Effektivare Linux

# Effektivare Linux

– kom igång med kommandoraden

Gareth Anderson
Tobias Hagberg

HME Publishing

Den här boken är en översättning och omfattande bearbetning och utökning av *GNU/Linux Command-Line Tools Summary* av Gareth Anderson. Förlagan är tillgänglig från The Linux Documentation Project, `http://tldp.org/LDP/GNU-Linux-Tools-Summary/html/`.

Hemsidan för denna bok finns på `http://hme.se/el/`, där även källkoden finns tillgänglig i LaTeX-format.

Det engelska originalet av Gareth Anderson.
Översättning, bearbetning och utökning av Tobias Hagberg.
Faktagranskning av den andra utgåvan av Andreas Önnebring.



Tryckt i EU 2010

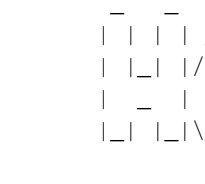
Andra utgåvan, första tryckningen

ISBN: 978-91-977260-9-2

# Innehåll

# Kapitel 1

# Förord



```
och välkommen till kommandoraden!
```

Den här boken vänder sig till dig som har använt Linux (eller en annan UNIX-variant, exempelvis Mac OS X) en tid och nu vill ta ett steg till: att börja använda alla de kraftfulla kommandona som gömmer sig bakom det snygga men inte alltid så effektiva grafiska gränssnittet.

Boken hjälper dig igång på allvar. Bläddra, testa, fundera och testa igen. Steg för steg kommer du att lära dig att få saker gjorda snabbare och (faktiskt) enklare än tidigare.

Det är förstås möjligt att läsa boken från pärm till pärm, men troligtvis är det inte så du kommer att få bäst nytta av den. Tänk på boken som en verktygslåda snarare än en komplett byggsats med ritningar. Vad du vill göra med verktygen är upp till dig. Med hjälp av boken, ett terminalfönster och lite envishet kan du göra i princip vad som helst.

För den erfarna användaren tjänar boken som en bra referens att ha i bokhyllan.

En stor del av bokens alla kommandon är utvecklade inom ramen för GNU-projektet, en pionjärorganisation och

en av de mest betydelsefulla drivkrafterna inom fri källkod. Kombinationen GNU/Linux erbjuder dig massor av kreativa och effektivitetshöjande möjligheter – en stor mängd små specialkommandon kan kopplas ihop för att utföra precis de uppgifter du vill, även sådana som är mycket komplexa.

Kommandona i boken är enkelt tillgängliga i Debian, Ubuntu, Fedora och de flesta andra Linuxvarianter. Tack vare projektet Fink kan du installera och använda de flesta av kommandona i Mac OS X. Boken visar dig hur du installerar kommandona på dessa plattformar.

Den här boken bygger vidare på *GNU/Linux Command-Line Tools Summary* av Gareth Anderson. Förlagan, liksom den här bearbetningen, är licensierad i enlighet med GNU Free Documentation License och innehåller värdefulla bidrag från många personer (se kapitel 22 för mer information om förlagan).

Översättningen till svenska har gjorts av Tobias Hagberg. I samband med översättningen har också texten bearbetats – somligt har strykts, ändrats eller stuvats om, en hel del annat har lagts till. Till den andra utgåvan utökades texten med kommandon för PDF- och grafik-manipulering. Därtill bidrog Andreas Önnebring inför den andra utgåvan med en grundlig genomläsning, och föreslog många viktiga förbättringar.

Förhoppningen är att den här boken ska vara den mest lättillgängliga introduktionen till kommandoraden för svenskspråkiga läsare.

# Kapitel 2

# Introduktion

Den här boken innehåller ett stort antal mycket användbara verktyg som finns i GNU/Linux-baserade operativsystem, liksom i många andra UNIX-varianter.

Att berätta om alla kommandon som finns är en omöjlig uppgift. Målet med boken är att ge information om de mest använda och mest användbara verktygen. Varje beskrivning innehåller en snabb översikt av dess funktion och användbara alternativ för verktyget. Boken visar också var du hittar mer information om respektive kommando, om och när du skulle vilja fördjupa dig.

Verktygen i boken är helt kommandoradsbaserade och kräver inte ett grafiskt gränssnitt för att köras. Du kan lika gärna använda en helt textbaserad terminal utan grafikstöd som ett terminalfönster i det grafiska gränssnittet.

Notera också att några av verktygen i den här guiden är specifika för skalet Bash (the Bourne-Again-SHell). Skalet Bash är standard i Linux och Mac OS X, även om flera andra skal finns att tillgå för båda operativsystemen. Verktyg som är specifika för andra skal beskrivs inte.

För vissa verktyg som kräver fördjupande förklaringar finns en uppsättning mini-handledningar i slutet av boken.

Ordet ”verktyg” används omväxlande med ordet ”kommando”. Båda har samma betydelse i den här guiden. En mer utförlig förklaring finns i ▷ sektion 3.1.

## 2.1 Till vem vänder sig den här boken?

Vem som helst som är intresserad av att lära sig mer om några av de viktigaste kommandoradsverktygen i operativsystemet GNU/Linux och andra UNIX-baserade operativsystem har nytta av den här boken.

Även om kommandoradsgränssnittet (*Command-Line Interface* på engelska, förkortas ofta CLI) är svårare att lära sig än den grafiska miljön är det ofta det snabbaste och effektivaste sättet att använda en dator. För många systemadministratörer, programmerare och avancerade användare är kommanoraden standardgränssnittet mot datorn.

Med den här boken kommer du att få hjälp att börja bekanta dig med några av de mest användbara kommandona i Linux. Om du tar investerar tid att lära dig mer om kommandoradsgränssnittet nu vinner du mycket tid längre fram.

Den här boken är främst avsedd för nybörjare och medelavancerade användare som vill upptäcka vilka kommandoradsverktyg som finns tillgängliga. Avancerade användare kan dra nytta av den som en kommandoradsreferens.

## 2.2 Typografiska konventioner

Följande typografiska konventioner används i den här guiden:

**Kommandosyntax och -exempel** Visar helt enkelt hur du vanligtvis använder ett kommando. Ofta används verkliga exempel istället för förklaringar av kommandosyntaxen.

Ordet ”kommandosyntax” följs av ett exempel som visar hur du skriver in kommandot i ett skal.

Standardsyntaxen för verktyg är vanligtvis följande:

```
kommando -alternativ argument
```

**Kommandonamn, -alternativ och -argument i brödtexten** Körbara kommandon, alternativ till kommandon, argument till kommandon eller sökvägen till filer på datorn (se kapitel 3 för förklaringar av dessa termer) visas i löpande text så här: **`ls -l /home/bertil`**

**Originaltermer på engelska** *Kursivering* används för att markera text, liksom för engelska översättningar eller originalnamn.

**Tangentbordsgenvägar** Kombinationer av tangenter fogas samman med + (plustecken) mellan de tangenter som måste tryckas ned samtidigt. Till exempel: CTRL+Z betyder att du ska hålla ned tangenten med bokstäverna Ctrl och samtidigt trycka på Z-tangenten.

**Uppmärksamhetsvarningar** Används på olika ställen för att notifiera eller varna dig som läsare. Följande typer finns:

**OBS** Observera följande text; används för att visa viktig information om ett verktyg.

**TIPS** Följande ger ett särskilt användbart tips om hur du kan använda ett verktyg.

**Kommandoinformation** Avsnittet om ett kommando inleds med information om kommandot, enligt nedan:

---

**$ kommandonamn** kort beskrivning

---

**DEBIAN, UBUNTU:** **paketnamn** **FEDORA:** **paketnamn**

---

**HJÄLP:** **man kommandonamn**

---

- Kommandonamnet är det du skriver vid prompten för att köra kommandot. $ betyder det att det går bra att köra som en vanliga användare, # betyder det att endast root-användaren kan använda det.
- En kort beskrivning av kommandot följer därpå.
- Använd paketnamnet tillsammans med ditt pakethanteringsverktyg (se kapitel 7) för att installerat kommandot på datorn.
- På sista raden ser du vilket kommando du kan köra för att se mer information och få hjälp.

## 2.3 Läs boken så här

För dig som precis har börjat bekanta dig med skalet är kapitel 3 den bästa utgångspunkten. Där får du en introduktion till terminaler, virtuella terminaler, skal och kommandopromtpen. Där får du också en lista med några viktiga kommandon som gör det möjligt för dig att förflytta dig i filsystemet samt kopiera, flytta och radera filer. I kapitlet ingår också information om jokertecken och smarta knep som gör arbetet framför datorn snabbare och säkrare. Efter att ha läst på om grunderna i kapitel 3 kan du fritt utforska bokens alla kommandon. Testa flitigt och läs i boken och i manual-sidorna för de kommandon som intresserar dig. Du kommer att bli varm i kläderna förr än du anar.

Om du någon gång börjar känna dig begränsad rekommenderas kapitel 19 och kapitel 20. I de kapitlen får du lära dig det du behöver för att jonglera med kommandon på högsta nivå. När du har lärt dig allt om reguljära uttryck och skalets finesser för att dirigera indata och utdata från kommandon finns det inget – säger inget – som kan stoppa dig.

# Kapitel 3

# Kom igång med skalet

Det här kapitlet innehåller mycket användbara grundkunskaper. Det innehåller också smarta tips om tangentbordsgenvägar och funktioner som gör arbetet snabbare.

## 3.1 Filosofin bakom UNIX-verktyg

UNIX (och därmed GNU/Linux) är uppbyggt för att vara modulärt och flexibelt. Istället för ett fåtal stora program som klarar mycket finns en mycket stor mängd små specialiserade program som kan lite. Tricket är sedan att kombinera delarna – som att bygga lego ungefär.

Ett verktyg är ett vanligt enkelt program, vanligtvis utformat för ett visst specifikt syfte. Det kallas i den här boken för ett kommando.

Filosofin bakom Unix-verktyg växte fram under skapandet av operativsystemet UNIX, efter den banbrytande uppfinningen av rör (*pipe* på engelska), | (se kapitel 19 för utförlig information om hur du använder rör).

I korthet gav rör möjligheten att skicka utdata från ett program som indata till ett annat. Filosofin var att skapa små program som löste specifika uppgifter istället för att skapa stora program som klarade mycket. Mer komplexa uppgifter kunde lösas genom att koppla ihop de olika småprogrammen med rörledningar.

Alla grundläggande verktyg i UNIX-systemet utformades så att de skulle kunna fungera tillsammans.

Med den här filosofin kunde programmerare undvika att skriva kod i sina större program som redan hade skrivits av någon annan (man kan kalla det kod-återvinning). Till exempel använder en mängd olika program samma kommandoradsbaserade stavningskontroll.

Den här filosofin lever vidare idag i GNU/Linux och andra UNIX-baserade operativsystem (som NetBSD, FreeBSD, OpenBSD, Mac OS X och så vidare).

## 3.2 Terminaler och skal

I boken används många för linuxnybörjaren nya koncept, exempelvis 'skal', 'terminal' och 'kommandorad'. Det här kapitlet ger en snabb översikt som bringar ordning i floran av nya termer.

**Terminal** Förr i världen kopplades flera så kallade terminaler till en och samma stordator. En terminal är helt enkelt en skärm och ett tangentbord som är kopplade till en dator. Terminalen gör det möjligt för en användare att interagera med datorn – genom att läsa text och skriva in fjärrkommandon. I operativsystemet används namnet tty, kortform för *teletyping* på engelska, ungefär fjärrinmatning på svenska. När du startar datorn i textläge använder du en terminal för att interagera med datorn.

**Virtuell terminal** Sannolikheten är stor att du använder ett grafiskt gränssnitt på din dator. För att skriva in kommandon utan att lämna den grafiska miljön kan du öppna en virtuell terminal, det vill säga ett fönster på din dator som ser ut som och som fungerar som en vanlig terminal. Detta kallas virtuell terminal eller terminalfönster. Det finns en uppsjö olika varianter på terminalfönster, exempelvis **`xterm`**, **`kterm`** (standard i KDE) eller **`gnome-terminal`** (standard i GNOME). Vilket du använder spelar ingen roll, användningssättet och finesserna är i allt avgörande desamma.

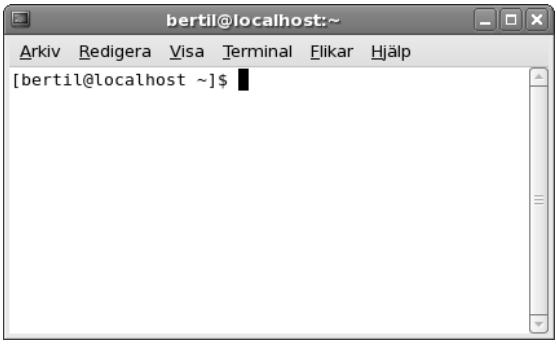
**Skal** Oavsett om du använder en terminal eller en virtuell terminal behövs ett skal för att tolka kommandona

du skriver in och visa resultatet på skärmen. Ett skal är helt enkelt ett program som körs i terminalen och som gör det möjligt för datorn att kommunicera med användaren, liksom för att låta kommandon kommunicera med varandra. Skalet innehåller också en mängd trevliga finesser som underlättar arbetet, bland annat en smart kommanoradshistorik. Det finns många olika skal som du kan använda, i Linux är dock skalet **`bash`** standardskalet. Den här boken förutsätter att du använder **`bash`**.

