usb, men kommer att göra det inom kort.

Datorn bör ha en CD-läsare eller ett nätverkskort med koppling till ett ställe där din distribution finns. Det går att installera på flera olika sätt utan CD-läsare, men då CD-läsare anses vara standard idag utgår jag från att an CD-läsare finns i ditt system. Skulle så unte vara fallet hänvisar jag till manualen till din Linuxdistrubution.

19.4 Samla information om ditt system

Linux är (ännu) inte lika bra som MS-Windows på att identifiera din hårdvara. Mycket fixar Linux själv men inte allt. Anledningen till att det är så är att tillverkarna som regel inte skriver drivrutiner til sin hårdvara. Det kan tyckas konstigt eftersom de minskar sin kundbas men så är det i alla fall. Vidare så har flera tillverkare ingen lust att delge Linuxutvecklare information om hur hårdvaran fungerar. Vilket i sin tur gör det svårt eller nästan omöjligt att skriva egna drivrutiner. På detta område händer mycket när detta skrivs (april -99).

Följande skall du veta om systemet innnan du startar installationsprogrammet:

- Hårddiskens typ (IDE, SCSI).
- Finns SCSI-kort, i så fall modell.
- Nätverkskort, typ, tillverkare.
- Grafikkort, typ, tillverkare, mängd minne.
- Ljudkort, typ, tillverkare.
- Bildskärm, modell och/eller dess data, Horisontell och vertikal frekvens, interlaced eller ej, vilka upplösningar den klarar osv.
- skrivare, märke, modell och eventuellt om den emulerar någon annan skrivare.

Installerar du på en dator som står i ett tcp/ip nät skall du också ta reda på din ip-adress, vilken subnätmask som gäller, vilken DNS-server kan du använda (om sådan finns) och om det finns en gateway och i så fall dess ip-nummer. Står din dator på ett ställe där ett annat nätverksprotokoll används bör du tala med systemadministratören om vad som gäller.

Hårddisken får inte vara komprimerad med något program som till exempel `dblspace`. Dessa program måste ju köras för att man skall kunna förstå vad som finns på hårddisken. Skulle du idag ha en komprimerad hårddisk rekommenderar jag starkt att du

köper en till. Hårddiskar är så pass billiga, och stora, idag att ingen skall behöva sitta med en komprimerad hårddisk.

19.5 Se upp med Windows-hårdvara

Idag finns det viss hårdvara som bara fungerar i Windows. Detta beror på att dumsnåla tilverkare utelämnar processorn i hårdvaran och istället skriver en emulator för Windows som med hjälp av systemets processor gör den utelämnade processorns job. Detta gäller idag fämst modem och skrivare. Dessa skrivare och modem är idag omöjliga att få igång under Linux. Passa dig för sådana. Skall du köpa något nytt så fråga i butiken om det går att använda i Linux. På så sätt får du garantier för att det fungerar och butiken blir upplystom att Linux finns och att fok köper saker till det. Förhoppningsvis frågar butiksägaren i sin tur sina leverantörer om stöd för Linux och så är hjulet i rullning ...

19.6 Skapa installationsdisketter

Till några av de stora distributionerna behöver man inga startdisketter. Antingen finns det en .bat-fil som man kan köra från DOS eller så är CD-skivan bootbar. Det vill säga man kan boota från skivan istället för från hårddisk eller diskett. Detta klarar de flesta moderna moderkort.

I vissa fall måste man ha disketter. Om du köper en packeterad distribution får du oftast med disketterna. Men om du köper en spegling av ett ftp-arkiv eller laddat ned din distribution från internet måste du själv skapa disketterna. Det samma gäller om du skall installera direkt via ett nätverk. Hur detta går till beror lite på vilket operativsystem du nu har. Har du inget operativsystem kan du skapa disketterna hos en kompis.

Om du har ett MS-DOS eller Windows system kan du använda programmet `rawrite`. `rawrite` finns med på skivan med din distribution. Du startar programmet genom att ange dess namn, alternativt dubbelklicka på det i filhanteraren/utforskaren. Först frågar programmet efter en källfil. Sedan efter i vilken enhet disketten sitter. Figur 19.1 visar en körning med `rawrite` i Windows 95.

Är du lycklig nog att redan ha tillgång till ett linux- eller unixsystem kan du använda kommandot `dd` för att skapa disketterna. Kommandot

```
keron:~$ dd if=bare.i of=/dev/fd0
```

gör samma sak som rawrite i exemplet ovan på en Linuxmaskin. På vissa Unixdialekter måste fler argument skickas till programmet. Det är inte säkert (eller ens troligt) att diskettenheten heter /dev/fd0 som den gör i Linux.

Du behöver som regel minst två disketter. En som bootar upp datorn med Linux. Sedan behöver du en diskett med ett litet filsystem. Disketterna brukar kallas för boot- och root- disketter. I vissa fall behöver du en tredje diskett som innehåller drivrutiner för

Figur 19.1: `rawrite` i Windows 95

`rawrite` i Windows 95. Här skrivs imagen `bare.i` till en diskett i enhet `A:` (Installation av Slackware 3.6)

specialsaker du kan behöva. Men oftast använder man en root-disk som är anpassad efter sina behov. Läs mer on detta i manualen till din distribution.

19.7 Partitionera din hårddisk

Linux måste installalleras på minst egen partiton[3]. Detta kan du åstadkomma på något av följande vis. Frigör utrymme på en av dina befitliga partitioner eller hårddiskar, partitionera om din hårddisk eller montera in en ny hårddisk.

Det är att föredra att göra nåt av de två första. Det är säkrare att flytta all data från en partition till en annan eller att sätta i en ny hårddisk än att dela en befintlig. Men även att dela en befintlig partition är möjligt utan dataförlust.

Den tomma partitionen bör vara minst 400MB. En bekväm storlek för en distribution är ca 1GB[4].

19.7.1 Partitionera en tom disk

Om du skall installera Linux på en tom disk (eller disk med data som ej skall vara kvar) kan du starta installationen direkt. Installationsprogrammet kommer då att starta ett program som låter dig partitionera disken. Detta program heter inte sällan `fdisk`. Vissa distributioner har egna verktyg för partitionering, men eftersom `fdisk` alltid (hmm, nåja) finns tillgängligt så beskriver jag det här. `fdisk` liknar programmet med

[3] En partition är enkelt förklarat en del av en hårddisk som fungerar som en egen hårddisk.
[4] Hur kan ett OS ta 1GB? –Det gör det inte, du får ju även med alla applikationer du behöver.

samma namn som du kanske känner igen från DOS och dess vänner. Du kan även starta upp maskinen med en bootdiskett och köra ett partitionsprogram från disketten. Man brukar rekommendera att man använder ett partitioneringsprogram som hör till det operativsystem man tänker installera. Här beskrivs lite kort hur Linux `fdisk` fungerar. Jag börjar med att starta fdisk. Då jag startat fdisk så ber jag programmet om hjälp med kommandot m.

```
keron:~# /sbin/fdisk /dev/hda

Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Nedan beskrivs de alternativ som du måste använda (och som inte helt förklarar sig själv):

a toggle a bootable flag Detta används för att ange om en partition skall vara bootbar eller ej.

t change a partition's system id Detta används för att markera för dina operativsystem vilken typ av partition som det är. Detta ändrar inget i partitionen ytan markerar den bara som en viss typ. Kommandot l visar dig vilka typer som finns.

p print the partition table Du kan alltid kolla med P hur din partitionstabell ser ut. Kolla denna innan du avslutar fdisk med w.

q quit without saving changes Är du det minsta osäker på att du gjort allt rätt avsluta med detta kommando.

w write table to disk and exit Då du är riktigt säker på att du fått till allt som du vill, avsluta med w.

Nedanstående körning visar hur man skapar en linuxpartition och en swap partition på en tom disk (I detta fall en Zip-skiva).

```
keron:~# /sbin/fdisk /dev/sda

Command (m for help): p

Disk /dev/sda: 64 heads, 32 sectors, 96 cylinders
Units = cylinders of 2048 * 512 bytes

   Device Boot    Start      End   Blocks   Id  System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-96): 1
Last cylinder or +size or +sizeM or +sizeK ([1]-96): +16M

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (18-96): 18
Last cylinder or +size or +sizeM or +sizeK ([18]-96): 96

Command (m for help): a
Partition number (1-4): 2

Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap)

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 83

Command (m for help): p

Disk /dev/sda: 64 heads, 32 sectors, 96 cylinders
Units = cylinders of 2048 * 512 bytes

   Device Boot    Start      End   Blocks   Id  System
/dev/sda1             1       17    17392   82  Linux swap
/dev/sda2   *        18       96    80896   83  Linux native

