From DOS/Windows to Linux HOWTO - Swedish Version

By Guido Gonzato, guido@ibogfs.cineca.it Svensk översättning: Linus Åkerlund, uxm165t@tninet.se
v1.3.0, 15 April 1998. Svensk översättning, 4 juni 1998

Den här HOWTOn är tillägnad alla de (snart före detta?) DOS- och Windows-användare som just har tagit steget och bestämt sig för att byta till Linux, den fria UNIX-klonen. Givet likheterna mellan DOS och UNIX, så är syftet med detta dokument att hjälpa läsaren att föra över sina kunskaper om DOS och Windows till Linux-miljön, för att kunna bli produktiv så fort som möjligt.

Innehåll

1 Inledning

1.1 Är Linux rätt för dig?

Så du vill byta från DOS/Windows till Linux? Bra idé, men akta dig: det kanske inte är användbart för dig. Enligt min åsikt finns det inget sådant som ”den bästa datorn” eller ”det bästa operativsystemet”: det beror på vad man ska göra. Det är därför jag inte tror att Linux är det bästa lösningen för alla, även om det, tekniskt sett, är överlägset många kommersiella operativsystem. Du kommer att tjäna en enorm massa på att använda Linux, om du behöver mjukvara för programmering, Internet, TeX... teknisk mjukvara i allmänhet, men om du mest behöver kommersiell mjukvara, eller om du inte känner för att lära dig att skriva kommandon, leta någon annanstans.

Linux är inte (nu) så enkelt att använda och konfigurera som Windows eller Mac, så förbered dig på att ”hacka” en del. Trots dessa varningar, låt mig påpeka att jag är 100% säker på att om du tillhör den rätta användar-typen så kommer Linux att bli ditt dator-Nirvana. Det är upp till dig. Och kom ihåg att Linux + DOS/Windows kan samexistera på en och samma maskin, i alla fall.

Förutsättningar för den här HOWTOn, jag kommer att anta att

- du känner till de grundläggande DOS-kommandona och -koncepten;

- Linux, eventuellt med X-Window-systemet, är korrekt installerat på din PC;
- ditt skal—motsvarigheten till `COMMAND.COM`—är `bash`.

Notera att, om inte annat anges, så är all information i det här dokumentet riktad mot det ”gamla dåliga DOS”. Det finns ett avsnitt om Windows, men kom ihåg att Windows och Linux är helt och hållet olika, i motsats till DOS, som ett slags sämre släkting till UNIX. Lägg också märke till att det här dokumentet varken är en komplett inledning, och ej heller en konfigurerings-handledning!

1.2 Det är det. Berätta mera

Du installerade Linux och de program du behövde på din PC. Du gav dig själv ett användarkonto (om du inte gjorde det, skriv `adduser` *nu!*) och Linux fungerar. Du har just skrivit in ditt användarnamn och lösenord, och nu tittar du på skärmen och tänker: ”Jaha, vad ska jag göra nu?”

Förtvivla inte. Du är nästan färdig att göra samma saker som du förut gjorde i DOS/Win, och en hel del till. Om du körde DOS/Win istället för Linux, skulle du hålla på med några av följande saker:

- köra program och skapa, kopiera, titta på, ta bort, skriva ut, ändra namn på filer;
- CD-a, MD-a, RD-a och DIR-a dina kataloger;
- formattera disketter och kopiera filer till och från dem;
- skräddarsy systemet;
- skriva .BAT-filer och program i ditt favorit-språk;
- den återstående 1%.

Du kommer bli lättad när du inser att dessa saker kan åstadkommas under Linux, på ett sätt som i mycket påminner om DOS. Under DOS använder den genomsnittlige användaren väldigt få av de mer än hundra tillgängliga kommandona: detsamma gäller, till en del, även för Linux.

1.2.1 Inledande kommentarer

Det bästa sättet att lära sig något är att gå in i det. Du uppmanas starkt att experimentera och leka med Linux: du kan inte skada systemet på det sättet. Några påpekande:

- först, hur du tar dig ur det. För att avsluta Linux: om du ser en textläges-skärm, tryck CTRL-ALT-DEL, vänta på att systemet fixar till sitt innanmäte och upplyser dig om att allt är okej och stäng sedan av din PC. Om du arbetar under X-Window-systemet, tryck först CTRL- ALT-BACKSPACE, sedan CTRL-ALT-DEL. Slå aldrig av eller ”resetta” din PC direkt: det kan skada ditt filsystem;
- i motsats till DOS så har Linux inbyggda säkerhets-mekanismer, p.g.a. sin fleranvändar-natur. Filer och kataloger har rättigheter (permissions), och därför kan vissa av dem inte tillgås av den vanlige användaren; (see avsnittet 3.3 (Rättigheter och ägarskap)). DOS, å andra sidan, låter dig ta bort hela innehållet på hårddisken;
- det finns en speciell användare, som kallas ”root”: system-administratören, med fulla förmågar att ge liv åt eller ta död på maskinen. Om du arbetar på din egen PC, så kommer du vara root också. Att arbeta som root är *farligt*: vilket misstag som helst kan skada eller förstöra systemet, precis som i DOS/Win. Arbeta inte som root om det inte är absolut nödvändigt;

- mycket av komplexiteten hos Linux kommer från dess extrema konfigurabilitet: i stort sett varje funktion och varje applikation kan ställas in genom en eller flera konfigurerings-filer. Komplexitet är priset man betalar för kraft;
- omdirigering och ”piping” är en sido-funktion under DOS, en väldigt viktig sådan, och mycket mera kraftfull, under Linux. Enkla kommandon kan bindas samman för att åstadkomma komplicerade saker. Jag föreslår att du lär dig hur du kan använda dem.

1.2.2 Skaffa hjälp

Det finns många sätt att få hjälp med Linux. De viktigaste är:

- *läs dokumentationen*—jag menar det. Även om HOWTOn du läser kan fungera som en introduktion till Linux, så finns det flera böcker som du verkligen bör läsa: Matt Welsh ”Linux installation and getting started” (`<http://sunsite.unc.edu/mdw/LDP/gs/gs.html>`), Larry Greenfields ”Linux user guide” (`<ftp://sunsite.unc.edu/pub/Linux/docs/linux-doc-project/users-guide>`), och Linux FAQ (`<http://sunsite.unc.edu/mdw/FAQ/Linux-FAQ.html>`). Du kan få ha dåligt samvete tills du åtminstone har läst en av dem;
- dokumentationen till de paket som finns installerade på din maskin finns oftast i underkataloger under `/usr/doc/`;
- för att få en del hjälp om de ”interna kommandona”, skal-kommandona, skriv `help` eller, ännu bättre, `man bash` eller `info bash`;
- för att få hjälp om ett kommando, skriv `man kommando`, vilket plockar fram manualen (”man-sidan”) om `kommando`. Alternativt kan du skriva `info kommando`, vilket anropar, om den finns, info-sidan om `kommando`. Info är ett hypertext-baserat dokumentations-system, kanske inte så intuitivt i början. Slutligen kan du pröva `apropos kommando` eller `whatis kommando`. Med alla de här kommandona kan du trycka ”q” för att avsluta.

1.3 Konventioner

Genom det här verket kommer exemplen att följa detta format: <...> är ett argument som krävs, medan

```
[...]
```

är ett valfritt. Exempel:

```
$ tar -tf <fil.tar> [> omdir_fil]
```

`fil.tar` måste anges, men omdirigering till `omdir_fil` är valfritt.

”LMS” betyder ”var vänlig läs man-sidan för mer information”. Jag kan inte tillräckligt understryka hur viktigt det är att läsa dokumentation.