**Kommandoprompt** Med kommandorad menas helt enkelt den rad i en terminal eller virtuell terminal som skalet visar och på vilken du kan skriva in kommandon. Tänk på raden som en uppmaning (*prompt* på engelska) att skriva in något. När du har skrivit in kommandot och tryckt på ENTER tolkar skalet det du skrivet, kör kommandot och visar resultatet i den terminal eller virtuella terminal som du använder. Det första som syns på kommandoraden är en prompt, som ger dig information om var du befinner dig i filstrukturen och låter dig mata in kommandon. Läs mer om kommandoprompten i ▷ sektion 3.3.

## 3.3 Kommandoprompten

När du startar ett skal ser du en kommandoprompt. Exakt hur prompten ser ut beror på vilken Linuxdistribution och vilket skal du använder.




Användaren ”bertil” inloggad på datorn ”localhost”, i hemmappen.

Så kan kommandoprompten se ut:

```
användarnamn@datornamn:aktuell_mapp$
```

Exempel:

```
bertil@skalman:~/dokument$
```

Detta betyder kort och gott att den aktuella användaren heter ”bertil”, att datornamnet är ”skalman” och att du för tillfället befinner dig i mappen ”dokument”, som ligger i din hemmapp. Det lilla tecknet `~` är ett alias för den aktuella användarens hemmapp (mer om detta i ▷ sektion 3.4.3). Sist i prompten är ett dollartecken, $, som anger att den aktuella användare är en vanlig användare utan administrationsrättigheter. Om du är inloggad som administratören på en dator, med root-användaren, anges detta med ett brädgårdstecken, # (läs mer i ▷ sektion 3.3.2).

### 3.3.1 Skriva in kommandon

Det är enkelt att skriva in och köra ett kommando. Skriv helt enkelt kommandonamnet efter prompten, lägg till eventuella alternativ som kommandot stöder och avsluta med eventuella argument. Alternativ ändrar beteendet hos kommandot, argumentet är det som kommandot ska köras på (exempelvis en fil, sökväg eller text).

Skriv kommandon på följande sätt, avsluta alltid med ett tryck på tangenten ENTER:

```
kommandonamn -alternativ argument
```

Detta gäller bokens alla kommandon, även om tangenttryckningen på ENTER har uteslutits från instruktionerna för att undvika onödiga upprepningar.

Ett praktiskt exempel:

```
ls -l /home/bertil
```

Detta kör kommandot **`ls`** med alternativet **`-l`** och argumentet **`/home/bertil`**, vilket visar en utförlig lista med alla filer i hemmappen för användaren ”bertil”. Läs mer om **`ls`** i ▷ sektion 8.1.2.


Resultatet av kommandot **`ls`**: innehållet i ”bertils” hemmapp.

För att ange flera alternativ till ett kommando på en och samma gång går det lika bra att slå ihop dem som att ange dem separerade från varandra.

Till exempel fungerar

```
ls -al /home/bertil
```

precis lika bra som

```
ls -a -l /home/bertil
```

I vissa fall använder olika kommandon samma alternativ med samma betydelse, men huvudregeln är att varje kommando har en unik uppsättning alternativ. För en del kommandon kan alternativ inte anges alls.

**OBS** Lägg märke till att alternativ ibland kallas ’flagga’ eller ’väljare’. Dessa ord är alla synonymer i kommandoradssammanhang.

### 3.3.2 Undvik root-användaren

Kör kommandona i den här boken som en vanlig användare, inte som systemadministratören root. Detta gäller särskilt dig som är nybörjare och experimenterar mycket. Om du är inloggad som en vanlig användare kan du inte oavsiktligen ställa till med problem på systemet – datorn låter dig inte göra det. Som root-användaren har du däremot oinskränkt makt över minsta del av datorn. Om du som root-användare

av misstag skriver fel kan det få onödigt tråkiga konsekvenser.

Vissa kommandon måste köras som root-användaren för att fungera. Detta anges i sådana fall tydligt.

I Bash ser du om du är inloggad som en vanlig användare genom att prompten avslutas med ett dollartecken, $ (vanlig användare). Om du är inloggad som root-användaren visas detta med tecknet # (root-användaren).

Exempel:

```
root@skalman:~#
```

**TIPS** Om du under läsningen av den här boken vill experimentera med olika kommandon och inställningar på datorn kan du skapa skapa en särskild testanvändare på datorn. Då lämnar du din riktiga hemmapp i fred, och kan experimentera utan att oroa dig. Se ▷sektion 11.4.2 och ▷sektion 11.2 för mer information.

#### Använd sudo

I vissa operativsystem är root-användaren helt inaktiverad. Detta gäller exempelvis i Mac OS X och Ubuntu Linux. Istället kan vanliga användare tillfälligt få root-rättigheter genom att skriva in kommandot **sudo** före det önskade kommandot.

Kommandosyntax:

```
sudo kommandonamn
```

Om du är systemadministratör på datorn ger **sudo** dig rätt att köra det önskade kommandot, annars talar **sudo** om för dig att du inte har rätt att köra kommandot.

Mer information om kommandot finns i ▷sektion 11.3.

### 3.3.3 exit

Avslutar ditt skal (vilket oftast leder till att den virtuella terminalen stängs).

Exempel:

```
exit
```

TIPS Du kan också använda CTRL+D eller kommandot **logout** för att logga ut ur ett inloggningsskal.

### 3.3.4 Grundläggande kommandon

Börja gärna med att ta en närmare titt på följande kommandon om du är nybörjare och vill komma igång med att utforska systemet.

**ls** Se ▷sektion 8.1.2.

**cd** Se ▷sektion 8.1.1.

**cp** Se ▷sektion 8.4.5.

**mv** Se ▷sektion 8.4.4.

**rm** Se ▷sektion 8.4.2.

**mkdir** Se ▷sektion 8.4.1.

**rmdir** Se ▷sektion 8.4.3.

**find** Se ▷sektion 8.2.1.

## 3.4 Specialtecken och jokertecken

Det finns ett antal specialtecken som du behöver känna till när du jobbar med skalet. Dessa specialtecken och jokertecken fungerar med alla kommandon som du kör i skalet.

### 3.4.1 . (punkt)

En punkt betyder kort och gott ”här”. Den anger med andra ord den aktuella mappen.

Om du till exempel står i mappen ”/home/bertil” och vill kopiera en fil till just den här mappen kan du använda kommandot **cp** (kortform av *copy* på engelska, se ▷sektion 8.4.5).

```
cp /tmp/fil.txt .
```

istället för

```
cp /tmp/fil.txt /home/bertil
```

Båda exemplen ovan kopierar ”fil.txt” från mappen ”/tmp” till den mapp du för tillfället står i, men den övre är enklare och snabbare.

### 3.4.2 .. (två punkter)

Två punkter anger föräldramappen. Du använder två punkter för att ange en mapp i hierarkin (närmare roten i filträdet).

Om du till exempel står i mappen ”/home/bertil” och vill gå till ”/home” kan du använda kommandot **cd** (kortform av *change directory* på engelska, se ▷sektion 8.1.1) med två punkter som argument:

```
cd ..
```

Detta ställer dig i mappen ”/home”.

Om du står i mappen ”/home/bertil/musik/mp3” och vill hoppa till ”/home/bertil/dokument” kan du skriva

```
cd ../../dokument
```

Om du står i mappen ”/home/bertil/dokument” och vill se en detaljerad lista över filerna i ”/home/bertil” kan du använda kommandot **ls** (kortform av *list* på engelska, se ▷sektion 8.1.2).

```
ls -l ..
```

### 3.4.3 ~ (tilde)

Tilde-tecknet används som ett alias för användarens hemmapp.

Om ditt användarnamn till exempel är ”bertil” kan du istället för att skriva in **cd /home/bertil** helt enkelt knappa in **cd ~**. Om du vill hoppa till en mapp i hemmappen som heter dokument kan du skriva **cd ~/dokument**.

**TIPS** ~ (tilde) kan också användas som en genväg till andra användarens hemmappar. Skriv då ~användarnamn, så hoppar du till hemmappen för den användaren.

### 3.4.4 ? (frågetecken)

Jokertecknet ? representerar ett enda tecken, vilket som helst. Till exempel matchar ”hd?” både hda, hdb, hdc eller vilket annat tecken som helst från a till z, liksom vilken siffra som helst mellan 0 till 9.

Exempel:

```
ls dokument-?.doc
```

Detta visar alla filer i den aktuella mappen vars namn börjar med ”dokument-”, som därefter innehåller exakt ett tecken (vilket som helst) och som slutar med ”.doc”.

Läs mer om hur du kan använda detta jokertecken i skalet, eller för att skapa reguljära uttryck i olika program, i kapitel 20.

### 3.4.5 * (asterisk)

Detta är det vanligast förkommande jokertecknet, med betydelsen matcha allt. Asterisken representerar ett godtyckligt antal tecken (inklusive inget tecken alls). Till exempel matchar ”cd*” både cd, cda, cdrom, cdrecord – allt som börjar med cd inklusive enbart cd. Exemplet ”m*l” matchar både ml, mall och multilateral – allt som börjar med m och slutar med l, med andra ord.

Exempel:

```
ls *.doc
```

Detta visar alla filer som slutar med ändelsen .doc i den aktuella mappen.

Läs mer om hur du kan använda detta jokertecken i skalet, eller för att skapa reguljära uttryck i olika program, i kapitel 20.

### 3.4.6 \ (bakstreck)

Tecknet \ kan användas före speciella tecken som mellanslag och jokertecken för att förhindra att Bash försöker tolka, ”expandera”, dem. Om du ber skalet utföra något på en fil eller mapp som i sitt namn innehåller ett specialtecken kan du använda bakstrecket för att skydda det från att först tolkas av skalet.

Ett sådant speciellt tecken är mellanslag. Du kan byta till en mapp vars namn innehåller mellanslag så här:

```
cd namn\ med\ mellanslag
```

Utan bakstrecken skulle ovanstående inte fungera eftersom skalet Bash skulle försöka byta till en mapp med endast namnet ”namn” (resten av raden skulle ignoreras).

**TIPS** Istället för att använda bakstreck före mellanslag i filnamn kan du kapsla in hela filnamnet med enkla citationstecken.

Exempel:

```
cd 'namn med mellanslag'
```

Båda sätten fungerar lika bra!

**OBS** Lägg märke till att TAB-tangenten (automatisk ifyllning av namn och kommandon) automatiskt använder bakstreck före mellanslag, du slipper skriva in dem manuellt.

## 3.5 Redigera kommandon

### 3.5.1 Komplettera automatiskt

Skriv in de första bokstäverna i namnet på ett kommando och tryck på tangenten TAB, så kompletterar Bash automatiskt med det som återstår av namnet åt dig. Du kan använda funktionen för att fylla i kommandonamn (*automatic commmand completion* på engelska), liksom när du navigerar och jobbar i filsystemet, exempelvis när du hoppar mellan mappar, kopierar filer och så vidare (*automatic filename completion* på engelska).

### 3.5.2 Tangentbordsgenvägar

Det finns många tangentbordsgenvägar i GNU/Linux som du kan använda för att förenkla redigeringen av kommandon i skalet och för att snabba upp arbetet. Nedan är en lista med några av dem (se också CTRL+R i sektionen om kommandoradshistoriken).

**CTRL+A och CTRL+E**

Dessa tangentbordsgenvägar används för att gå till början eller slutet av en rad i terminalen. Använd CTRL+A för att gå till början på raden, använd CTRL+E för att hoppa till slutet av raden.

Normalt fungerar tangenterna markerade HOME och END likadant om du är van vid dessa.

**CTRL+K**

Den här tangentbordsgenvägen används för att klippa ut eller radera det som finns framför markören.

**CTRL+Y**

Den här tangentbordsgenvägen kan användas för att klistra in det du senaste klippte ut (med CTRL+K eller CTRL+W).

**CTRL+W**

Den här tangentbordsgenvägen kan användas för att klippa ut eller radera hela ord.

### 3.5.3 Kommandoradshistoriken

Alla kommandon du skriver in sparas av Bash. Du kan bläddra bland de kommando du har skrivit tidigare genom att trycka på uppåtpil- och nedåtpil-tangenterna på tangentbordet. Tryck ENTER för att utföra ett kommando du har bläddrat fram till, eller använd vänsterpil- och högerpil-tangenterna för att redigera kommandot.

Du kan också använda det i Bash inbyggda kommandot **`history`**. Med kommandot kan du visa alla kommandon du har använt. Använd det så här:

```
history n
```

Detta skriver ut de senaste n (där n står för godtyckligt antal) kommandona. Skriv in **`history`** (utan alternativ) för att se hela historiklistan.

Du kan också skriva in **`!n`** för att köra kommando nummer n. Använd **`!!`** för att köra det senaste inskrivna kommandont.