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
Re-read table failed with error 16: Device or resource busy.
Reboot your system to ensure the partition table is updated.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
keron:~#
```

19.7.2 Partitionera en disk med data

Ta säkerhetskopior på dina data innan du börjar

Har du ett fungerande system på maskinen som du vill ha kvar efter installationen av linux skall du redan nu ta backup på den. Jag kommer inte att skriva detta flera

gånger så gör det nu! Du skall också ha en startdiskett för ditt nuvarande system. Hur du skapar en sådan framgår av manualen till ditt system.

Du har väl tagit backup?

fips

fips är ett program som följer med de flesta moderna distributioner. Det kan dela en hårddisk i flera partitioner utan att förstöra befintligt data. Att dela partitioner utan att förstöra data är ett väldigt svårt och onaturligt förfarande i datavärlden, du har väl tagit backup. Anledningen till att jag tjatar om backup är att du inte skall komma till mig och klaga. Jag kan nämna att jag aldrig själv har förlorat data vid ompartitionering med `fips`, och jag har använt det många gånger...

För att du skall kunna använda fips måste du ha tillgång till ett fungerande DOS eller Windows system. Eller åtminstone en bootdiskett. Du behöver också ett defragmenteringsprogram för att samla datat på disken på ett ställe. DOS/Windows defrag fungerar utmärkt. Vidare behöver du en bootbar diskett för att kunna skapa en "det-gick-åt-skogen-laga-diskett". Du har väl tagit backup?.

Börja med att utföra en defragmentering av din disk. Detta skall du göra även om defragmenteringsprogrammet säger att en defragmentering inte är nödvändig. Detta krävs för att samla all information på disken i början på den. Det ger ju en sammanhängande yta i slutet på disken. Det fips gör är att dela partitionen i två delar så att all befintlig information hamnar på den ena och den andra förblir tom.

Starta sedan programmet **fips** från en diskett eller från CD-ROM. Du måste först boota datorn i DOS-läge, det vill säga fråm em DOS bootdiskett eller välja *Starta datorn i MS-DOS läge?* i dialogrutan *Avsluta Windows* i Windows 9x. Så här ser `fips` ut då det startar:

***** Fips screen skall in här *******

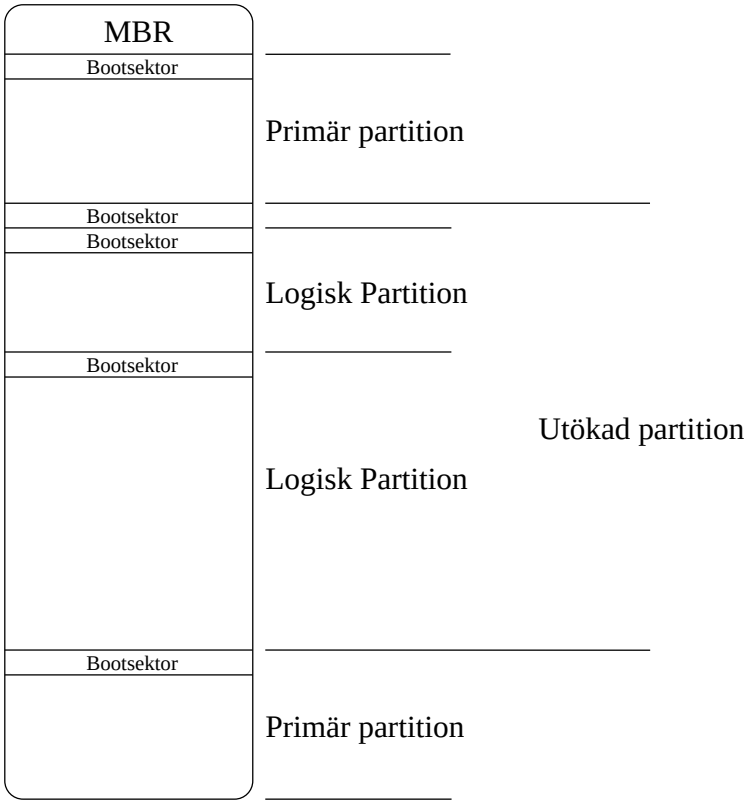
Du behöver bara dela disken en gång så att du får en tom partition. Denna kan du sedan dela i två (eller fler) delar med fdisk eller ett liknande program från installationsprogrammet till din distribution. Så nu kan du starta installationen.

19.7.3 Hur många partitioner skall jag skapa?

Detta behandlas mer utförligt i Kapitel 29 på sid 227

Nu när du har en tom partition för Linux är det dags att dela den i ytterligare partioner för att installera Linux. Detta kan du som sagt göra från installationsprogrammet.

Hur många och hur stora partitioner som skall skapas är en av de svåraste frågor du ställs inför då du skall installera Linux. Ett minimikrav är två partitioner till Linux, men jag rekommenderar att du skapar flera. Första gången du installerar Linux kan det dock vara enklast att bara ha två partitioner. Vill du till exempel ha Windows på samma dator måste du dela in hårddisken i tre delar (eftersom du redan skapat en partition för Linux gäller det att bara skapa nya partitioner i den och inte röra den,



Figur 19.2: Skiss av partitionerad hårddisk

eller de, partitioner som hör till ditt nuvarande system.). Figur 19.2 visar en partitionerad (delad) hårddisk. Partitionerna till Linux bör vara ungefär dubbla storleken av ditt RAM till swap-partitionen och minst 400-600 MB till den partition som skall innehålla Linux filsystem. Vill du göra en riktigt proffsig installataion skall du göra fler partitioner, läs i Kapitel 29.

19.8 umsdos

Det finns ett speciellt filsystem till Linux som kallas umsdos. Med detta filsystem kan du installera Linux på en dator som kör DOS eller Windows utan att partitionera om hårddisken. Linux installeras då i en katalog på disken. Detta fungerar bra men gör att systemet blir lite segt. Om du kommer att arbeta mycket i Linux är detta inte att rekommendera. Är du intresserad tycker jag att du skall titta på distrubutionen slackware och deras Zip-slack[5].

[5] Se: `http://www.slackware.org`

19.9 Starta installationen

När du skall till att starta själva Linuxinstallationen kan man göra det på två olika sätt beroemde på dator och distribution. En modern dator klarar ofta av att starta från CD-ROM enheten. Detta är den enklaste metoden. Du anger bara att din dator skall starta från CD-ROM[6], sätter i Linux-skivan och startar datorn. Du kommer då direkt in i installationsprogrammet.

Alternativt startar du med den startdiskett som du skapat. Eventuellt så byter du till root-disketten då systemet säger till dig att göra det.

Installationsprogrammen är idag så pass enkla att det nästan är fånigt att berätta om dem. Men här beskriver jag lita av vad som brukar upplevas som krångligt.

19.9.1 Mountpoints

I installationsprogrammet kommer du förr eller senare (snarare förr) till ett stadium då programmet vill veta var du skall montera dina partitioner. Detta steg förekommer inte i till exempel Windows och upplevs därför ofta som svårt. Om du skapade två partitioner till Linux skall den största monteras som ”/” och den andra som swap. Om du i `fdisk` markerade swappartitionen som swap så förstår installationsprogrammet att du vill ha den som swap. Du känner enklast igen partitionerna på deras storlekar. Men det är även lätt att känna igen dem på namnen. Linux namnger partitionerna enligt följande:

- IDE
  - /dev/hda = Första hårddisken på första IDE anslutningen
  - /dev/hda1 = Första partitionen på första disken
  - /dev/hda2 = Andra partitionen på första disken
  - /dev/hdb = Andra disken på första IDE anslutningen
  - /dev/hdb1 = Första partitionen på andra disken
  - /dev/hdc = Första disken på andra IDE anslutningen
  - osv..
- SCSI
  - /dev/sda = Första SCSI-disken
  - /dev/sdb = Andra SCSI-disken
  - i övrigt som IDE ovan.

Mer om alternativa mountpointer (monterinspunkter) kan du läsa i kapitel 29

[6] Denna inställning görs i datorns BIOS-konfi gurering, se din dators eller moderkorts manual

19.9.2 LILO (Vid installation)

LILO(LInux LOader) är en så kallad boot-loader. Det är ett litet program som startas i stället för ett operativsystem vid uppstart av en dator. boot-loadern kan sedan välja vilket operativsystem som skall laddas. Finns bara Linux på datorn konfigurerar man LILO så att det startar Linux direkt. I detta skede nöjer vi oss med att konstatera att LILO konfigureras automatiskt av installationsprogrammet. Mer om hur du kan konfigurera LILO kommer längre fram i boken. Vill du att LILO skall starta då datorn startar, som är det vanligaste väljer du att LILO skall installeras på *Master Boot Record* av din primära hårddisk. Har du ett annat operativsystem som du vill behålla bör du för säkerhets skull ha en boot-diskett till detta.

19.9.3 Vad skall jag installera?

Oftast vet man inte som nybörjare vilka program man behöver. Det är väldigt många program och alla har underliga namn. Oftast kan man välja färdiga paket av program beroende på vad man skall använda maskinen till. Det är klokt att välja ett av dessa alternativ istället för att välja program för program eftersom det tar lång tid och du riskerar att säga nej till något program som du behöver.

Passa dig för att välja alternativet “server” eller liknande. Dessa alternativ kan förutsätta att inget annat OS kommer att finnas på datorn och tar gela din hårddisk i anspråk!

19.10 Installera över ett nätverk

Att installera över ett nätverk är inte krångligare än att installera från en CD. Du måste skapa startdisketter. Se till att du har rätt startdisketter. I vissa fall måste du ha speciella disketter för att installera över nätverk. Du måste också ha en diskett med PCMCIA stöd om du skall installera via ett PCMCIA nätverkskort. Alla dessa disketter finns med din Liunuxdistribution. Vidare måste du veta varifrån du skall installera, samt vilket protokoll du skall använda. Det vanligaste är att installera via ftp men om du installerar från ett lokalt nätverk kan du även installera via nfs.

Detta sätt att installera är att föredra om du har tillgång till den typ av ansluning som krävs. Du slipper ha en CD-skiva liggande som i alla fall blir gammal på någon eller några månader. Du har ju i alla fall tillgång till de senaste filerna via din nätverksanslutning! Denna installering tar valnigtvis lite längre tid än att göra det via CD i synnerhet om du installerar via Internet men. som sagt, det kan det vara värt. Jämfört med att tanka hem alla filer och sedan installera så går det ändå fortare. Du behöver dessutom bara tanka hem de filer som du verkligen behöver.

Kapitel 20

Installera och konfi gurera X

läs först kapitel 6 på sidan 93 innan du läser detta kapitel. I det kapitlet beskrivs vad X är och kort hur det fungerar och används.

Har du inte X på din maskin så rekommenderar jag att du skaffar det. Har du ingen speciell anledning att **inte** köra X så finns det ingen anledning att sitta och köra i konsolläge. Du vill förmodligen inte köra X om

- Maskinen används som en server
- Du har en väldigt klen maskin

I det första fallet kan X installeras för att underlätta underhåll vid maskinen. X skall dock inte köras då det inte används. Oftast sker mycket av det nödvändiga underhållet från en annan maskin vid vilken det ofta är bekvämare att sitta och arbeta. Serverutrymmen har en tendens att vara mycket kalla och bullriga. En maskin som bara kör som någon typ av server behöver inte heller ha varken tangentbord eller bildskärm. I en sådan situation är naturligtvis X helt onödigt.

Du vill absolut köra X om

- Någon skall jobba vid maskinen. En sådan maskin kallas en arbetsstation.

X följde med största sannolikhet med din distribution. Det är dessutom förmodligen också installerat. Vet du inte kan du prova att ge kommandot **`startx`** eller **`xinit`**. Finns inte dessa kommandon så är troligen inte X installerat. Kommer det en massa text och slutligen ett felmeddelande så är X installerat men inte konfigurerat.

20.1 Installera och konfi gurera

X kommer som de flesta andra program som antingen källkod eller som binärfiler. X följer med största sannolikhet med din distribution och installerades dessutom troligen

från början. Om det inte gjorde det så kan du tanka hem den på `ftp://ftp.xfree86.org/pub/XFree86`(Läs filen `MIRRORS` för att få reda på var det finns en spegling nära dig). I den katalogen finns det både källkod och binärfiler till en mängd arkitekturer. Senaste versionen av XFree86 då detta skrivs är 3.3.5. Jag rekommenderar dock inte att du, om du är ovan, installerar X från något annat än din leverantör av Linux. Vill du ha X (det vill du) och inte har det tycker jag att du skall skaffa dig en modern Linuxdistribution som inkluderar X, vilket de flesta gör.

Läs mer om hur du installerar program i kapitel 21 på sidan 197

20.1.1 Välja X-server

Du måste någon gång under installationen välja en X-server. Det finns flera olika X-servrar till olika grafikkort. Oftast väljer du X-server genom att välja ditt grafikkort i en lista. Men det ka ändå vara bra att veta vilken X-server man vill använda. På sidan `url.till.serverlistan`[1] finns en lista över stödda kort och vilken server som skall användas till vilket kort. Är du förundrad över vad en X-sevrer är bör du kanske läsa om kapitel 6.

Du bör alltid installera den server du vill använda och *VGA_16* servern eftersom den används av en del grafiska konfigureringsverktyg som till exempel `XF86Setup`. Anledningen till att dessa verktyg använder den servern är att den stöds av alla moderna skärmar/grafikkort.

Ofta då du instellerar en modern Linuxdistribution behöver du bara välja ditt grafikkort i en lista och den rätta servern installeras. I vissa fall hittar installationsprogrammet ditt grafikkort självt och du behöver inte alls bry det om det!

20.2 `XF86Config`

Bilaga B på sidan 305 visar konfigureringsfilen `XF86Config` för min bärbara dator.

Konfigurationsfilen för X heter `XF86Config`[2]. Denna finns i `/etc/` eller i vissa fall `/etc/X11/`. Denna fil består av olika delar.

20.2.1 Files

Under Files anges sökvägen till olika filer som X behöver.

FontPath

Sökvägen till fonter. Se Bilaga B för exempel.

[1] jag hänvisar till sidan istället för att inkludera data i denna bok. Detta för att listan hela tiden ändras och för att spara på papper om du skriver ut, eller köper, denna bok.

[2] Att kunna syntaxen för hela XF86Confi g får anses som överkurs. Är du inte intresserad kan du hoppa över detta stycke och läsa om konfi gureringsprogrammen nedan.

RGBPath

Sökvägen till den så kallada RGB databasen. RGB-Databasen är en databas som innehåller alla definierade färgnamn och deras RGB värden. RGB-värdet[3] är ett värde som består av tre tal. Varje tal är en byte stort (0-255). Och representerar färgstyrkan av färgerna Röd Grön och Blå. Filen heter oftast rgb.txt, i denna kan du se alla färgnamn som är definierade på din maskin. Vet du inte var den finns kan du ju titta i din `XF86Config`.

Det finns mer att säga om dessa värden. XXXXX

ModulePath

Sökvägen till eventuella moduler som kan länkas dynamiskt.

20.2.2 Module

Dynamiskt länkade moduler. Dessa används inte så flitigt. Används i samband med *XImput* (se nedan). Anges bara filnamnet utan sökväg, söks katalogerna i ModulePath.

20.2.3 ServerFlags

Denna sektion använd för att styra X servern på det sätt man önskar.

XXXX Skall fylla på mer här.

20.2.4 Keyboard

Här konfigureras tangentbordet. Det finns flera alternativ för denna sektion. De rubriker som finns är:

- Protocol [Standard | Xqueue]
  - Standard – Används i de flesta fall.
  - Xqueue – Brukar speciell händelse kö (eventQueue).
- AutoRepeat värde:

20.2.5 Pointer

Konfiguration av pekdonet (musen)

[3] Du som har gjort hemsidor i HTML är redan bekant med sådana värden.

20.2.6 Monitor

Bildskärm.

20.2.7 Device

Grafikkort

20.2.8 Screen

Skärmbild.

20.2.9 XInput

Möjlighet till extra inmatningsenheter.

20.3 Setup verktyg

Det är inte speciellt smart att från början skriva en `XF86Config`-fil. Det finns många bra verktyg som gör det för dig. Här presenterar jag de tre vanligaste. Det är ingen ide att lära dig alla tre. Lär dig den som finns tillgänglig på ditt system. Jag rekommenderar dem i den ordning jag tar upp dem här.

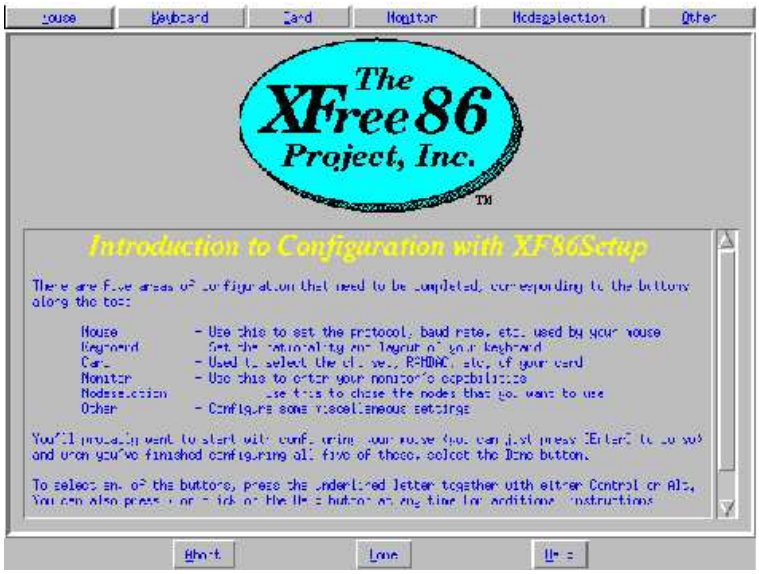
Detta är, som de flesta, ett område på vilket stor utveckling sker. Det är svårt att hålla en bok uppdaterad. Det är mycket möjligt att din distribution har en modernare variant av den här typen av verktyg. Det som beskrivs här gäller i alla fall i stort i de flesta verktyg.

20.3.1 XF86Setup

XF86Setup är ett snyggt grafiskt verktyg för att skapa en XF86Config fil. Det startar direkt i det grafiska läget. Detta är möjligt eftersom det använder vga servern. Vga grafik klarar så gott som alla bildskärmar och grafikkort. Hur XF86Setup ser ut visas i figur 20.1.

20.3.2 Xconfi gurator

Xconfigurator är RedHat's verktyg för att skapa en XF86Config fil. Det startar automatiskt vid installation av senare versioner av RedHat. Om du missade det vid installationen kan du köra det igen. Detta kräver att det finns installerat. Finns det inte installerat kan du enkelt installera det med till exempel `glint` eller `gnorpm`, mer



Figur 20.1: XF86Setup

Då man trycker på knapparna i överkanten så visas en inställningssida för det man valt. Om inte musen fungerar på en gång kan man använda Tab *och piltangenterna för att ställa in musen.*

om detta står i kapitel 21. Gränssnittet är detsamma som du känner igen från installationen av RedHat. Hur Xconfigurator ser ut visas i figur 20.2. Xconfigurator probar[4] Om Xconfigurator hittar ditt grafikkort så är det bra.

Skärminställningen brukar vara det som är krångligast. Finns inte din skärm med i listan över skärmar måste du välja läget själv. Ett bra tips om du inte vet vad din skärm klarar av är att börja uppifrån i listan. De översta raderna är de “snällaste” för din skärm. Sedan blir de vassare och vassare ju längre ned i listan du går. Tänk på att ett för hårda inställningar kan skada din skärm. Moderna CRT[5] skärmar är inte så ömtåliga som det står i varningstexterna. Men det skadar aldrig att vara försiktig. Framför allt skall man vara försiktig om man har en gammal skärm eller en LCD-skärm[6].

20.3.3 xf86config

xf86config är ett textbaserat verktyg. Det är ganska självförklarande. Det går ut på att man skall svara på ett antal frågor. Programmet skapar sedan en XF86Config-fil. Gå igenom frågorna noga och läs all information du får så skall det inte vålla några problem.

[4] Att “proba” innebär att mjukvaran försöker fråga hårdvaran saker för att lista ut vad det är.
[5] CRT=Cathode Ray Tube, Den vanligaste typen av bildskärm
[6] LCD=Liquid Chrystal Display, Vanlig i bärbara datorer och platta skärmar

Figur 20.2: Xconfigurator

RedHat's Xconfigurator, här har programmet identifierat grafikkortet på min bärbara dator.

20.3.4 xvidtune

Ibland händer det att bilden blir lite för liten, eller hamnar lite snett på skärmen. Det kan man fixa med hjälp av `xvidtune`. Då programmet startar får man en varningstext. Denna gör gällande att din skärm förmodligen snart kommer att gå sönder. Nu är det inte sådan stor risk att detta händer med en modern skärm så tryck på "OK" för att komma vidare.

Figur 20.3 visar hur `xvidtune` ser ut. Det finns en del knappar som används för att styra programmet.

Left, Right, Wider, Narrower Används för att justera bilden i sidled.

Up, Down, Shorter, Taller Används för att justera bilden i höjdled.

Quit Avslutar programmet

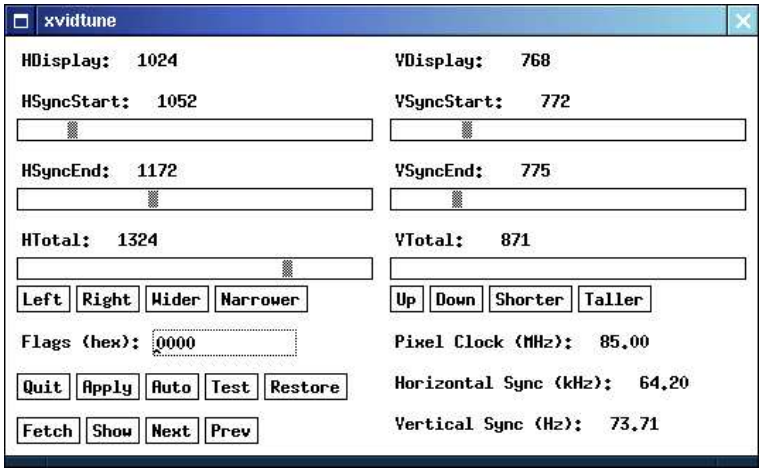
Apply Verkställer ändringarna så du kan se resultatet på din skärm. Denna inställning blir då den du kommer tillbaka till då du trycker på "Restore".

Auto Detta är en knapp som antingen är aktiverad eller ej. Då den är aktiverad så uppdateras skärminställningarna direkt då du ändrar på dem.

Test Ändrar läget till det du valt tamporärt.

Restore Ändrar tillbaka till inställningen som var då du senast tryckte "Apply".

Fetch Hämtar informatio från X-servern om vilket läge den för tillfället kör i och ställer in reglagen efter detta.



Figur 20.3: `xvidtune`

Så här ser `xvidtune` ut. Programmet används för att finjustera inställningarna för din skärm och skapa en "modeline" rad till din `XF86Config`-fil.

Show Skriver ut en "modeline"-rad i det terminalfönster du startade programmet. Denna går att direkt klistra in i din `XF86Config`-fil.

Next Byter till nästa "mode" om du har flera modes konfigurerade i din `XF86Config`-fil.

Prev Byter till föregående "mode".

Skulle bilden bli oläslig, det vill säga bara flimmer eller ränder, så kan du trycka på R så återställs skärmen till det den var då du senast tryckte på "Apply". Då du hittat en instlällning spm passar trycker du på "Show". Då visas hur "modelinen" ser ut. Starta din favorit editor och klistra in raden på lämpligt ställe i `XF86Config`.

Kapitel 21

Installera ny programvara

Att installera program i Linux är numera en trivial sak. Skaffar man ett binärpaket av den önskade programvaran till sin distribution är det en smal sak att installera denna. Man kan dock stöta på problem om man har otur. Det kan till exempal bero på att man vill installera ett program som i sin tur kräver andra program. Vissa program leveras dessutom bara som källkod, vilket gör det krångligare att installera dem. Som tur är så kommer de allra flesta program med bra instruktioner om hur man skall installera dem.

21.1 Varför är det så krångligt?

Det korta svaret på den frågan är att det inte är krångligt. Fast det vore kanske lite att blunda för verkligheten. Vi börjar med att behandla vad som kan gå fel.

För det första så finns ju de flesta program man installerar i sitt Linuxsystem tillgängliga på Internet. Detta gör det naturligt att man själv laddar hem sina program. Det finns då naturligtvis en risk att filen eller filerna blir korrupta. Detta problem finns i alla operativsystem.

Då man använder binära (redan kompilerade) paket av ett program kan man råka ut för *failed dependencies* eller olösta beroenden som det kanske skulle heta på svenska. Detta beror i huvudsak på två saker. För det första så är det ganska vanligt att program är så kallade *frontends* eller fasader till andra program. Då krävs det ju att grundprogrammet är installerat. Det förekommer även att ett program använder sig av andra program. Man kan ju då fråga sig varför inte allt som behövs inkluderas i programpaketet. Detta beror ofta på att man vill hålla de nedladdningsbara paketen så små som möjligt. Vanligare är kanske att det inte alls är samma personer eller företag som utvecklar de inblandade programmen.

Man kan även få *failed dependencies* av andra skäl. Vissa program är dynamiskt länkade mot olika bibliotek som normalt finns på ett linuxsystem. Detta gör man dels för att minska storleken på filerna och för att öka prestandan. Det fungerar så att när ett program kompileras (görs körbart) så byggs inte allt in i programmet utan man räknar

med att vissa funktioner redan finns på målsystemet. Den som är insatt i Windows känner igen detta från de så kallade `.dll`-filerna som finns i det operativsystemet. Att storleken på filerna miskar är självklart eftersom inte allt följer med det nya programmet. Prestandan ökas genom att om flera program använder samma bibliotek så behöver detta bara laddas en gång i minnet. Detta leder naturligtvis till att man sparar på minne jämfört om alla program skulle lagra samma funktioner en gång var. Vad är då problemet? Jo, för det första måste dessa bibliotek (som varierar från program till program) finnas tillgängliga på ditt system. Detta är som regel inget större problem. Ett större, och besvärligare, problem uppstår om programmet du vill installera kräver nyare versioner av de bibliotek du har. Om vi återigen jämför med Windows så kan man ju konstatera att ett program för Windows98 inte fungerar i Windows3.0 detta är bland annat för att biblioteksfilerna inte stämmer överens. I Windows är dett dock inget problem eftersom uppdateringar inte sker så ofta. Om du har Windows95 så skaffar du program till det. I Linux sker utvecklingen kontinuerligt. Detta gör att din distribution ganska snabbt och obemärkt blir gammal. Programtillverkarna vill snabbt använda de nya funktionerna som inte finns i din något äldre installation. Du får då problem med att installera programmet och måste installera nyare versioner av andra paket innan du kan installera programmet du vill ha.

Hoppas nu att detta har gjort det lite klarare för dig. Detta låter väldigt krångligt, men om du använder de verktyg som följer med ditt system så skall det inte innebära några problem.

21.2 deb

Debians paketformat kallas `deb`. Dessa filer har således suffixet `.deb`.

21.3 rpm

`RPM` är RedHats paketsystem.

21.4 tgz

Tgz är bara filerna packade med tar och gzip. Detta används av bland anna Slckware som har ett verktyg som heter pkgtool för att hantera dessa.

21.5 Källkod

Det krångligaste och långsammaste sättet att installera är att installera från källkod. Eftersom många program till Linux (och Linux självt) släpps med öppen källkod så är det naturligt att installera från källkod. Naturligtvis så måste programmet då först kompileras vilket tar lite tid och inte är helt enkelt alla gånger. Den första meningen

i detta stycke kan göra att man får den uppfattningen att jag inte gillar att installera källkod. Så är dock inte fallet. Det är även det lärorikaste, mest äkta och roligaste sättet. Syftet med den första meningen var att varna läsaren för att det kan vara krångligt att installera från källkod. Mitt tips är alltså; Vill du installera och köra programvaran direkt utan krångel, skaffa en binär version av programmet i det paketformat som hör till din distribution!

Hur man installerar från källkod är dock lite krångligt att beskriva. Det beror på att det finns så många olika sätt. Det är ju helt upp till programmeraren hur hans skapelse skall kompileras och vilka verktyg som behövs. Här går jag generellt igenom vad som brukar behöva göras.

Först måste du packa upp program-distributionen på ett lämpligt ställe. Jag rekommenderar i enlighet med filsystemstandarden att du packar upp källkoden i `/usr/local/src/programm` alternativt `/usr/src/programnamn-1.2.3/`. Källkoden kommer nästan alltid som en tarad och gzippad alternativt bzippad fil. Hur du packar upp dessa beskrivs i avsnitt 7.8 på sidan 114.

Då du packat upp källkodsträdet skall du **alltid** läsa en fil som heter `README`, `INSTALL` eller liknande. I denna fil står det hur kompileringen och installationen går till och vad du behöver. Det är väldigt jobbigt och irriterande för din lokala linuxguru att få frågor om sådant som du enkelt kunnat läsa dig till i sådana filer.

Ett vanligt sätt är att det i källkodsträdets rot finns en fil som heter `configure`. Denna skall du börja med att köra (**`./configure`**). Detta skript kollar att din maskin har allt som behövs för att kompilera programmet. Det skapar en fil som heter `Makefile` i källkodsträdets rot (och ofta en i alla underkataloger). Mer om Makefilen kan du läsa i kapitel 43 på sidan 275.

Jag rekommenderar att du nu läser Makefilen (även on du inte förstår den). I början finns det ofta variabler som det kan vara bra att veta vad de är satta till och iband kan det vara bra att ändra dem. Framför allt är det intressant att veta i vilka kataloger programmet kommer att installeras. Jag rekommenderar att du installerar alla program du kompilerar själv i katalogen `/usr/local` och dess underkataloger. På så sätt så vet du var alla dina egenhändigt kompilerade program hamnar. Det kan underlätta vid en uppgradering eller säkerhetskopiering.

Nästa steg, om föregående fungerade, är att starta kompileringen. Detta görs med kommandot **`make`** med det nya programmets källkodsträds rot som aktuell katalog. Nu kompileras programmet. Detta brukar ta sin lilla tid så du kan gott gå och ta dig en kopp kaffe eller umgås lite med din familj under tiden.

När programmet är kompilerat skall det flyttas till de ställen där filerna skall ligga för att du enkelt skall kunna använda programmet. Dett görs vanligtvis med kommandot **`make install`** i samma katalog som tidigare. Nu kopieras filerna dit de skall vara.

Denna korta beskrivning är en generell beskrivning av hur det brukar gå till att installera källkodspaket. Det behöver inte alls vara så att det fungerar så här med det program du vill installera. Däremot står det alltid i någon fil i källkodsträdet rot hur man gör. Läs denna fil och lär dig hur man gör innan du börjar.

Kapitel 22

Uppgradera ditt system

I detta kapitel skall vi gå igenom saker man bör tänka på då ett system skall uppdateras. Detta kan också belysa varför man bör dela in hårddisken(arna) i flera partitioner.

22.0.1 Olika typer av uppgradering

Man kan uppgradera sitt system på flera olika sätt. Antingen vill man bara uppdatera vissa väsentliga program och låta resten vara orört. Eller så vill man uppdatera hela sitt system till en senare version av samma distribution. Den hårdaste typen av uppgradering innebär att man tar bort hela sitt system och installerar en senare version av samma distribution eller byter till en annan distribution.

De flesta distributioner har idag möjligheten att uppdatera sig själva. Det vill säga skall du updatera från en version av en distribution till en nyare av samma kan du köra installationsprogrammet och i installationen välja att du vill uppgradera. Det finns flera anledningar till att ändå ta bort allt och göra om installationen från början. Har du dock konfigurerat och justerat ditt system till att passa just din maskin och inte gjort det på ett smidigt sätt så kan uppgraderingen bli väligt omfattande.

Ett sista sätt att uppgradera sitt system är att helt enkelt lägga till flera program. Det gör man enklast genom att anväda de program som finns till sin distibution, till exempel `rpm` eller `dpkg`. Men vill man göra det som en riktig Linuxfantast så skall man kompilera sina egna program, lite jobbigare, lite svårare men mycket roligare ...

Man bör vara lite restriktiv på hur ofta man uppgraderar. För det första så kommer det nya versioner av distributionerna väldigt ofta. En uppgradering av hela systemet tar en hel del tid. Man bör alltså väntra med att uppdatera systemet till dess att man tycker att det man har är omodernt. Är man intresserad av att ha ett säkert system bör man dock hålla koll på de säkerhets fixar som kommer till alla distributioner då och då.

En annan varning som kan vara värd att fundera på om man är lat är att vänta ett tag efter att en ny version av sin distribution kommit ut. Detta för att slippa de barnsjukdommar som en ny version ofta kan lida av. Genom att låta andra installera först kan man räkna med att de hittar felen och att distributören hinner fixa dessa tills det blir din

tur. Man bör vra extra försiktig med stora uppdateringar av disributionerna. Till exempel när RedHat uppdaterades från 4.2 till 5.0 introduserades en hel del fel. De flesta av dessa hade fixats till 5.1 som dock snabbt uppdaterades till 5.2. De som väntade med sin version 4.2 till dess att 5.2 hade kommit ut besparades en del bekymmer.

22.1 Uppgradera vissa program

Idag leveraras de flesta program som binära filer. För att hålla reda på alla filer som hör till ett visst program används ett system som innebär att alla program levereras i paket. Varje paket innehåller de filer som hör till programmet. Dessutom innehåller paketet information om var i filsystemet de olika filerna skall ligga. Det finns även information om vilka andra program, bibliotek mm. som behövs för att programmet skall gå att köra. Det finns flera sådana paketystem.

22.1.1 rpm-format

rpm (RedHat Packet Manager) är det största systemet. Det används bland annat av RedHat och S.u.S.E. För att adninistrera dessa paket, som har filnamn som `paket-xx.xx.x.rpm` där x-en står för versionsnumret, finns det flera olika verktyg. De två vanligaste är `rpm` som är ett kommandoradsprogram och `glint` som är ett grafiskt gränssnitt till `rpm`. Rpm är ett smidigt system som är lätt att lära sig.

XXX Mer info om hur rpm används

22.1.2 deb-format

deb-formatet är ett format som används av Debian. Deb-formatet liknar i många avseenden rpm men har vissa fördelar.

XXX Mer om fördelarna med Deb

Nackdelen med deb-formatet är att det idag är rätt så omkört av rpm. Allt flera stöder rpm. Detta gör att program som släpps som binärfiler oftast först kommer som rpm. Nu är det så att rpm inte är bundet ett visst system utan kan köras på de flesta moderna Distributioner. Kan behövas lite trixande i Slackware som har en annan systemdesign än de flesta andra stora distributioner. XXXX Stämmer det forfarande. XXX Skall skriva mer om detta i “Mer om Linux”

För att administrera deb-paket finns precis som med rpm både kommandoradsprogrammet `dpkg` och det grafiska `dselect`. `dselect` är också textbaserat och inte lika snyggt och lättanvänt som `glint`.

22.2 Uppgadera till en nyare version

22.3 Byta distribution

Om du skall helt byta distribution. Det kan vara en bra ide om du vill testa flera olika. Det är inte alltid så att den du provar först passar dig bäst. Om du skall byta kan du ha sparat mycket arbete genom att ha varit smart då du partitionerade din/dina diskar. Har du till exempel `/home` och `/usr/local` på egna partitioner så blir bytet en barnlek.

22.3.1 Ta backup

Eftersom detta innebär att du raderar ditt nuvarande system kan det vara klokt att se till att du har backup på hela ditt system. Då kan du ju återställa det om något skulle gå galet. Det kan vara smidigt att ha backup på `/etc` lätt tillgänglig då du installerat ditt nya system eftersom du kan vilja kika lite på hur filerna i denna katalog såg ut innan ominstallationen. Om du väljer att bara ta backup på delar av ditt system skall du noga tänka efter vad som behövs kopieras. `/usr` innehåller per definition bara sådant som hörde till din förra distribution och behöver således inte kopieras. Undantaget är `/usr/local` som i allra högsta grad måste kopieras eftersom den innehåller saker som du trixat till själv. Om den sitter på en egen partition är det bara att ange var den skall monteras i det nya systemet. Har du tur behöver du sedan inte göra något åt den. Detsamma gäller `/home`. På `/home` bör du ta flera kopior. Det är ingen trevlig känsla att inse att man trixat bort `/home`. I `/var/` skall du ta backup på de kataloger som innehåller saker som Mail och eventuella skrivarköer. Loggar och dylikt behöver du inte kopiera om du inte har ett system för att arkivera dem.

22.3.2 Starta installationen

22.4 installera program

22.5 Uppgradera kärnan

Linuxkärnan kommer med jämna mellanrum ut i nya versioner. hur du uppdaterar kärnan beskrivs i kapitel 2.4.

Kapitel 23

Kompilera om kärnan

Att kompilera om kärnan är inget man gör om det inte behövs. De kärnor som följer med distributionerna är stabila. Det finns naturligtvis tillfällen då det kan vara värt att uppgradera sin kärna. Exempel på sådana tillfällen kan vara är.

- Stöd för en viss hårdvara man har finns bara i nyare kärnor.
- Man vill lägga till stöd för mer hårdvara.
- Man är helt enkelt nyfiken och vill se hur man kompilerar om kärnan.

Det kan också vara på sin plats att säga att trots att detta är en kritisk och farlig uppgift så är det inte svårt. Tänk dig bara för så skall allt gå bra. Läs igenom hela detta kapitel **innan** du börjar så att du vet vad som skall göras. Gå sedan tillbaka hit då du skall till att börja.

För att kompilera om sin kärna är det följande som skall göras:

- Skaffa allt som behövs, inklusive källkoden till en ny kärna.
- Spara undan det nuvarande ifall något skulle gå snett.
- Packa upp det nya källkodsträdet.
- Konfigurera den nya kärnan.
- Kompilera kärnan och eventuella moduler.
- Installera den nya kärnan och modulerna.
- Uppdatera boothanteraren att starta den nya kärnan, men också se till att den kan boota den gamla utifall …

23.1 Vad behövs

För det första så måste du ha källkoden till den version[1] av kärnan du vill kompilera. Källkoden tar du naturligtvis hem från Internet. Källkoden är en väldigt stor fil (ca 15 megabyte komprimerad) Se källförteckningen för en lista på bra ftp-arkiv. Källkoden ligger tarad och gzippad i en fil som heter linux-x.x.x.tar.gz. Vidare behöver du de rätta versionerna av de program som behövs för att kompilera kärnan. Information om vilka de är finns i filen README som följer med källkoden. När du har allt detta är det bara att sätta igång. Tycker du att det verkar krångligt så kan jag trösta dig med att du förmodligen redan har allt som behövs.

Första gången du kompilerar om kärnan kan det vara bra att utgå från den konfiguration som din nuvarande kärna har. Framför allt om den i stort fungerar bra. Då skall du först installera det källkodsträd som följer med din distribution.

23.2 Spara din gamla konfi guration

Din gamla kärnkonfiguration ligger sparad i en fil `.config` i källkodsträdets rot. Denna kan vara bra att spara undan. Källkodsträdets rot är `/usr/scr/linux`. Denna katalog kan vara en länk till en annan men den skall alltid heta `/usr/src/linux`. Det kan vara en god ide att spara undan hela källkodsträdet. En konvention är att spara källkoden i en katalog `/usr/src/linux-x.x.x` där x:en representerar kärnversionen. Vet du inte vilken version av kärnan du har för närvarande kan du ta reda på det med kommandot **`uname -r`**.