När prompten i ett kommando-exempel är ”#”, så kan kommandot endast utföras av root.

1.4 Översättarens anmärkningar

Uppdaterade dokumentet 13/11-98, genom att byta ut översättningen av "permissions" till "rättigheter", istället för det sämre "tillåtelser"

2 För de otåliga

Vill du köra igång på direkten? Ta en titt på den här tabellen:

```
DOS                     Linux                   Anmärkningar
------------------------------------------------------------------------------

BACKUP                  tar -Mcvf enhet kat/    helt annorlunda
CD katnamn\             cd katnamn/             nästan samma syntax
COPY fil1 fil2          cp fil1 fil2            dito
DEL fil                 rm fil                  varning - ingen "undelete"
DELTREE katnamn         rm -R katnamn/          dito
DIR                     ls                      inte riktigt samma syntax
DIR fil /S              find . -namn fil        helt annorlunda
EDIT fil                vi fil                  tror inte du kommer gilla det här
                        emacs fil               det här är bättre
                        jstar fil               känns som DOS edit
FORMAT                  fdformat,
                        mount, umount           ganska annorlunda syntax
HELP kommando           man kommando            samma filosofi
MD katnamn              mkdir katnamn/          nästan samma syntax
MOVE fil1 fil2          mv fil1 fil2            dito
NUL                     /dev/null               dito
PRINT fil               lpr fil                 dito
PRN                     /dev/lp0,
                        /dev/lp1                dito
RD katnamn              rmdir katnamn/          nästan samma syntax
REN fil1 fil2           mv fil1 fil2            inte med flera filer
RESTORE                 tar -Mxpvf enhet        annorlunda syntax
TYPE fil                less fil                mycket bättre
WIN                     startx                  enorm skillnad!
```

Om du behöver mer än en tabell med kommandon, fortsätt läsa.

3 Filer och program

3.1 Filer: inledande anmärkningar

Linux har ett filsystem, vilket betyder ”katalog-strukturen och filerna i den”, som liknar DOS väldigt mycket. Filer har filnamn som följer vissa regler, lagras i kataloger, vissa är körbara och vissa av dem har har kommando-parametrar. Vidare kan du använda ”wildcard- tecken”, omdirigering och ”piping”. Det finns bara några små skillnader:

- under DOS är filnamnen i så kallat 8.3-format; t.ex. `NOTENOUG.TXT`. Under Linux har vi det bättre. Om du installerade Linux, med ett fil- system som ext2 eller umsdos, så kan du använda längre filnamn (upp till 255 tecken) och med mer än en punkt i dem: t.ex. `This_is.a.VERY_long.filename`. Observera att jag använde både stora och små bokstäver: faktiskt...
- så är det så att stora och små bokstäver i filnamn är annorlunda. `FILENAME.tar.gz` och `filename.tar.gz` är därför två olika filer. `ls` är ett kommando, `LS` är en felskrivning;

o Windows 95-användare kommer naturligtvis att vilja använda långa fil- namn under Linux. Om ett filnamn innehåller mellanslag (vilket inte rekommenderas, men är möjligt), så måste du innesluta filnamnet i citationstecken när du vill referera till det. T.ex.:

```
$ # följande kommando skapar en katalog som heter "Mina gamla filer"
$ mkdir "Mina gamla filer"
$ ls
Mina gamla filer    bin     tmp
```

Vidare så finns det vissa tecken som inte bör användas: några av dessa är `!*$&`.

- det finns inga obligatoriska filändelser som .COM och .EXE för program, eller .BAT för batch-filer. Körbara filer är markerade med en asterisk,`*` , på slutet av namnet, när du kör kommandot `ls -F`. T.ex.:

  ```
  $ ls -F
  I_am_a_dir/   cindy.jpg    cjpg*   letter_to_Joe    my_1st_script*  old~
  ```

  Filerna `cjpg*` och `my_1st_script*` är körbara; ”program”. Under DOS slutar säkerhetskopiornas namn med .BAK, medan de under Linux slutar med ett tilde, ”`~`”, som gömda filer. Exempel: filen `.I.am.a.hidden.file` dyker inte upp efter `ls`-kommandot;

- parametrar till DOS-program skrivs som `/parameter`, medan Linux använder `-parameter` eller `--parameter`. Exempel: `dir /s` blir `ls -R`. Observera att många DOS-program, som `PKZIP` och `ARJ`, använder UNIX-liknande parametrar.

Du kan nu gå vidare till avsnitten 3.4 (Översätta kommandon från DOS till Linux), men om jag var du skulle jag läsa vidare.

3.2 Symboliska länkar

UNIX har en filtyp som inte existerar under DOS: den symboliska länken. Du kan tänka på den som en pekare till en fil eller en katalog, och den kan användas istället för filen eller katalogen den pekar på; det liknar genvägarna i Windows 95. Exempel på symboliska länkar är `/usr/X11`, vilken pekar på `/usr/X11R6`, och `/dev/modem`, vilken pekar på antingen `/dev/cua0` eller `/dev/cua1`.

För att skapa en symbolisk länk:

```
$ ln -s <fil_eller_katalog> <länknamn>
```

Exampel:

```
$ ln -s /usr/doc/g77/DOC g77manual.txt
```

Nu kan du hänvisa till `g77manual.txt` istället för `/usr/doc/g77/DOS`. Länkar ser ut så här i kataloglistningar:

```
$ ls -F
g77manual.txt@
$ ls -l
(diverse saker...)          g77manual.txt -> /usr/doc/g77/DOC
```

3.3 Rättigheter (permissions) och ägarskap

DOS-filer och -kataloger har följande attribut: A (arkiv), H (dolda), R (endast läsbara) och S (system). Endast H och R finns under Linux: dolda filers namn börjar med en punkt, och för R-attributet, läs vidare.

Under UNIX har en fil ”rättigheter” och en ägare, som i sin tur till- hör en ”grupp”. Titta på det här exemplet:

```
$ ls -l /bin/ls
-rwxr-xr-x  1  root  bin  27281 Aug 15 1995 /bin/ls*
```

Det första fältet innehåller rättigheterna för filen `/bin/ls`, vilken tillhör root, gruppen är bin. Om vi bortser från den återstående informationen, lägg på minnet vad `-rwxr-xr-x` betyder, från vänster till höger:

`-` är filtypen (- = vanlig fil, d = katalog, l = länk osv.); `rwx` är rättigheterna för filens ägare (läs (read), skriv (write) och kör (execute)); `r-x` är rättigheterna för filägarens grupp (läs, kör); (Jag kommer inte att avhandla grupp-begreppet, du kan överleva utan det, så länge du är nybörjare ;-) `r-x` är rättigheterna för alla andra användare (läs, kör).

Katalogen `/bin` har också rättigheter: see avsnitt 4.2 (Katalogers rättigheter) för mer detaljer. Det är p.g.a. detta som du inte kan radera filen `/bin/ls`, om du inte är root: du har inte skriv-rättighet att göra detta. För att ändra en fils rättigheter, använd kommandot:

```
$ chmod <vemXrättighet> <file>
```

där vem är `u` (användare (user), alltså användaren), `g` (grupp), `o` (annan (other)), X är antingen `+` eller `-`, rättighet är `r` (läs), `w` (skriv) eller `x` (kör). Vanliga exempel på `chmod` är de följande:

```
$ chmod +x fil
```

vilket sätter kör-rättighet för filen.

```
$ chmod go-rw fil
```

Det här tar bort läs- och skriv-rättigheterna för alla utom ägaren.

```
$ chmod ugo+rwx fil
```

Detta ger alla läs-, skriv- och kör-rättigheterna.

```
# chmod +s fil
```

det här skapar en så kalla ”setuid”- eller ”suid”-fil—en fil som alla kan köra med dess ägares rättigheter. Dessa kommer du typiskt att stöta på som root-suid-filer.