**!-n** kör kommandot n steg bakåt i listan (därmed blir **!-1** samma sak som **!!**).

**!sträng** kör de senaste kommandot som börjar med texten ”sträng” (där sträng betyder godtycklig text, kan vara vad som helst) och **!?sträng?** kör de senaste kommandot som innehåller ordet ”sträng”. Exempel:

```
!cd
```

Kör det senaste kommandot du skrev in som börjar med **cd**.

”kommandonamn !*” kör ”kommandonamn” med de argument du använde med ditt förra kommando. Detta är användbart när du stavar kommandonamnet fel, till exempel. Om du skrev:

```
emasc /home/bertil/program.java /tmp/test.java
```

för att använda Emacs för att öppna de två angivna filerna kommer du misslyckas eftersom kommandonamnet är felstavat. Om du då skriver:

```
emacs !*
```

kommer du köra emacs med de argument du senaste skrev in på kommandoraden. Det är med andra ord detsamma som:

```
emacs /home/bertil/program.java /tmp/test.java
```

Använd CTRL+R för att göra en bakåtvänd sökning i historiken, *reverse-i-search* på engelska. Om du till exempel upprepa det du skrev in senaste gången du använd kommandot **snort** skriver du:

```
CTRL+R snort
```

Då ser du följande i terminalfönstret:

```
(reverse-i-search)`':
```

Skriv in det du söker och använd sedan CTRL+R för att bläddra bakåt bland sökträffarna i historiken.

Tryck CTRL+R så många gånger du behöver för att hitta det du söker. Tryck då ENTER för att köra kommandot.

Du kan också använda högerpil- och vänsterpil-tangenterna för att åter sätta in kommandot på kommandoraden, så att du kan redigera det.

# Kapitel 4

# Söka hjälp

Det här kapitlet ger information om hur du söker hjälp i ett GNU/Linux-system. Det finns nästan alltid någon form av bra hjälp för verktygen på datorn.

## 4.1 -?, -h, --h, --help

Pröva alternativen **`-?`**, **`--h`**, **`--help`** och **`-h`** med ett kommando.

Exempel:

```
ls --help
```

Nästan alla kommandon visar översiktligt information om dess användning med något av dessa alternativ. Tyvärr finns det ingen enhetlig standard kring vilket av alternativen som används, utan du måste pröva dig fram. Det är ofta värt besväret – den inbyggda hjälpen brukar vara mer kortfattad än den som du kan se med **`man`** eller **`info`**, så det är en bra idé att börja med den.

## 4.2 man

**$ man** Visar manual-sidor för program

**DEBIAN, UBUNTU:** **man** **FEDORA:** **man**

**HJÄLP:** **man man**

Detta kommando visar kortfattad information om ett program från det inbyggda manualsystemet på datorn.

Kommandosyntax:

```
man programnamn
```

Pröva till exempel att läsa manual-sidan för **man** (det vill säga **man**-kommandots egen man-sida):

```
man man
```

Observera att manual-sidorna i allt väsentligt endast finns tillgängliga på engelska, även om vissa sidor har översatts till svenska.

**OBS** Stäng ned manual-visaren med tangenten Q.

Manual-sidorna är indelade i flera så kallade sektioner, från 1–9. Vissa kommandon har endast en manual-sida, andra har flera i olika sektioner. Du kan använda kommandot **apropos** (▷ sektion 4.5) för att ta reda på vilka sektioner du ska kolla i.

Indelningen i sektioner ser ut så här:

1. kommandon
2. systemanrop (för programmerare)
3. funktioner (för programmerare)
4. enhetsfiler under `/dev`
5. filformat
6. spel
7. diverse
8. admin-kommandon
9. linuxkärnan (för programmerare)

En man-sida med ett visst namn kan finnas i flera sektioner samtidigt. Ibland kan du se referenser till mansidor med tillhörande sektion inom parentes. Till exempel refererar ”time(1)” till ls-manualen i sektion 1.

Syntaxen för att se manual-sidan i en viss sektion är:

```
man sektionsnummer programnamn
```

Exempel:

```
man 7 time
```

Detta visar man-sidan i sektion 7 för kommandot **time**, innehållet i sektion 1 är helt annorlunda.

```
man -k sträng
```

Du kan också söka efter en sträng (godtycklig text) i alla manualsidor. Då frågar programmet om du vill se varje sida som det hittar. Omgärda strängen du söker efter med dubbla citattecken ” och ” om den innehåller mellanslag.

**OBS** Observera att sökningen tar mycket, mycket lång tid, eftersom programmet söker efter strängen i *alla* man-sidor på datorn.

```
man -f kommando
```

Detta visar uppgifter om kommandot. För att det ska fungera måste root-användaren på datorn först köra kommandot **makewhatis**.

**OBS** **man -f kommando** ger samma resultat som att använda kommandot **whatis** (se nedan).

## 4.3 info

---

**$ info** Visar info-sidor för program

---

**DEBIAN, UBUNTU:** **info** **FEDORA:** **info**

---

**HJÄLP:** **man info, info info**

---

Visar en mer detaljerad manual med hyperlänkar om ett visst kommando. Detta fungerar endast med vissa kommandon.

Kommandosyntax:

```
info programnamn
```

## 4.4 whatis

**$ whatis** Visar korta beskrivningar av program

**DEBIAN, UBUNTU:** **man-db** **FEDORA:** **man**

**HJÄLP:** **man whatis**

Visar en enradsbeskrivning om programmet. Programnamnet måste skriva in exakt, annars visas ingenting alls. Använder sig av whatis-databasen (se nedan).

Kommandosyntax:

```
whatis programnamn
```

```
makewhatis
```

Skapar den så kallade whatis-databasen som kommandona **apropos**, **whatis** och **man -f** behöver. Efter att databasen en gång har skapats behöver kommandot inte köras igen.

**OBS** Observera att kommandot måste köras av root-användaren, och att det tar ganska lång tid att skapa databasen.

## 4.5 apropos

**$ apropos** Söker efter beskrivningar och namn i manual-sidor

**DEBIAN, UBUNTU:** **apropos** **FEDORA:** **apropos**

**HJÄLP:** **man apropos**

```
apropos
```

Söker i whatis-databasen efter strängar, på likande sätt som **whatis**, men med den skillnaden att det söker efter och skriver ut alla resultat som matchar strängen (eller en

del av strängen). Kommandot behöver en whatis-databas för att fungera (se ovan).

Kommandosyntax:

```
apropos sträng
```

**OBS** Att köra kommandot **apropos** ger samma resultat som **man -k**.

## 4.6 Ytterligare källor till hjälp

Utöver de vanligaste hjälp- och dokumentationskällorna **man**, **info** och den inbyggda hjälpen i kommandon finns det information att hämta på några ställen till.

Till exempel kan det under `/usr/share/doc/paketnamn` finnas ytterligare information att hitta, exempelvis HTML-format. Dessa sidor kan du titta på med en webbläsare.

För kommandon som är inbyggda i skalet Bash kan man se hjälptexter med hjälp av **help kommandonamn** direkt vid kommandoprompten.. Om du vill veta om ett kommando är inbyggt i skalet eller är ett eget program kan du köra kommandot **type** följt av kommandots namn (se ▷ sektion 8.3).

Dessutom finns en uppsjö dokumentation på nätet, som är enkelt åtkomlig tack vare sökmotorer.

# Kapitel 5

# Skal- och miljövariabler

Det är tack vare en miljövariabel som du kan starta ett program bara genom att skriva dess namn, utan att känna till i vilken mapp det ligger. Miljövariabler har också många andra användningsområden.

När skalet startas läser det in så kallade miljövariabler i datorns minne som olika kommandon och program kan använda sig av. Varje miljövariabel innehåller ett visst värde, som kan vara exempelvis information om standardspråk på datorn, sökvägen till körbara program och kommandon, sökvägen till användarens hemmapp och så vidare (se nedan för en lista med exempel).

Så fort ett program behöver kolla upp sökvägen till den aktuella användarens hemmapp kan det kontrollera miljövariabeln `$HOME`.

Det är miljövariabeln `$PATH` som håller reda på sökvägen till mappar som innehåller körbara program och kommandon. Det är denna miljövariabel som gör det möjligt för Bash att hitta hela sökvägen till och starta ett kommando, exempelvis `/bin/ls` när du skriver **`ls`**. Hade det inte varit för att variabeln `$PATH` innehöll sökvägen `/bin` hade Bash inte hit-

tat kommandot alls (du hade då behövt ange `/bin/ls` för att köra det).

Som namnet *variabel* antyder kan värdet av en miljövariabel fritt ändras av användaren. En användare eller ett program kan dessutom själv skapa valfria egna miljövariabler när som helst.

Miljövariabler går i arv. Nya processer som skapas i skalet (exempelvis när du kör ett kommando) får tillgång till samma variabler som skalet, som därmed kan sägas vara *förälderprocess* till alla andra processer som startas i det. Däremot kan en föräldraprocess inte ärva miljövariabler i andra riktningen, från en barnprocess.

Här är några vanliga miljövariabler på datorn:

**`$PATH`** anger de sökvägar skalet letar i efter körbara program/kommandon

**`$HOME`** anger sökvägen till användarens hemmapp

**`$TERM`** anger vilken terminal eller virtuell terminal som används

**`$PS1`** anger hur kommandoprompten ser ut

Vanligtvis läser Bash in standarduppsättningen med variabler från filen `~/.bashrc` (i användarens hemmapp) när skalet startar. (Punkten framför namnet indikerar att filen är osynlig, läs mer om hur du visar sådana filer i ▷sektion 8.1.2.) Om du vill att en miljövariabel ska finnas kvar när du startar skalet efter omstart lägger du till miljövariabeln på en tom rad i filen `~/.bashrc` i din hemmapp. Om du önskar att miljövariabeln ska fungera för alla användare på datorn kan du lägga till den till filen `/etc/profile`.

## 5.1 export

**`$ export`** Exporterar en skalvariabel till en miljövariabler

**DEBIAN, UBUNTU:** **`bash`** **FEDORA:** **`bash`**

**HJÄLP:** **`man bash`**

Du kan skapa eller ändra en befintlig variabel i ditt Bashskal på den här formen:

```
VARIABEL=värde
```

Exempel:

```
FAVORITMAT=hamburgare
```

Ännu så länge finns variabeln endast i det aktuella skalet. Om du vill göra denna skalvariabel till miljövariabel använder du kommandot **`export`**.

Kommandosyntax:

```
export VARIABEL
```

Exempel:

```
export FAVORITMAT
```

Genom att exportera skalvariabeln har du skapat en miljövariabel. Använd kommandot **`printenv`** för att se värdet av miljövariabeln.

**TIPS** Du kan göra båda operationerna ovan i ett svep. Om du skriver

```
export FAVORITMAT=hamburgare
```

skapas miljövariabeln i ett steg.

Nu kanske variabeln `$FAVORITMAT` inte är så särskilt användbar, men nedanstående exempel kan komma till större användning.

Miljövariabeln `$PATH` består av en lista med mappar som innehåller körbara filer. Sökvägarna skiljs åt med ':', kolon. Om du själv har installerat ett program på datorn i en mapp som inte finns i miljövariablen `$PATH`, exempelvis i mappen `/home/bertil/program`, kan du lägga till den till listan genom att skriva följande:

```
export PATH=$PATH:/home/bertil/program
```

Detta lägger till den privata programmappen till slutet av listan med sökvägar i den befintliga variablen `$PATH`. `$PATH` innehöll sökvägarna `/bin:/usr/bin:/usr/local/bin` tidigare innehåller den efter att du har kört ovanstående `/bin:/usr/bin:/usr/local/bin:/home/bertil/program`.

Alla program du installerar i den privata programmappen kan du starta genom att skriva kommandonamnet, utan att ange hela sökvägen.

Om du vill ändra utseendet på kommandoprompten från standardutseendet till något kompaktare kan du testa att köra följande:

```
export PS1="\u \$ "
```

Detta ändrar prompten så att den enbart innehåller användarnamnet följt av dollartecken.

Om du vill göra en miljövariabel permanent lägger du till den till filen `~/.bashrc` i din hemmapp.

## 5.2 printenv

---

**$ printenv** Visar en eller flera miljövariabler

---

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

---

**HJÄLP:** **man printenv**

---

Använd **printenv** för att se värdet av en specifik eller alla miljövariabler.

Kommandosyntax:

```
printenv miljövariabel
```

Exempel:

```
printenv HOME
```

Detta visar vilken som är användarens hemmapp.

```
printenv
```

Detta visar alla miljövariabler.

**OBS** Du kan även använda kommandona **env** och **set** (se ▷ sektion 6.2) för att se miljövariabler och deras värden. Med dessa två kommandon kan du också ställa in miljövariabler.

# Kapitel 6

# Knep och tips

Det här kapitlet innehåller ett antal knep som gör tiden framför kommandoraden lite effektivare och säkrare.

## 6.1 alias

**$ alias** Sätta alias för kommandon i Bash

**DEBIAN, UBUNTU:** **bash** **FEDORA:** **bash**

**HJÄLP:** **man bash**

Ett alias är ett annat namn för ett kommando. Du kan ställa in dina alias för att exempelvis använda ett eller flera extraalternativ, eller helt förändra beteendet för ett kommando. Du kan ställa in dina alias under en pågående session med hjälp av alias-kommandot eller permanent i filen `~/.bashrc` i din hemmapp.