```
$ uname -r
2.2.14
```

För att spara undan vitt källkodsträd skulle jag alltså ge kommandot:

```
# cd /usr/src
# mv linux linux-2.2.14
```

Kontrollera innan du gör detta att `linux` inte redan är en länk till, i det här fallet, `linux-2.2.14`. Skulle så vara fallet behöver du bara ta bort länken.

23.3 Packa upp källkoden

Okej, nu skall net nya källkodsträdet packas upp. Se till att den tarade och gzippade filen ligger i `(usr/src` packa sedan upp den med kommandot:

```
# tar xzf linux-x.x.x.tar.gz
```

[1] Läs avsnitt 2.4 på sid 39 för information om kärnversioner

Se till att det inte finns någon fil linux i katalogen `/usr/src` innan du gör detta, finns en sådan fil kommer den att skrivas över.

Nu har källkoden packats upp i katalogen `/usr/src/linux`. Där skall den ligga för att kompileringen skall lyckas. Anledningen till detta är att katalogerna `/usr/include/linux` och `/usr/include/asm` i själva verket är länkar till källkodsträdet. Du kan kontrollera att dessa är riktiga:

```
$ ls -l /usr/include/{linux,asm}
lrwxrwxrwx 1 root root 24 Oct 13 01:42 /usr/include/asm -> ../src/linux/include/asm/
lrwxrwxrwx 1 root root 26 Oct 13 01:42 /usr/include/linux -> ../src/linux/include/linux/
$
```

23.4 Konfigurera

23.5 Kompilera

23.6 Installera

23.7 Konfigurera LILO

23.8 Prova

Kapitel 24

LILO

24.0.1 `/etc/lilo.conf`

24.1 loadlin.exe

Ibland kan det vara smidigt att kunna boota Linux från DOS. Ett exempel kan vara om du installerat om DOS och fått din master boot record överskriven och inte hittar din Linuxbootdiskett[1].

[1] En bootdiskett har du väl som du har kollat att den fungerar?

Kapitel 25

Allmänt

Nu har vi gått igen så pass mycket att du skall ha tillgång till ett fungerande Linuxsystem. Nu ger vi oss in på lite systemadministration.

25.1 Starta en annan användare, `su`

För det första så skall man inte köra som användaren root mer än nödvändigt. Detta gäller även systemadministratören, som naturligtvis också skall ha ett vanligt användarmamn. Men ibland så måste man ju köra som root för att kunna ändra vissa filer eller köra vissa program. Vill man tillfälligt byta användare så finns det ett program ör det som heter su. Su kan användas för att starta användaren root men även för att starta andra användare. Syntaxen i det enklaste fallet är **`su användarnamn`**. Utelämnas användarnamnet så antas att man vill byta till användaren root. För att aktuell användare skall startas så måste man ange den användarens lösenord. Är man root och vill starta en annan användare så behöver man inte ange något lösenord. Root kan ju i alla fall göra vad han vill. Se nedanstående exempel.

```
$ whoami
marcus
$ su
Password:
# whoami
root
# su hans
$ whoami
hans
$ exit
# exit
$ whoami
marcus
```

Om man vill att ett login skal skall startas för den nya användaren kan man lägga till

flaggan - eller -l. För att starta ett login skal som användaren root ger man således kommandot **su -**.

25.2 Praktiska kommandon

I detta avsnitt behandlas några kommandon som är praktiska för systemadminitsratörer. De flesta av dessa kommandon är också mycket användbara för användare. Så detta stycke är läsvärt för alla. Här finns i denna version av boken också några kommandon som inte riktigt passar in någon annan stans.

25.2.1 uname

Uname är ett program som ger dig information om vilket system du kör. Det är bra att använda i tillexempel initieringsfiler som .bashrc om du har en och samma hemmakatalog på flera olika plattformar eller om du skriver ett skript som kan komma att exekveras på olika plattformar. Du kan då skriva ett vilkor som säger att om systemet är Linuxgör si annars så. Det gär naturligtvis att göra på många olika sätt beroende på vad man vill åstadkomma.

Utan flaggor visar uname vilket operativsystem som är aktuellt, med flaggan **-a** visas all information uname kan bidra med. Flaggan **-r** visar vilken maskintyp som gäller. Flaggan **-p** visar vilken processor systemat har. Flaggan **-r** visar operativsystemets versionsnummer. Slutligen visar flaggan **--help** information om alla flaggor som finns.

25.2.2 stat

Visar information om filer, eller rättare sagt inoder.

25.2.3 df

Df (Disk Free?) visar hur mycket utrymme du har på dina diskar. Utan flaggor visar df information om alla monterade diskar. Man kan även välja att bara visa information om en viss typ av filsystem. Detta gör man då med flaggan **-t em filsys** där *filsys* är den filsystemstyp man vill visa. Fylsystemstypen kan vara samma som man kan ange som argument till mount, se sidan 230 för mer information om dessa. Man kan även istället för att visa information om diskutrymme visa information om *inoder* detta gör man med flaggan **-i**. Vad inoder är beskrivs på sidan**??**. Df är ett mycket användbart kommando.

25.2.4 du

Du (Disk Usage) är ett kommando som med fördel kan användas för att spåra vilka kataloger som äter upp ditt diskutrymme. Detta är användbart även för användare som börjar fylla sin diskkvot. Utan flaggor visar du hur stor varje fil i aktuell katalog är i kilobytes. Katalogfiler visas hur mycket plats de, och alla filer i dem, tar. Alla underkataloger till kataloger i aktuell katalog visas också. Vill man att även filer i underkataloger skall visas ger man flaggan **-a**. Oftast när man gör sådana listningar så blir de väldigt långa. Man vill då minska den information man får. Det gör man med flaggan -s som i summarize. Då visas bara hur stor varje katalog är, inga underkataloger visas. Gör man detta utan att ange några argument till du visas bara hur stor plats aktuel katalog tar. Vill man visa andra kataloger kan man skicka dessa som argument till du. Till exempel visar **du -s *** storleken på alla filer och kataloger i aktuell katalog. Nu kanske man vill veta hur stor några kataloger är tillsammans, då kan man använda flaggan **-c** som visar alla arguments totala storlek.

Som standard visas storleken i kilobytes. Man kan få en mer lättläst utskrift med flaggan **-h**. Då visas datat i ett läsvänligt format. Nämligen i bytes, kilobytes, megabytes, etc beroende på hur stor filerna är.

25.2.5 top

25.2.6 free

Programmet free visar information om hur maskinens minne används. Utan flaggor visar free informaitonen i kilobytes. Om man ger flaggan **-s <tid>** där tid är en siffra för tid i sekunder så ligger programmet och uppdaterar sin information men angivet tidsintervall. Programmet använder filen meminfo i proc-filsystemet. Du kan med andra ord få samma information (inte lika snyggt ordnad) med kommandot **cat /proc/meminfo**.

25.3 uptime

uptime och w

25.4 ulimit

25.5 mount kommandot

Kommandot mount används för att montera filsystem.

Kapitel 26

Starta och stoppa systemet

Detta kan för en före detta Windows användare verka lite krångligt. Varför skall det behövas ett helt kapitel om hur man startar och stoppar systemet. Och visst ligger det något i det. Det skulle inte behöva vara så krångligt. Det är det inte heller. Moderna desktop managers har ofta ett alternativ som låter dig bara välja ett alternativ i en meny så tas maskinen ner snyggt och prydligt. Det kan dock vara nyttigt att veta vad som händer.

Anledningen till att det är så krångligt är att Linux inte konstruerades för att ständigt startas och stoppas. Linux är i första hand ett operativsystem för maskiner som kör som olika servrar. Dessa maskiner brukas ofta av ett stort antal användare. Dessa användare förväntar sig att maskinen alltid skall finnas tillgänglig. Dessa maskinen skall naturligtvis inte startas och stoppas utan stå på dygnet runt. Faktum är att även desktop datorer med fördel kan stå på dygnet runt. Då kan man ju köra vissa job på nätterna som annars skulle slöa ner datorn om de kördes på dagen.

Först beskrivs hur man praktiskt startar och stoppar systemet. Sedan beskrivs på en mer teoretisk nivå vad som händer då systemet går upp, respektive ned.

26.1 Starta systemet

Att starta, eller ta upp, ett Linuxsystem är mycket enkelt. Är bara LILO, eller vilken bootloader man valt att använda, rätt konfigurerad så är det bara att trycka på knappen så startar systemet. Om du använder datorn från en annan dator och inte har fysisk tillgång till Linuxmaskinen måste du se till att maskinen startar utan handpåläggning. Du kan då administrera, och starta om, maskinen från din arbetsplats.

26.2 Stoppa systemet

Man kan inte bara stänga av datorn med strömbrytaren. Att göra så kan alvarligt skada filsystemet. Detta beror på att flera processer kan vara igång samtidigt. Alla dessa processer kan ha filer öppna. Alla dessa processer måste avslutas innan strömmen kan slås av. En server som innehåller kritiska data skall förutom att ha regelbundna backuprutiner även förses med avbrottsfri kraft (ups) så att systemet kan tas ned snyggt även som det skulle bli strömavbrott. Vidare kan ju flera användare samtidigt vara inloggade på maskinen. Man vill ju då ge dem en chans att spara undan sitt arbete och logga ut innan maskinen går ned.

Det finns flera sätt att ta ned en maskin. Det korrekta är att använda kommandot **`shtdown`**. Detta kommando anropas med två argument. Dels en tid i minuter som talar om om hur lång tid systemet skall gå ned. Det andra argumentet är ett meddelande till användarna så att de vet varför systemet skall går ned. Meddelandet till användarna kan uteslutas men en tid, eller ordet *now* måste anges. Exempel:

```
# /sbin/shutdown +10 ¨Systemet går ned på grund av hårdvaruuppdatering¨
```

Tiden kan vara en tidpunkt på formen *timme:minut* eller på formen *+minuter*. I det sera fallet tas maskinen ned om *minuter* minuter. Meddelandet skickas till alla användare så att de skall kunna spara undan sitt arbete i tid.

Man kan även till `shutdown` skicka ett antal flaggor. De vanligaste är `-h` som i *halt* vilket innebär att maskinen stoppas efter det att den gått ned. Använd detta då du vill stänga av maskinen. En annan flagga är `-r` som i *reboot*. Med den flaggan startas maskinen om.

Det finns tre andra kommandon `halt`, `reboot` och `poweroff` som gör precis vad det låter som att de gör. Dessa program kan verka smidiga, men jag rekommenderar att man använder `shutdown`.

Vissa distributioner fungerar så att man kan i konsolen ge en trefinger-salut (Ctrl+Alt+Delete) för att boota om datorn. I Linux kan man dock ge denna tangentkombination vilken betydelse som helst. Den skall alltså inte användas för att boota om systemet.

Då systemet har stoppats så står det ett meddelande på skärmen som kan lyda på lite olika sätt. Vanligen står det *System Halted*. Då man ser det meddelandet kan man lugnt stänga av strömmen till maskinen. Detta gäller även om man använder *Shutdown*-funktionen i display manager som till exempel `kdm` eller `gdm`.

26.3 Uppstarten mer i detalj

Uppstarten av ett Linuxsystem kan verka lite krånglig. En anledning till detta kan vara att den varierar från system till system. I alla system startas först ett litet program som heter `init`. `Init` har alltid *PID* 1, det vill säga det första programmet. Alla andra program är på ett eller annat sätt barn till `init`. Init startar sedan upp systemet enligt vad som står i filerna i katalogen `/etc/rc.d`. Det finns två skolor för hur uppstarten går till, *System 5 (SysV) init* och *BSD init*. SysV är vanligast och andänds

av till exempel RedHat, Debian, Caldera, med flera. Men det är mycket bra att känna till den äldre BSD init. Den används av Slackvare och av alla BSD varianter. Här behandlas kort båda två. Detta att det finns två skolor, och Linux använder båda skapar vissa problem. Det är inte så lätt att känna igen sig på till exempel ett Slackware system då man är inskolad på ett RedHat system. Flera leverantörer, till exempel RedHat, skapar dessutom egna rutiner för hur saker och ting skall skötas. Detta gör att det även kan vara krångligt att gå från RedHat till exempelvis Debian som båda är SysV system. Skall du administrera ett system så bör du noga läsa dokumentationen för just det systemet och utforska filerna i `/etc`. Här finns mycket att lära. Är du en nyfiken användare bör du installera flera olika system och själv studera skillnaderna. Extra roligt blir det om du i jämförelsen lägger in en kommersiell UNIX eller varför inte en fri *BSD variant.

26.4 Körnivåer, (eng. Runlevels)

Både SysV och BSD init använder sig av något som kallas *körnivåer* (eng. *runlevels*). En körnivå kan beskrivas som ett systems läge. Systemet kan bete sig olika i de olika körnivåerna. Det finns 6 stycken olika körnivåer. Tre stycken är fördefinierade och skall inte ändras. Dessa tre är runlevel 0 (Halt), runlevel 1 (singel user mode) och runlevel 6 (reboot). De övriga (2-5) är lägen för flera användare (multiuser) och dessa kan du definiera hur du vill. Vanligt är att en används för X, en för systemet med nätverk och en för systemet utan nätverk. Men du kan definiera dem hur du vill. De olika körnivåerna definieras i filen `/etc/inittab`. Denna fil är konfigureringsfilen till `init` som ju var det första programmet som startades efter initiering av kärnan. I denna fil definieras också vilken körnivå som skall vara standard. Vill du inte köra i den körnivå som är standard kan du vid system starten skicka önskad körnivå som ett argument till kärnan. Mer om detta kan du läsa i kapitlet om boothanteraren `LILO` på sidan 209.

Du kan på de flesta system få reda på den aktuella körnivån med hjälp av kommandot `runlevel` som du på flera system kan hitta i `/usr/sbin` eller `/sbin`.

26.4.1 SysV init

26.4.2 BSD init

Kapitel 27

Boot disketter, underhållsdisketter

Då du installerar ditt Linux så får du i de flesta distributioner en fråga am du vill skapa en boot diskett. Den frågan skall du svara *Ja* på. En boot diskett kan vara mycket bra om till exempel LILO av någon anledning skulle sluta att fungera. Det finns ytterligare en diskett som jag tycker att du skall skapa. Denna följer inte med ditt system utan är en helt egen Linuxdistribution. Det finns flera sådana att välja på. Alla kan du hämta på Internet.

Vad är så bra med en sådan linusdistribution då? Jo om det skulle bli något fel på din rotpartition så kan det vara mycket svårt att få igång systemet. Då kan det vara bra att ha en annan Linux distribution på en diskett som kan boota helt oberoende av ditt "stora" system. På denna diskett skall du naturligtvis se till att ha de verktyg som du kan behöva för att reparera ditt vanliga system.

Det finns ett par sådana system som jag provat och tycker är bra.

27.1 DLX Linux

DLX Linux är ett litet system som jag använt många gånger. Det är väldigt smidigt att ha till hands om man till exempel har glömt root-lösenordet eller har trasslat till det för sig i `/etc/fstab`.

27.2 Trinux

Kapitel 28

Hantera användare

Användare är ett grundläggande begrepp i ett Linuxsystem. Alla filer har olika rättigheter och vissa användare får göra vissa saker. Alltså måste systemet veta vem det är som för tillfället använder maskinen. Därför måste du alltid identifiera dig med ett användarnamn och ett lösenord för att kunna använda en Linuxmaskin.

28.1 Ett bra lösenord

Man bör uppmuntra sina användare till att använda bra lösenord. Användare med högre rättigheter måste ha bra lösenord. Ett bra lösenord består av 6–8 tecken och är en blandning av små, stora bokstäver coh siffror. Det finns flera bra sätt att hitta på, och komma ihåg, bra lösenord. Här visar jag mina två favoritsätt:

Skapa en mening: Ett bra knep. Till exempel skulle meningen “Jag är rik år 2000”. Kunna förkortas ner till JaRa2k vilket är ett hyfsat bra lösenord (Man kan även ta titeln till, eller refrängen i, en favoritlåt).

Förvräng ett ord: Ett nästan lika bra knep. Ta ett ord, till exempel räkning och sväng till det till exempelvis, R4kn1n9. Detta är något knivigare tycker jag personligen.

Jag skriver inte denna bok för att lära folk hur man väljer lösenord. Huvudsaken är att du inte väljer ditt, din frus eller mans, dina barns eller din hunds namn. Dessa är nämligen väldigt lätta att gissa, och knappast ovanliga . . .

28.2 Ändra sitt lösenord

Man bör ändra sitt lösenord då och då. Vissa system kräver att man gör det. För att ändra sitt lösenord använder man programmet `passwd`. Om du vill ändra ditt eget lösenord så startar du programmet utan argument. Du måste då först ange dett nuvarande lösenord följt av det önskade två gånger. Det syns inget på skärmen då du skriver lösenorden. Om du är root så kan du ändra allas lösenord. Då startar man `passwd`

med användarnamnet som argument. Root behöver inte ange nuvarande lösenord utan anger bara det nya lösenordet två gånger.

Vill du tvinga dina användare att regelbundet ändra sina lösenord så finns det versioner av mjukvaran som fixaar detta. Detta är dock inget jag rekommenderar eftersom dina användare då tvingas att byta lösenord då de inte planerat att göra det. Detta leder till att det nya lösenordet ofta blir dåligt och lätt för en inbrytare att knäcka.

28.3 Om man glömt sitt lösenord

En användare som glömt sitt lösenord måste vända sig till systemadministratören för att han, eller hon, skall byta lösenord. Administratören använder då `passwd` för att skapa ett nytt lösenord till användaren.

Det pinsammaste som kan hända (nästan) är att man glömmer bort lösenordet till root. I Linux är detta dock enkelt att fixa. Man kan göra på lite olika vis. Målet är att i filen `/etc/passwd` ta bort lösenordet för root. Sedan startar man om maskinen och loggar in som root. Nu behöver man inte ange något lösenord. Sedan använder man `passwd` för att sätta ett nytt lösenord.

Hur kommer man då åt filen `/etc/passwd`? Antingen kan man boota från en diskett med ett litet Linuxsystem på. Mer om sådana disketter behandlas i kapitel 27. I detta lilla system monterar du rotpartitionen från det vanliga systemet. Nu kan du ändra `/etc/passwd`. Alternativt så kan du boota Linux i så kallat *single user mode*. Det gör du genom att när LILO startar ange det du vanligen anger för att starta Linux följt av ordet **single**. Nu startar systemet direkt som användare root. Inga andra användare kan logga in på systemet.

Men, kanske du nu tänker, detta låter inte så säkert. Vem som helst kan ju bli root på ett system. Det är förvisso sant. Men för att kunna bli root måste man ha fysisk kontakt med maskinen. I det läget finns det inget system som är säkert. Det går att göra ovanstående procedur betydligt svårare. Till exempel genom att i datorns bios (på pc) ta bort möjligheten att starta från en diskett och att konfigurera så att man inte kan skicka parametrar till lilo. Hur detta går till beskrivs i kapitel 24. I de flesta bios kan du sätta ett lösenord för att boota datorn. Sätt inte detta lösenord om datorn kör någon typ av server och står i ett rum i källaren. Då måste du ju söka upp datorn och skriva lösenordet om du bootat om datorn eller om det, till exempel, varit strömavbrott.

> **Varning!**
> *Kom ihåg att en dator till vilken en eventuell fiende har fysisk tillgång aldrig blir säker. Ovanstående kan göra det krångligare men inte omöjligt.*

28.4 Skuggade lösenord (eng. Shadow Passwords)

För att öka säkerheten i ett system kan man använda sig av något som kallas *skuggade lösenord*. Traditionellt så sparas de krypterade lösenorden i en fil `/etc/passwd`. Denna fil har accessrättigheterna `-rw-r--r--`, det vill säga den är läsbar för alla. Detta gör att det är relativt enkelt för en inkräktare att få tag på de krypterade lösenor-

den. Dessa är genererade med en algoritm som inte går att reversera. Inkräktaren kan alltså inte få reda på alla lösenord. Vad inkräktaren däremot kan göra är att testa en massa ord och av dem generera krypterade lösenord. Om han hittar ett, av honom, genererat krypterat lösenord som matchar ett av de som finns i lösenordsfilen så har han tillgång till ett lösenord till systemet. Naturligtvis provar han inte alla ord manuellt utan det finns program som gör det. Det kan vara bra att på ditt eget system testa ett av dessa program för att se om du har några osäkra lösenord. Har du det bör du uppmuntra dessa användare att byta till bättre lösenord.

Ett sätt att skydda dina krypterade lösenord är att använda skuggade lösenord. Då sparas de krypterade lösenorden i en annan fil som kan ges striktare accessrättigheter, vanligen `-r--------`. I lösenordsfilen markerar man detta med ett `x` istället för det krypterade lösenordet.

Hur får man detta att fungera? –Jo. XXXXXXXXXXX

28.5 Användare som brukar finnas i ett system

I ett Linuxsystem brukar det direkt efter installation finnas några användare. Dessa varierar från distribution till distribution. Här tittar vi på några exempel.

28.6 Lägga till användare

28.6.1 Manuellt

Då det finns flera bra verktyg för att lägga till användare kan detta anses som överkurs. Det är dock nyttigt att kunna då det ibland är nödvändigt att styra till exempel vilket **uid** en användare får.

En användare är för systemet bara en rad i en fil som heter `/etc/passwd`. Vidare måste personen ha en hemmakatalog om han skall kunna utföra något nyttigt arbete. Till den katalogen brukar en del initieringsfiler kopieras då kontot skapas. Varje användare tillhör också en grupp. Denna grupp måste finnas i filen `/etc/group`. Slutligen måste användaren ges ett lösenord för att kunna logga in i systemet. Följande måste alltså göras då ett användarkonto skapas.

- Skapa en rad i `/etc/passwd`.
- Eventuellt skapa, eller ändra, en rad i `/etc/group`.
- Skapa användarens hemmakatalog.
- Kopiera standardfiler till användarens hemmakatalog.
- Kontrollera att alla rättigheter och ägare på filer stämmer.
- Ge användaren ett lösenord.

Nu skall vi gå igenom hur man gör detta.

`/etc/passwd`

`/etc/group`

Skapa hemmakatalog

Kontrollera/Sätta rättigheter

Ge användaren ett lösenord

Nu när vi vet att användarkontot fungerar kan vi ge användaren ett lösenord. Det gör vi som vanligt med kommandot **`passwd användarnamn`**. Detta kan man inte göra manuellt eftersom lösenordet måste krypteras med en envägsalgoritm.

28.6.2 Använda verktyg

Att göra som det beskrevs i stycket ovan för att skapa en användare är onödigt. Det är andå inte helt onödigt vetande. Mem det enkla vanligta visat att skapa en användare beskrivs i detta avsnitt.

`adduser` och `passwd`

För att lägga till användare använder man lämpligen programmet `adduser`. I vanliga fall är det bara att köra programmet med den nye användarens användarnamn som argument. Efter detta behöver den nye användaren ett lösenord. Detta skapas med programmet `passwd`. Se nedanstående exempel.