Ett kortare sätt att referera till rättigheter är med nummer: `rwxr-xr-x` kan uttryckas som 755 (varje bokstav korresponderar till en bit: `---` är 0, `--x` är 1, `-w-` är 2, `-wx` är 3 osv.). Det ser svårt ut, men med en del träning kommer du att förstå konceptet.

root, som super-användare, kan ändra allas filers rättigheter. LMS.

3.4 Filer: Översättning av kommandon från DOS till Linux

Till vänster finns DOS-kommandona; till höger finns deras motsvarigheter under Linux.

```
ATTRIB:         chmod
COPY:           cp
DEL:            rm
MOVE:           mv
REN:            mv
TYPE:           more, less, cat
```

Omdirigerings- och rörlednings-operatorer: < > >> |

”Wildcards”: `* ?`

`nul: /dev/null`

`prn, lpt1: /dev/lp0 or /dev/lp1; lpr`

3.4.1 Exempel

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>ATTRIB +R FILE.TXT             $ chmod 400 file.txt
C:\GUIDO>COPY JOE.TXT JOE.DOC           $ cp joe.txt joe.doc
C:\GUIDO>COPY *.* TOTAL                 $ cat * > total
C:\GUIDO>COPY FRACTALS.DOC PRN          $ lpr fractals.doc
C:\GUIDO>DEL TEMP                       $ rm temp
C:\GUIDO>DEL *.BAK                      $ rm *~
C:\GUIDO>MOVE PAPER.TXT TMP\            $ mv paper.txt tmp/
C:\GUIDO>REN PAPER.TXT PAPER.ASC        $ mv paper.txt paper.asc
C:\GUIDO>PRINT LETTER.TXT               $ lpr letter.txt
C:\GUIDO>TYPE LETTER.TXT                $ more letter.txt
C:\GUIDO>TYPE LETTER.TXT                $ less letter.txt
C:\GUIDO>TYPE LETTER.TXT > NUL          $ cat letter.txt > /dev/null
        n/a                             $ more *.txt *.asc
        n/a                             $ cat section*.txt | less
```

Notes:

- `*` är smartare under Linux: `*` matchar alla filer, utom de dolda; `.*` matchar alla dolda filer (men också den aktuella katalogen, `.`, och föräldra-katalogen, `..`: varning!); `*.*` matchar endast de som har en punkt i mitten, följt av andra tecken; `p*r` matchar både ”peter” och ”piper”; `*c*` matchar både ”picked” och ”peck”;
- när du använder `more`, tryck <SPACE> för att bläddra genom filen, ”q” för att avsluta. `less` är mer intuitivt och låter dig använda pil-tangenterna.
- det finns ingen `UNDELETE`, så *tänk dig för* innan du tar bort något;

- utöver DOS < > >>, så har Linux `2>` för att omdirigera felmeddelanden (stderr); vidare så omdirigerar `2>&1` stderr till stdout, medan `1>:&2` omdirigerar stdout till stderr;
- Linux har ett annat "wildcard":

  `[]`

  . Användning:

  `[abc]*`

  matchar filer som börjar med a, b och c;

  `*[I-N1-3]`

  matchar filer som slutar med I, J, K, L, M, N, 1, 2 och 3;
- `lpr` <fil> skriver ut en fil i bakgrunden. För att kolla statusen på utskriftskön, använd `lpq`; för att ta bort en fil från utskriftskön, använd `lprm`;
- det finns ingen DOS-liknande `RENAME`; `mv` *.xxx *.yyy fungerar alltså inte. Du skulle kunna pröva följande enkla skal-program; se avsnittet 8.1 (Skal-program: .BAT-filer på anabola) för detaljer.

```
#!/bin/sh
# ren: byt namn på flera filer enligt flera regler

if [ $# -lt 3 ] ; then
  echo "användning: ren \"mönster\" \"ersättning\" filer..."
  exit 1
fi

OLD=$1 ; NEW=$2 ; shift ; shift

for file in $*
do
  new=`echo ${file} | sed s/${OLD}/${NEW}/g`
  mv ${file} $new
done
```

  Varning: det beter sig inte som DOS `REN`, eftersom det använder reguljära mönster (regular expressions), som du fortfarande inte vet något om. Om du bara vill byta ut filändelser, använd det så här: `ren "htm$" "html" *htm`. Glöm inte `$`-tecknet.

- använd `cp -i` och `mv -i` för att få en varning när en fil kommer att bli överskriven.

3.5 Köra program: multitasking och sessioner

För att köra ett program, skriv in dess namn, som du skulle göra under DOS. Om katalogen (avsnittet 4 (Kataloger)) där programmet är lagrat finns med i din sökväg (avsnittet 7.1 (System-initiering)), så kommer programmet att startas. Undantag: i motsats till DOS, så startas inte ett program i den aktuella katalogen, under Linux, om den katalogen inte finns med i sökvägen. Lösning: om `prog` är ditt program, skriv `./prog`.

Så här ser en typisk kommando-rad ut:

```
$ command [-s1 [-s2] ... [-sn]] [par1 [par2] ... [parn]] [< input] [> output]
```

där `-s1`, ..., `-sn` är program-”switchar”, `par1`, ..., `parn` är program- parametrar. Du kan ge flera kommandon på en rad:

```
$ kommando1 ; kommando2 ; ... ; kommandon
```

Det var allt för denna gång om att köra program, men det är lätt att gå bortom detta. En av huvudanledningarna till att använda Linux är att det är ett multi-taskande operativsystem: det kan köra flera program (från och med nu: processer) samtidigt. Du kan köra igång processer i bakgrunden och fortsätta arbeta med en gång. Vidare låter dig Linux ha flera sessioner: det är som att ha flera datorer som arbetar samtidigt!

- För att byta till session 1..6 på de virtuella terminalerna, tryck <ALT-F1> ... <ALT-F6>
- För att starta en ny session i samma virtuell terminal, utan att lämna den nuvarande, skriv `su - <loginnamn>`. Exempel: `su - root`. Detta är användbart t.ex. när du behöver utföra något, som bara root kan göra.
- För att avsluta en session, skriv `exit`. Om det finns stoppade jobb (mer om det senare), så blir du varnad.
- För att starta en process i bakgrunden, lägg till en ampersand ”`&`” på slutet av kommando-raden:

  ```
  $ prognamn [-switchar] [parametrar] [< indata] [> utdata] &
  [1] 123
  ```

  skalet identifierar processen med ett jobb-nummer (t.ex.