Alias-kommandot (som är inbyggt i Bash) visar dina aktuella alias.

```
alias
```

Detta visar alla alias som du använder för närvarande. Många distributioner ställer in ett antal alias från början, så det är en god idé att köra kommandot för att se vilka de är.

Du kan använda kommandot **unalias** för att ta bort ett alias. Du kan temporärt inaktivera av ett alias genom att lägga till ett \ (bakstreck) framför kommandot (se nedan). Då körs originalkommandot istället för aliaset.

Exemplet nedan visar hur en alias-sektion i din ~/.bashrc-fil kan se ut:

```
# fråga innan filer skrivs över,
# ange var filerna kopieras
alias cp='cp -vi'

# fråga innan borttagning
alias rm='rm -i'

#fråga innan filer skrivs över vid flyttning
alias mv='mv -i'
```

Du kan när som helst lägga till valfritt antal alias i filen. Nytillagda rader blir aktiva nästa du startar ett skal. Du kan alltså enkelt testa dina ändringar genom att öppna en ny terminal.

Tecknet bakstreck används för att köra ett kommando och förbigå eventuella alias.

Om **rm** exempelvis görs till ett alias för **rm -i** blir resultatet körs **rm -i** varje gång du skriver **rm**.

Men om du skriver **\rm** ignorerar skalet aliaset och kör **rm**, precis som du skrev in det. Då kommer kommandot inte fråga dig om du verkligen vill radera det du kör kommandot på.

Tänk på att aliaset för ta-bort-kommandot rm finns av en anledning. Om du använder kommandot oförsiktigt kan du råka radera filer som du inte vill ta bort.

Använd därför endast **\rm** om du vet vad du gör. Det är inte helt enkelt att återskapa raderade filer, **rm** raderar direkt utan att först lägga filerna i en papperskorg.

Läs mer om \ i ▷sektion 3.4.6.

## 6.2 set

**$ set** Läs och sätt variabler i Bash

**DEBIAN, UBUNTU:** **bash** **FEDORA:** **bash**

**HJÄLP:** **man bash**

**set** är ett av de många inbyggda kommandona som finns i Bash.

Om du anger alternativet **-x** skriver Bash ut namnet på det kommando som kommer att köras, precis innan det körs.

Detta är användbart när du vill förstå exakt vad som händer med vissa kommandon, exempelvis om det innehåller jokertecken eller speciella tecken som orsakar problem, eller krångliga alias.

Exempel:

```
set -x
```

Detta säger åt Bash att slå på funktionen att visa exakt vad som körs när du skriver in ett kommando. Om du till exempel skriver **ls** skriver skalet ut **+ ls -F --color=auto** innan det körs (exakt vad som visas beror på vilka alias, om något, som används på ditt system).

Detta betyder att kommandot egentligen är ett alias för **ls** med alternativen **-F** och **--color=auto**.

Använd **set +x** för att stänga av funktionen.

Kör kommandot **help set** för att se information om kommandots många alternativ, eller läs i Bash-manualen.

## 6.3 echo

**$ echo** Visar det du just skrev in i terminalen

**DEBIAN, UBUNTU:** **bash** **FEDORA:** **bash**

**HJÄLP:** **help echo**

Ett litet kommando som visar sina argument i terminalen.

Exempel:

```
echo "hej världen"
```

Då visas ”hej världen” i terminalen.

Pröva även alternativet **-e**. Då kan du använda escapetecken, \ (bakstreck) tillsammans med vissa tecken för att formatera visningen på skärmen. Exempel: **\t** för tabulator, **\n** för ny rad.

Exempel:

```
echo -e "hej\n hopp\n nytt\t hopp"
```

Ett användningsområde för **echo** är att förebygga olyckor. Genom att skriva in **echo** följt av det eller de kommandon du vill köra kan du undvika att råka ut för något du inte hade tänkt dig. Med **echo** ser du också hur jokertecken expanderar, vilket gör att du förstår exakt vad som händer innan du faktiskt kör kommandot.

Exempel:

```
echo rm -R *
```

Detta skriver ut vad som skulle ha raderats om du hade kört **rm**-kommandot med argumentet *. Om du sätter **echo** före ett kommando gör du det harmlöst (det expanderar endast eventuella jokertecken så att du ser vad som annars skulle ha skett).

## 6.4 script

---

**$ script** ”Spelar in” allt som visas i skalet

---

**DEBIAN, UBUNTU:** **bsdutils** **FEDORA:** **util-linux**

---

**HJÄLP:** **man script**

---

Kommandot **script** skapar en avskrift av en hel skalsession. Kommandot skriver en kopia av din session till en fil, inklusive kommandon du skriver och resultatet av dem.

Aktivera inspelningen genom att skriva **script**. Avbryt inspelningen genom att skriva **exit**.

Som standard sparas avskriften i en fil med namnet `typescript` i den mapp du stod i när du startade **script**.

## 6.5 reset

**$ reset** Intialisera terminalen

**DEBIAN, UBUNTU:** **ncurses-bin** **FEDORA:** **ncurses**

**HJÄLP:** **man reset**

**reset**-kommandot nollställer den aktuella terminalen. Det är användbart om ett program gör att texten i din terminal blir konstig. Skriv in kommandot, så fixar det saken.

Exempel:

```
reset
```

## 6.6 init=/bin/sh

Om du någon gång råkar ut för att du inte kan starta och logga in på din dator kan du genskjuta den normala startsekvensen och logga in med skalet **sh** på datorn (som är mindre och lättare än Bash), innan några andra program och processer hinner starta. Du kan göra detta genom att ge argumentet **init=/bin/sh** till systemet innan det startar.

De flesta Linuxdatorer använder startladdaren (*boot loader* på engelska) GRUB, och du kan redigera startargumenten i GRUB om du trycker ned ESC-tangenten precis när GRUB laddas i samband med datorstart.

# Kapitel 7

# Installera program

Det här kapitlet hjälper dig igång med de två vanligaste kommandoradsverktygen för att ladda ned och installera program i binärformat.

Kommandona nedan hjälper dig att snabbt och smidigt installera färdigkompilerade versioner av många av bokens kommandon på din dator.

## 7.1 apt-get (Debian, Ubuntu, Fink)

**OBS** Detta avsnitt gäller dig som har en Debian Linux-dator, eller använder en Debian- och dpkg-baserad, exempelvis Ubuntu. Avsnittet är också relevant för dig som använder Fink i Mac OS X.

Det bakomliggande pakethanteringssystemet i Debian Linux kallas dpkg (*Debian package management system* på engelska). Utöver dpkg, som bland annat innehåller en databas över alla tillgängliga och installerade program (och beroenden mellan olika paket) finns ett antal verktyg för att enkelt och smidigt ladda ned, kontrollera och installera paket i det så kallade deb-formatet på datorn.

Det vanligaste kommandoradsverktyget för att hantera dpkg kallas **`apt-get`**. Verktyget är standard i Debian och Debian-baserade distributioner som exempelvis Ubuntu.

Om du använder Mac OS X kan du med hjälp av Fink-projektet också använda **apt-get** för att hantera installation av UNIX-verktyg i binärformat på Macen. Instruktioner för hur du kommer igång med Fink finns på projektet hemsida, `http://fink.sourceforge.net`.

### 7.1.1 apt-get

**# apt-get** Pakethanterare för kommandoraden

**DEBIAN, UBUNTU: apt FEDORA: apt**

**HJÄLP: man apt-get**

**apt-get** laddar ned paket direkt från servrar på nätet, så kallade förråd (*repository* på engelska). Vilka förråd som är aktiva styrs av innehållet i filen `/etc/apt/sources.list`.

Kommandot måste köras som root-användaren (eller genom att använda **sudo**) och kräver en fungerande internetuppkoppling eller tillgång till ett paketförråd exempelvis på cd-skiva för att fungera.

#### Kontrollera om det finns uppdaterade paket

Kommandonsyntax för uppdatering av paketlistor i aktiverade förråd:

```
sudo apt-get update
```

#### Installera nya versioner av alla uppdaterade paket

Kommandosyntax:

```
sudo apt-get dist-upgrade
```

Detta laddar ned och installerar alla uppdaterade paket som finns tillgängliga i paketförråden.

#### Installera ett nytt paket

Kommandosyntax:

```
sudo apt-get install paketnamn
```

Exempel:

```
sudo apt-get install wget
```

Detta laddar automatiskt ned, installerar och konfigurerar programmet **wget**, med hjälp av **sudo** körs kommandot med administratörsrättigheter.

#### Ta bort ett paket

Du kan också använda **apt-get** för att ta bort paket från datorn.

Exempel:

```
sudo apt-get remove wget
```

Detta avinstallerar **wget**, men låter konfigurationsfiler vara kvar.

För att ta bort ett installerat paket, samt helt eliminera alla spår av det på datorn, inklusive inställningsfiler, kan du lägga till alternativet **--purge**.

Exempel:

```
sudo apt-get --purge remove wget
```

### 7.1.2 apt-cache

**$ apt-cache** Hanterare för paket-cachen för apt

**DEBIAN, UBUNTU:** **apt** **FEDORA:** **apt**

**HJÄLP:** **man apt-cache**

Med **apt-cache** kan du fritext-söka i aktiva paketförråd. Detta är exempelvis bra om du vill hitta ett paket som innehåller ett visst kommando eller program, men inte vet namnet på paketet.

Du behöver inte administratörsrättigheter för att köra kommandot, eftersom du endast söker efter filer och inte modifierar systemet som med **apt-get**.

Kommandosyntax:

```
apt-cache search sträng
```

Exempel:

```
apt-cache search mahjong
```

Detta söker efter paket som innehåller spelet mahjong. Du kan sedan installera önskat paket med hjälp av **apt-get install**.

## 7.2 yum (Fedora, RedHat, openSUSE)

**OBS** Detta avsnitt gäller dig som har en Fedora Linux-dator, eller använder en annan RPM-baserad distribution, exempelvis RedHat eller openSUSE.

### 7.2.1 yum

**# yum** RPM-pakethanterare för kommandoraden

**DEBIAN, UBUNTU: yum FEDORA: yum**

**HJÄLP: man yum**

**yum** laddar ned paket direkt från förråd på nätet. Aktiva förråd anges i filen `/etc/yum.conf`.

Kommandot måste köras som root-användaren (eller genom att använda **sudo**) och kräver en fungerande internetuppkoppling eller tillgång till ett paketförråd exempelvis på cd-skiva för att fungera.

Varje gång du kör **yum** kontrollerar det om det finns uppdaterade paket i något av paketförråden.

#### Installera nya versioner av alla uppdaterade paket

Kommandosyntax:

```
yum update
```

Detta laddar ned och installerar alla uppdaterade paket som finns tillgängliga i paketförråden.

#### Installera ett nytt paket

Kommandosyntax:

```
yum install paketnamn
```

Exempel:

```
yum install wget
```

Detta laddar automatiskt ned, installerar och konfigurerar programmet **wget**.

### Ta bort ett paket

Kommandosyntax:

```
yum remove paketnamn
```

Exempel:

```
yum remove wget
```

Detta avinstallerar **wget**, men låter konfigurationsfiler vara kvar.

### Söka efter paket

Kommandosyntax:

```
yum list "*games*"
```

Detta söker efter paket vars namn innehåller order ”games”. Du kan sedan installera önskat paket med hjälp av **yum install**.

# Kapitel 8

# Hantera filträdet

Det här kapitlet förklarar ett antal kommandon som du kan använda för att flytta dig i filsystemet och hantera filer. Här förklaras också hur du kan söka efter filer och hur du kan döpa om flera filer åt gången.

## 8.1 Förflytta dig i filträdet

### 8.1.1 cd

| | |
|---|---|
| **$ cd** | Gå till en annan mapp |
| **DEBIAN, UBUNTU:** | **bash** **FEDORA:** **bash** |
| **HJÄLP:** | **man bash** |

Används för att gå till en annan mapp (cd står för *change directory* på engelska).

Använd **cd ..** för att hoppa ned en mapp i hierarkin (närmare roten i filträdet).

En punkt, **.**, anger den nuvarande mappen, två punkter, **..**, anger föräldramappen.

**cd –** gör att du hoppar till föregående mapp (tänk på den som en ångrafunktion).

Du kan använda **cd** med absoluta och relativa sökvägar (*path* på engelska).

**Absoluta sökvägar** En absolut sökväg känns igen på det inledande snedstrecket, /, (*forward slash* på engelska).

Om du till exempel vill hoppa till /boot/grub skriver du:

```
cd /boot/grub
```

Detta är en absolut sökväg eftersom du börjar vid roten i filträdet och rör dig uppåt från den punkten. Det spelar ingen roll var du befinner i filträdet när du skriver kommandot, du kommer till samma plats ändå.

**Relativa sökvägar** En relativ sökväg saknar det inledande snedstrecket. Använd en relativ sökväg när du utgår från den mapp du befinner dig i. Detta fungerar oberoende av var du är i filträdet.