```
namatj:~#
namatj:~# /usr/sbin/adduser kalle   <-- Skapar användaren "kalle"
namatj:~# /usr/bin/passwd kalle     <-- Ger "kalle" ett lösenord
New UNIX password:                  <-- Här skriver jag lösenordet
BAD PASSWORD: it does not contain enough DIFFERENT characters
Retype new UNIX password:           <-- Här skriver jag det igen
passwd: all authentication tokens updated successfully
namatj:~#
```

I exemplet ovan varnas jag för att jag använt ett för enkelt lösenord. Detta skall inte behöva ske om man använt tipsen ovan. Detta är bara varningar om du är systemadministratör och lösenordet accepteras i alla fall. Observera att jag också angett fullständiga filnamn till programmen. Detta är som regel inte nödvändigt eftersom dessa bör finnas i användaren **root**s $PATH.

28.7 Ta bort användare

Att ta bort användare är lika enkelt som att lägga till dem. Även här finns det verktyg som kan hjälpa dig. Det finns dock flera saker att tänka på.

Det första du bör göra är att skriva något ogiltigt i lösenordsfältet för användaren, eller ta bort användarens rad i `/etc/password`. På så sätt kan ingen använda kontot. Då en tid gått eller då du är säker på att kontot inte används tar du en kopia på (eller byter namn på) hemmakatalogen och raderar den. En anledning till att du skall ha en backup på den är att någon annan kanske använder någon fil i den hemmakatalog som skall tas bort. På så sätt så märker du även detta.

> **Varning!**
> *Det är inte kul att berätta för en användare att du tagit bort alla hennes filer oavsett hur det gått till . . .*

28.7.1 Manuellt

Att ta bort användare manuellt är enkelt. Det som skall göras är samma sak som då användaren lades till fast tvärt om. Användaren skall tas bort ur `/etc/passwd` och eventuellt skall användarens egna grupp tas bort ur `/etc/group` använder du skuggade lösenord skall du också städa filen med de krypterade lösenorden, ofta `/etc/shadow`.

28.7.2 Använda verktyg

28.8 Meddela dina användare

Ibland vill du skicka meddelande till dina användare. Du kanske vill upplysa om att systemet kommer att vara nere torsdag kväll eller något annat som berör. Detta kan du gör i filen `/etc/motd` (Message Of The Day). Det du skriver i denna fil visas då varje gång en användare loggar in.

Innan en användare logar in visas också ett meddelande. I alla fall om användaren loggar in på en konslol eller via telnet. Denna fil brukar kallas `/etc/issue`. Som standard brukar denna fil identifiera systemet så att en användare vet vad det är för typ av system. Filen brukar bärför skrivas om varje gång systemet bootar. På så sätt kan den till exempel alltid visa information om vilken version av kärnan som används. Om du uppdaterar kärnan så uppateras denna fil automatiskt. Denna uppdatering måste du ta bort eller ändra så att den passar dina ändamål om du vill att den skall visa ett, för din maskin, unikt meddelande. Var, och om, denna uppdatering sker varierar från distribution till distribution och tas alltså inte upp i denna bok. Leta i uppstartfilerna i katalogen `/etc/rc.d/`. Har du en bra distribution som gör en sådan uppdatering så bör den vara dokumenterad så att du kan hitta den. Ett hett tips bör vara `/etc/rc.d/rc.local`. Mer om uppstartfilerna hittar du i kapitel 26.

28.9 Avancerade accessrättigheter

Inledningsvis behandlades filers accessrättighter. Här behandlas mer avancerade sådana.

28.9.1 Filtyper

- d l b c p

28.9.2 `chgrp` och `chown`

28.9.3 SUID och SGID

Kapitel 29

Hårddiskar – Kataloger

Detta kapitel behandlar hur ett Linux filsystem kan, och bör, spridas över flera hårddiskar eller partitioner.

I installationskapitlet talade vi om att Linux borde installeras på flera olika partitioner eller diskar. Här behandlar vi hur detta går till på ett teoretiskt sätt. Hur det i praktiken går till behandlas i delen “Systemadministration”.

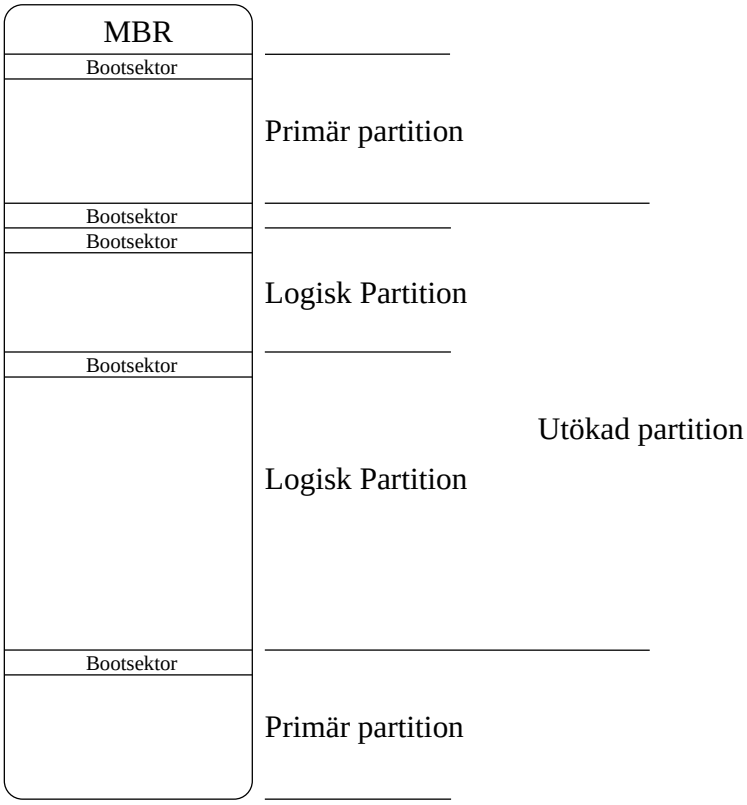
29.1 Begrepp

29.1.1 Hårddisk och partition

Vad en hårddisk är behöver jag väl knappast beskriva. En modern hårddisk är ju som bekant ofta flera gigabyte stor. Det är sällan praktiskt att ha så stora filsystem. En hårddisk kan indelas i flera olika partitioner. Det har en mängd fördelar. Varje partition fungerar som en hårddisk för sig. Partitionerna är helt oberoende av varandra och kan ha olika filsystem och olika operativsystem precis som om de vore egna hårddiskar. Man kan skapa partioner med olika program beroende på vilket operativsystem man skall ha. Inte sällan heter dessa program `fdisk`.

Det finns tre typer av partitioner. Primära, utökade och logiska. Detta har att göra med att en gammal begränssning gör att det bara går att ha fyra partitioner på en disk. Detta har man kommit runt på så sätt att man låter en av de fyra primära partitionerna vara en utökad partition. I denna utökade partition kan man sedan skapa logiska partitioner. Det skiljer inget i prestanda mellan dessa två typer av partitioner.

Information om alla partioner på disken finns i disken *Master Boot Reckord* (MBR). MBR är det första blocket på disken. MBR innehåller ett litet program som, vid systemstart, läser in partitionstabellen och kontrollerar vilken partition som skall boota systemet. Detta markeras genom att man sätter en “boot-flagga” till denna partition. Programmet läser sedan första sektorn på den partitionen. Denna sektor kallas *Bootsektorn* och är alltid den första i partitionen. Ett exempel på en partitionerad hårddisk



Figur 29.1: Exempel på partitionerad hårddisk

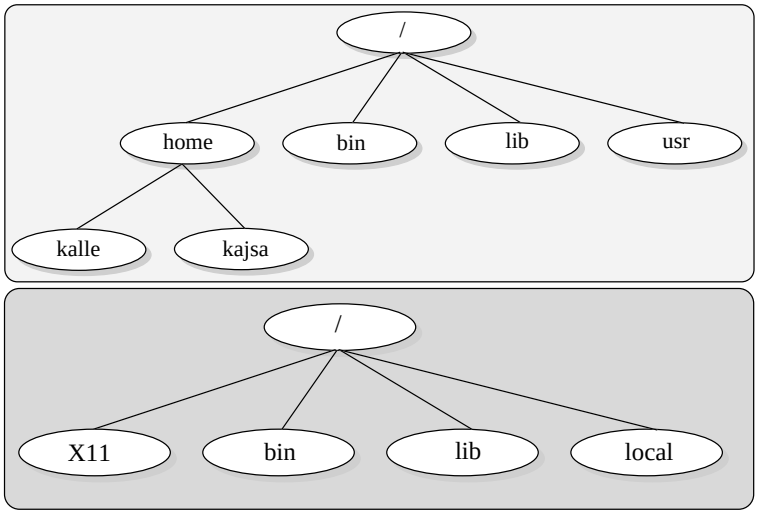
visas i figur 29.1.

I detta dokument kommer jag att blanda orden disk och partition lite hur som helst. Men vilket som går bra i alla fall.

29.1.2 Katalog och filsystem

Ett Linux filsystem består av bara en enda rotkatalog. Till skillnad från MS-DOS som har en rotkatalog för varje enhet. UNIX har ju inga enhetsbokstäver, men kan naturligvis använda flera enheter i alla fall. I stället för att ge varje partition en enhetsbokstav så *monteras* (eng mount) den på en katalog i filsystemet. Katalogen där partitionen monteras kallas *monteringspunkt* (eng. mountpoint). Varje partition har naturligtvis ett eget filsystem. Studera de två filsystemen i figur 29.2. De har båda en egen katalogstruktur.

I figur 29.3 har det undre filsystemet monterats på katalogen `/usr` i det övre filsystemet. Observera hur innehållet i `/usr` har bytts ut mot innehållet i det undre filsystemet. Observera att monteringen är helt osynlig för användarna. Användarna behöver

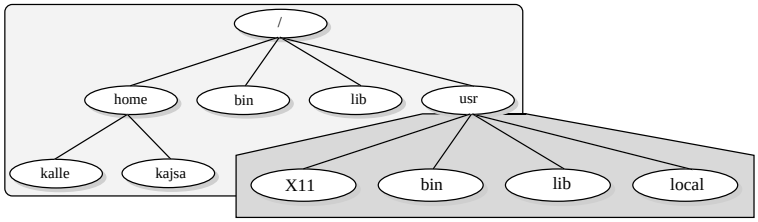


Figur 29.2: Två stycken filsystem

Bilden visar två stycken filsystem. De befinner sin på två olika diskar eller partitioner.

aldrig hålle reda på vilken disk en fil ligger (det är ju i alla fall onödigt vetande för dem). Det räcker att veta filnamnet för att återfinna filen.

Precis på samma sätt går man tillväga om man vill titta på något flyttbart media. Man monterar det på en katalog, sparar eller hämtar data och sedan monterar man av enheten med kommandot **`umount`**.



Figur 29.3: Två stycken filsystem

Här har det under filsystemet monterats på katalogen `/usr` i det övre filsystemet.

29.2 tune2fs

29.3 mount och umount

29.4 Formatera diskar och disketter

29.5 mtools

29.5.1 Enheter och kataloger

I installationsavsnittet talades kort om att olika enheter monteras på olika kataloger i Linux filsystem. Vi gör en kort jämförelse med MS-DOS. I MS-DOS har alla filsystem en enhetsbokstav (A:, C:, D: etc.) i Linux har man inte sådana enhetsbokstäver utan alla filsystem "monteras" in i ett enda stort filsystem. Detta är ett mycket bra system eftersom användaren inte behöver fundera på vart en fil han/hon söker. Om en diskett till exempel heter A: i MS-DOS så kan den heta `/floppy` i ett Linuxsystem. Alltså bara en vanlig underkatalog i Linux. Just disketten var kanske ett dumt exempel eftersom disketthantering i Linux är onödigt krånglig. Men om användaren söker en fil som i MS-DOS ligger på en av 6 st nätverksenheter (alla med olika enhetsbeteckningar) så förstår man att det blir enklare i ett Linuxsystem där alla enheter bildar ett enda stort filsystem.

29.5.2 Montera (olika) filsystem

Linux kan hantera en hel del olika typer av filsystem. Det är inte som t.ex. MS-DOS bundet till att använda sitt eget. Ett Linuxsystem kan utan problem till exempel dela en hårddisk med till exempel en Windows-installation. För att se vilka filsystem din installation för tillfället stöder kan du titta i `proc`-filsystemet med följande kommando:

```
cat /proc/filesystems
```

29.5.3 /etc/fstab

För att automatisera monteringen av de olika filsystemen, och tillåta användare att göra detta skall vi titta på en fil som heter `/etc/fstab`. Denna inehåller en lista på hur olika enheter skall tillåtas att monteras i filstrukturen. Även information om hur, och av vem, de får monteras. En av mina `/etc/fstab` filer finns i bilaga C.

Syntaxen i `fstab` är enkel. Rader som börjar med # är kommentarer. Annars avser varje rad ett filsystem. Varje rad består av 6 fält. Varje fält separeras med <tab> eller ett eller flera mellanslag. Varje fält skall innehålla följande information:

Första fältet innehåller den enhetsfil på vilken filsystemet finns.

Andra fältet skall tala om vart detta filsystem skall monteras.

Tredje fältet skall ange vilken typ av fisystem som filsystemet har.
Fjärde fältet Anger olika inställningar. Vanliga alternativ (som åtskiljs med komma) är:

- ro eller rw, ReadOnly eller ReadWrite
- auto eller noauto, Om filsystemet skall monteras automatiskt vid uppstart eller ej.
- user, Anger att montering får ske av användare
- userquota, Om quota skall aktiveras på filsystemet
- XXXXXXXXXXXXXX — Fyll på med mera

Femte fältet Anger om programmet “dump” skall “dumpa” enheten eller ej. Dump är ett program som används för säkerhetskopiering. Är värdet “0” kommer enheten inte att dumpas, detsamma gäller om femte fältet utelämnas. Vill man att enheten skall dumpas kan man sätta det till “1”

Sjätte fältet Används av `fsck` för att avgöra i vilken ordning de skall kontrolleras vid uppstart av maskinen. Rootfilsystemet skall ha värdet “2”. Övriga filsystem skall ha värdet “1”. Utelämnat fält eller värde “0” gör att filsystemet inte testas alls.

Kapitel 30

Filsystemet

I detta avsnitt behandlas frågor om vad som skall finnas var i ett Linux filsystem. Det finns inga benhårda regler för vad som gäller utan man kan ganska fritt själv bestämma vart saker och ting skall ligga. Detta kapitel kan dock ses som en rekommendation om var saker och ting skall placeras. Genom att följa dessa regler får du ett system som är lätt att underhålla och som gör att du känner igen dig på främmande system. Detta avsnitt baseras på *Filesystem Hierarchy Standard – v2.0* (FHS-2.0). Denna standard gäller nu mera inte bara Linux utan alla Unix-liknande operativsystem. De flesta stora Linuxdistributioner är FHS-kopatibla. Detsamma bör gälla för de flesta programpaket.

30.1 Varför finns det en standard?

Varför finns det en standard? Skulle man inte kunna få lägga sina filer vart man vill? – Jo visst, det får du. Om du är, eller blir, en van administratör så kommer du med största sannolikhet att skaffa dig en egen stil. Du kommer att placera dina filer efter eget tycke och smak. Detta är inget problem, utan det är så det fungerar. Denna standard hindrar dig inte heller från att placera dina filer vart du vill. Standarden är till för att du skall veta var ett programpaket som du installerar lägger sina filer och var alla olika filer ligger då du just installerat ett nytt system. Sedan är det naturligtvis bra att hålla sig till konventionerna, det kommer att hjäpa dig att uppgradera, säkerhetskopiera och underhålla ditt system. Nedan beskriner jag de viktigaste katlogerna var för sig. Jag beskriver även lite kort vad som karakteriserar ett Linux filsystem. Vill du ha ytterligare information så hänvisar jag till *Filesystem Hierarchy Standard* (Se källförteckningen)

30.2 Filsystemets egenskaper

Linux filsystem är ett hierarkiskt fisystem. Hela filsystemet finns samlat under en katalog (roten /). En katalog i filsystemet kan vara en monteringspunkt för en annan partition. Filsystemet på den monterade hårddiskpartitionen blir då en del av filsyste-

	delbara	odelbara
statiska	/usr /opt	/etc /boot
dynamiska	/var/mail /var/spool/news	/var/run /var/lock

Tabell 30.1: Exempel på de fyra filtyperna.

met. Detta har till skillnad från MS-DOS den fördelen att en anväande inte behöver veta vilken disk en fil ligger. Det går även att utöka filsystemets storlek med fler diskar utan att användaren märker det.

30.2.1 Fyra typer av filer

Ur filsystemets synpunkt finns det fyra typer av filer. Dessa olika typer bör ej finnas på samma diskpartition. Det är dock fullt möjligt att ha dem på en gemensam partition om man vill.

Alla filer i ett filsystem kan delas in i två delar, *delbara* och *odelbara*. En delbar fil är en fil som kan delas mellan flera olika maskiner, till exempel via ett nfs-filsystem. Ett bra exempel på delbara filer är användares hemma-kataloger. En användare kan ha samma hemmakatalog på flera olika maskiner. Ett exempel på odelbara filer är filerna i katalogen `/etc`. Denna katalog innehåller filer som bara gäller den egna maskinen. ett filsystem med *delbara* filer kan således delas mellan flera maskiner i ett nätverk. Detta kan spara diskutrymme och underlätta underhåll eftersom bara en version av filerna är nödvändig.

Vidare kan filerna delas in i två delar till, *statiska* och *dynamiska*. *Statiska* filer är filer som inte ändras annat än vid systemunderhåll. det vill säga vid administratörens inblandning. Dynamiska filer är sådana som ändras vid vanligt användande. Varför detta är en så stor skillnad för filsystemet är att ett filsystem med bara *statiska* filer kan monteras *read-only*. Ett exempel kan vara ett sådant filsystem till exempel skulle kunna finnas på en CD-ROM (inte så vanligt men ett bra exempel på hur ett *read-only* filsystem skulle kunna användas.).

Dessa två grupper kansammanfattas i tabell 30.1. I varje ruta ger jag exempel på kataloger som passar.

30.3 rot-filsystemet `/`

Roten är den allra högsta nivån i ett Linux filsystem. Det filsystem som den ligger på måste innehålla alla verktyg som behövs för montera de andra filsystemen. Vidare måste det finnas verktyg för att återställa filsystem från backup. Allt för att göra det enkelt vid ett haveri. De kataloger som delar filsystem med roten och alltså alltid finns vid uppstart brukar vara `/bin`, `/sbin` och `/lib` `/etc`.

Rot-filsystemet brukar ligga på en liten disk eller partition. Den är inte delbar. Filsystem som inte brukar ligga på rot-filsystemet är `/var`, `/usr` och `/home`.

30.4 /bin

I denna katalog fins de binärfiler som behövs under uppstarten av systemet. Det är program som kan nytjas av vanliga användare. Fast då efter systemstarten. Här skall de absolut nödvävdigaste kommandona finnas. Denna skall finna på rootfilsystemet. Vissa filer som ligger i `/bin` ligger där eftersom de av tradidtion har hamnat där.

Exempel på filer i `/bin`
mount, umount, chown, chgrp, rm, rmir, mv, ln, login, sync.

30.5 /sbin

Lika som ovan fast program som i regel inte används av vanliga användare. Det finns dock inget som hindrar dig från att låta vanliga användare använda dem om det skulla behövas. De flesta program brukar få exekveras av användare även om de kanske för begränsad funktionalitet då. Här skall det som behövs för systemunderhåll och liknande finnas. Denna katalog skall liksom `/bin` alltid finnas vid systemstart och skall således ligga på rootpartitionen. I /sbin eller /bin måste kommandot `mount` finnas för att kunna montera övriga filsystem.

Exempel på filer i `/sbin`
getty, swapon, swapoff, mkswap, halt, shutdown, fdisk, mkfs*[1], fsck*, ifconfig, route.

30.6 /boot

Filer som används av bootprogramvaran. Till exempel *LILO*. Denna måste på vissa PC-maskiner helt finnas före den 1024:e cylindern.

Den kompilerade Linuxkärnan skall finnas antingen i `/boot` eller i roten (/). Den kallas ofta vmlinuz.

30.7 /dev

Enhetsfiler (eng. Devicefiles). Detta är filer som representerar hårdvara. En enhetsfil är egentligen en drivrutin. Man kan säga att en enhetsfil består av två delar; En som uppåt liknar en vanlig fil till vilken man kan skriva eller från vilken man kan läsa. Men nedåt talar filen med den hårdvara den representerar. Enhetsfiler behandlas mer i kap 40 på sid 267.

[1] Med '*' menar jag att det finns flera kommandon som börjar på fsck, t.ex. fsck, fsck.ext2

MAKEDEV är ett kommando som skapar enhetsfiler. Finns inte en enhetsfil som borde finnas skall du prova att köra `/dev/MAKEDEV`.

Exempel på filer i `/dev`
MAKEDEV, hda, hdb, ttyS0, null, audio, fd0, sda.

30.8 `/etc`

I `/etc` finns konfigurationsfiler som hör till systemet. Denna katalog kommer du som systemadministratör att rota en hel del i. Denna katalog är inte delbar, utan hör till en viss maskin. Nu skall inga binärfiler finns i `/etc`. Tidigare har vissa administratörkommandon funnits i `/etc`, dessa skall nu finnas i `/sbin`.

Konfigurationsfilerna till X skall ligga i underkatalogen `/etc/X11`.

Program som ligger i katalogen `/opt`[2]. skall ha sina konfigurationsfiler i `/etc/opt`.

Exempel på filer i `/etc`
fstab, passwd, inittab, printcap,

30.9 `/home`

Användarnas hemmakataloger. Denna liger ofta på en egen partition. Finns det väldigt många användare kan denna läggas på flera partitioner. Då delas den upp i underkataloger som till exempel `/home/students` och `/home/teachers` eller något annat.

30.10 `/lib`

Här finns de bibliotek som behövs an programmen i `/bin` och `/sbin`. Denna måste också ligga på rootpartitionen.

Exempel på filer i `/lib`

30.11 `/opt`

Katalogen `/opt` är till för program som är tillägg till systemet. Det vill säga användarprogram och dylikt. Varja program under `/opt` skall ligga i sin egen underkatalog. Till exempel skulle programmet `program` ligga i `/opt/program`. `/opt` skall vara delbar. Därför skall filer till programmen i `/opt` som inte är delbara ligga

[2] Läs mer om katalogen `/opt` i detta kapitel.

i /etc/opt och /var/opt. Dessa kataloger skall delas uopp på samma sätt som /opt, exemplet ovan alltså /etc/opt/program och /var/opt/program.

30.12 /lost+found

Ibland kan det hända att systemet avslutas så abrupt att skador sker på filsystemet. Det kan vara till exempel ett strömavbrott, eller en ilsken operatör som drar ur sladden.

30.13 /mnt

En moteringspunkt för tillfälliga monteringar. Kan delas upp i underkataloger för olika enheter. Till exempel /nmt/cdrom /mnt/floppy. Personligen tycker jag att dessa två exempel är dåliga. Jag lägger hellre till katalogerna /floppy och /cdrom direkt i roten och har /mnt bara för mycket tillfälligt och sporadiskt monterade filsystem.

30.14 /proc

/proc är en rolig katalog. På denna monteras ett filsystem som inte är ett fysiskt filsystem. Detta filsystem skapas av kärnan i minnet. Detta filsystem representerar datastrukturer i kärnan. Ursprungligen visades information om processer i detta filsystem. Därav namnet. Nästan hela filsystemet är inte skrivbart utan kan bara läsas. /proc är ett stort filsystem. Men det tar inget diskutrymme, detta beror på att det representerar andra delar av hårdvaran. Till exempel finns en fil /proc/kcore som är lika stor som ditt minne. Den kan du om du vill kopiera till en katalog och undersöka. Då du kopierar den ser du att den är lika stor som ditt minne men i /proc tar den inget diskutryme.

30.15 /root

Användaren **root**s hemmakatalog.

30.16 /tmp

Temporära filer. Denna katalog är skrivbar för alla användare. Innehållet i katalogen bör raderas vid start av systemet.

30.17 /usr

/usr är som regel en monteringspunkt för ett stort filsystem. Enligt FSSTND skall detta filsystem kunna monteras ReadOnly. Det är i detta filsystem som alla program finns installerade. De flesta filer i detta system följer med distributionen och behöver aldrig ändras. Det kan vara bra att ha i åtanke vid backup. Undantaget är /usr/local som måste säkerhetskopieras.

30.17.1 /usr/local

I detta filsystem (eller katalog) kan systemadministratören (du?) leka lite som man vill. Här lägger man med fördel program som är lokala för systemet. Det vill säga program som inte följer med din Linuxdistribution. Det kan vara program du tankat hem och kompilerat själv eller lokalt utvecklade program. Katalogstrukturen i /usr/local bör likna den i rootkatalogen. Det är en fördel om /usr/local/bin ligger långt fram i dina användares $PATH miljövariabler. På så sätt kan du enkelt styra om kommandon med till exempel symboliska länkar utan att böka i några andra kataloger än /usr/local/bin. Denna katalog kan man med fördel lägga på ett eget filsystem. Detta för att enkelt kunna ta backup på det. Men framför allt är det lämpligt vid en systemuppgradering[3].

30.17.2 /usr/X11R6

Program som behöver, och tillhör, X. Du känner nog igen katalogerna under denna katalog vid det här laget.

30.18 /opt

30.19 /var

Denna katalog har man också med fördel på en egen partition. Denna katalog innehåller filer som varierar i storlek och innehåll. Dessa kan inte finnas under /usr eftersom /usr skulle kunna monteras readonly. Här hittar man mail, news, skrivarköer, loggar mm. Denna måste naturligtvis monteras read-write. Det vill säga både läs och skrivbar.

30.20 Tips

När du flyttar och lägger till filer, var noga med att följa något system. Det kan vara ditt eget, eller ett annat. Bara du sedan vet var du har filerna. Skriv gärna en log för systemet dar du beskriver vad du ändrat. Har du inte placerat ditt fisystem på ett smart

[3] Se Uppgradering på sid 201

sätt över diskar och partitioner kan en sådan log vara ovärderlig vid en uppdatering av systemet.

Jag rekommenderar att du, enligt standarden som beskrivs i detta kapitel, så långt som det är möjligt gör dina ändringar i `/usr/local`. Helst bör denna ligga på en egen partition för att underlätta vid systemuppgradering. Se även till att `/usr/local/bin` ligger långt fram i dina användares `$PATH`.

30.21 sudo

Ibland vill man som administratör ge vissa medarbetare tillåtelse att köra vissa kommandon som root utan att vilja ge dem användaren roots lösenord. Detta är möjligt med hjälp av ett program som heter `sudo`. Sudo finns inte installerad i alla distributioner. Men går naturligtvis att hämta på de vanliga platserna.

Om du som användare skall använda `sudo` så ger du det kommando du vill som vanligt fast anger **`sudo`** före. Om du till exempel vill ange kommandon **`shutdown -r now`** så anger du kommandot **`sudo shutdown -r now`**. Nu frågas du efter ett lösenord. Då anger du *ditt* lösenord. Har du kört `sudo` de senste 5[4] minuterna så behöver du inte upprepa lösenordet. Vet du att de fem minuterna håller på att ta slut kan du köra **`sudo -v`** så får du fem nya minuter utan att köra något kommando.

Om du kört `sudo` och sedan går ifrån din terminal skall du köra kommandot **`sudo -k`**. Nu måste du på nytt ange ditt lösenord för att kunna köra sudo kommandon.

Man kan även med hjälp av `sudo` köra kommandon som andra användare än root även om det oftast inte är lika användbart. Man använder då kommandot **`sudo -u användare kommando`**.

Okej, att använda `sudo` är inte så svårt. Men hur anger man vem som får göra vad?

`Sudo` har en konfigureringsfil. Den heter `sudoers` och ligger vanligtvis i `/etc` men kan ligga på andra ställen. I denna kan du konfigurera precis vad var och en får göra. Du kan även ange grupper av användare och även grupper av kommandon.

Man bygger upp filen så att man skapar grupper av användare. Till exempel kan gruppen MAY_SHUTDOWN innehålla användare som får boota om en maskin och kanske PRT_ADMIN kan innehålla de som får sköta om skrivare. Se nedanstående exempel

```
User_Alias MAY_SHUTDOWN=joakim,kalle,alexander
User_Alias PRT_ADMIN=kajsa,mimmi,klarabella
User_Alias ADMIN=jocke,ludvig
```

Sedan är det smidigt att gruppera upp kommandon på liknande sätt. Man gör till exempel en grupp av kommandon för att ta ner ett system i en grupp. Vi kan kalla den SHUTDOWN. Man gör likadant med de kommandon man vill.

[4] 5 minuter är ett standardvärde som kan ändras men det brukar vara 5 minuter

```
Cmnd_Alias PRINTERS=/usr/bin/lprm,/usr/bin/lpq
Cmnd_Alias SHELLS=/bin/sh,/bin/csh,/bin/tcsh,/bin/ksh,/bin/bash,/bin/vi
Cmnd_Alias SU=/bin/su
Cmnd_Alias MISC=/bin/rm,/bin/cat
Cmnd_Alias SHUTDOWN=/bin/halt,/bin/reboot,/bin/shutdown
```

Man kan även styra vilka användare som för exekvera kommandon som vilka användare. Oftast är man bara intresserad av att ge vissa användare möjlighet att köra vissa kommandon som root. För att göra det kan man skapa gupper av anvädare på liknande sätt som ovan.

```
Runas_Alias OP=root,operator
```

En annan bra möjlighet man har med `sudo` är att man kan låta vissa användare köra kommandon som root på vissa maskiner, eller vissa grupper av maskiner. Man kan ju till exempel vilja ge kalle möjligheten att starta om maskinerna i korridor 14.

```
Host_Alias KORRIDOR_14=balder,tor,mumin,hugin
Host_Alias BYGGNAD_3=192,168,100,0
Host_Alias REMOTE_=alvedon,ipren,magnecyl
```

Nu skall man bara koppla ihop användarna med de kommandon de får köra, som vilka de får köra och på vilka maskiner de får göra det. Det är inte så avancerat som det kanske låter. Det är dags att nämna att filen `sudoers` skall editeras med ett speciellt kommando, **`visudo`** som startar `vi` och öpnnar konfigureringsfilen för `sudo`.

```
kalle KORRIDOR_14=(OP) SHUTDOWN
MAY_SHUTDOWN ALL=(OP) SHUTDOWN
ADMIN ALL=(ALL) ALL,!SHELL,!SU
```

I exemplet ovan får kalle starta om datorerna i korridor 14, anvädarna i gruppen MAY_SHUTDOWN får köra kommandona i kommandogruppen SHUTDOWN. användarna i gruppen ADMIN får köra alla kommandon utom de i kommandogruppen SHELL och SU. Observera att de grupper jag refererar till är det grupper som definieras i filen. Inga andra grupper. Det som står inom parentes är vilka användare som man får köra som.

Hur är det med säkerheten? Jo den frågan är i allra högsta grad befogad. För det första så måste man tänka efter vad användarna får köra. Vitsen med `sudo` är ju att inte behöve ge en administratör fullständiga root rättigheter. Mem om man, på ett eller annat sätt, kan köra ett skal som root så har man samma möjligheter som root. Man kan ju ge kommandon till skalet som körs som root. Det farliga är de “dolda” skalen. Om du till exempel ger en användare möjligheten att köra `vi` som root så kan den användaren köra kommandot :shell i vi och på så sätt få root access. I exemplet ovan kan flera av användarna få root access genom till exempel programmet `emacs`.

Vidare kan en användare som for köra `chmod` som root mycket enkelt få root access.

> **Varning!**
> *Var alltid restrikiv då du delar ut kommandon via* `sudo`*. Undvik ALL.*

Mer läsning om detta finns i manualsidorna sudo(8), sudoers(5) visudo(8).

Kapitel 31

Konfi gurera skrivare

Det första problemet som många användare upplever då de installerat Linux är att skrivaren inte fungerar. Även detta blir dock enklare och enklare för varje distribution som kommer ut.

Ett av problemen är att inte alla skrivare stöds av Linux. Många skrivare måste ställas om för att emulera en annan skrivare och andra förlitar sig på mjukvara som bara finns till Windows. De senare kallas Windowsskrivare och är av ondo. Undvik dem om du köper en skrivare. Ett allmänt tips är att innan du köper en skrivare kontrollera med din handlare att skrivaren går att använda i Linux.

> **Tips!**
> *Observera innan du gör vad som står i detta kapitel att de flesta moderna distributionerna av Linux har ett program som enkelt och utan krångel sätter upp dina skrivare. Använd dessa. Läs i manualen till din Linuxdistribution. Har du inte något bra konfigureringsprogram eller är intresserad av hur det går till på riktigt så skall du läsa detta kapitel.*

Principen för hur utskrift fungerar i Linux är att man skickar en fil till skrivaren med ett kommando. Skrivaren är som allt annat i Linux en fil, även om det inte märks så tydligt. För att skrivaren skall kunna förstå vad den skall göra med filen som den får så måste filen vara i ett format som skrivaren känner igen. De flesta skrivare känner igen en ren textfil men sedan slutar likheterna. De flesta program i Linux producerar en PostScript fil då något grafiskt skall presenteras. En bra och trevlig skrivare känner direkt igen en PostScript fil och skriver ut denna. Nu är det tyvärr så att de skrivare som gör det ofta är dyra och inte så vanliga hos hemmaanvändaren eller det lilla kontoret. De vanligaste skrivarna har sitt eget språk eller ett annat halvstandardiserat språk. Problemet blir då, hur skall vi översätta PostScript filerna till ett språk som skrivaren förstår. Jo vi använder ett filter. Ett filter om tar en PostScriptfil och omvandlar den till en skrivaranpassad fil och skickar den till skrivaren. Extra trevligt vore det ju om detta skedde utan att användaren märkte det. Så att användaren skickar filen till skrivaren. Och skrivaren skriver ut. Precis så fungerar det i de flesta installationer. Då filen skickas till skrivaren skickas den egentligen till filtret, som omvandlar filen till skrivarformat och sedan, i sin tur, skickar den till skrivaren. Hur du får detta att fungera beskrivs i detta kapitel.

31.1 Förberedelser

Om du skall ansluta en skrivare till parallellporten på din dator måste stöd för detta kompileras in i kärnan eller ladas som en modul. Detta är redan gjort i de flesta moderna distributioner. Du kan kolla om detta är gjort genom att köra kommandot **`cat /proc/devices`**. Om du har stöd för en parallellprotsansluten skrivare skall du se *lp* listas bland enheterna. Om den inte gör det kan du kontrollera att lp finns som en modul. Annars måste du kompilera om kärnan och säga ja eller modul till *Parallell printer support*. Hur detta går till beskrivs i kapitel 23 på sidan 205.

Grundläggande för att skrivaren skall kunna användas är att den är ansluten och att man kan skicka tecken till den. Det kan du prova genom att testa något av dessa kommandon

```
echo "Kalle"> /dev/lp0
echo "Kalle"> /dev/lp1
```

Skrivaren skall rycka till och eventuellt skriva ut Kalle. Kom sedan ihåg om det var lp0 eller lp1 som skrivaren reagerade på.

31.2 Konfigurera skrivardemonen (`lpd`)

31.3 Ghostscript

31.4 Dela din skrivare med andra

I dag är det vanligt att man delar på sina enheter med andra. Detta är både kostnadseffektivt och enkelt. Linux har redan färdigt stöd för just detta.

31.5 Använda en fjärrskrivare

Med en fjärrskrivare menar jag en skrivare som inte sitter ansluten till din dator. I denna section beskriver jag hur du kan använda en nätverksskrivare ansluten till en UNIX maskin eller en Windowsmaskin. Du måste naturligtvis ha rättigheter att skriva ut på skrivaren för att detta skall fungera.

Kapitel 32

Håll reda på dina systemloggar

Linux håller reda på vad som händer i systemet och skriver detta i en hel bunt med loggfiler. I dessa kan du få mycket information om vad som händer i ditt system och hur det mår. Detta är i första hand intressant om du är systemadministratör för ett system med andra användare än dig själv och din familj. Det är också ett bra sätt att upptäcka att du haft dataintrång i ditt system.

32.0.1 Lär känna dina loggar

Genom att dagligen kolla dina loggar får du en känsla för hur de skall se ut när allt är som det skall. Då är det lättare att upptäcka om något inte är som det skall.

Kapitel 33

Systemklockan

Klockan är inte bara bra för att veta när det är dags att gå på lunch. Den är också en väsentlig del för systemets funktion.

I en dator så sitter det en klocka som går på batteri. Det gör att den fortsätter att gå även om systemet stängs av eller av annan anledning blir strömlöst. Då ditt system bootas startas en mjukvaru klocka som Linux använder istället för den fysiska klockan i datorn. Klockorna synkroniseras i uppstarten och går sedan helt oberoende av varandra. Anledningen till detta är att det är krångligt och långsamt att konstant kolla vad hårdvaruklockan visar.

Hårdvaruklockan är traditionellt i UNIX-sammanhang satt till standardtiden *Greewich Mean Time, GMT*. Detta gillar inte andra omerativsystem såsom MS-DOS med flera. Därför är det vanligt att man ställer hårdvaruklockan till lokal tid.

Vid installationen får man i de flesta fall en fråga om klockan är satt i GMT eller *Universal Time (Coordinated), UT(C)* som det också kallas. Har man missat det vid installationen så gör man på lite olika sätt beroende på vilket system man har. Kärnan använder dock alltid GMT-tid så även om din klocka är ställd på lokal tid så skall du tala om vilken tidszon du befinner dig i. I Sverige kallas tidszonen *Central European Time, CET* då vi har normaltid och *Central European Summer Time, CEST* då vi har sommartid. I installationsprogram och dyligt får man oftast välja vilket land man bor i så fixar allt sig automatiskt.

Man kan börja med att titta på filen `/etc/localtime` som är en länk. Man kan även kolla i sina uppstartsfiler, till exempel `/etc/rc.sysinit`. För att se hur systemet får reda på om klockan är satt till UTC eller ej. I RedHat görs detta i filen `/etc/sysconfig/clock/`.

33.0.2 Ställa klockan

Systemklockan i Linuxdrivs av avbrott som generas av maskinens hårdvara. Då Linux är ett äkta multitaskande system kan det under hård belastning hända att klockan sackar efter eftersom systemet inte hinner serva avbrotten inom den tid det skall ta. Detta

är ett mindre problem för hemmadatorer som startas om då och då. Men ibland måste klockan ställas.

Linux är mycket beroende av klockan. Att ställa klockan är inte så snällt. Det kan uppstå konstiga fel då klockan har justerats. Var alltid beredd på att få boota om systemet efter det att klockan ställts. Det allra bästa sättet att ställa klockan är att boota om datorn och ställa klockan i datorn BIOS. Men nu skall vi visa hur man ställer klockan inne i systemet.

Vi har tidigare sett att **`date`** ger aktuell tid. Man kan även använda `date` för att ställa systemklockan. Syntaxen för hur man anger tiden är lite krånglig. Man anger tiden i ett format "MMDDhhmm" där MM är månaden, DD är datumet och hhmm är timmar och minuter. Man kan även bygga på med år och slutligen sekunder efter en punkt. Studera följande exempel:

```
$ date
Thu Sep 30 19:37:12 CEST 1999
$ date 031712031973.06
date: cannot set date: Operation not permitted
Sat Mar 17 12:03:06 CET 1973
$
```

I exemplet ovan försökte jag sätta tiden till den tid jag föddes (inte på sekunden rätt men ganska nära). I exemplet ser vi att en vanlig användare inte får ställ om systemklockan. Man måste vara root för att kunna göra det. Nu finns det smartare sätt att göra detta. Man kan till exempel synkronisera klockan med en maskin som man vet har en klocka som går rätt.

För att synkronisera sin klocka med en annan dator kan man använda programmet `rdate` eller `netdate` här tittar vi på `rdate`. `Rdate` tar två flaggor **`-p`** eller **`-s`** Flaggan **`-p`** är default och behöver inte anges. Vidare tar `rdate` ett argument. Argumentet är namnet på den maskin som tiden skall hämtas ifrån. Flaggan **`-p`** visar tiden från den (eller de) maskiner som listas som argument. Flaggan **`-s`** ställer systemets klocka efter den andra datorns. För att kunna göra detta måste du naturligtvis vara root. Vi tar ett exempel på det också:

```
# rdate namatj
[namatj]        Thu Sep 30 19:55:17 1999
# rdate -s namatj
#
```

En dator som du kan synkronisera klockan mot här i Sverige är `timer.sunet.se`. Vill du att andra datorer skall kunna synkronisera sina klockor mot din maskin skall du se till att ha följande rader i `/etc/services`

```
daytime 13/tcp
daytime 13/udp
```

Den intresserade ser att synkroniseringen sker på port 13. Och mycket riktigt ser det ut så här då man telnettar till en maskin på port 13 som tillåter det:

```
$ telnet namatj 13
Trying 192.168.100.2...
Connected to namatj.
Escape character is '^]'.
Thu Sep 30 20:00:04 1999
Connection closed by foreign host.
$
```

33.0.3 NTP – network time protocol

33.0.4 Ställa hårdvaruklockan

Nu har vi ställt systemklockan. Men eftersom den synkroniseras med hårdvaruklockan vid nästa uppstart av systemet måste vi även ställa hårdvaruklockan. Detta gör man med komamndot `clock` som med flaggan **`-w`** synkroniserar hårdvaruklockan med systemklockan.

Alltså för att ställa klockan gör man enklast så här (observera att du naturligvis kör `clock` sist):