  ```
  [1]
  ```

  ; se nedan), och med en PID (Process identification Number; 123 i vårt exempel).
- För att se hur många processer som körs, skriv `ps -ax`. Detta kommer att visa en lista på den aktuella processerna.
- För att döda en process, skriv `kill <PID>`. Du kanske behöver döda en process, om du inte vet hur man ska stänga den på rätt sätt... Om du är root, så kan du döda andras processer. Ibland kan en process endast dödas med `kill -SIGKILL <PID>`.

  Utöver detta låter dig skalet stoppa eller tillfälligt pausa en process, skicka en process till bakgrunden och hämta fram en process från bakgrunden till förgrunden. I denna kontext kallas processer för ”jobb”.
- För att se hur många jobb som finns, tryck `jobs`. Här identifieras jobben med sina jobb-nummer, inte sina PID.
- För att stoppa en process, som körs i förgrunden, tryck <CTRL-C> (det fungerar inte alltid).
- För att pausa en process som körs i förgrunden, tryck <CTRL-Z> (dito).
- För att skicka en pausad process till bakgrunden, skriv `bg <jobb>` (den blir ett jobb).
- För att plocka fram ett jobb till förgrunden, skriv `fg <jobbgt;/`. För att förgrunda det senast bakgrundade (är mitt språk underligt?) jobbet, skriv bara `fg`.
- För att döda ett jobb, skriv `kill <%jobb>`, där <jobb> kan vara 1, 2, 3...

Om du använder dessa kommandon kan du formattera en disk, zippa en bunt filer, kompilera ett program och packa upp ett arkiv, allt på samma gång, och fortfarande ha prompten tillgänglig. Försök med det under DOS! Och försök med Windows, bara för att se skillnaden i prestanda (om det inte krashar, naturligtvis).

3.6 Köra program på avlägsna datorer

För att köra ett program på en avlägsen maskin, vars IP-adress är `remote.machine.edu`, skriver du:

```
$ telnet remote.machine.edu
```

Efter att du loggat in är det bara att starta ditt favoritprogram. Jag behöver naturligtvis inte ens nämna att du måste ha ett användar- konto på den andra maskinen.

Om du har X11 så kan du till och med köra X-applikationer på en avlägsen dator, och visa det på din X-skärm. Säg att `remote.machine.edu` är den avlägsna X-datorn och `local.linux.box` är din Linux-maskin. För att, från `local.linux.box`, köra ett X-program, som finns på `remote.machine.edu`, kan du göra följande:

- kör igång X11, starta en `xterm` eller annan terminal-emulator, och skriv sedan:

  ```
  $ xhost +remote.machine.edu
  $ telnet remote.machine.edu
  ```

- efter att du loggat in, skriv:

  ```
  remote:$ DISPLAY=local.linux.box:0.0
  remote:$ prognamn &
  ```

  (istället för `DISPLAY...`, kan du bli tvungen att skriva: `setenv DISPLAY local.linux.box:0.0`. Det beror på den andra maskinens skal.)

Och voila! Nu kommer `prognamn` startas på `remote.machine.edu` och visas på din maskin. Försök dock inte med det här över en PPP-lina, det är alldeles för långsamt för att vara användbart.

4 Använda kataloger

4.1 Kataloger: inledande anmärkningar

Vi har sett skillnaderna mellan filer under DOS och Linux. Vad gäller kataloger, så heter rot-katalogen \ under DOS, under Linux heter den `/`. På samma sätt är nästade kataloger åtskilda med \ under DOS, och med `/` under Linux. Exempel på fil-sökvägar:

```
DOS:    C:\PAPERS\GEOLOGY\MID_EOC.TEX
Linux:  /home/guido/papers/geology/middle_eocene.tex
```

Som vanligt är `..` föräldra-katalogen (parent directory) och `.` är den aktuella katalogen. Kom ihåg att systemet inte låter dig köra `cd`, `rd` eller `md` överallt, där du vill. Varje användare börjar i sin hemkatalog, t.ex. `/home/guido`.

4.2 Katalog-rättigheter

Kataloger har också rättigheter. Det vi lärt oss i avsnittet 3.3 (Rättigheter och ägarskap) gäller även för kataloger (användare, grupp och andra). För en katalog innebär `rx` att du kan `cd`-a till den katalogen, och `w` innebär att du kan ta bort filer i den (beroende på filens rättigheter också, naturligtvis), eller katalogen själv.

Till exempel, för att förhindra andra användare från att tjuvkika i `/home/guido/text`:

```
$ chmod o-rwx /home/guido/text
```

4.3 Kataloger: översätta kommandon från DOS till Linux

```
DIR:            ls, find, du
CD:             cd, pwd
MD:             mkdir
RD:             rmdir
DELTREE:        rm -R
MOVE:           mv
```

4.3.1 Exempel

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>DIR                            $ ls
C:\GUIDO>DIR FILE.TXT                   $ ls file.txt
C:\GUIDO>DIR *.H *.C                    $ ls *.h *.c
C:\GUIDO>DIR/P                          $ ls | more
C:\GUIDO>DIR/A                          $ ls -l
C:\GUIDO>DIR *.TMP /S                   $ find / -name "*.tmp"
C:\GUIDO>CD                             $ pwd
       n/a - se not                     $ cd
       dito                             $ cd ~
       dito                             $ cd ~/temp
C:\GUIDO>CD \OTHER                      $ cd /other
C:\GUIDO>CD ..\TEMP\TRASH               $ cd ../temp/trash
C:\GUIDO>MD NEWPROGS                    $ mkdir newprogs
C:\GUIDO>MOVE PROG ..                   $ mv prog ..
C:\GUIDO>MD \PROGS\TURBO                $ mkdir /progs/turbo
C:\GUIDO>DELTREE TEMP\TRASH             $ rm -R temp/trash
C:\GUIDO>RD NEWPROGS                    $ rmdir newprogs
C:\GUIDO>RD \PROGS\TURBO                $ rmdir /progs/turbo
```

Noter:

1. När du använder `rmdir` så måste katalogen som ska tas bort vara tom. För att ta bort en katalog och allt dess innehåll, använd `rm -R` (på egen risk).

2. Tecknet "~" är en genväg för namnet på din hemkatalog. Kommandona `cd` och `cd ~` tar dig båda till din hemkatalog, från den katalog där du för tillfället är; kommandot `cd ~/tmp` tar dig till `/home/ditt_hem/tmp`.

3. `cd -` "tar tillbaks" det senaste `cd`.

5 Floppy-diskar, hårddiskar och liknande

Det finns två sätt att administrera enheter under Linux: DOS-sättet coh UNIX-sättet. Välj själv.

5.1 Administrera enheter på DOS-sättet

Det flesta Linux-distributioner kommer med Mtools-paketet, en uppsättning kommandon som fungerar precis som sina motsvarigheter under DOS, men börjar med ett "m": alltså `mformat`, `mdir`, `mdel`, `mmd` osv. De kan till och med bevara långa filnamn, men inte fil-rättigheter. Om du konfigurerar Mtools, genom att editera en fil som heter `/etc/mtools.conf` (ett exempel ges), så kan du komma åt DOS/Win-partitioner, CD-ROM-spelaren och Zip-driven.

För att formattera en tomdisk, så funkar dock inte `mformat`-kommandot. Du måste, som root, köra följande kommando först:

```
# fdformat /dev/fd0H1440
```

Observera att du inte kan komma åt filerna på en diskett med ett kommando som, t.ex. `less a:file.txt`! Detta är en nackdel med DOS-sättet att montera disketter.

5.2 Administrera enhter på UNIX-sättet

UNIX har ett annat sätt att administrera enheter än DOS/Win. Det finns inga separata volymer som A: eller C:; en disk, vare sig det är en diskett eller något annat, blir en del av det lokala filsystemet, efter en operation som kallas "montering". När du är klar med den diskett, så måste du "avmontera" den, innan du tar ut den.