Om du till exempel befinner dig i root-användarens hemmapp och vill gå till mappen ”/root/musik” kan du skriva:

```
cd musik
```

Lägg märke till att det saknas ett snedstreck i ovanstående **cd**-kommando. Om du hade infogat ett **/** framför ”musik” hade det blivit en absolut sökväg, då hade du utgått från roten i filträdet.

### 8.1.2 ls

**$ ls** Visa mappinnehåll

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man ls**

Visa filer och mappar. Genom att skriva **ls** visar du filer och mappar, förutom gömda mappar och filer. (En mapp eller fil är gömd om den inleds med en punkt, ’.’.)

**ls** visar allt i mappen du står i. Alternativ är helt frivilliga.

Kommandosyntax:

```
ls -alternativ sträng
```

Detta visar filer med namn som matchar strängen. Strängen kan innehålla jokertecken för att visa flera filer. Se ▷sektion 3.4 för mer information om jokertecken.

Exempel på alternativ till kommandot:

```
ls -l
```

Alternativet **-l** betyder för att **ls** visar resultatet utförligare än utan alternativet. Både rättigheter, filstorlekar, ändringsdatum och ägare visas.

```
ls -a
```

Alternativet **-a** betyder att alla filer, inklusive gömda filer som börjar med en punkt, '.', visas.

```
ls -d
```

Visar enbart mappar, inte filer (se exempel nedan).

```
ls -F
```

Med alternativet **-F** kan du lägga till tecken för olika typer av filer, exempelvis '*' (asterisk) för körbara filer.

```
ls -S
```

Med alternativet **-S** sorteras utdatat i bokstavs- och storleksordning.

```
ls -R
```

Alternativet **-R** gör att kommandot visar innehållet rekursivt, det vill säga också inkluderar allt innehåll i mappar i mappen.

Med hjälp av ls -d kan du visa mappar som matchar en sträng exakt, eller använda jokertecken för att visa flera. Skriv **ls -d */** för att visa alla mappar i mappen du befinner dig i. Om dina alias är konfigurerade så kan du eventuellt använda **lsd** istället för **ls -d */**.

Exempel med ls -d:

```
ls -d */
```

Visar alla mappar som finns i mappen du befinner dig i.

```
ls -d sträng*
```

Visar alla mappar som börjar med ”sträng”.

```
ls -d /usr/*/*/doc
```

Visar alla mappar som är två nivåer ovanför mappen ”/usr/” och som innehåller en mapp med namnet ”doc”. Det här knepet är mycket användbart ibland.

**TIPS** Du kan eventuellt också använda **`l`**, **`la`** (visa alla) eller **`ll`** (visa detaljerad lista) om dina alias är inställda för det.

### 8.1.3 pwd

**`$ pwd`** Visa namnet på aktuell mapp

**DEBIAN, UBUNTU:** **`coreutils`** **FEDORA:** **`coreutils`**

**HJÄLP:** **`man pwd`**

**`pwd`** (förkortning av *print working directory* på engelska). Visar den absoluta sökvägen till mappen du befinner dig i.

Kommandosyntax:

```
pwd
```

Detta ger dig den kompletta sökvägen till mappen du befinner dig i, exempelvis ”/usr/local/bin” om du befinner dig i den mappen.

### 8.1.4 tree

**`$ tree`** Visa mappinnehåll som en trädstruktur

**DEBIAN, UBUNTU:** **`tree`** **FEDORA:** **`tree`**

**HJÄLP:** **`man tree`**

Visar ett ”grafiskt” träd med sin början i mappen du befinner dig i. Detta kommando letar sig rekursivt igenom alla mappar som finns i mappen du befinner dig i.

Med andra ord visar det alla filer i alla mappar, liksom alla filer i mappen du står i.

Det finns en stor uppsättning alternativ till **tree**, se manual-sidan för detaljerad information om dessa.

Kommandosyntax:

```
tree
```

eller

```
tree -alternativ /valfri/sökväg/att/visa
```

## 8.2 Söka efter filer

### 8.2.1 find

**$ find** Söker efter filer i en mappstruktur

**DEBIAN, UBUNTU:** **findutils** **FEDORA:** **findutils**

**HJÄLP:** **man find**

**find** är ett verktyg som letar efter filer på hårddisken. Kommandot har ett stort antal alternativ som kan användas för att anpassa sökningen. Se kommandots manual- och info-sidor för fördjupad information.

Lägg märke till att **find** fungerar bra med vanliga jokertecken (se ▷sektion 3.4), liksom med reguljära uttryck (se kapitel 20).

Exempel:

```
find / -name filnamn
```

Med detta exempel söker du efter en fil men namnet ”filnamn” och börjar sökningen i roten i filträdet (den söker i alla mappar, inklusive monterade filsystem).

Alternativet **-name** tar hänsyn till versaler och gemener. Om du vill hitta all förekomster oberoende av teckentyp kan du använda alternativet **-iname** istället.

Använd alternativet **-regex** eller **-iregex** för att använda reguljära uttryck (första varianten gör skillnad på gemena och versala tecken, det andra inte).

Alternativet **-exec** ger möjlighet till mer avancerade sökningar. Med alternativet körs ett kommando på de filer som hittas (exempelvis för att flytta, ta bort eller göra något annat med sökresultaten).

Börja med att använda **find** för att göra en vanlig sökning som ett test, lägg sedan till **-exec** i slutet av raden och skriv därefter följande:

kommando_att_köra (1), sedan {} (måsvingeparenteser) (2), och slutligen argumenten (exempelvis en ny mapp) samt ett semikolon, ; (3).

En detaljerad beskrivning:

**(1)** Detta är verktyget som du vill köra på filerna som **find** hittar. Om du till exempel vill ta bort allt det hittar använder du **-exec rm -f**.

**(2)** Måsvingeparentesen används av **find** för att benämna filen som **find** har hittat. Om det till exempel har hittat en fil med namnet ”shopping.doc” byts {} ut mot shopping.doc. Därefter fortsätter den med att byta ut {} mot varje fil som den har hittat. Måsvingarna skyddas vanligen med bakstreck (\) eller en apostrof ('), så att Bash inte expanderar dem (det vill säga försöker tolka dem som några särskilda kommandon, exempelvis jokertecken).

**(3)** Detta är tecknet som används för att signalera slutet på kommandona. Det används ofta med ett skyddande bakstreck (\) eller apostrofer för att hindra Bash från att försöka tolka det.

Exempel:

```
find / -name '*.doc' -exec cp '{}' /tmp/ ';'
```

Ovanstående kommando försöker hitta filer med filändelsen ”.doc” och kopierar dem till mappen ”/tmp”. Kommandot är kanske inte särskilt användbart, men det visar på vad som är möjligt att göra. Observera apostroferna som hindrar Bash från att försöka tolka tecknen som något annat.

**OBS** I ▷sektion 19.5 hittar du exempel på hur du kan använda **find** tillsammans med **xargs** för att bygga avancerade kommandon.

Det kan vara knepigt att lista ut hur man kan exkludera vissa mappar med **find**, men det kan vara mycket bra att känna till hur man gör om du till exempel vill söka på hårddisken men inte på monterade filsystem. Använd

alternativet **-path** för att exkludera en specifik mapp, och alternativet **-prune** för att exkludera mappar i mappen.

Exempel:

```
find / -path '/mnt/win' -prune -o -name "string" -print
```

Det här exemplet söker i hela filträdet exklusive en mapp med namnet ”/mnt/win” och dess undermappar. Alternativet **-path** fungerar med jokertecken. Notera att det går bra att lägga till valfritt antal **-path** med argument till kommandot.

Det finns en stor mängd alternativ som du kan använda med **find**, se manualen (och info-sidorna) för mer information.

### 8.2.2 locate

**$ locate** Söker efter filer med hjälp av en indexerad databas

**DEBIAN, UBUNTU:** **mlocate** **FEDORA:** **mlocate**

**HJÄLP:** **man locate**

Med kommandot **locate** kan du mycket snabbt hitta en fil eller en mapp som du söker. Kommandot skriver ut en lista med alla filer på systemet som matchar din sökning. Träffarna måste inte vara exakta, allt som innehåller sökordet visas.

**OBS** I många Linuxdistributioner (inklusive Ubuntu och Fedora) är kommandot **locate** ett alias för **mlocate**. Kommandot **mlocate** är en mer modern variant av original-**locate**, men funktionen är helt och hållet densamma.

Kommandosyntax:

```
locate sträng
```

**OBS** Innan du kör **locate** måste du uppdatera databasen med filer på systemet. Detta gör du genom att köra **updatedb** (▷sektion 8.2.3) som root.

### 8.2.3 updatedb

**# updatedb** Uppdaterar databasen för **locate**

**DEBIAN, UBUNTU:** **mlocate** **FEDORA:** **mlocate**

**HJÄLP:** **man updatedb**

Kommandot **updatedb** uppdaterar den indexerade databas som **locate** använder för att snabbt hitta det du söker.

Observera att du måste köra kommandot **updatedb** som root-användaren.

Exempel:

```
sudo updatedb
```

Detta kör kommandot **updatedb** med administratörsrättigheter.

### 8.2.4 whereis

**$ whereis** Söker efter filer i en mappstruktur

**DEBIAN, UBUNTU:** **util-linux** **FEDORA:** **util-linux**

**HJÄLP:** **man whereis**

Kommandot **whereis** söker efter körbara filer, källkod och manualsidor för ett visst program. Det visar endast exakta träffar, så om du bara känner till en del av namnet måste du söka efter det först, exempelvis med **locate**.

Kommandosyntax:

```
whereis programnamn
```

### 8.2.5 which

**$ which** Visa var ett kommando finns

**DEBIAN, UBUNTU:** **debianutils** **FEDORA:** **which**

**HJÄLP:** **man which**

Det här kommandot är nästan samma sak som **whereis**, bortsett från att det endast söker efter körbara filer (pro-

gram). Det söker endast i miljövariabeln `$PATH` i användarens skal.

Använd alternativet **`-a`** för att se alla förekomster av ett visst programnamn i din `$PATH` (om det finns fler än ett ser du alla).

Kommandosyntax:

```
which programnamn
```

## 8.3 type

**`$ type`** Visar typ av kommando

**DEBIAN, UBUNTU:** **`bash`** **FEDORA:** **`bash`**

**HJÄLP:** **`man bash`**

Visar om ett kommando är inbyggt i skalet eller ett fristående program, liksom var det finns i filsystemet. Visar också om ett kommando är ett alias för ett annat kommando.

Kommandosyntax:

```
type kommando
```

Exempel:

```
type ls
```

## 8.4 Jobba med filer och mappar

### 8.4.1 mkdir

**`$ mkdir`** Skapa en mapp

**DEBIAN, UBUNTU:** **`coreutils`** **FEDORA:** **`coreutils`**

**HJÄLP:** **`man mkdir`**

Mappar kallas ibland för kataloger eller bibliotek, men i den här boken använder vi termen mapp som oftare används i Windows- och Mac OS-sammanhang.

Använd kommandot **`mkdir`** för att skapa en mapp.

Kommandosyntax:

```
mkdir foo
```

Detta skapar en mapp med namn ”foo” i den mapp du befinner dig i.

Använd **`mkdir -p`** för att skapa undermappar automatiskt.

Exempel:

```
mkdir -p /home/bertil/plugg/matte
```

Detta skapar mappen ”matte” och ”plugg” i bertils hemmapp (hade bertils hemmapp inte funnits hade det också skapats).

### 8.4.2 rm

**`$ rm`** Radera fil(er)

**DEBIAN, UBUNTU:** **`coreutils`** **FEDORA:** **`coreutils`**

**HJÄLP:** **`man rm`**

Använd **`rm`** för att radera en eller flera filer och/eller mappar. Du kan använda vanliga jokertecken med det här kommandot (se ▷ sektion 3.4).

Kommandosyntax:

```
rm -alternativ fil_eller_mapp
```

Du kan förstås använda vanliga jokertecken för att radera flera filer åt gången.

Använd alternativet **`-R`** eller **`-r`** för att ta bort rekursivt, det vill säga ta bort allt som finns i en mapp, inklusive andra mappar. Pröva också alternativet **`-f`** för att tvinga borttagning, det vill säga utan att fråga dig innan kommandot faktiskt tar bort något.

Exempel:

```
rm -rf temp-filer
```

Detta tar bort mappen ”temp-filer”, inklusive allt innehåll.

**TIPS** På vissa system, som exempelvis Mandrake finns det ett alias för **rm** som pekar på **rm -i** (vilket gör att du får frågan om du vill radera varje enskild fil som kommandot försöker ta bort). Om du vill förbigå aliaset kan du använda **\rm -r mapp** (med hjälp av \ slår du temporärt av aliaset).

### 8.4.3 rmdir

**$ rmdir** Radera en tom mapp

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man rmdir**

Om du vill ta bort en mapp som innehåller filer använder du **rm -R mapp**, läs ovan för mer information om **rm -r**.

Kommandosyntax:

```
rmdir mapp
```

Detta tar endast bort mappen om den är tom, annars visas ett felmeddelande.