```
$ rdate -s timer.sunet.se
$ clock -w
```

Kapitel 34

Begränsa användares tillgång

I ett system med många användare kan det ofta vara bra att begränsa användarnas tillgång till diskutrymme. Annars är det lätt hänt att vissa användare medvetet, eller omedvetet tar upp en väldig massa diskutrymme vilket leder till att andra inte kan ha lika mycket. Du som administratör får till slut skäll för att diskarna hela tiden blir fulla. Din hjälpreda i dessa fall är `quota`. Med `quota` kan du ställa in hur mycket användarna får spara i sina hemmakataloger.

34.1 Hur `quota` fungerar

Här skall jag beskriva hur quota fungerar.

34.2 Hur får jag det att fungera?

Här skall jag beskriva hur jag får det att fungera så som jag beskrev ovan.

34.2.1 `quota`

34.2.2 `quotaon`

34.2.3 `quotaoff`

Kapitel 35

Processhantering

Detta kapitel behandlar mer avancerad processhantering än vad som gicks igenom i avsnitt 7.14.

Varje program som körs i Linux har ett unikt processnummer. En speciell process är programmet `init`. Då Linux startar bootas först kärnan. Då kärnan är laddad staras alltid `init`. `Init` tar sedan hand om uppsarten av datorn. Processnummren tilldelas processena i den ordning de startas, `init` har således alltid processnummer (PID) 1. Alla processer i Linux har en förälder, undantaget är `init` som är allas förälder. Man kan se alla processer i Linux som ett träd. Där `init` är roten och alla subprocesser sedan bildar trädet. Det speciella med detta träd är att det får bara finnas ett träd. Det vill säga, det får inte finnas några föräldralösa procecessrer som inte passar in i trädet. Alltså måste en förälder som dör se till att sina barn antingen dödas eller tas om hand.

Kommandon **`pstree`** visar hur processträdet ser ut för en maskin för ögonblicket. Där kan man mycket riktigt se att alla processer är barn till `init`. Nedan visas hur `pstree` kan se ut:

```
namatj:~/linuxboken% pstree -u marcus
.xsession--wmaker-+-rxvt--zsh-+-emacs
|                 `-pstree
`-wmclock
namatj:~/linuxboken%
```

I exemplet ger jag kommandot **`pstree -u marcus`** vilket innebär att bara marcus processer visas. Man ser att .xsession tydligen var det första jag startade (det skedde då jag loggade in). Sedan har en fönsterhanterare (wmaker) startats. Därefter delas trädet. wmaker har två barn, wmklock (en klocka som nu visar att det är natt) och rxvt. Rxvt är ett terminalfönster som jag brukar använda. I det har jag startat emacs (som jag skriver detta i) och naturligtvis pstree. Om man startar `pstree` med argumentet **`-p`** så läggs även information om processernas PID till i utskriften. Det kan många gånger vara smidigt.

Antag att du startat ett antal program i ett terminalfönster. Alla dessa är då barn till

terminalfönstret. Om terminalfönstret dör (till exempel om du trycker på en "döda-knapp" i din fönsterhanterare) så kommer alla barn till terminalfönstret också att dö eftersom de då blir föräldralösa och operativsystemet ser till att inga sådana får förekomma. Om du istället avslutar terminalfönstret med commandot **`exit`** så märker du att barnen lever kvar och fungerar som om inget hade hänt. Hur gick det till? Jo, `exit` ser till att alla barn ärvs av processen med PID 1 (`init`). Detta kan du skåda med programmet `pstree`.

> **Tips!**
> *Ta för vana att avsluta dina terminalfönster med `exit` så riskerar du inte att slå ihjäl några program du vill skall fortsätta att exekveras.*

Här skall i ett senare skede förklaras mer ingående vad som händer med föläldrar och barn, wait(), mm. Samt zombies.

Huvudprogrammet för att kontrollera processer är `ps`.

Ett annat användbart program är `top`

Hur man behandlar ett program som slutat fungera, `kill` `xkill`.

Kapitel 36

Linux i nätverk

Detta kapitel handlar om Linux i nätverksmiljö. Det är i sådan miljö Linux kommer till sin fulla rätt. Mycket av det som står här kan du även använda på en fristående dator. Naturligtvis vill du använda Internet. Men du kan även ha mycket glädje av att köra en Web-server på din lokala burk. Det är et utmärkt sätt att träna sig på att, till exempel, lära sig skriva CGI-script. Det får du ju troligtvis inte göra hos din Internetleverantör. Du lär dig dessutom naturligtvis hur man sätter upp en web-server.

Första delen av detta kapitel behandlar sådant som du kan ha nytta av om du bara har tillgång till internet via modem. Den andra delen behandlar mer avancerade saker som du kan utnyttja i ett eget nätverk.

36.1 Grundläggande

Då man skall koppla upp en Linuxmaskin mot ett nätverk är det oftast ett TCP/IP nätverk som man har i åtanke.

36.1.1 Användbara verktyg

Dessa kommandon är avancerade nätverkskommandon. Du kan dock använda dem även för att kolla din modemförbindelse.

`ping`

`Ping` är ett litet trevligt, och mycket användbart, program. Med `ping` kan du kolla om ditt nätverk fungerar och att dina, eller andras, datorer svarar på nätverksanrop.

`netstat`

`route`

I detta avsnitt behandlas bara hur du kan använda `route` för att titta på din routingtabell. Det behandlas mer ingående nedan.

36.2 tcp-ip, kort-kort variant

Här beskriver jag hur tcp-ip fungerar. Detta är verkligen bara det abslolut nödvändiganste. Jag går (i alla fall inte i denna version) in på hur paketen ser ut, vad som skickas i dem, olika lager mm.

Alla datorer i ett tcp/ip nät (till exempel Internet) har en egen unik adress. Denna består av ett tal på formen aaa.bbb.ccc.ddd. Bokstäverna är fyra tal mellan 0 och 255. I nästa version av ip (v6) kommer adresserna att se annorlunda ut, mer om det senare. Internet består av flera mindre nätverk. Till exempel kan det lokala nätverket på ett företag vara en del av Internet. Adressen är uppbyggd hierarkist så att talen längst till vänster talar om vilket nät som datorn hör till och siffrorna till höger identifierar själva datorn. Hur stol del av adressen som specifierar ett nätverk beror på hur stort nätverk som datorn står i. Denna numeriska adress kallas ip-nummer. Och det är bara den som datorerna använder då de “talar” med varandra. Men denna adress är nästan hopplös för oss människor att lära oss. Tänk själv hur många du skulle kunna memorera. Därför har man kommit på det smarta att använda det så kallade domännamnssystemet. Det går ut på att alla datorer får ett namn som är lättare för människor att komma ihåg. Ett exempel på ett sådant namn är `www.sunet.se`. Men som jag sade tidigare så kan datorerna bara använda den numeriska adressen. Vi skulle behöva något som kan översätta den “mänskliga” adressen tilll en numerisk datorvänlig. Just detta sköter de så kallade namnservrarna om. Det enda de gör är att översätta datronamn till ip-nummer. Alla namnservrar innehåller en tabell med namn och ip-nummer. Om jag frågan till exempel en namnserver om ett ip-nummer till en viss maskin så kollar namnservern om den finns med i sin tabell. Finns inte maskinen där så frågar namnservern en annan namnserver.

Din dator måste alltså veta vad namnservern heter. Naturligtvis måste maskinen veta ip-nummret till en namnserver.

36.3 Uppringd Internet

Idag är det vanligast att man som privatperson kopplar upp sig mot Internet med modem. Förhoppningsvis vilar snart modemen och disketterna på samma kyrkogård som alla skrivmaskiner.

Att koppla upp mot en Internetleverantörs (ISP – Internet Service Provider) modempool är inte så svårt. Det är i huvudsak fem saker som skall göras.

- Skapa en seriell förbindelse
- Logga in (Autentisering)
- Ev starta ppp på ISPns server
- starta ppp på din dator
- Sätta ppp-koplingen till *Default Route* på din dator

Här beskriver jag varje del i mer detalj. Jag går inte så djupt in på detta nu utan hoppas att modem och modempooler snart tillhör historien (De står bra på bredvid disketten . . .). Man behöver dessutom inte kunna detta på så djup teoretisk nivå.

Den första delan sköts av ett preogram som heter ppp. Allt det andra sköts av ett program som heter chat.

När du är uppkopplad vill du kunna hitta maskiner med annat än ip-nummer. Då åste du ha tillgång till en namnserver. Det får du av din internetleverantör. ip-nummret till denna lägger di in i filen `/etc/resolv.conf`. Filen kan se ut så här:

```
search
nameserver 192.71.220.10
```

I detta fall har jag en nanmserver, men man kan ha flera. De måste då stå på olika rader som alla ser ut som den ovan (Fast olika ip-nummer).

36.4 RedHats `netcfg`

Att koppla upp sig mot sin internetleverantör med RedHats `netcfg` är en barnlek. XXXXXXXX

36.5 wvdial

Har du inte en distribution som enkelt låter dig konfigurera din internetuppkoppling med modem så kan du prova wvdial. Flera distributioner (däribland RedHat) har bra program för att konfigurera detta. Men om det saknas eller du inte för det att fungera så rekommenderar jag `wvdial`. `Wvdial` försöker lista ut hur uppkopplingen skall gå till och lyckas oftast.

`Wdial`s konfigureringsfil brukar heta `wvdial.conf` och ligger i `/etc`. Med `wvdial` följer ett program för att skapa denna fil. Programmet heter `wvdialconf`. Då du startat programmet letar det rätt på ditt modem och testar fram en bra init-sträng till det. Sedan måste du själv editera filen och skriva in ditt användarnamn och lösenord hos din internetleverantör. En `wvdial.conf` kan se ut så här:

Har du en bärbar dator eller vill konfigurera olika upprigningar så kan du göra det i configureringsfilen till wvdial, ofta `wvdial.conf` i katalogen `/etc`

36.6 KabelTV-modem

Det blir allt vanligare att privatpersoner koplar upp sig med specialbyggda modem som fungerar mot kabeltvnätet. Om dessa har jag inga större kunskaper. På de ställen jag har installerat sådana tingastar har de fungerat som en vanlig ethernetkoppling på den sida som är mot datorn. Detta gör att man konfigurerar dem som vilken nätverksanslutning som helst, ip-numer, namnserver och eventuellt gateway.

36.7 Junkbuster

Ett program som är speciellt smidigt då man har en modemkoppling till Internet är `Junkbuster`. Junkbuster är en proxyserver som filtrerar bort saker från internetsurfandet som du inte vill åt. Den skyddar dig dessutom mot cookies och annat du inte vill utsätta dig för. Den var från början tänkt som ett sätt att skydda din integritet på nätet men används idag mest för att filtrera bort reklam för att ditt surfande skall gå så effektivt som möjligt.

36.7.1 Fax-tjänst

I anvsnitt 15.10 på sidan 163 beskrivs hur en faxtjänst används. Här beskrivs hur du sätter upp en sådan.

Vad stod i denna fil XXXXXXXXXXXXXX

36.8 Samba, SMB

Programsviten Samba gör det möjligt för din Linux-maskin att vara en fil- och skrivarserver i ett nät med Windowsklienter.

36.9 Webserver – Apache

I detta avsnitt skall vi bahandla web-servern Apache. Apache är den vanligaste webservern på Internet Den är fri och lätt att konfigurera. Du har den kanske till och med installerad på din maskin utan att veta om det.

Vi har ju nu behandlat Apache. Du bör också veta att det finns andra web-servrar. En mycket trevlig svensk web-server är en som heter Roxen.

36.10 Namnserver (DNS) – Bind

Ett av de viktigaste systemen i Internet är domännamnssystemet DNS. Detta klapitel beskriver hur du blir en del i det.

Det är aven mycket bra att ha en egen namnserver i sitt egna lilla nätverk hemma.

Om du råkar ut för att någon nätaktivitet tar väldigt lång tid eller att ett nätrelaterat program inte startar som det skall så har du förmodligen problem med din DNS konfiguration.

36.11 DHCP server

Det mest använda protokollet för att automatiskt konfigurera maskiner i ett nätverk ar DHCP (Dynamic Host Configuration Protocol???).

Att i sitt nätverk ha en egen DHCP server är mycket smidigt. Framför allt aom man till och från har gäster i sitt nätverk (då avser jag inbjudna gäster . . .).

36.12 Mer läsning

Net-3-HOWTO är ett dokument om Linux i nätverk från Linux Documantation Project (LDP) som är mycket läsbart.

ISP-Hookup-HOWTO är ett dokument om hur man anluter till en iternetleverantör med Linux från Linux Documantation Project (LDP) som är mycket läsbart.

Tanenbaum, 1996 är ett bra bok om datornät.

Kapitel 37

Vanliga fel, och hur man löser dem

I detta kapitel tar jag upp vanliga fel som inte behandlats tidigare i boken. Detta är nästan lite av en faq än så länge. Jag har försökt att flika in de mesta där det hör hemma. Men detta är det som "blev över". Kommer kanske att ändra detta i senare versioner av denna bok.

37.1 `LILO` har försvunnit

Detta problem är inte helt ovanligt. Det kan bero på att du har installerat eller uppgraderat ett annat operativsystem på datorn. Du måste lösa problemet genom att starta Linux på ett alternativt sätt och i Linux köra kommandot `/sbin/lilo` som root.

Detta är enkelt om du har en boot-diskett til ditt system. I så fall är det bara att sätta i den och starta om datorn. Då du för loginprompten loggar du in som **root** och kör `lilo`.

37.2 Jag VILL att `LILO` skall försvinna

Oj då. Hoppas att detta beror på att du skall starta Linux på ett annat sätt. Men förmodligen vill du starta ett annat operativsystem. I DOS/Win kan du enkelt fixa detta med programmet `fdisk`. Du gör så här. Starta DOS (eller Windows). Ge kommandot `c:\> fdisk /mbr`. Nu skall ditt problem vara ur världen. Vad som hände var att Windows `fdisk` skrev över *Master Boot Reckord* (Där LILO huserade) med den information som behövs för att starta det andra operativsystemet.

37.3 Jag kan inte "umounta" CD-romen eller annan enhet

Detta beror på att någon använder den. Problemet är att hitta vad som håller i enheten. Kolla först så att du inte har en katalog på den monterade enheten som aktiv katalog i något terminalfönster. Kolla också att inget program är öppet som kan använda enheten.

Hittade du inget? Det finns hjälp att tillgå. Programmet `fuser`, vanligtvis i `/usr/sbin` kan tala om för dig vad som håller enheten.

Kapitel 38

Konfigurera cron och at

`cron` och `at` beskrivs i användardelen av denna bok. I detta kapitel beskrivs hur du som systemadministratör får dessa att fungera och hur du kan använda dem för att underlätta ditt dagliga arbete.

38.1 cron

`cron` installerades då du installerade ditt system. `Cron` består av en daemon, `crond` som startas av ett lämpligt script i `/etc/rc.d` använder du SysV så heter den vanligen `/etc/rc.d/init.d/crond`. Använder du ett BSD init system så startas den vanligen i `/etc/rc.d/XXXXXX`.

Användarnas `crontab`-filer ligger vanligen i `/var/spool/cron` användarna kan inte själv direkt ändra sina crontabfiler untam måste använda sig av SUID root programmet `crontab`. Detta gör att du kan (skall!) Ha följande rättigheter på crontabfilerna

```
\# ls -ld /var/spool/cron/ \\
drwx------   2 root    root        1024 Nov  2 22:11 /var/spool/cron/
\# ls -l /var/spool/cron/marcus \\
-rw-------   1 root    marcus       407 Nov  2 22:11 /var/spool/cron/
\#
```

I de flesta implementationer av `crond` så finns det en crontab för systemet i `/etc`. Den exekverar alla skript i `/etc/cron.d/*`.

`Crond` håller reda på alla ställen där den skall leta efter kommandon och vaknar upp varje minut och kollar igenom om det är något den skall göra. `Cron` kollar också om tidsstämpeln på någon av filerna har ändrats. Har den ändrats på någon fil så läses den in igen. På så sätt behöver aldrig `crond` startas om så fort någon ändrat sin `crontab`.

Kapitel 39

Ta backup

Det kan väl knappast påtryckas hårt nog. Backup är viktigt. Det finns flera verktyg att använda för att ta backup. Det som du som vanlig användare bör kunna hantera är `tar` (Tape ARchiver).

39.0.1 `tar`

`Tar` har som de flesta verkyg till Unix en mängd växlar (eller flaggor). Vill du lära dig allt rekommenderar jag **`man tar`**. De nödvändigaste tänker jag behandla här.

Kapitel 40

Enhetsfi ler

I detta kapitel behandlas hur du adresserar olika enheter i Linux. En sak som är rolig i Linux är att allt är filer. Så all hårdvara som du har representeras i systemet som en fil. Detta är väldigt praktiskt och inbjuder till experimenterande ...

Utförlig information om alla olika devicefiler och vad de gör kan du hitta i filen `/usr/src/linux/Documentation/devices.txt` som bör finnas i ditt system. Gör den inte det kan du installera källkodsträdet till Linuxkärnan eller läsa den på Internet på adressen `ftp://ftp.kernel.org/pub/linux/docs/device-list/`

Kapitel 41

Mer om viruellt minne (swap)

Linux bör ha tillgång till swapminne. Även om man har mycket minne.

I installationsdelen behandlades hur man skapar en swappartition i samband med installationen. Här behandlas swapminnet mer ingående. Dessutom behandlas hur man i efterhand lägger till swappartitioner om man skulle upptäcka att den man gjorde inte räcker till.

I detta kapitel visas också hur man kan skapa en fil som används som swapminne.

41.1 `vmstat`

`vmstat` visar statistik över användningen av det virtuella minnet. `Vmstat` är ett avancerat kommando. Men det ger en hel del information.

41.2 Swapaktivering vid systemstart

Du skapade ju en swappartitione då du installerade ditt Linux. Sedan fungerade det bara. Här skall vi titta lite på vad systemet gör med din swapfil då det startar upp.

41.3 Lägga till swappartition

41.4 Lägga till swapfi l

Man kan om man till exepmel bara tillfälligt vill ah lite extra minne lägga till en fil som används som swap. Detta fungerar bra men har vissa begränsningar. Att använda en fil i ett befintligt filsystem gör också att det går något långsammare eftersom ett

filsystem, med allt vad det innebär, ligger mellan kärnan och swapfilen. Det är dock väldigt flexibelt och enkelt så det kan vara bra att ta till ibland. Minnen är idag dock så pass billiga att minnesbrist borde inte vara något stort problem.

Så här skapar man en swapfil:

XXXXXXXXXXXXXX