Fysisk formattering av en diskett är en sak, och att skapa ett filsystem är en annan. DOS-kommandot `FORMAT A:` gör båda sakerna, men under Linux är de separata kommandon. För att formattera en diskett, se ovan; för att skapa ett filsystem, gör följande:

```
# mkfs -t ext2 -c /dev/fd0H1440
```

Du kan använda `minix`, `vfat`, `dos` eller andra format, istället för `ext2`. Så fort disken är preparerad, montera den med kommandot

```
# mount -t ext2 /dev/fd0 /mnt
```

och specificera det korrekta filsystemet, om du inte använder `ext2`. Nu kan du komma åt filerna på disketten. Allt du brukade göra med A: eller B: kan du nu göra, fast med `/mnt` istället. Exempel:

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>DIR A:                         $ ls /mnt
C:\GUIDO>COPY A:*.*                     $ cp /mnt/* .
C:\GUIDO>COPY *.ZIP A:                  $ cp *.zip /mnt
C:\GUIDO>EDIT A:FILE.TXT                $ jstar /mnt/file.txt
C:\GUIDO>A:                             $ cd /mnt
A:>_                                    /mnt/$ _
```

När du är klar, så *måste* du avmontera disketten, innan du tar ut den. Använd kommandot

```
# umount /mnt
```

Det är naturligtvis så att du bara behöver köra `fdformat` och `mkfs` på oformatterade disketter, inte sådana du redan har använt. Om du vill använda enhet B:, hänvisa till den som `fd1H1440` och `fd1`, istället för `fd0H1440` och `fd0`, i exemplen ovan.

Jag behöver egentligen inte nämna det, men det som är tillämpligt på disketter, är det även på andra enheter; om du t.ex. vill montera en till hårddisk eller en CD-ROM-spelare. Så här monterar du en CD-ROM:

```
# mount -t iso9660 /dev/cdrom /mnt
```

Det där var det ”officiella” sättet att montera diskar, men det finns ett litet knep vi kan ta till. Eftersom det är något av ett irritations- moment att behöva vara root för att montera en floppy-disk eller CD-ROM, så kan varje användare tillåtas att montera dem, på följande sätt:

- som root, gör följande:

```
# mkdir /mnt/a: ; mkdir /mnt/a ; mkdir /mnt/cdrom
# chmod 777 /mnt/a* /mnt/cd*
# # se till så att CD-ROM-enheten är den rätta
# chmod 666 /dev/hdb ; chmod 666 /dev/fd*
```

- lägg de följande raderna till `/etc/fstab`:

```
/dev/cdrom      /mnt/cdrom  iso9660 ro,user,noauto          0       0
/dev/fd0        /mnt/a:     msdos   user,noauto             0       0
/dev/fd0        /mnt/a      ext2    user,noauto             0       0
```

För att montera en DOS-diskett, en ext2-diskett och en CD-ROM, kan du nu skriva:

```
$ mount /mnt/a:
$ mount /mnt/a
$ mount /mnt/cdrom
```

`/mnt/a`, `/mnt/a:`, och `/mnt/cdrom` kan nu användas av alla användare. Kom ihåg att det är en stor säkerhetsrisk att låta vem som helst montera enheter, om det nu är något du bryr dig om.

Två användbara kommandon är `df`, vilket ger information om de monterade filsystemen, och `du katnamn`, vilket rapporterar hur mycket ytrymme som tas upp av varje katalog.

5.3 Ta säkerhets-kopior

Det finns flera paket som kan hjälpa dig med detta, men det absolut minsta du kan göra för att ta en säkerhetskopia på flera volymer är (som root):

```
# tar -M -cvf /dev/fd0H1440 kat_att_kopiera/
```

Se till att du har en formatterad diskett i stationen och flera andra redo. För att få tillbaks grejerna, sätt i den första disketten i stationen och skriv:

```
# tar -M -xpvf /dev/fd0H1440
```

6 Windows då?

”Motsvarigheten” till Windows är det grafiska systemet X11. I motsats till Windows och Mac, så skapades inte X11 för att vara enkelt att använda eller för att se snyggt ut, utan bara för att erbjuda grafiska möjligheter på UNIX arbetsstationer. Det finns några huvudsakliga skillnader:

- Medan Windows ser ut och känns på samma sätt över hela världen, så gör inte X11 det: det finns mycket mer du kan ställa in i det. X11s utseende och känsla kommer av en huvudkomponent som kallas ”fönster- hanterare”; det finns många att välja mellan. Den vanligaste är `fvwm`, grundläggande men trevlig och minnessnål, `fvwm2-95` och `The Next Level`, plus flera andra. Fönster-hanteraren anropas vanligtvis från en bil som heter `.xinitrc`;
- din fönster-hanterare kan konfigureras så att ett fönster beter sig som i, öh, Windows: du klickar på det och det hamnar i förgrunden. En annan möjlighet är att det hamnar i förgrunden när muspekaren befinner sig i det (”fokus”). Placeringen av fönster på skärmen kan också vara antingen automatisk eller interaktiv: om en konstig ram dyker upp, istället för ditt program, vänster-klicka där du vill att det ska dyka upp;
- de flesta saker kan skräddassys, genom att modifiera en eller flera konfigurerings-filer. Läs dokumentation om din fönster-hanterare: konfigureringsfilen kan vara `.fvwmrc`, `.fvwm2rc95`, `.steprc` osv. Ett exempel på en konfigureringsfil hittar du i typfallet i `/etc/X11/fönster-hanterar-namnet/system.fönster-hantera-namnet`;
- X-applikationer är skrivna med hjälp av speciella bibliotek (”widget sets”); eftersom flera olika är tillgängliga, så ser applikationerna olika ut. Det mest grundläggande av dessa är Athena widgets (2-D-utseende; `xdvi`, `xman`, `xcalc`); andra använder Motif (`netscape`), ytterligare andra använder Tcl/Tk, XForms, Qt och allt vad det nu är. Vissa, inte alla, av dessa bibliotek ger ungefär samma utseende och känsla som Windows;
- hm, inte riktigt. Känslan kan, olyckligtvis, vara inkonsekvent. Om du t.ex. markerar en textrad, genom att använda musen, och trycker <BACKSPACE>, så räknar du med att raden ska försvinna, eller hur? Så fungerar det inte med Athena–baserade applikationer, men det gör det med Motif, Qt, Gtk och Tcl/Tk;
- hur scrollnings-listerna och storleksförändringarna fungerar beror på förnster-hanterare och ”widget set”. Tips: om du upptäcker att scrollnings-listerna inte uppträder som du förväntat dig, prova att använda den mittersta mus-knappen, eller de två knapparna tillsammans, för att flytta dem;
- applikationer har inte en ikon som standard, men de kan ha många. De flesta fönster-hanterar har en meny, som du kommer åt genom att tryck på bakgrunden (”rot-fönstret”); denna meny kan naturligtvis skräddarsys. För att byta ut rot-fönstrets utseende kan du använda `xsetroot` eller `xloadimage`;
- urklippsarean kan endast innehålla text, och uppträder konstigt. Så fort du har markerat text, så finns den i urklippsarean: flytta någon annanstans och tryck ned mittenknappen, för att klistra in den. Det finns ett program, `xclipboard`, som ger dig flera urklipps-buffrar;
- dra och släpp är ett alternativ, och är bara tillgängligt om du använder X11-applikationer som stödjer det.

För att spara minne ska man använda applikationer som använder samma bibliotek, men det kan vara svårt att göra detta i praktiken.

K Desktop Enviroment är ett projekt som siktar på att få X11 att se ut och uppföra sig konsekvent, som Windows; det är fortfarande på tidig beta-nivå, men tro mig, det är suveränt. Rikta din webb-läsare mot `<http://www.kde.org>`.

7 Skräddarsy systemet

7.1 Systemets initierings-filer

Två viktiga filer under DOS är `AUTOEXEC.BAT` och `CONFIG.SYS`, vilka används för att initiera systemet, ange några miljö-variabler, såsom PATH och FILES, och eventuellt köra igång några program eller batch- filer, då du startar systemet. Under Linux finns det ett flertal initierings-filer, av vilka vissa är sådana som du nog inte ska ge dig på, innan du vet exakt vad du sysslar med. Jag ska berätta vilka de viktigaste är, i alla fall:

```
FILES                                 NOTES

/etc/inittab                          rör inte den nu!
/etc/rc.d/*                           dito
```

Om allt du behöver göra är att ange

`$$PATH`

`[sökväg]`

och andra miljö- variabler, eller om du vill byta ut login-meddelandet, eller automatiskt köra igång ett program efter inloggningen, kan du ta en titt på följande filer:

```
FILER                                 ANMÄRKNINGAR

/etc/issue                            anger pre-login-meddelandet
/etc/motd                             anger post-login-meddelandet
/etc/profile                          anger $PATH och andra variabler osv.
/etc/bashrc                           anger alias och funktioner osv.
/home/your_home/.bashrc               anger dina alias + funktioner
/home/your_home/.bash_profile   eller
/home/your_home/.profile              anger miljö + startar dina program
```

Om de senare filerna existerar (observera att de är dolda filer), så kommer de att läsas in efter att du loggat in, och kommandona i dem kommer att utföras.

Exempel—titta på denna `.bash_profile`:

```
# Jag är en kommentar
echo Miljö:
printenv | less   # ekvivalent med SET-kommandot under DOS
alias d='ls -l'   # lätt att förstå vad ett alias är
alias up='cd ..'
echo "Påminner dig om att sökvägen är "$PATH
echo "Idag är det `date`"  # använder utdatan från kommandot 'date'
echo "Ha det så trevligt, "$LOGNAME
# Det följande är en "skal-funktion"
ctgz() # Lista innehållet i ett .tar.gz-arkiv
{
  for file in $*
  do
    gzip -dc ${file} | tar tf -
```

```
  done
}
# slut på .profile
```

`$PATH` och `$LOGNAME` är, ja, du gissade rätt, miljö-variabler. Det finns många andra att leka med; LMS för applikationer som `less` och `bash`.

7.2 Programs initierings-filer

Under Linux kan i stort sett allt skräddarsys, så att det passar dig. De flesta program har en eller flera initierings-filer som du kan fippla med, ofta i stil med `.programnamnrc`, i din hemkatalog. De första du kommer vilja modifiera är:

- `.inputrc`: används av `bash` för att definiera tangent-bindningar;
- `.xinitrc`: används av `startx` för att initialisera X-Window-systemet;
- `.fvwmrc`: används av fönster-hanteraren `fvwm`.
- `.joerc`: används av editorn `joe`;
- `.jedrc`: används av editorn `jed`;
- `.pinerc`: används av e-postprogrammet `pine`;
- `.Xdefault`: används av många X-program.

För alla dessa och andra du stöter på senare, LMS. Som en avslutande anmärkning, låt mig rekommendera att du tar en titt på Configuration HOWTO på `<http://sunsite.unc.edu/mdw/HOWTO/Config-HOWTO.html>` .

8 Lite programmering

8.1 Skal-program: .BAT-filer på anabola

Om du använde .BAT-filer för att skapa genvägar till långa kommando- rader (jag gjorde det en massa), så kan du göra detta genom att ange lämpliga alias-rader (se exempel ovan) i `profile` eller `.profile`. Men om dina .BAT-filer var mera komplicerade så kommer du älska skal- programmerings-språken som finns i varje skal: de är kraftfulla som QBasic, om inte kraftfullare. De har variabler, konstruktioner som while, for, case, if... then... else och en massa andra egenskaper: de kan vara bra alternativ till ”riktiga” programmerings-språk.

För att skriva ett skal-program, motsvarigheten till .BAT-filer under DOS, så är allt du behöver göra att skriva en vanlig ASCII-fil, som innehåller instruktioner, spara den och göra den körbar, med kommandot `chmod +x <skal-program-fil>`. För att köra den, skriv dess namn.

Ett varningens ord är här på sin plats. System-editorn heter `vi`, och enligt mina erfarenheter så finner de flesta nybörjare den väldigt svår att använda. Jag tänker inte förklara hur du ska använda den, för jag gillar den inte och använder den inte, så nu vet du det. Det räcker med att jag säger följande här:

- för att infoga text, skriv ”i”, sen texten;
- för att ta bort tecken, skriv <ESC>; sen ”x”;

- för att avsluta `vi` utan att spara, skriva <ESC>; sen `:q!`
- för att spara och avsluta, skriv <ESC>, sen `:wq`.

En bra editor för nybörjare är `joe`: anropar du den genom att skriva `jstar`, så får du samma tangentkombinationer som DOS-editorn. `jed` i WordStar- eller IDE-läge är annu bättre. Se avsnittet 9.4 (Var du får tag på program) för att få reda på var du kan få tag på dessa editorer.

Att skriva skal-program under `bash` är ett så stort ämne att det tar en bok själv, och jag tänker inte gå djupare in på ämnet. Jag ger bara ett exempel på ett skal-program, från vilket du kan lära dig vissa grundläggande regler:

```
#!/bin/sh
# sample.sh
# jag är en kommentar
# ändra inte första raden; den måste vara där
echo "Systemet är: `uname -a`" # använd utdatan från kommandot
echo "Mitt namn är $0" # inbyggda variabler
echo "Du gav mig följande $# parametrar: "$*
echo "Den första parametern är: "$1
echo -n "Vad heter du? " ; read ditt_namn
echo observera skillnaden: "hej $ditt_namn" # citerar med "
echo observera skillnaden: 'hej $ditt_namn' # citerar med '
DIRS=0 ; FILES=0
for file in `ls .` ; do
  if [ -d ${file} ] ; then # om filen är en katalog
    DIRS=`expr $DIRS + 1`  # DIRS = DIRS + 1
  elif [ -f ${file} ] ; then
    FILES=`expr $FILES + 1`
  fi
  case ${file} in
    *.gif|*jpg) echo "${file}: grafik-fil" ;;
    *.txt|*.tex) echo "${file}: text-fil" ;;
    *.c|*.f|*.for) echo "${file}: källkods-fil" ;;
    *) echo "${file}: vanlig fil" ;;
  esac
done
echo "det finns ${DIRS} kataloger och ${FILES} filer"
ls | grep "ZxY--!!!WKW"
if [ $? != 0 ] ; then # sista kommandots exit-kod
  echo "ZxY--!!!WKW inte funnen"
fi
echo "det räcker... skriv 'man bash' för mer info."
```

8.2 C själv

Under UNIX är C system-språket, vare sig du gillar det eller ej. Massor av andra språk (FORTRAN, Pascal, Lisp, Basic, Perl, awk...) finns också tillgängliga.

Om vi förutsätter att du kan lite om C, så kommer här några regler för de av er som har blivit bortskämda med Turbo C++ eller någon av dess släktingar under DOS. Linux C-kompilator heter `gcc` och saknar alla de "flashiga" egenskaper som vanligtvis finns med i DOS motsvarigheter: ingen IDE, on-line hjälp, integrerad avlusare osv. Det är bara en rå kommando-rads-kompilator, som är väldigt kraftfull och effektiv. För att kompilera det gamla goda `hello.c`, skriver du:

```
$ gcc hello.c
```

vilket skapar en körbar fil som heter `a.out`. För att ge det körbara programmet ett annat namn skriver vi

```
$ gcc -o hola hello.c
```

För att länka in ett bibliotek i programmet, lägg till switchen -l<biblioteksnamn>. T.ex., för att länk in matematik-biblioteket, skriv:

```
$ gcc -o mathprog mathprog.c -lm
```

(`--l<biblioteksnamn>`-switchen tvingar `gcc` att länka in biblioteket `/usr/lib/lib<bibltioteksnamn>.a`, så `-lm` änkar in `/usr/lib/libm.a`).

Så långt är allt gott. Men när du har ett program som består av flera källkods-filer, så kommer du att behöva använda `make`-verktyget. Låt oss förutsätta att du har skrivit en uttrycks-tolkare: dess källkods-fil heter `tolkare.c` och #inkluderar två "header"-filer, `tolkare.h` och `xy.h`. Sen vill du använda rutinen `tolkare.c` i ett program, säg `kalk.c`, vilket i sin tur #inkluderar `tolkare.h`. Vilken röra! Vad måste du nu göra för att kompilera `kalk.c`?

Du blir tvungen att skriva en så kallad `makefile`, vilken talar om för kompilatorn vilka "beroenden" som finns mellan källkod och objekt-filer. I vårt exempel:

```
# Det här är en makefile, vilken används för att kompilera kalk.c
# Använd TAB-tangenten där det är lämpligt!

kalk:     kalk.o tolkare.o
<TAB>gcc -o kalk kalk.o tolkare.o -lm
# kalk beror av två objekt-filer: kalk.o och tolkare.o

kalk.o:   kalk.c tolkare.h
<TAB>gcc -c kalk.c
# kalk.o beror av två källkods-filer

tolkare.o:   tolkare.c tolkare.h xy.h
<TAB>gcc -c tolkare.c
# tolkare.o beror av tre källkods-filer

# slut på makefile.
```

Spara denna fil som `Makefile` och skriv `make` för att kompilera ditt program; som alternativ kan du spara den som `kalk.mak` och skriva `make -f kalk.mak`, och naturligtvis ska du LMS. Du kan anropa hjälp om C-funktionerna, som avhandlas av man-sidorna, sektion 3, t.ex.:

```
$ man 3 printf
```

För att avlusa dina program, använd `gdb`. `info gdb` lär dig hur du ska använda det.

Det finns en massa bibliotek tillgängliga där ute; bland de första du kommer vilja använda finns `ncurses`, som tar hand om olika effekter i text-läge, och `svgalib`, som kan fixa grafik. Om du känner dig till- räckligt tuff för att tackla X-programmering, finns det bibliotek som de ovan nämnda XForms, Qt, Gtk och många andra, med vilka du lätt kan skriva X11-program. Ta en titt på <http://www.xnet.com/~blatura/linapp6.html> .

Många editorer kan fungera som ett IDE; `emacs` och `jed`, t.ex., och även ge sådana funktioner som syntax-markering, automatisk indentering osv. Alternativet är att skaffa `rhide`-paketet från <ftp://sunsite.unc.edu:/pub/Linux/devel/debuggers/>. Det är en kopia av Borlands IDE, och det finns vissa möjligheter att du kommer att gilla det.

9 Den återstående 1%

Mer än 1%, faktiskt...

9.1 Använda tar & gzip

Under UNIX finns det vissa, väldigt ofta använda, applikationer för att arkivera och komprimera filer. `tar` används för att arkivera: det är som `PKZIP`, men utför ingen komprimering, det bara arkiverar. För att skapa ett nytt arkiv, skriv:

```
$ tar -cvf <arkiv_namn.tar> <fil> [fil...]
```

För att packa upp filer från ett arkiv:

```
$ tar -xpvf <arkiv_namn.tar> [fil...]
```

För att lista innehållet i ett arkiv:

```
$ tar -tf <arkiv_namn.tar> | less
```

Du kan komprimera filer med `compress`, men det är gammalt och bör inte användas längre, eller `gzip`:

```
$ compress <fil>
$ gzip <fil>
```

som skapar en komprimerad fil med ändelsen .> (`compress`) eller .gz (`gzip`). Dessa program kan bara komprimera en fil i taget. För att packa upp, skriv:

```
$ compress -d <fil.Z>
$ gzip -d <fil.gz>
```

LMS.

Verktygen `unarj`, `zip` och `unzip` (PK??ZIP-kompatibla) finns också tillgängliga. Filer med ändelsen `.tar.gz` eller `.tgz` (arkiverade med `tar` och sedan komprimerade med `gzip`) är lika vanlig i UNIX-världen, som .ZIP-filer är under DOS. Så här listar du innehållet i ett `.tar.gz`- arkiv:

```
$ tar -ztf <file.tar.gz> | less
```

9.2 Installera program

Först av allt måste jag påpeka att det är roots jobb att installera applikationer. Vissa Linux-applikationer distribueras som `.tar.gz`-arkiv, vilka i typfallet innehåller en katalog som heter `paketnamn/`, och innehåller filer och/eller underkataloger. En bra regel är att installera dessa paket från `/usr/local` med kommandot

```
# tar -zxf <arkiv.tar.gz>
```

och sedan läsa README- eller INSTALL-filen. I många fall distribueras paketet som källkod, vilket innebär att du måste kompilera det för att skapa binärfilerna; ofta räcker det att skriva `make` och sedan `make install`. Du behöver naturligtvis `gcc`- eller `g++`-kompilatorerna.

Andra arkiv kan du bli tvungen att packa upp från `/`; det är fallet med Slackwares `.tgz`-arkiv. Andra arkiv innehåller filerna, men ingen underkatalog. Lista alltid innehållet i en katalog, innan du installerar det.

Debian- och Red Hat-distributionerna har sina egna arkiv-format: `.deb` och `.rpm` respektive. Det senare börjar bli vitt accepterat: för att installera ett `.rpm`-paket, skriv

```
# rpm -i paket.rpm
```

9.3 Tips du inte klarar dig utan

Kommando-komplettering: tryck <TAB> när du skriver ett kommando, för att komplettera kommandoraden. Exempel: du är tvungen att skriva `gcc this_is_a_long_name.c`; att skriva `gcc thi<TAB>` räcker. (Om du har andra filer som börjar med samma tecken, skriv tillräckligt många för att göra namnet unikt.)

Bakåtscrollning: om du trycker <SHIFT + PAGE UP> (den gråa knappen) så kan du scrolla bakåt ett par sidor, beroende på hur mycket grafik-minne du har.

Återställa skärmen: om du råkar köra `more` eller `cat` på en binär-fil, så fylls skärmen upp av skräp. För att fixa det, skriv (utan att se) `reset`, eller följande tecken-sekvens: `echo CTRL-V ESC c RETURN`.

Klistra in text: i konsollen, se nedan; i X, klicka och dra för att markera texten i en `xterm`, klicka sedan med mitten-knappen (eller båda knapparna tillsammans, om du har en mus med två knappar) för att klistra in. `xclipboard` (endast för text) finns också; bli inte förvirrad av dess långsamma reaktioner.

Använda musen: Om du har installerat `gpm`, en musdrivrutin för konsollen, så kan du klicka och dra för att markera text, sedan högerklicka för att klistra in markerad text. Fungerar även över olika virtuella terminaler.

Meddelanden från kärnan: ta en titt på `/var/adm/messages` eller `/var/log/messages`, som root, för att se vad kärnan har att säga dig, inklusive uppstarts-meddelanden. Kommandot `dmesg` är också praktiskt.

9.4 Var du kan hitta program

Om du undrar om det finns program som kan ersätta dina gamla för DOS/Win, så förslår jag att du kollar igenom de Linux-resurser som finns: <ftp://sunsite.unc.edu/pub/Linux> , <ftp://tsx-11.mit.edu/pub/linux> , and <ftp://ftp.funet.fi/pub/Linux> . Ett annat suveränt ställer är ”Linux applications and utilities page”, på <http://www.xnet.com/~blatura/linapps.shtml> .

9.5 Några saker du inte kunde göra

Linux kan göra en himla massa saker som var krångliga, svåra eller omöjlig att göra med DOS/Win. Här kommer en kort lista, för att reta din aptit:

- `at` låter dig köra program vid en angiven tidpunkt;
- `awk` är ett enkelt men kraftfullt språk för att manipulera datafiler (och inte bara det). T.ex., om `data.dat` är en datafil med flera fält, så skriver