### 8.4.4 mv

**$ mv** Flytta/döp om en fil eller mapp

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man mv**

Flytta en fil eller mapp till ett nytt ställe eller döp om en fil eller mapp.

Exempel:

```
mv filenamn1 filenamn2
```

Döper om filnamn1 till filnamn2.

För att flytta en fil eller mapp skriver du:

```
mv originalfil_eller_mapp ny_plats
```

Observera att det går bra att använda vanliga jokertecken vid flyttning (ej omdöpning). Läs mer i ▷sektion 3.4.

**TIPS** Du kan också flytta och samtidigt döpa om en fil i ett enda kommando. Skriv till exempel:

```
mv /etc/config.txt /backup/config-kopia.txt
```

Detta flyttar filen ”config.txt” till ”/backup” och döper om den till ”config-kopia.txt” i ett svep.

### 8.4.5 cp

**$ cp** Kopiera en fil eller mapp

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man cp**

Kopiera en fil. Kommandot har en uppsättning användbara alternativ, som **-R** (eller **-r**), vilken kopiera mappar (och mappar däri) rekursivt.

Kommandosyntax:

```
cp -alternativ fil_eller_filer ny_plats
```

Exempel:

```
cp fil1 fil2
```

Detta kopierar helt enkelt fil1 till fil2 (i samma mapp).

```
cp /tmp/fil1 ~/fil2 /mnt/win_c
```

Det sista argumentet är destinationsmappen; ovanstående kommando kopierar två filer från två olika platser till ”/mnt/win_c”.

```
cp -R mapp_och_eller_filer ny_plats
```

Detta kommando kopierar mappar (och alla mappar däri) och/eller filer till ”ny_plats”.

Observera att det går bra att använda vanliga jokertecken vid flyttning av flera filer samtidigt. Läs mer i ▷sektion 3.4.

### 8.4.6 ln

**$ ln** Skapar en länk mellan filer

**DEBIAN, UBUNTU:** `coreutils` **FEDORA:** `coreutils`

**HJÄLP:** `man ln`

Skapar en länk till en fil (fungerar ungefär som genvägar i Windows och Mac OS). Det finns två typer av länkar:

**Hårda länkar** Hårda länkar är pekare till en fil. Flera hårda länkar kan peka på samma fil.

Filen försvinner först efter att alla hårda länkar har raderats. Om du raderar originalfilen men inte de hårda länkarna finns de hårda länkarna kvar, liksom fildatan.

Exempel:

```
ln originalfil länkfil
```

Detta skapar en hård länk med namnet ”länkfil” till filen ”originalfil”. Du måste radera båda för att helt ta bort filen.

**Symboliska länkar** Symboliska länkar är ”mjuka” länkar liknande genvägar i Mac OS och Windows. Symboliska länkar skapas genom att skriva **`ln -s`**. Om du tar bort originalfilen blir den symboliska länken trasig.

Exempel:

```
ln -s originalfil länkfil
```

Detta skapar en symbolisk länk med namnet ”länkfil” till filen ”originalfil”. Om du raderar originalfilen fungerar den symboliska länken inte längre.

Fördelen med symboliska länkar är att länken kan peka på en fil på ett annat filsystem, till skillnad från hårda länkar som endast kan peka på filer på samma filsystem.

### 8.4.7 shred

**$ shred** Skriver över och raderar filer

DEBIAN, UBUNTU: **coreutils** FEDORA: **coreutils**

HJÄLP: **man shred**

Kommandot **shred** tar bort en fil säkert, utan möjlighet att återskapa filen senare. Kommandot skriver över filen med nonsensdata för att eliminera alla spår av filen. Detta gör det omöjligt för specialmjukvara (och oftast även specialhårdvara) att återskapa filen. Var försiktig med kommandot, eftersom du därmed aldrig kommer kunna återskapa informationen du har kört kommandot på.

Ett farligt exempel kan se ut så här:

```
shred -n 2 -z -v /dev/hda1
```

Ur info-sidan för shred:

”Detta säger åt **shred** att skriva över partitionen hda1 två gånger med slumpmässig data (-n 2) och sedan skriva över den igen med nollor (-z) medan den visar dig vad den gör (-v). Naturligtvis måste du byta ut /dev/hda1 till den partition du vill radera. Eftersom varje skrivomgång kan ta lång tid görs detta endast två gånger med slumpdata istället för standardvärdet 25. Du kan justera antalet skrivningar till din egen nivå av paranoia och tid du kan undvara. Då **shred** skriver på en låg nivå spelar det ingen roll vilken typ av filsystem du har på partitionen – inget blir möjligt att återskapa. När **shred** är klar kan du stänga ned maskinen och sälja eller slänga den utan att behöva oroa dig. [. . . ] Dock är det inte helt säkert att hårddiskar du kör **shred** på helt töms på känslig data. Till exempel markerar och undviker de flesta hårddiskar automatiskt trasiga disksektorer, och dessa sektorer blir osynliga för **shred**. Om de trasiga sektorerna innehåller känslig data kan inte **shred** radera något.”

OBS Kommandot **shred** fungerar inte med alla filssystem. Kommandot fungerar inte med journalförande filsystem som jfs, reiserfs, xfs och andra moderna filsystem (**shred** fungerar dock med ext3 om journal-läget först stängs av).

### 8.4.8 du

**$ du** Uppskattar storleken på filer och mappar

**DEBIAN, UBUNTU:** `coreutils` **FEDORA:** `coreutils`

**HJÄLP:** `man du`

Visar information om hårddiskanvändningen (**du** står för *disk usage* på engelska).

Kommandosyntax:

```
du -alternativ filer_och_eller_mappar
```

Om du använder kommandot utan alternativ och argument fungerar det rekursivt med utgångspunkt i den mapp du står i. Några alternativ du kan använda med **du**:

```
du -c
```

Detta gör så att **du** efter storleken på varje enskild mapp visar en totalsumma av alla argument.

```
du -s
```

Summerar varje enskilt argument utan att visa informationen rekursivt.

```
du -h
```

Skriver ut resultatet i enklare läsbar form (*human readable* på engelska), exempelvis 1M (megabyte) istället för 1 048 576 (byte).

Om du använder alternativen **-hs** för en mapp visas totala storleken på mappen och alla andra mappar däri.

Exempel:

```
du -hs *
```

Detta kommando visar storleken på alla filer i den mapp du står i, liksom mapparna däri. Det visar resultatet i enklare läsbar form.

### 8.4.9 file

**$ file** Visar filtypen hos en fil

**DEBIAN, UBUNTU:** **file** **FEDORA:** **file**

**HJÄLP:** **man file**

Det här kommandot utröner vad slags typ en fil är, exempelvis binär-, jpeg-, bmp-, ASCII-text- eller C header-fil, och så vidare. **file** är ett mycket användbart kommando om behöver identifera filtypen för filer på hårddisken som saknar filändelse eller har fel filändelse.

Kommandosyntax:

```
file filnamn
```

### 8.4.10 stat

**$ stat** Visar information om filer

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man stat**

Ger detaljerad information – statistik – om en fil, inklusive vem som äger filen, filrättigheter, hur mycket utrymme den tar upp på hårddisken, när filen ändrades på hårddisken eller när innehållet lästes. Har också många avancerade alternativ och användningsområden. Se manual-sidan.

Exempel på enkel användning:

```
stat filnamn
```

### 8.4.11 dd

**$ dd** Konvertera och kopiera filer

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man dd**

Kopierar data på en mycket låg nivå, kan användas för att kopiera hela hårddiskar, cd-avbildningar med mera (se ▷sektion 8.4.11).

Kommandot kan också utföra konverteringar av filer och ändra den så kallade blockstorleken när den skriver filer. Se manual-sidan för mer avancerad information.

Det är frivilligt att ange blockstorlek, **bs** (*block size* på engelska) och antal block, **count** (*count* på engelska). Du kan också använda kommandot på filer istället för enheter.

**OBS** Kommandot **dd** är avancerat och kan vara svårt att använda. Det är också mycket kraftfullt, så var försiktig när du använder det.

Kommandosyntax:

```
dd if=/dev/xxx of=/dev/xxx bs=xxxx count=x
```

**OBS** Kommandot arbetar på mycket låg nivå och kan användas för att skriva över viktiga delar av en hårddisk, som exempelvis den så kallade Master Boot Record (MBR), som är de första 512 byten på hårddisken. MBR är nödvändig för att datorn ska kunna starta operativsystemet.

### Duplicera enheter/skivor

Vanligtvis kopierar du data med hjälp av kommandot **cp**, men det är inte alltid det räcker. Kanske vill du till exempel kopiera en hel dvd-skiva till en iso-fil, eller klona innehållet på en hel hårddisk till en annan? Då är kommandot **dd** ett utmärkt hjälpmedel.

Till skillnad från **cp**, som kopierar en eller flera filer, använder du **dd** för att kopiera hela block på filsystemsnivå. Det gör det möjligt att använda **dd** för att klona hela hårddiskar, eller göra kopior av cd-/dvd-skivor till iso-filer, med mera.

Kommandot har en lite speciell kommandosyntax, där **if=** används för att ange inmatningsfilen (*input file* på engelska och **of=** används för att ange utmatningsfilen (*output file* på engelska).

Exempel:

```
dd if=/dev/fd0 of=diskettavbild
```

I ovanstående exempel används **dd** för att klona disketten i enheten ”/dev/fd0” till filen ”diskettavbild”.

```
dd if=diskettavbild of=/dev/fd0
```

I detta exempel klonas innehållet i filen `diskettavbild` till disketten i enheten ”/dev/fd0”. Observera att eventuellt befintligt innehåll skrivs över.

```
dd if=/dev/sda of=/dev/sdb
```

I ovanstående exempel klonas hela hårddiskinnehållet på ”/dev/sda” till hårddisken”/dev/sdb’. Resultatet blir en exakt kopia av hela hårddisken.

```
dd if=/dev/cdrom of=/tmp/cdrom.iso
```

Ovanstående klonar innehållet på cd-skivan i enheten ”/dev/cdrom” till skivavbildningen ”cdrom.iso”.

### 8.4.12 touch

**$ touch** Ändra tidstämpeln på filer

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man touch**

Det är kommandot kan användas för att skapa tomma filer med **touch filnamn**. Den kan också uppdatera tidsstämplarna på filer (tänk på kommandot som att du rör vid filen (*touch* på engelska).

Ändra tid och/eller datum på en fil:

```
touch -t 05070915 min_rapport.txt
```

Detta ändrar tidsstämpeln på filen ”min_rapport.txt” så att det ser ut som om du hade skapat den 7 maj (0507) klockan 9:15 (0915). Om du kör kommandot utan alternativ sätts aktuell tid på filen.

Istället för att använda enkla siffror för att ändra tiden kan du också använda alternativ som liknar de för kommandot **date**.

Exempel:

```
touch -d '5 May 2000' en_fil.txt
```

Du kan också använda **--date=** istället för **-d**. Titta också närmare på informationen om kommandot **date** (▷ sektion 9.2.1) för fler exempel på hur du kan använda dessa al-

ternativ. Syntaxen för datum och tid är exakt desamma som för **date**.

### 8.4.13 split

**$ split** Dela upp en fil i bitar

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man split**

Delar filer i mindre delar. Detta kan vara praktiskt till exempel om du måste bränna ned en fil på en dvd-skiva, men filen är för stor för att rymmas på en skiva. Lösningen är att dela upp filen i så många delar som behövs och bränna de delarna på varsin skiva.

Använd alternativet **–b xx** för att dela upp en fil i delar som är xx byte stora. Pröva också alternativet **–k** för kilobyte och **–m** för megabyte.

Detta kan vara användbart om du önskar dela upp stora filer för att kunna ta dem med dig på disketter eller andra lagringsmedia.

Kommandosyntax:

```
split –alternativ filnamn
```

Det kan också komma till användning om du vill dela upp textfiler till givna storlekar. Utan några alternativ delar kommandot upp en textfil i 1000-radiga delar.

Exempel:

```
split filnamn
```

Detta delar upp filen i 1 000-radiga delar och ger delarna namnet ”xaa” (första delen av filen), ”xab” (andra delen av filen), ”xac” (tredje delen av filen) och så vidare, tills det inte finns något mer av ursprungsfilen att dela upp.

**OBS** Kommandot är tämligen meningslöst om man inte senare kan sätta ihop delarna igen. Tursamt nog kan kommandot **cat** (▷ sektion 12.2.4) hjälpa till med det. I textfilsexemplet ovan kan du sätta hop filerna igen med kommandot **cat x* > ihopslagen_fil**.

## 8.5 Verktyg för massomdöpning och -flyttning

I GNU/Linux finns det några olika sätt att utföra massomdöpning på. Det finns också ett perl-skript som är specialiserat på att döpa om filändelserna på filer.

Nedan redovisas tre olika sätt att döpa om flera filer åt gången, med kommandona **mmv**, **rename** (ett perl-skript) och ett alternativ som involverar ett skalskript för Bash.

### 8.5.1 mmv

---

**$ mmv** Flytta, kopiera, foga till och länka flera filer

---

**DEBIAN, UBUNTU:** **mmv** **FEDORA:** **mmv**

---

**HJÄLP:** **man mmv**

---

Kommandot **mmv** (förkortning av *mass move* på engelska) är ett verktyg för mass-flyttning, -kopiering och -omdöpning och som använder vanliga jokertecken.