```
dd if=/dev/zero of=swap bs=1M count=50
mkswap swap
swapon swap
...
swapoff swap
rm swap
```

Del VIII

Programmering

Kapitel 42

Programmering

I denna del går jag igenom lite om programmering. Jag kommer inte att gå igenom något programmeringsspråk utan bara som kan vara lite speciellt för Linux och annat jag tycker är intressant. Framför allt avsnittet om `make` är intressant även för de som inte programmerar.

Linux är en underbar plattform att programmera från. Inte mist eftersom du redan från början har alla de verktyg du behöver. Ovavsett om du vill programmera i Java, C, C++, Pascal, Perl, Ada, Fortran, Prolog, Lisp, ja nästan vad du vill . . .

Kapitel 43

Make

Make är ett finurligt program. Det används vanligvis vid programmering, men kan användas till annat också. Till exempel används make för att typsätta denna bok.

Antag att du sitter och programmerar ett program som består av ett flertal filer. För att kompilera och provköra detta program måste du köra ett antal krångliga och långa program. Du ledsnar snart på att skriva dessa kommandon om och om igen. Du börjar fundera på om det finns något enklare sätt. Det första du kanske kommer på är att skriva ett skript som löser din uppgift. Du programmerar på och kompilerar med ditt skript. Allt är bra och du är lycklig. Ditt program växer och blir större. Det tar längre och längre tid att kompilera det. Då upptäcker du att det naturligtvis går fortare om du bara kompilerar om de filer du ändrar och låter de andra vara. Du funderar och konstaterar att det blir jobbigt att skriva ett nytt skript för varje fil du uppdaterar. Tänk om det fanns ett program som bara kompilerar de filer du har ändrat och sedan länkar ihop ditt program. Naturligtvis finns det ett sådant program. Det heter `make`.

`Make` använder sig av en fil som heter `Makefile` i vilken det beskrivs vad programmet skall göra. Du behöver inte specifiera för `make` vilken `Makefile` den skall använda utna den tittar alltid i aktuell katalog efter en `Makefile` därför är det noga att du har rätt katalog aktiv då du ger kommandot **`make`**.

Här skall jag förklara hur man skriver enkla Makefiler.

Kapitel 44

Länkare

Efter att ett program har kompilerats så skall det länkas. Det innebär att den kompilerade källkoden skall sättas ihop med andra färdigkompilerade delar av programmet som följer med ditt system.

Kapitel 45

Debuggers

Ibland (nåja oftast) stöter man på problem då man programmerar. Vissa fel man gjort kan vara hopplösa att hitta. Då kan man ta en debugger till sin hjälp. Det finns redan minst en sådan installerad i ditt Linuxsystem.

45.1 gdb

45.2 xxgdb

45.3 ddd

Kapitel 46

Versionshantering

46.1 cvs

46.2 rcs

Kapitel 47

Specifi ka språk

47.1 C

UNIX skrevs om helt 1973. Detta gjorders då i programspråket C. Linux är från början skrivet i C. Därför är Linux mycket nära besläktat med C.

47.2 C++

Många, dock inte männen bakom, ser C++ som ett bättre C. Det är vanligt att man får frågan om det finns någon så kallad integrerad miljå till Linux. Xemacs, emaxs, kdevelop.

47.3 Java

Det finns interpreterande språk och kompilerande språk ...

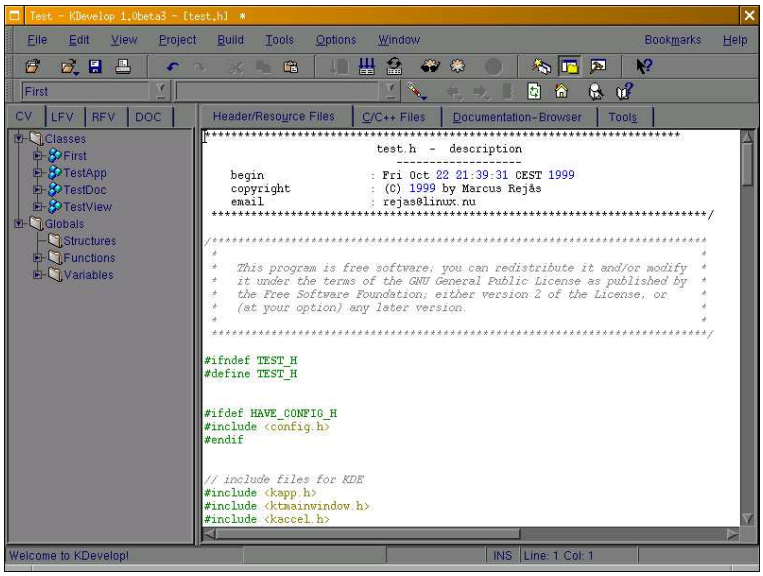
47.4 Perl

Perl är ett interpreterande språk.

Övningsuppgifter

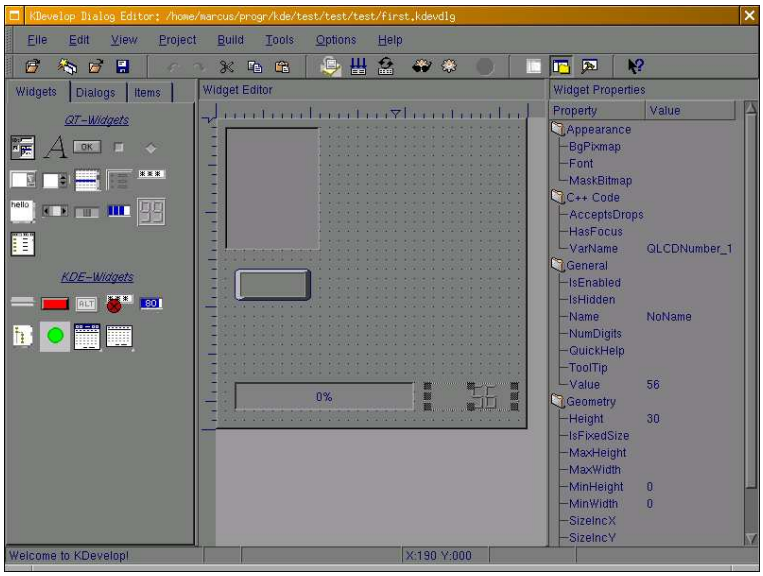
Uppgift 47.1

Uppgift 47.2



Figur 47.1: KDevelop, integrerad programmeringsmiljö

KDevelop är en integrerad miljö för att, i C++, skriva KDE program. Man kan även skriva andra program i det.



Figur 47.2: KDevelop, Dialog Editorn

KDevelop har en integrerad dialog editor. I den kan man dra och släppa widgets i dialogrutan. Då man är nöjd ber man miljö generera C++ kod för sin nyskapade dialog. Smidigt!

Del IX

GNU och Free Software Foundation

Kapitel 48

GNU och GNU General Public License

48.1 GNU Projektet

GNU projektet startatdes av **Richard Stallman** 1983. Det var alltså långt innan utvecklingen av Linux hade börjat.

På 70-talet då Stallman jobbade på MIT jobbade de på ett sätt som innebar att man delade med sig av all programvara. Hade man skrivit ett program som utförde något på ett bra sätt så var det självklart att man dealde med sig av det. På samma sätt bad man att få se källkoden om man såg ett program som gjorde något finurligt. Detta gällde inte bara inom MIT utan även mellan programmerare på olika företag.

I början på 80 talet började företagen få upp ögonen för att mjukvaran var något att tjäna pengar på. Man började då att hemlighålla källkod inom företaget. Detta uppskattades inte av programmerare (i synnerhet inte av Stallman) eftersom det innebar mer arbete för dem. Det kändes nu jobbigt att sitta och utveckla något som man visste att kompisen på det andra företaget redan gjort men inte får dela med sig av.

Stallman talar om ett samhälle av datoranvändare och programmerare. Ett samhälle där alla hjälper varandra för att på ett effektivt sätt driva utvecklingen frammåt. Varför sitta och utveckla en sak som någon annan redan gjort? Stallman nämner att med det nya systemet blev det inte tillåtet att hjälpa sin granne utan att vara en “pirat”.

Stallman ställdes inför ett svårt moralisk beslut, antingen bara fortsätta och rätta sig efter det nya spelreglerna. Eller helt hoppa av datorbranchen och göra något annat. Eller det tredje, försöka bygga upp det sammhälle som en gång var.

Stallman beslöt att det tredje alternativet var det bästa. På så sätt skulle hans kunskaper varken missbrukas (som i det första fallet) eller slösas (som i det andra). Han ställde frågan, vilka program behövs för detta samhälle och kan jag skriva dem? Han kom fram till att ett operativsystem stod överst på listan.

Figur 48.1: GNU projektets logo

Gnu projektets logo är just en Gnu. Den kan se ut på många olika vis. Här an en mer konstnärlig variant.

Stallman som utvecklade just operativsystem konstaterade att han hade de kunskaper som behövdes. Han var rädd att hans projekt skulle hamna i MITs klor om han jobbade kvar där så han sade upp sig och startade GNU projektet. Förkortningen GNU följer en hacker tradition att göra rekursiva forkortningar och står för **G**NU's **N**ot **U**NIX.

Han satte upp som mål för projektet att skapa ett fritt operativsystem, kallat GNU operating system. Detta operativsystem skulle inte bara innehålla en kärna utan skulle, som alla stora operativsystem, innehålla assemblerare, kompilatorer, texteditorer, mailprogram mm. Han började med att definiera var "fri programvara" är. Fri programvara, enligt Stallman, är;

- Du har tillstånd att köra programmet i vilket syfte som helst.
- Du för modifiera programmet hur du vill för att det skall passa dina behov (detta medför att källkoden måste finnas tillgänglig).
- Du för distribuera kopior, antingen gratis eller mot en avgift.
- Du kan distribuera modifierade versioner så att samhället kan dra nytta av dina förbättringar.

Det gäller att göra skillnad på fri och gratis. Det är tillåtet att ta betalt för att distribuera fri programmvara (även i syfte att käna pengar). Faktum ar är att det är på så sätt utvecklingen av fri programvara finansieras.

I januari -84 slutade Stallman på MIT för att starta sitt projekt. Han började att utveckla en kompilator (nu känd som GCC). Innan den blev klar hade han börjat att utveckla ett program som kom att bli mycket betydelsefullt, nämligen GNU Emacs. GNU Emacs började Stallman utveckla i september -84 och den började bli användbar i början av -85. Naturligvis lade han ut den på en ftp-sajt så att alla som ville kunde komma åt den. Nu var det så att ryktet om denna fria editor spred sig, men inte alla hade tillgång till Internet (Visste du, till exempel, ens vad det var 1985?). Så han erbjöds sig att spara ned programvaran på band om man skickade honom ett band och porto.

I samma veva började naturligtvis pengarna att ta slut. Stallman behövde ett sätt att tjäna pengar på sin fria mjukvara. Han började ta betalt för att kopiera mjukvaran till band och kunde på så sätt fortsätta att utveckla den.

GNU projektet behövde nu någon typ av programvarilucens som gjorde det omöjligt att omvandla fri programvara till kommersiell, icke fri sådan. Den metod som användes kallas för *copyleft*. Copyleft fungerar på samma sätt som *copyright* men vänder hela konceptet. Iställer för att privatisera mjukvara ser copyleft till att den alltid förblir fri.

Copyleft implementerades i GNU General Public Lincense (GNU GPL, eller bara GPL). Denna behandlas i avsnitt 48.3 på sidan 290

Idag är i princip allt klart till GNU operating system utom själva kärnan. Dock finns ju Linuxkärnan att tillgå (Linuxkärnan är inte GNU mjukvara, men släpps under GNU GPL). Med hjälp av denna kan man sätta ihop GNU system sum fungerar och används av mijontals användare världen över. Utvecklingen av GNUs kärna (kallad HURD) har inte på något vis avstannat utan fortsätter och kommer att släppas i en stabil version inom en snar framtid.

48.2 Free Software Foundation (FSF)

GNU projektet behöver pengar för att kunna utveckla sin mjukvara. För denna del startades Free Software Foundation (FSF). FSF tog över distributionen av mjukvara (Emacs mm) på band och började sälja manualer med mera till den fria mjukvaran. FSF tar också emot donationer, både pengar och hårdvara.

XXXXXXXXXXXXXX

48.3 GNU Gneral Public License (GPL)

Många program som släpps till Linux släpps under något som heter GNU General Public License, eller kort GPL. Denna licens är framtagen av GNU projektet. GPL är den licens som garanterar att Linux alltid kommer att finnas fritt tillgängligt. Fritt och gratis är i detta sammanhang inte riktigt samma sak. Det finns flera program som inte kostar något men som ändå inte är fria. Det är program som är gratis men som du inte får med källkoden till. Du kan med andra ord inte se hur programmet är gjort och du kan inte ändra i det så att du får det att fungera som du vill. Det är ett exempel på ett gratis program som inte är fritt. Ett fritt program distribueras med kälkoden öppen. Du kan se hur programmet fungerar. Det finns inga hemligheter i programmet som du inte kan se. Vill du ändra programmet så att det passar dig bättre ändrar du i källkoden. Detta är ett fritt program. Du kan få betala för att få ett fritt program. Programmet är ändå fritt. de flesta fria program brukar vara både fria och gratis.

48.3.1 GPL på svenska

GPL går kortfattat ut på följande:

- GPL Begränsar inte användandet av mjukvaran. Du får använda det hur du vill.
- Du får vidaredistribuera programmet, gratis eller mot avgift, förutsatt att
  - Du distribuerar det under GPL (gäller även om du ändrat programmet).
  - Du gör källkoden tillgänglig till de du distribuerar programmet. Detta kan göra på tre sätt
    1. Skicka med komplett källkod.
    2. Skicka med ett intyg där du lovar att inom tre år skicka källkoden till godtycklig tredje part mot avgift som max motsvarar dina utlägg för detta.
    3. Skicka med ett intyg, som du mottog, enligt punkt två. Detta har vissa begränsningar, läs mer i licensen.
- Du får ändra programmet.
- Du får distribuera dina alster, baserade på programmet.

GPL finns som bilaga till denna bok. På originalspråk i bilaga E och en svensk översättning i bilaga F.

Del X

Avrundning

Kapitel 49

Andra fria operativsystem

Nu när du kommit så här långt i denna boken har du förhoppningsvis lärt dig något om Linux. Du utforskar förmodligen ditt system för att lära dig hur det är uppbyggt. Jag kommer ihåg känslan när jag för första gången installerade Slackware Linux hemma på min dator. Det var nämligen faktiskt inte så länge sedan.

Först tillbringade jag två dagar med att tanka hem de paket jag ville ha. Att jag valde just Slackware som min första distribution var att den på ftp-sajten låg upplagd i kataloger där varje katalog rymdes på en diskett. Så jag tankade ner de disketter jag behövde och kopierade över på disketter och fick på så sätt de installationsdisketter som jag ville ha. Att detta inte var så smidigt visste jag ju inte då. Själva nedtankningen tog som sagt ungefär två dagar (inte dygnet runt dock) trots att jag hade ett alldeles nytt svindyrt 28k8 modem som gick som blixten.

Till slut kunde jag starta installationen. Jag fick en mängd frågor om olika programvaror –vill du installera awk-x-x (rekommenderas)? Kruxet var ju förstås att jag aldrig hade hört talas om programmen och vad de gjorde. Problemet var ju dessutom att diskarna då kostade en förmögenhet så man hade inte råd att skicka in något onödigt.

Trots allt så gick installationen klockrent. Jag fick bolla en hel den med Slackwares **`pkgtool`** för att ta bort och lägga till paket. Men jag fick snart ett fungerande system. Vid det har laget hade jag fortfarande MS-DOS installerat och startade det så fort jag skulle göra något nyttigt. Min Linux-installation använde jag bara för att utforska och lära mig mera. Ju mer jag lärde mig desto mer av nyttoarbetet flyttades över till Linux. Som en följd av detta blev det längre och längre mellan de gånger MS-DOS startades. Till slut kunde jag konstatera att jag inte startade MS-DOS mer.

Varför skrev jag denna nostalgiska text här i slutet på denna bok? –Ja inte vet jag egentligen. Jo det var ju det. Den känslan jag kände då jag trevande utforskade Linux och som du kanske känner nu (och förhoppningsvis uppskattar, vissa tycker ju bara att det är fustrerande) kan man få känna fler gånger. När du behärskar ditt Linux-system och helst skulle vilja vara en kunskapstörstande nybörjare igen rekommenderar jag starkt att du testar något av dessa andra fria operativsystem. För det är ju inte så att Linux är det enda fria operrativsystemet på marknaden (även om det kan verka så i media).

Jag presenterar här slutligen några andra fria operativsystem som du, på samma sätt som Linux, kan tanka hem från någon lämplig ftp-plats på Internet.

49.1 GNU HURD

GNU HURD har ännu inte nått ett läge då det betraktas som stabilt. Det här operativsystemet var ju målet med GNU projektet på det startades i början av 80-talet. HURD distribueras naturligtvis under GNU General public License.

49.2 FreeBSD

49.3 OpenBSD

49.4 NetBSD

Del XI

Bilagor

Bilaga A

GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the
   Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications
   adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice
   giving the public permission to use the Modified Version under the
   terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections
   and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add
   to it an item stating at least the title, year, new authors, and
   publisher of the Modified Version as given on the Title Page. If
   there is no section Entitled "History" in the Document, create one
   stating the title, year, authors, and publisher of the Document as
   given on its Title Page, then add an item describing the Modified
   Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
   it was based on. These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
   Preserve the Title of the section, and preserve in the section all
   the substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles. Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
   or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant. To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version. Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity. If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the
Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its
license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warrany Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bilaga B

XF86Config

```
Section "Files"
   RgbPath    "/usr/X11R6/lib/X11/rgb"
   FontPath   "/usr/X11R6/lib/X11/fonts/misc:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/URW"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/Type1"
   FontPath   "/usr/X11R6/lib/X11/fonts/Speedo"
   FontPath   "/usr/X11R6/lib/X11/fonts/misc"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/URW"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi"
EndSection

Section "ServerFlags"
EndSection

Section "Keyboard"
   Protocol        "Standard"
   XkbRules        "xfree86"
   XkbModel        "pc101"
   XkbLayout       "fi"
EndSection

Section "Pointer"
   Protocol        "PS/2"
   Device          "/dev/mouse"
   Emulate3Buttons
EndSection



Section "Monitor"
    Identifier "ThinkPad 560E"
    VendorName "IBM"
    ModelName "800x600 TFT"
    HorizSync 24.21-50
    VertRefresh 37.83-76
    Modeline "640x480" 25.18 640 664 760 800   480 491 493 525
    ModeLine "800x600" 25.18 800 808 1024 1040 600 600 626 640 +hsync +vsync
EndSection

Section "Device"
    Identifier "Trident 9382"
    VendorName "Trident"
```

```
    BoardName "9382"
    Chipset "cyber9382"
    ClockChip "tgui"
    VideoRam 1024
EndSection

Section "Screen"
    Driver "SVGA"
    Device "Trident 9382"
    Monitor "ThinkPad 560E"
    DefaultColorDepth 16
    Subsection "Display"
        Depth     8
        Modes     "800x600" "640x480"
    EndSubsection
    Subsection "Display"
        Depth     16
        Modes     "800x600" "640x480"
    EndSubsection
EndSection

Section "Screen"
    Driver "VGA16"
    Device "Trident 9382"
    Monitor "ThinkPad 560E"
    SubSection "Display"
Depth     4
Modes     "800x600" "640x480"
    EndSubSection
EndSection
```

Bilaga C

fstab