  ```
  $ awk '$2 ~ "abc" {print $1, "\t", $4}' data.dat
  ```

  ut fälten 1 och 4 i varje rad i `data.dat`, vars andra fält innehåller "abc".
- `cron` är användbart för att utföra vissa saker med jämna mellanrum, på speciella datum och tidpunkter. Skriv `man 5 crontab`.
- `file` <`filnamn`> talar om vilken sort fil `filnamn` är (ASCII-text, körbar, arkiv osv.);
- `find` (se även avsnittet 4.3 (Kataloger: att översätta kommandon från DOS/Win till Linux)) är ett av de mest kraftfulla och användbara kommandona. Det används för att hitta filer som matchar flera beskrivningar och utför vissa saker med dem. Ett gerenellt sätt att använda det är:

  ```
  $ find <katalog> <uttryck>
  ```

  där <uttryck> innehåller sökningskriterier och åtgärder. Exempel:

  ```
  $ find . -type l -exec ls -l {} \;
  ```

  hittar alla filer som är symboliska länkar och visar vart de pekar.

  ```
  $ find / -name "*.old" -ok rm {} \;
  ```

  hittar alla filer som matchar mönstret och tar bort dem, efter att ha frågat om lov först.

  ```
  $ find . -perm +111
  ```

  hittar alla filer vars rättigheter matchar 111 (körbara).

  ```
  $ find . -user root
  ```

  hittar alla filer som tillhör root. Massor av möjligheter här—LMS.
- `grep` hittar text-mönster i filer. T.ex. så listar

  ```
  $ grep -l "geology" *.tex
  ```

  filerna *.tex som innehåller ordet "geology". Varianten `zgrep` fungerar på gzippade filer. LMS;
- **reguljära mönster** är ett komplicerat men väldigt kraftfullt sätt att utföra säknings-operationer på text. T.ex. så matchar

  `^a[^a-m]X{4,}txt$`

  en rad som börjar med "a", följt av vilket tecken som helst, utom någon i intervallet a-m, följt av fyra eller flera "X" och slutar med ".txt". Du använder reguljära mönster med avancerade editorer, `less` och många andra program. `man grep` för en introduktion.
- `script` <`skal-programs-fil`> skickar skärm-innehållet till `skal-programs-fil`, tills du ger kommandot `exit`. Användbart för avlusning;
- `sudo` tillåter användare att utföra vissa av roots uppgifter (t.ex. att formattera och montera diskar; LMS);
- `uname -a` ger dig information om ditt system;
- Följande kommandon kan vara praktiska: `bc`, `cal`, `chsh`, `cmp`, `cut`, `fmt`, `head`, `hexdump`, `nl`, `passwd`, `printf`, `sort`, `split`, `strings`, `tac`, `tail`, `tee`, `touch`, `uniq`, `w`, `wall`, `wc`, `whereis`, `write`, `xargs`, `znew`. LMS.

9.6 Vanliga filändelser och relaterade program

Du kommer att stöta på massor av filändelser. Bortsett från några av de mer ovanliga (d.v.s. typsnitt osv.), så kommer här en lista på vad som är vad:

- `1 ... 8`: man-sidor. Om du är från Mars, och alltså inte har det, skaffa `man`.
- `arj`: arkiv gjorda med `arj`.
- `dvi`: utdata-filer från TeX (se nedan). `xdvi` för att titta på dem; `dvips` för att omvandla dem till PostScript (`.ps`).
- `gz`: arkiv gjorda med `gzip`.
- `info`: info-fil (typ som ett alternativ till man-sidor). Skaffa `info`.
- `lsm`: Linux Software Map-fil. Det är en ren ASCII-fil som innehåller en beskrivning av paketet.
- `ps`: PostScript-fil. För att titta på den, eller skriva ut den, skaffa `gs` och, valfritt, `ghostview` eller `gv`.
- `rpm`: Red Hat-paket. Du kan installera det på ditt system med hjälp av paket-hanteraren `rpm`.
- `taz, tar.Z`: arkiv skapade med `tar` och komprimerade med `compress`.
- `tgz, tar.gz`: arkiv skapade med `tar` och komprimerade med `gzip`.
- `tex`: text-fil som ska skickas vidare till TeX, ett kraftfullt typsättnings-system. skaffa paketet `tex`, vilket kommer med de flesta distributioner: men akta dig för NTeX, vilken har korrupta typsnitt, och kommer med vissa versioner av Slackware.
- `texi`: texinfo-fil, kan producera både TeX- och info-filer (cp. `info`). Skaffa `texinfo`.
- `xbm, xpm, xwd`: Grafik-filer. Skaffa `xpaint`.
- `Z`: arkiv skapade med `compress`.

9.7 Konvertera filer

Om du behöver skicka text-filer mellan DOS/Win och Linux, så se upp med ”slut-på-raden”-problemet. Under DOS avslutas varje text rad med CR/LF (vagnretur och ny rad), medan det under Linux endast är LF. Om du vill försöka editera en DOS-text under Linux, så kommer antagligen varje rad sluta med ett konstigt ”M”-tecken; en Linux-textfil under DOS, kommer var en enda lång rad, utan stycken. Det finns några verktyg, `dos2unix` och `unix2dos`, för att konvertera filerna.

Om dina filer innehåller accenterade tecken, se till så att de är skapade under Winders (med t.ex. Write eller Notepad) och inte under DOS; annars komer alla accenterade tecken att bli förstörda.

För att konvertera Word- och WordPerfect-filer till ren text, så är det lite krångligare, men möjligt. Du behöver ett av verktygen som finns på CTAN-sajten: ett är `<ftp://ftp.tex.ac.uk>` . Skaffa paketet `word2x` från katalogen `/pub/tex/tools`, eller testa ett av paketen i katalogen `/pub/tex/support`. Jag har bara testat `word2x`, ocoh det fungerar ganska bra.

10 Slutet, för tillfället

Gratulerar! Du har nu greppat en liten del av UNIX och är redo att börja jobba. Kom ihåg att din kunskap om systemet fortfarande är begränsad, och att du förväntas träna mera med Linux, för att kunna använda det utan alltför stora problem. Men om allt du behöver göra är att skaffa en bunt applikationer och börja jobba med dem, så är jag säker på att det jag har tagit med här räcker till.

Jag är säker på att du kommer att trivas med Linux och fortsätta lära dig mer om det, det gör alla. Jag är också säker på att du aldrig kommer vilja gå tillbaks till DOS! Jag hoppas att jag har gjort mig förstådd och gjort mina 3-4 läsare en tjänst.

10.1 Upphovsrätt

10.2 Tillkännagivanden

”From DOS to Linux HOWTO” är skriven av Guido Gonzato, `guido@ibogfs.cineca.it`. Många tack till Matt Welsh, författaren till ”Linux installation and getting started”, till Ian Jackson, författaren till ”Linux frequently asked questions with answers”, till Giuseppe Zanetti, författaren till ”Linux”, till alla människor som skickat mig e-post med förslag, och speciellt till Linus Torvalds och GNU som gav oss Linux.

Det här dokumentet levereras som det är. Jag har lagt ner mycket ansträngningar på att skriva det så att det stämmer så bra som möjligt, men du får använda informationen i det på egen risk. Under inga omständigheter skall jag hållas ansvarig för några skador, som resulterar ur användningen av detta arbete.

All respons är välkommen. Känn dig välkommen att kontakta mig med frågor, förslag, ”flames” osv.

Ha det kul i Linux och livet,

Guido =8-)