Exempel:

```
mmv \*.JPG \#1.jpg
```

Det första jokertecknet, *, avgränsar allt som slutar på ".JPG" och döper om ändelsen på varje enskild fil (#1 matchar det första jokertecknet) till ".jpg". Bakstrecket, '\', skyddar tecknen från att tolkas av skalet redan innan programmet hunnit titta på dem.

Varje jokertecken du använder kan du matcha mot ett motsvarande #x (där x är ett löpnummer, det första är 1).

I paketet som innehåller kommandot **mmv** följer också en uppsättning andra kommandon: **mcp** (*mass copy* på engelska), **mad** (*mass append* på engelska) och **mln** (*mass link* på engelska). Dessa kommandon fungerar med många av alternativen för **mmv**. Läs mer om kommandona i respektive manual-sida.

### 8.5.2 rename

**$ rename** Döp om flera filer

DEBIAN, UBUNTU: **rename-utils** FEDORA: **-**

HJÄLP: **man rename**

Kommandot **rename** är ett perl-skript som kan användas för att döpa om flera filer samtidigt. Kommandot använder reguljära uttryck.

Ett exempel för att ändra filändelserna för alla filer i den aktuella mappan från ".JPG" till ".jpg":

```
rename 's/\.JPG$/.jpg/' *.JPG
```

OBS Om skriptet inte finns tillgängligt i ett installationspaket för din distribution kan du ladda ned det från nätet. Hämta det från `http://search.cpan.org/~pederst/rename/`.

### 8.5.3 bash-skript

Att göra ett Bash-skript kan vara ett annat sätt att döpa om filer. Du kan själv skriva en liten uppsättning instruktioner (ett så kallat skript) som gör jobbet. Ett skript som det följande kan vara effektivt för omdöpning av filer om du inte har tillgång till **mmv** eller **rename**.

Ett sätt att döpa om jpg-filerna vi använde i ovanstående exempel är följande:

```
for i in *.JPG;
do mv $i `basename $i JPG`jpg;
done
```

Den första raden instruerar Bash att för varje fil med ändelsen ".JPG" (endast versaler, eftersom UNIX-filsystem gör skillnad på versaler och gemener) stoppa in filnamnet i variabeln 'i' (det är ett helt godtyckligt namn, du kan använda något annat om du vill).

I den andra raden används först kommandot **basename** för att strippa bort ändelsen JPG, varpå kommandot **mv** lägger till ändelsen jpg. (Lägg märke till att vi i skriptet använder oss av kommandosubstitution. Läs mer om det på ▷sektion 19.3. Information om kommandot **basename** finns i manual-sidan.)

Skriptet går på så sätt genom varje fil – en fil i taget – tills det inte finns några fler filer att döpa om. Sista raden säger åt Bash att skriptet är färdigt, varpå det avslutas.

Ett alternativt sätt att göra samma sak är följande:

```
for i in *.JPG;
do mv $i ${i%%.JPG}.jpg;
done
```

Ovanstående skript döper om filerna med en inbyggd funktion i Bash.

**TIPS** Läs mer om hur du kan skapa smarta Bash-skript i den fria boken *Advanced Bash scripting guide* (`http://tldp.org/LDP/abs/html/`) av Mendel Cooper.

# Kapitel 9

# Få information om ditt system

När något går galet är det bra att kunna ta reda på vad det underliggande problemet är. Det här kapitlet innehåller kommandon som hjälper dig att få information om ditt system.

## 9.1 Verktyg för systeminformation

### 9.1.1 time

| | |
|---|---|
| **$ time** | Mäter resursanspråk för program |
| **DEBIAN, UBUNTU:** | **time** **FEDORA:** **time** |
| **HJÄLP:** | **man time** |

**time** är ett verktyg som mäter hur lång tid det tar för ett program att köra. Det mäter också processoranvändningen och visar statistik.

**OBS** Information om hur du ändrar tid och datum på datorn finns i ▷sektion 9.2.1.

Använd **time -v** (v står för utförlig, talrik, *verbose* på engelska) för att se ännu mer detaljerade information om programmet du testar.

Exempel:

```
time programnamn alternativ
```

### 9.1.2 dmesg

**$ dmesg** Visa startmeddelanden från kärnan

**DEBIAN, UBUNTU:** **util-linux** **FEDORA:** **util-linux**

**HJÄLP:** **man dmesg**

Kommandot **dmesg** används för att visa den så kallade kernel ring buffer (till den skriver kärnan och kärnmoduler meddelanden när datorn startar). Under felsökning ger kommandot **dmesg** ofta bra information om vad som kan ha gått galet.

Testa att skriva:

```
dmesg
```

Detta visar dig informationen som kärnan skrev till *kernel ring buffer* vid start.

### 9.1.3 df

**$ df** Visa information om diskanvändning

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man df**

Visar information om hårddiskutrymmet på datorn. Använd alternativet **-h** för att be **df** att visa resultatet i mer lättläst form: istället för 1 024 kilobytes säger kommandot istället 1MB.

Kommandosyntax:

```
df -alternativ /dev/hdx
```

Alternativet som pekar ut vilken hårddisk som ska kollas är helt frivilligt. Om du inte anger något särskilt filsystem

visar kommandot utrymmet på alla filsystem. Du kan också ange ett mappnamn.

Exempel:

```
df -h /data/backup
```

### 9.1.4 who

**$ who** Visa vilka som är inloggade

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man who**

Visar information om vilka användare som är inloggade eller har varit inloggade på datorn, inklusive tidpunkten då de loggade in.

Kommandosyntax:

```
who
```

. . . i all enkelhet.

### 9.1.5 w

**$ w** Visa vilka som är inloggade och vad de gör

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man w**

Visar information om vilka som är inloggade på systemet och vad de gör för tillfället (det vill säga vilka processer som de kör). Det liknar kommandot **who** men visar delvis annorlunda information.

Kommandosyntax:

```
w
```

### 9.1.6 users

**$ users** Visa vilka som är inloggade och vad de gör

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man users**

Även detta kommando är mycket likt **who**, bortsett från att det endast visar vilka användare som är inloggade för tillfället. Inga alternativ kan användas med kommandot.

Kommandosyntax:

```
users
```

### 9.1.7 last

**$ last** Visa senast inloggade användare

**DEBIAN, UBUNTU:** **sysvutils** **FEDORA:** **sysvutils**

**HJÄLP:** **man last**

Visar information om när användare på datorn loggade in och ut. Detta inkluderar information om när datorn startades om.

Skriv helt enkelt följande för att köra kommandot:

```
last
```

### 9.1.8 lastlog

**$ lastlog** Visa de senaste inloggningarna för användare

**DEBIAN, UBUNTU:** **login** **FEDORA:** **login**

**HJÄLP:** **man lastlog**

Visar en lista på användare och vid vilken tidpunkt (dag och klockslag) som det loggade in på systemet.

Kommandosyntax:

```
lastlog
```

### 9.1.9 whoami

**$ whoami** Visa vem jag är

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man whoami**

Visar användaren vilket namn som användes för inloggning. Är bra för att råda bot på den identitetsförvirring som kan uppstå vid användning av kommandon som **su** (se kapitel 11). Det finns inga alternativ som kan användas med kommandot.

Skriv helt enkelt:

```
whoami
```

### 9.1.10 free

**$ free** Visa information om minnesanvändningen på systemet

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man free**

Visar information om minnesanvändningen (total, fritt minne (*free* på engelska), använt minne (*used* på engelska), cachat minne (*cached* på engelska), växlingsutrymme (*swap* på engelska). Använd alternativet **-t** för at visa totalsummor av allt. Använd alternativet **-m** för att se informationen i megabyte.

Exempel:

```
free -tm
```

Detta visar den totala minnesanvändningen i megabyte.

### 9.1.11 uptime

**$ uptime** Visa hur länge systemet varit igång

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man uptime**

Visar hur länge datorn har varit påslagen. Det visar också antalet användare och statistik om processoranvändningen.

**TIPS** Kommandot **w** använder sig av **uptime** för att visa tiden datorn har varit igång. Du kan alltså lika gärna använda dig av **w** för att se informationen.

### 9.1.12 uname

**$ uname** Visa systeminformation

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man uname**

Kommandot **uname** används för att visa information om systemet: typ av operativsystem, kärnversion och så vidare.

Några alternativ till **uname**:

**-a** Visa all tillgänglig information.

**-m** Visa information som är relaterad till själva datorn.

**-n** Visa bara datorns värdnamn.

**-r** Visa versionsnumret för den aktuella kärnan.

**-s** Visa namnet på operativsystemet.

**-p** Visa processortypen.

Kommandosyntax:

```
uname -alternativ
```

## 9.2 Datum, tid och kalendrar

Det finns ett kommando som du kan använda för att ändra både datum och tid på UNIX-likande operativsystem, **date**. På din dator hittar du också ett enkelt kalenderverktyg, **cal**. Om du händelsevis inte söker information om något av dessa utan hur du ändra tidsstämpeln på en fil, se istället ▷ sektion 8.4.12.

### 9.2.1 date

**$ date** Visa/ställ in systemtid och -datum

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man date**

Ger dig dagens datum och aktuellt klockslag. Används också för att ställa in datum och tid.

Skriv datumet i formatet MM:DD:YYYY (amerikansk standard) för att ställa in tiden. MM är månaden, DD är dagen och YYYY är året.

```
date 01:01:2000
```

Skriv följande för att ställa in tiden (alternativet **-s** anger att du vill ställa in tiden):

```
date -s hh:mm:ss
```

Ett annat användbart alternativ som du kan använda är **--date="string"**.

Exempel:

```
date --date="3 months 1 day ago"
```

Visar datumet 3 månader och 1 dag sedan räknat från dagens datum. Observera att **--date="x month x day ago"** ger samma resultat som **-d "x month x day ago"**.

```
date -d "3 days"
```

Visar datumet tre dagar in i framtiden räknat från dagens datum.

### 9.2.2 cal

**$ cal** Visa en kalender (och påskdagen)

**DEBIAN, UBUNTU:** **bsdmainutils** **FEDORA:** **util-linux**

**HJÄLP:** **man cal**

Om du skriver **cal** visas en kalender för den innevarande månaden på skärmen. Du kan använda några olika alternativ för att anpassa hur kalendern ser ut. Som stan-

dard visas kalendern med söndag som första dagen i veckan, ange alternativet **-m** för att istället visa måndag som första veckodag. Se manual-sidan för mer information.

Exempel:

```
cal -m -y year
```

Om du lägger till alternativet **-y** visas en kalender för det innevarande året.

```
cal -m 2 2009
```

Detta visar en kalender för februari 2009, med måndag som första veckodag.

**cal** visar kalendern med dagarna som överskrift. Varianten **ncal** visar istället kalendern med dagarna i en kolumn. Varianten **ncal** kan också med alternativet **-e** visa vilket datum påskdagen infaller ett visst år.

Exempel:

```
ncal -e 2021
```

## 9.3 Visa information om partitioner

Det finns ett antal olika sätt att söka information om hårddiskar och partitioner i systemet, bland annat med kommandot **fdisk**. Se även ▷sektion 9.4.

### 9.3.1 fdisk

**# fdisk** Visa information om och partitionera hårddiskar

**DEBIAN, UBUNTU:** **util-linux** **FEDORA:** **util-linux**

**HJÄLP:** **man fdisk**

Med **fdisk** och alternativet **-l** kan du visa information om de inkopplade hårddiskarna och deras partitioner på systemet (exempelvis partitionstypen). Observera att du måste vara root-användaren för att kunna köra kommandot.

Om du till exempel vill se information om enheten sda1 skriver du så här:

```
fdisk -l sda1
```

Du kan också använda **fdisk** för att partitionera hårddiskar, se manual-sidan för mer information.

## 9.4 Visa systeminformation med /proc

**/proc** Visar information från kärnan som filer

**MER INFO: man proc**

Filerna i mappen `/proc` (*process information pseudo filesystem* på engelska) visar diverse information om systemet. Man kan se på filerna i `/proc` som ett fönster mot kärnan och den information den använder.

Exempel:

```
cat /proc/cpuinfo
```

Visar information om processorn i datorn.

```
less /proc/modules
```

Använd ovanstående för att se information om vilka kärnmoduler som för närvarande används i systemet.

```
less /proc/x
```

där x är numret på en process. Under `/proc` hittar du en undermapp för var och en av processerna som körs på systemet. Denna mapp innehåller läsbara virtuella textfiler som innehåller all tänkbar information om processen.

Du kan se information om olika partitioner genom att titta i den korresponderande delen av filsystemet `/proc`, antingen under `/proc/ide` eller `/proc/ideN/hdX`, där N motsvarar ett nummer och X en bokstav (med början på a).

Exempel:

```
cd /proc/ide0/
```

Om du har en eller flera IDE-hårddisk inkopplad på datorn finns det i den här mappen utförlig information enheten eller enheten i olika undermappar.

Läs mer genom att köra **man proc**.