```
#
# /etc/fstab
# Marcus Rejas 990115
#
# Changelog:
# -----------------------------------------------------------------------------
#
#
# These partitions are initially mounted
#
/dev/hda2               /                       ext2    defaults        1 1
/dev/hda8               /cd-img                 vfat    defaults        0 0
/dev/hda7               /home                   ext2    defaults        1 2
/dev/hda6               /usr                    ext2    defaults        1 2
/dev/hda4               /usr/local              ext2    defaults        1 2
/dev/hda5               swap                    swap    defaults        0 0
/dev/hda1               /win                    vfat    defaults        0 0
none                    /proc                   proc    defaults        0 0
#
# These are mountable for users
#
/dev/fd0                /floppy                 vfat    user,noauto     0 0
/dev/fd0                /floppy_ext2            ext2    user,noauto     0 0
/dev/sda1               /zip                    vfat    user,noauto     0 0
/dev/sda1               /zip_ext2               ext2    user,noauto     0 0
keron:/                 /keron                  nfs     user,noauto     0 0
```

Bilaga D

lilo.conf

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/etc/lilo.msg
prompt
timeout=50

image=/boot/vmlinuz
        label=lx
        root=/dev/hda2
        read-only

image=/boot/vmlinuz-old
        label=lxo
        root=/dev/hda2
        read-only

image=/boot/vmlinuz-test
        label=lxt
        root=/dev/hda2
        read-only

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

Bilaga E

GNU General public license (GPL)

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

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.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. 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.

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

NO WARRANTY

11. 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.

12. 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) 19yy  <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) 19yy 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.

Bilaga F

GNU GPL - Svensk översättning

Inofficiell översättning av GNU General Public License version 2.
Denna översättning har version 0.1.3, se nedan för mer info.

Översättarens förord till denna svenska version

Detta är en översättning av GNU General Public License (GNU GPL), den mjukvarulicens som används för många program från Free Software Foundation och mycket av övrig fri mjukvara på nätet. Syftet med denna översättning är att, för personer som har svenska som modersmål, underlätta förståelsen av vad GNU GPL innebär.

Använd inte denna översättning som licens för era egna program. Ge istället ut dem under den engelska licensen (och skicka då ev. med denna översättning för att underlätta förståelsen). Att använda den svenska licensen kan medföra mängder med problem för dig och alla andra som ska vidareutveckla programmet.

Jag är teknolog på datatekniklinjen på Chalmers och inte någon professionell översättare eller expert på amerikansk lagstiftning. Nedanstående licens är således inte rättsligt bindande på något sätt. Det är texten i den ursprungliga engelskspråkiga versionen av licensen som ska gälla.

Då mina kunskaper i engelska språket har sina begränsningar är jag mycket tacksam om jag får rapporter om brister i översättningen, så att jag kan rätta till ev. fel. Jag har försökt att följa den engelska ordalydelsen av licensen när jag ansett att så varit möjligt inom ramen för det svenska språket.

Skicka förslag på hur översättningen kan förbättras till mig så att vi med tiden kan få en helt korrekt översättning.

Jag har infört ett versionsnummer på översättningen. Tills vi når version 1.0.0 är detta dokument att betrakta som en alfaversion. Enda syftet är just nu att texten ska korrekturläsas, felaktiga översättningar rättas och dåliga eller oklara översättningar ska förbättras. Denna och ev. senare versioner kommer finnas åtkomliga någonstans via länk från min hemsida. Och från Swe-doc projektets hemsida på http://www.swe-doc.linux.nu.

Tack på förhand:

Jörgen Granstam <Jorgen.Granstam@abc.se>
http:/www.abc.se/~m8810

GNUs ALLMÄNNA PUBLIKA LICENS

Version 2, June 1991

Förord

Licenserna för de flesta mjukvaror är utformade för att ta ifrån dig friheten att dela med dig av dem och modifiera dem. I motsats är GNUs Allmänna Publika Licens avsedd att garantera dig friheten att dela med dig av och modifiera fri mjukvara--att se till att mjukvaran är fri för alla dess användare. Denna Allmänna Publika Licens gäller för det mesta av Free Software Foundations mjukvara och för vilket annat program som helst, vars upphovsman åtar sig att använda den. (Viss annan Free Software Foundation mjukvara täcks av GNUs Biblioteks Allmänna Publika Licens istället.) Du kan tillämpa den på dina program också.

När vi talar om fri mjukvara, avser vi frihet, inte pris. Vår Allmänna Publika Licens är designad för att säkerställa att du har friheten att distribuera kopior av fri mjukvara (och ta betalt för denna service om du så önskar), att du får källkoden eller kan få tag på den om du vill ha den, att du kan modifiera mjukvaran eller använda delar av den i nya fria program; och att du vet att du kan göra dessa saker.

För att skydda dina rättigheter, måste vi införa restriktioner som förbjuder någon att förvägra dig dessa rättigheter eller be dig ge upp rättigheterna. Dessa restriktioner innebär vissa ansvarstaganden från dig om du distribuerar kopior av mjukvaran, eller om du modifierar den.

Till exempel, om du distribuerar kopior av ett sådant program, vare sig du gör det gratis eller mot en avgift, så måste du ge mottagarna alla de rättigheter som du har. Du måste se till att de också får eller kan få tag på källkoden. Och du måste visa dem dessa villkor så att de känner till sina rättigheter.

Vi skyddar dina rättigheter i två steg: (1) upphovsrättsskyddar mjukvaran, och (2) erbjuder dig denna licens som ger dig laglig rätt att kopiera, distribuera och/eller modifiera mjukvaran.

Dessutom, för varje upphovsmans skydd och vårt eget, vill vi försäkra oss om att alla förstår att det inte finns någon garanti för denna fria mjukvara. Om mjukvaran har modifierats av någon annan och skickats vidare, vill vi att dess mottagare ska känna till att de inte har originalet, så att problem som introducerats av andra inte avspeglar sig på de ursprungliga upphovsmännens anseende.

Slutligen, alla fria program hotas ständigt av mjukvarupatent. Vi önskar undvika faran att distributörer av ett fritt program på egen hand skaffar patent som i praktiken gör programmet till deras egendom. För att förhindra detta har vi klargjort att patent som tas antingen måste kunna användas fritt av alla eller inte tas alls.

De exakta bestämmelserna och villkoren för kopiering, distribution och modifikation följer.

BESTÄMMELSER OCH VILLKOR FÖR KOPIERING, DISTRIBUTION OCH MODIFIKATION

0. Denna Licens gäller för godtyckligt program eller annat arbete som innehåller en notis ditsatt av upphovsrättsinnehavaren som säger att det får distribueras enligt bestämmelserna i denna Allmänna Publika Licens. "Programmet" nedan, avser godtyckligt sådant program eller arbete, och ett "arbete baserat på Programmet" betyder antingen Programmet eller något därav härlett arbete under upphovsrättslagen: det vill säga, ett arbete innehållandes Programmet eller en del av det, antingen oförändrat eller med modifieringar och/eller översatt till ett annat språk. (Hädanefter ingår översättning utan begränsning i termen

"modifikation".) Varje licensinnehavare tituleras "du".

Andra aktiviteter än kopiering, distribution och modifikation täcks inte av denna Licens; de ligger utanför dess omfång. Själva handlingen att köra Programmet saknar restriktioner, och utdata från programmet täcks bara om dess innehåll utgörs av ett arbete baserat på programmet (oberoende av att ha skapats genom att köra programmet). Huruvida detta är sant beror på vad programmet gör.

1. Du får kopiera och distribuera oförändrade kopior av Programmets källkod som du fick den, på valfritt medium, förutsatt att du iögonenfallande och korrekt på varje kopia publicerar en korrekt upphovsrättsnotis och avsägelse från garanti; lämnar alla notiser som refererar till denna Licens och avsaknaden av någon garanti intakta; och ger alla andra mottagare av programmet en kopia av denna Licens tillsammans med programmet.

   Du får ta ut en avgift för den fysiska handlingen att överföra kopian och du får om du så vill erbjuda ett garantiskydd i utbyte mot en avgift.

2. Du får modifiera din kopia eller kopior av Programmet eller någon del av det, och på så sätt skapa ett arbete baserat på Programmet, och kopiera och distribuera sådana modifikationer eller arbeten under bestämmelserna i Avsnitt 1 ovan, förutsatt att du också uppfyller alla dessa villkor:

   a. Du måste tillse att de modifierade filerna har iögonenfallande notiser som talar om att du modifierat filerna samt datum för ändringarna.

   b. Du måste tillse att ett arbete som du distribuerar eller publicerar, i sin helhet eller i delar innehållande eller härlett från Programmet eller någon del därav, blir licensierat som helhet utan avgift till alla tredjeparter under villkoren i denna Licens.

   c. Om det modifierade programmet normalt läser kommandon interaktivt då det körs, måste du tillse att det, när det startas och körs för sådan interaktiv användning på det mest vanliga sättet, skriver ut eller visar ett tillkännagivande som innefattar en korrekt upphovsrättsnotis och en notis om att det inte finns någon garanti (eller annars, som säger att du erbjuder garanti) och att användare får distribuera Programmet vidare under dessa villkor, och som talar om för användaren hur en kopia kan få ses av denna Licens. (Undantag: om Programmet självt är interaktivt men inte normalt visar ett sådant tillkännagivande, behöver inte ditt arbete baserat på Programmet visa ett tillkännagivande).

   Dessa krav gäller det modifierade arbetet som helhet. Om identifierbara delar av det arbetet inte är härledda från Programmet, och rimligen kan anses oberoende och separata arbeten i sig själva, så gäller denna Licens och dess bestämmelser för dessa delar när du distribuerar dem som separata arbeten. Men när du distribuerar samma delar som en del av en helhet som är baserad på Programmet, måste distributionen av helheten göras enligt bestämmelserna i denna Licens, vars rättigheter åt andra licensinnehavare utsträcker sig att omfatta hela arbetet, och sålunda till var och en av delarna oavsett vem som skrev den.

   Avsikten med detta avsnitt är sålunda inte att göra anspråk på rättigheter eller bestrida dina rättigheter till arbete skrivet i sin helhet av dig; avsikten är snarare att utöva rätten att styra distributionen av härledda eller samlade arbeten baserade på Programmet.

   Dessutom, att endast lägga ett annat arbete ej baserat på Programmet tillsammans med programmet (eller med ett arbete baserat på Programmet) på en lagringsvolym eller ett distributionsmedium medför inte att detta andra arbete omfattas av denna Licens.

3. Du får kopiera och distribuera Programmet (eller arbete baserat på det enligt Avsnitt 2) som objektkod eller körbar form enligt villkoren i Avsnitt 1 och 2 ovan förutsatt att du också gör något av följande:

   a. Medsänder det med den fullständiga motsvarande maskinläsbara källkoden, som måste distribueras under bestämmelserna i Avsnitt 1 och 2 ovan på ett medium vanligen använt för mjukvaruutbyte; eller,

   b. Medsänder det med ett skriftligt erbjudande, giltigt i minst tre år, att ge godtycklig tredje part, mot en avgift som inte motsvarar mer än din kostnad för att fysiskt genomföra källkodsdistributionen, en komplett maskinläsbar kopia av den motsvarande källkoden, för att distribueras enligt bestämmelserna i Avsnitt 1 och 2 ovan på ett medium vanligen använt för mjukvaruutbyte; eller,

   c. Medsänder det med den information du mottog angående erbjudandet att distribuera motsvarande källkod till dig. (Detta alternativ är tillåtet endast för ickekommersiell distribution och bara om du mottog programmet som objektkod eller i körbar form med ett sådant erbjudande, i överensstämmelse med Underavsnitt b ovan.)

   Med källkoden för ett arbete avses den form av arbetet som föredras för att göra förändringar i det. För ett körbart arbete innebär fullständig källkod all källkod för alla moduler som det innehåller samt alla tillhörande filer med gränssnittsdefinitioner, samt skripten som används för att styra kompilering och installation av den körbara filen. Emellertid, som ett särskilt undantag, behöver den distribuerade källkoden inte inkludera något som normalt distribueras (i antingen källkod eller binär form) med de viktigare delarna (kompilator, kärna, osv.) av operativsystemet på vilket den körbara filen kan köras, såvida inte den delen i sig är medsänd med den körbara filen.

   Om distribution av körbar fil eller objektkod görs genom att erbjuda tillgång att kopiera från en där för avsedd plats, så gäller ett motsvarande erbjudande av tillgång att kopiera källkoden från samma plats som distribution av källkoden, trots att tredje part inte behöver kopiera källkoden tillsammans med objektkoden.

4. Du får inte kopiera, modifiera, licensiera ut, eller distribuera Programmet på annat sätt än de som uttryckligen getts enligt denna Licens. Försök att på annat sätt kopiera, modifiera, licensiera ut eller distribuera Programmet är ogiltigt, och kommer automatiskt frånta dig dina rättigheter enligt denna Licens. Emellertid, parter som har mottagit kopior, eller rättigheter från dig enligt denna Licens fråntas inte sina rättigheter så länge dessa parter följer den.

5. Du behöver inte acceptera denna Licens, eftersom du inte har undertecknat den. Emellertid, inget annat ger dig tillstånd att modifiera eller distribuera Programmet eller dess härledda verk. Dessa handlingar är förbjudna i lag om du inte accepterar denna Licens. Därför, genom att modifiera eller distribuera Programmet (eller något arbete baserat på Programmet), visar du att du accepterar denna Licens för att få göra det, och alla dess bestämmelser och villkor för kopiering, distribution eller förändring av Programmet eller arbeten baserade på det.

6. Varje gång du distribuerar Programmet (eller något arbete baserat på Programmet), mottar mottagaren automatiskt en Licens från den ursprungliga licensutgivaren att kopiera, distribuera eller modifiera Programmet enligt dessa bestämmelser och villkor. Du får inte lägga till några ytterligare restriktioner på mottagarens utövande av de rättigheter som ges häri. Du är inte ansvarig att upprätthålla tredje parts tillmötesgående av denna Licens.

7. Om, som en följd av ett utslag i rätten eller anklagelse om

patentöverträdelse eller av någon annan annan orsak (ej begränsad till patentärenden), villkor ställs på dig (genom domstolsbeslut eller överenskommelse eller annat) som motsäger villkoren i denna Licens, så befriar det dig inte från villkoren i denna Licens. Om du inte kan distribuera så att du samtidigt uppfyller dina plikter enligt denna Licens och alla andra relevanta plikter, så får du som följd därav inte alls distribuera Programmet. Till exempel, om ett patent inte tillåter royalty-fri vidaredistribution av Programmet av alla de som direkt eller indirekt mottar kopior från dig, så är enda sättet du kan uppfylla både det och denna Licens att avstå helt från att distribuera Programmet.

Om någon del av detta avsnitt hålls för ogiltig eller ogenomdrivbar under någon särskild omständighet, är balansen av detta avsnitt avsedd att gälla och avsnittet som helhet avsett att gälla under andra omständigheter.

Det är inte avsikten med detta avsnitt att förorsaka att du överträder några patent eller andra egendomsrättsliga anspråk eller bestrider giltigheten av några sådana anspråk; detta avsnitt har som enda syfte att skydda integriteten för systemet för distribution av fri mjukvara, som realiseras genom allmänna Licenspraktika. Många människor har givit generösa bidrag till det breda omfång av mjukvara distribuerad genom detta system i tilltro till fortsatt tillämpbarhet av detta system; det är upp till upphovsmannen/donatorn att bestämma om han eller hon är villig att distribuera mjukvara genom något annat system och en licensinnehavare kan inte påtvinga detta val.

Detta avsnitt är avsett att göra det alldeles klart vad som tros vara konsekvenserna av resten av denna Licens.

8. Om distribution och/eller användande av Programmet är begränsat till vissa länder antingen på grund av patent eller upphovsrättsskyddade gränssnitt, så får upphovsrättsinnehavaren som släpper Programmet under denna Licens lägga till en uttrycklig geografisk begränsning för distributionen som utesluter dessa länder, så att distribution är tillåten endast i eller mellan länder som inte uteslutits. I sådana fall, innefattar denna Licens begränsningen som om den skrivits in i själva Licensen.

9. Free Software Foundation får publicera reviderade och/eller nya versioner av den Allmänna Publika Licensen då och då. Sådana nya versioner kommer vara i liknande anda som den nuvarande versionen, men kan skilja sig åt detaljer för att inrikta sig på nya problem och angelägenheter.

   Varje version ges ett utmärkande versionsnummer. Om Programmet specificerar ett versionsnummer av denna Licens som gäller för det och "valfri senare version", så har du möjligheten att följa bestämmelserna och villkoren antingen av den versionen eller av valfri senare version publicerad av Free Software Foundation. Om Programmet inte specificerar ett versionsnummer av denna Licens, får du använda valfri version som någonsin publicerats av Free Software Foundation.

10. Om du önskar införliva delar av Programmet i andra fria program vars villkor för distributionen är annorlunda, skriv till upphovsmannen och be om tillstånd. För mjukvara som Free Software Foundation har upphovsrätten till, skriv till Free Software Foundation; vi gör ibland undantag för detta. Vårt beslut kommer vägledas av de två målen att bevara den fria ställningen för alla derivat av vår fria mjukvara och främjandet av delandet och återanvändandet av mjukvara i allmänhet.

INGEN GARANTI

11. EFTERSOM PROGRAMMET ÄR LICENSIERAT UTAN KOSTNAD, FINNS DET INGEN GARANTI FÖR PROGRAMMET, I DEN UTSTRÄCKNING SOM TILLÅTS AV TILLÄMPBAR LAG. FÖRUTOM NÄR ANNAT SÄGS I SKRIFT TILLHANDAHÅLLER UPPHOVSRÄTTSINNEHAVARNA OCH/ELLER ANDRA PARTER PROGRAMMET "SOM DET ÄR" UTAN GARANTI AV NÅGOT SLAG, VARESIG UTTRYCKLIGEN ELLER

ANTYDD, INNEFATTANDE, MEN INTE BEGRÄNSAD TILL DE ANTYDDA GARANTIERNA OM HANDELSBARHET OCH PASSNING FÖR ETT VISST ÄNDAMÅL. HELA RISKEN BETRÄFFANDE KVALITET OCH PRESTANDA FÖR PROGRAMMET LIGGER HOS DIG. SKULLE PROGRAMMET VISA SIG VARA FELAKTIGT, ÄR DET UPP TILL DIG ATT STÅ FÖR KOSTNADEN FÖR ALL NÖDVÄNDIG SERVICE, REPARATION OCH RÄTTNING.

12. I INGET FALL SÅVIDA DET INTE KRÄVS AV TILLÄMPBAR LAG ELLER SKRIFTLIGEN ÖVERENSKOMMET KOMMER NÅGON UPPHOVSRÄTTSINNEHAVARE, ELLER NÅGON ANNAN PART SOM FÅR MODIFIERA OCH ELLER ÅTERDISTRIBUERA PROGRAMMET SÅSOM TILLÅTS OVAN, VARA SKYLDIG DIG FÖR SKADOR, INNEFATTANDE ALLA ALLMÄNNA, SPECIELLA, TILLFÄLLIGA ELLER DÄRAV FÖLJANDE SKADOR SOM UPPSTÅR GENOM ANVÄNDANDET ELLER OFÖRMÅGA ATT ANVÄNDA PROGRAMMET (INNEFATTANDE MEN INTE BEGRÄNSAT TILL FÖRLUST AV DATA ELLER DATA SOM GJORTS INKORREKT ELLER FÖRLUSTER DRABBANDE DIG ELLER TREDJE PART ELLER MISSLYCKANDE FÖR PROGRAMMET ATT FUNGERA TILLSAMMANS MED NÅGOT ANNAT PROGRAM, ÄVEN OM SÅDAN INNEHAVARE ELLER ANNAN PART HAR UNDERRÄTTATS OM MÖJLIGHETEN TILL SÅDANA SKADOR).

SLUT PÅ BESTÄMMELSER OCH VILLKOR

Hur Du Tillämpar Dessa Bestämmelser och Villkor på Dina Nya Program

Om du utvecklar ett nytt program, och du vill att det ska vara till största möjliga nytta för allmänheten, så är bästa sättet att uppnå detta att göra det till fri mjukvara som alla kan vidaredistribuera och modifiera under dessa bestämmelser.

För att göra så, lägg till följande notiser till programmet. Det är säkrast att lägga till dem i början på varje källkodsfil för att så effektivt som möjligt förmedla avsaknaden av garanti; varje fil bör ha åtminstone "upphovsrättsraden" och en hänvisning till var den fullständiga notisen kan återfinnas.

EN RAD FÖR ATT GE PROGRAMMETS NAMN OCH EN IDÉ OM VAD DET GÖR
Upphovsrätt (C) 19YY NAMNET PÅ UPPHOVSMANNEN

Detta program är fri mjukvara; du kan vidaredistribuera det och eller modifiera det under villkoren i GNUs Allmänna Publika Licens som den är publicerad av Free Software Foundation; antingen version 2 av Licensen eller (om du vill) valfri senare version.

Detta program distribueras med förhoppningen att det ska vara användbart, men UTAN NÅGON GARANTI; utan även den antydda garantin om HANDELSBARHET och PASSNING FÖR ETT VISST ÄNDAMÅL. Se GNUs Allmänna Publika Licens för närmare detaljer.

Du bör ha mottagit en kopia av GNUs Allmänna Publika Licens tillsammans med detta program; om inte, skriv till Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Lägg också till information om hur man kontaktar dig via elektronisk och vanlig post.

Om programmet är interaktivt, gör så att det skriver ut en kort notis liknande följande när det startas i interaktivt läge:

Gnomovision version 69, Upphovsrätt (C) 19YY NAMN PÅ UPPHOVSMAN
Gnomovision kommer ABSOLUT UTAN GARANTI; för detaljer knappa ‘visa g’. Detta är fri mjukvara, och du är välkommen att vidaredistribuera det under vissa villkor; knappa ‘visa u’ för detaljer.

De hypotetiska kommandona ‘visa g’ och ‘visa u’ bör visa de lämpliga delarna av GNUs Allmänna Publika Licens. Naturligtvis, får de kommandon du använder kallas något annat än ‘visa g’och ‘visa u’; de kan till och med vara musklick eller menyval--vad som än passar ditt program.

Du bör också få din arbetsgivare (om du arbetar som programmerare) eller din skola, om någon, att underteckna en "avsägelse från upphovsrätt" för programmet, om nödvändigt. Här är ett exempel; byt ut

namnen:

> Yoyodyne AB avsäger sig härmed alla upphovsrättsliga
> intressen i programmet ‘Gnomovision’ (vilket stöter
> på kompilatorer) skrivet av James Hacker.
>
> UNDERSKRIFT AV TY COON, 1 April 1989
> Ty Coon, Direktör Av Verkställande

Denna Allmänna Publika Licens tillåter inte införlivande av ditt program i ägda program. Om ditt program är ett underrutinbibliotek, kan du anse att det är mer användbart att tillåta länkning av ägd programvara till biblioteket. Om detta är vad du vill göra, använd GNUs Allmänna Publika Biblioteks Licens istället för denna Licens.

Bilaga G

Sök vidare information

Detta kapitel innehåller anvisningar till hur du går tillväga för att hitta ytterligare information. Och hur du kan be andra om hjälp då du kört fast. Detta kapitel får också tills vidare anses vara en källförteckning

G.1 Läsvärda dokument

Denna bok saknar än så länge en fullständig källförteckning. Jag har skrivit denna bok mestadels ur huvudet och på min bärbara dator på de mest underliga ställen. Därfår har jag tyvärr inte kontinuerligt gjort källanvisningar i boken.

De böcker och texter jag hänvisar till här har jag av en eller annan anledning läst och uppskattat.

Filesystem Hierarchy Standard – v2.0

Introduktion till GNU/Linux Göran Andersson och Kalle Andersson. En fri bok som jag tycker är väldigt bra. Den tar dock inte upp installationen av Linux, men har ett suveränt avsnitt om C-programmering. Klart läsvärt.

http://www.sslug.dk/gnulinx

G.2 Linux Dokumentation Project

G.2.1 swe-doc

DOS-Win-to-Linux-HOWTO Net-3-HOWTO ISP-Hookup-HOWTO

G.3 ftp-arkiv

G.3.1 Linux kärnan

Allt om linux-kärnan kan man läsa på webben. Det är ju nästan där den föddes. På följande websidor står det mesta om Linux-kärnan.

http://www.linuxhq.com http://www.kernel.org

Vi som bor i Sverige kan med fördel hämta Linux-kärnan från dess “ursprungliga” plats:

ftp://ftp.funet.fi/Linux/PEOPLE/Linus/

G.4 Riktiga källförteckningen

Litteraturförteckning

[Silberschatz och Galvin, 1998] Abraham Silberschatz och Peter Baer Galvin, OPERATING SYSTEM CONCEPTS, Addison Wesley Longman Inc., 1998, ISBN: 0-201-54262-5.

[Tanenbaum, 1996] Andrew S. Tanenbaum, COMPUTER NETWORKS 3:d ed., Prentice-Hall Inc., 1996, ISBN: 0-13-349945-6.

[Thomas och Farrow, 1989] Rebecca Thomas och Rik Farrow, UNIX ADMINISTRATION GUIDE FOR SYSTEM V, Prentice-Hall Inc., 1989, ISBN: 0-13-942889-5

[FSF, 199X] Rebecca Thomas och Rik Farrow, BASH MANUAL, XXXXXXXXX-XXXXXXXXXXXXXXXXXXXXXXX, ISBN: 0-13-942889-5

Bilaga H

Ordlista

A

Arbetsstation En arbetsstation kallas ofta en dator med ett UNIX system vid vilken arbete med annat än underhåll utförs.

B

BSD en variant av UNIX.

C

`cfdisk`

D

`diskdruid`

E

F

`fdisk` Program för att partitionera diskar.

FreeBSD En av de största fria BSD varianterna. Ett fritt opetativsystem. Mycket fina egenskaper.

Frontend Ett program som fungerar som en, ofta grafisk, fasad till ett annat program. Det finns många bra kommandobaserade program till Linux. Till vissa av dem finns frontends som gör att de kan användas genom att klicka på knappar istället för att ge kommandon via tangentbordet.

G

GID Group ID Varje användare tillhör minst en grupp. Varje grupp har ett namn detta är både ett namn och ett nummer. Detta nummer brukar kallas GroupID eller GID

H

I

Inode **I**nformation **node**.

J

K

L

M

N

NetBSD

O

OpenBSD

P

POSIX Den standard som de flesta UNIX system är byggda efter.

prompt Den markkör vid vilken kommandon till skalet ges. Kan se ut på många olika vis. Vanligtvis slutar den på $ eller % om du är vanlig användare och # om du är superanvändaren root

Q

R

root root är systemadninistratören. Den här användaren finns i alla UNIX- och Linuxsystem. Har alltid UID (User ID) noll (0). UID är mer bundet än namnet. Det vill säga användaren kan ha ett annat namn, men alltid UID 0.

rotfönster I X kallas bakgrunden för rotfönstret. Det vill säga det fönster som täcker hela skärmen. Kallas för skrivbordet i Windows-sammanhang. I rotfönstret kan man tilll exempel visa en snygg bakgrundsbild.

S

SGID Set Group ID

`stderr` (Standard Felutmatning) En enhetsfil som normalt är kopplad till den terminal du sitter vid. På denna fil skriver normalt program sina felmeddelanden. Normalt är denna kanal kopplad till samma ställe som `stdout` men det är möjligt att omdirigera dem var för sig för att skilja dem åt.

`stdin` (Standard inmatning) En enhetsfil som normalt är kopplad till tangentbordet. Program som tar sin input från `stdin` kan enkelt användas på höger sida om en så kallad pipe (|).

`stout` (Standard Utmatning) En enhetsfil som normalt är kopplad till den terminal du kör i. Program som ger sina utdata på `stdout` kan med fördel användas på vänster sida om en pipe (|). `stdout` kan enkelt omdirigeras med > operatorn.

skalskript En texfil innehållande kommandon till skalet. Filen kan exekveras som ett program. Vid exekveringen körs kommandona i filen som on de skrevs vid prompten.

Socket

Spegling Spegling (eng. Mirror) är då ett antal filer finns på flera ställen. Man säger då att de är speglade. Vanligtvis är ftp-arkiv är speglade så att de finns på flera olika ftp-servrar. Detta gör att alla inte behöver ladda ned från samma server. Vidare kan man spegla information i servrar eller vanliga datorer mellan olika diskar för att förhindra att data går förlorad vid en eventuell diskkrasch. Det är också vanligt att kataloger i ett nätverk av säkerhetsskäl, eller bekvämlighetsskäl, speglas.

Sticky bit

SUID Set User ID

T

U

UID User ID Varje användare har ett unikt namn. Detta namn är både ett namn och ett nummer. Numret kallas UserID eller bara UID. Användaren root har alltid UID 0.

UNIX

URL *Universal Resource Locator* är ett sätt att identifiera resurser på internet (och andra nätverk). En URL består av olika delar. Vi tittar på ett exempel `http://www.linux.org` Det som står före `://` är vilket protokoll som används (detta skall anges om det är skall vara en riktig url) vid kontakt med den angivna maskinen. Exempel på protokoll kan vara `http://`, `ftp://` eller `gopher://`. Den andra delen är datornamnet i det här fallet www.linux.org. Man kan även ange kataloger på den angivna maskinen. Det görs då efter datornamnet och katalogerna åtskiljs med snedstreck (/).

V, W

Widgets

X

X-klient

X-server

Y

Z

Å

Ä

Ö

Sakregister