# Kapitel 10

# Kontrollera systemet

Det här kapitlet berättar mer om kommandon du kan använda för att interagera med enheter på ditt system, liksom hur du kan hantera processer och tjänster/demoner.

## 10.1 eject

---

**`$ eject`** Mata ut löstagbara enheter

---

**DEBIAN, UBUNTU:** **`eject`** **FEDORA:** **`eject`**

---

**HJÄLP:** **`man eject`**

---

Kommandot **`eject`** säger åt en enhet att öppna/mata ut en skiva. Detta är användbart med cdrom- och dvd-skivor.

Med kommandot nedan kan du exempelvis mata ut en cd-skiva i cdrom-enheten:

```
eject /dev/cdrom
```

**OBS** Detta fungerar endast om användaren har rättigheter att montera/avmontera enheten eller partitionen. Se tips i ▷sektion 10.2 för mer information.

## 10.2 Montera och avmontera enheter

Kommandona nedan fungerar endast om användaren har tillräckliga rättigheter.

Om ditt system inte tillåter vanliga användare att montera partitioner är det enkelt att åtgärda. Öppna helt enkelt filen `/etc/fstab` som root-användaren och byt ut ordet ”defaults” mot ”user” eller lägg till ”user” i alternativlistan för de önskade partitionerna.

### 10.2.1 mount

**# mount** Montera enheter

DEBIAN, UBUNTU: **mount** FEDORA: **util-linux**

HJÄLP: **man mount**

Montera en enhet. Lägg till enheten till filsystemshierarkin (filträdet som börjar med `/`). Detta krävs för att du ska kunna komma åt enheten.

När du monterar en enhet måste du ange vilken typ av filsystem som enheten innehåller. Gör detta med alternativet **-t**.

Nedanstående är en lista med några av de vanligare filsystemen du kan ange med hjälp av alternativet **-t**

**ext2** Det vanligaste filsystemet på äldre Linuxmaskiner.

**ext3** En uppdaterad variant av ext2, används ofta på modernare Linuxdatorer.

**hfs** Används på äldre Apple Mac-system.

**hfsplus** Används i Mac OS X.

**iso9660** Vanligt filystem för cdrom-skivor.

**ntfs** Använs som standard i Windows 2000, XP och Vista.

**reiserfs** Ett alternativ filsystem som används av vissa Linuxdistributioner.

**vfat** Används som standard i Windows 9x.

**cifs** Används för att montera utdelade enheter i Windowsnätverk.

Se manual-sidan för kommandot för en komplett lista av filsystem som stöds.

Exempel:

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

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

**OBS** Du kan ofta utesluta alternativet **-t**, eftersom kommandot försöker lista ut enhetstypen själv.

Med kommandot kan du också montera skivavbildningar (oftast filer med ändelsen .iso, .img eller .cdr), så att du kan se och redigera innehållet på avbildningen.

Exempel:

```
mount -t iso /tmp/cd.iso /mnt/cdskiva -o loop
```

Detta monterar skivavbildningen (vanligtvis en cd- eller dvd-avbildning) så att du kan titta på och ändra innehållet på avbildningen, som ser ut och beter sig som vilken annan enhet som helst när den är monterad.

Det är också möjligt att montera utdelade mappar på Windowsdatorer med **mount**.

Exempel:

```
mount -t cifs //winxp/musik /mnt/musik
```

Detta monterar mappen ”musik” på datorn ”winxp” på ”/mnt/musik”.

Läs mer om hur du kan montera utdelade mappar på Windowsdatorer i nätverket i **man mount.cifs**.

### 10.2.2 umount

**# umount** Avmontera enheter

**DEBIAN, UBUNTU:** **mount** **FEDORA:** **util-linux**

**HJÄLP:** **man umount**

Avmontera en enhet. Kommandot **umount** tar bort enheten från filsystemshierarkin (filträdet som börjar med /). Innan du tar ut en diskett eller optisk skiva från sin enhet,

antingen med kommandot **eject** eller genom att trycka på utmatningsknappen på datorn, måste du avmontera den.

Kommandosyntax:

```
umount monteringspunkt eller enhet
```

Exempel på en monteringspunkt är `/mnt/floppy` eller `/mnt/cdrom`. Det går lika bra att ange enhetsnamnet istället för monteringspunkten, exempelvis `/dev/fd0` respektive `/dev/hdb`.

### 10.2.3 smbmount

**# smbmount** Montera Windowsenheter över nätverk

**DEBIAN, UBUNTU: smbfs FEDORA: –**

**HJÄLP: man smbmount**

Med **smbmount** kan du montera och använda utdelade mappar från Windowsdatorer i nätverket.

**OBS** För dig som kör Fedora: använd istället **mount -t cifs** för att montera utdelade mappar på Windowsdatorer. Se ▷sektion 10.2.1 och **man mount.cifs** för mer information.

Kommandosyntax:

```
smbmount //windowsdatorn/c /mnt/windows
```

där ”windows” är platsen du vill montera på, ”windowsdatorn” är namnet på eller IP-numret till den Windowsdator du vill ansluta till. ”c” anger namnet på den utdelade enhet eller mapp på Windowsdatorn du önskar montera.

Använd alternativet **-o** för att ange att du vill ansluta som en vissa användare på fjärrmaskinen.

Exempel:

```
smbmount -o bertil //192.168.0.234/c /mnt/windows
```

**OBS** Du kan bara använda datornamn på nätverket istället för IP-adresser om du har ställt in din dator så att namnen pekar på rätt IP-adresser. Denna information sparas i filen `/etc/hosts` på din dator. Här är ett exempel:

```
192.168.0.2 windowsdatorn
```

Denna rad anger att det finns en dator men namnet ”windowsdatorn” med IP-adress ”192.168.0.2”. Efter att du har lagt till denna rad kan du använda **ping**, **smbmount**, **ssh** och alla andra kommandon med namnet ”windowsdatorn” istället för IP-numret.

### 10.2.4 smbumount

**# smbumount** Avmontera Windowsenheter över nätverk

**DEBIAN, UBUNTU:** **smbfs** **FEDORA:** **–**

**HJÄLP:** **man smbmount**

För att avmontera Windowsenheter använder du samma syntax som med kommandot **umount**, eller så skriver du:

```
smbumount monteringspunkt
```

där ”monteringspunkt” är mappen på din dator du monterade fjärrenheten på, exempelvis ”/mnt/windowsdatorn”.

## 10.3 Stänga av/starta om systemet

### 10.3.1 shutdown

**# shutdown** Stänger ned datorn

**DEBIAN, UBUNTU:** **upstart-compat-sysv** **FEDORA:** **SysVinit**

**HJÄLP:** **man shutdown**

#### shutdown -now

Stänger omedelbart av systemet, men utan att bryta strömmen till datorn. På UNIX-system betyder detta byt till enanvändarläge (*single-user mode* på engelska), det vill säga att endast root-användaren har tillgång till datorn. Läget är avsett för systemunderhåll och -reparationer.

Till exempel tar följande dig till enanvändarläget:

```
shutdown now
```

**shutdown -h now**

Stänger ned datorn omedelbart (**-h** står för *halt*). Påbörjar nedstängningsprocessen omedelbart efter att du har tryckt på ENTER. Avbryt processen genom att trycka på CTRL+C. I slutet på kommandoraden kan du också ange ett meddelande som skickas till alla användare av systemet, till exempel:

```
shutdown -h now "Sluta tjatta, dags att sova."
```

Detta stänger helt ned systemet och skickar meddelandet till alla som är inloggade för tillfället.

**TIPS** Du kan istället för ”now” ange en tid då datorn ska stängas ned. Du kan skriva ”+x minutes” (där x står för valfritt antal minuter) eller ange tiden exakt. Om du till exempel vill stänga ned datorn 20:55 skriver du följande:

```
shutdown -h 20:55
```

**shutdown -r now**

Stänger ned och startar om datorn omedelbart (**-r** står för starta om, *reboot* på engelska). Precis som med **shutdown -h** kan du både ange tid för omstart och skicka ett meddelande till inloggade användare.

### 10.3.2 halt

**# halt** Startar om eller stänger ned datorn

**DEBIAN, UBUNTU:** **upstart-compat-sysv** **FEDORA:** **SysVinit**

**HJÄLP:** **man halt**

Samma sak som **shutdown -h**, men utan möjlighet att ange extra alternativ. Kommandot stänger helt enkelt av datorn omedelbart.

**OBS** På vissa system stänger **shutdown -h** och **halt** inte av strömmen till systemet. Använd i sådana fall istället kommandot **poweroff**.

### 10.3.3 reboot

**# reboot** Startar om eller stänger ned datorn

DEBIAN, UBUNTU: **upstart-compat-sysv** FEDORA: **SysVinit**

HJÄLP: **man reboot**

Samma sak som **shutdown -r**, men utan möjlighet att ange extra alternativ. Kommandot startar om datorn omedelbart.

### 10.3.4 CTRL+ALT+DEL

Tangentbordskombination som kan användas från en terminal för att starta om eller stänga ned datorn (vilket beror på hur din dator är inställd). Observera att detta inte fungerar när du är använder ett terminalfönster i den grafiska miljön. CTRL+ALT+DEL påbörjar omstart/nedstängning omedelbart, även om användaren inte är inloggad.

Du kan ändra beteendet av CTRL+ALT+DEL genom att redigera en speciell fil på datorn. På många vanliga Linuxdistributioner är det filen `/etc/inittab`, på Ubuntu Linux är det `/etc/event.d/control-alt-delete`. Här ett exempel på filinnehållet från Ubuntu:

```
# control-alt-delete - emergency keypress handling
#
# This task is run whenever the Control-Alt-Delete
# key combination is pressed.  Usually used to
# shut down the machine.

start on control-alt-delete

exec /sbin/shutdown -r now "Control-Alt-Delete pressed"
```

I ovanstående fil framgår att kommandot **shutdown -r now** används som standard på Ubuntu när CTRL+ALT+DEL trycks ned.

Observera att brädgårdstecknet # (*hash* på engelska) betyder att raden ignoreras när skriptet körs, det vill säga att raden endast är en kommentar. Om du sätter tecknet # framför raden som börjar med ”exec” körs inte kommandot – raden är ju förvandlad till en kommentar.

Du skulle också kunna ändra raden så att det startar ett visst annat kommando eller program. Om du helt enkelt ändrar raden **-r** till **-h** skulle datorn stängas ned istället för startas om.

## 10.4 Kontrollera processer

### 10.4.1 ps

---

**$ ps** Visar information om processer

---

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

---

**HJÄLP:** **man ps**

---

Visar en lista med processer som körs på systemet. Utan några alternativ visar **ps** en lista med enbart processer som har startats av den aktuella användaren.

Exempel på alternativ:

**-a** Visar alla processer för alla användare.

**-x** Visar alla processer som är igång, även de som inte har startats i ett terminalfönster (till exempel alla processer som körs när datorn startar).

**-u** Visar mer information, inklusive namn, processoranvändning, minnesanvändning och så vidare.

**-aux** Visar alla processer som körs, oavsett vilken användare som startade dem, samt utförlig information om dem. (Det fungerar lika bra att använda formen **ps aux** i enlighet med BSD-standarden, det vill säga utan minustecken framför alternativen. Läs mer om skillnaderna i användningen i manualsidan för kommandot.)

**-l** Visar en längre lista med information (**-l** står för ”lång”, *long* på engelska), inklusive värdena UID och nice. UID anger vilket användarnummer användaren som äger processen har. **nice** anger processens prioritet i förhållande till andra processer.

**--forest** Visar processhierarkin med en trädstruktur. På så sätt ser du hur de olika processerna på systemet förhåller sig till varandra. Du kan också pröva alternativet **pstree**.

Om du till exempel vill se alla processer som körs, samt visa extra information om dem, skriver du helt enkelt följande:

```
ps -aux
```

### 10.4.2 pstree

**$ pstree** Visar ett träd med processer

**DEBIAN, UBUNTU:** **psmisc** **FEDORA:** **psmisc**

**HJÄLP:** **man pstree**

Visar processerna i form av en trädstruktur (liknande den som kommandot **tree** gör för mappar).

Använd alternativet **-p** för att visa processernas id-nummer.

Exempel:

```
pstree -p
```

Detta visar alla processer samt deras id-nummer.

### 10.4.3 pgrep

**$ pgrep** Interagera med processer utifrån namn

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man pgrep**

Detta kommando är användbart för att ta reda på id-numret för en viss process som du bara känner till en del av namnet på.

Använd alternativet **-l** för att visa namnen på processerna. Använd **-u** tillsammans med ett användar-id för att begränsa resultatet enbart till den användaren.

Normalt sett visar **pgrep** endast pid-numret (processens id-nummer). Eftersom kommandot endast visar id-numret kan du använda det tillsammans med andra kommandon eller i skript.

Exempel:

```
kill $(pgrep mozilla)
```

Detta dödar alla processer vars namn börjar med ”mozilla”. Notera att detta ger samma resultat som om du hade använt **pkill** (se nedan).

Se ▷ sektion 19.3 om $( )-delen är obekant för dig.

För att visa en process id-nummer och namn kan du skriva:

```
pgrep -l processnamn
```

### 10.4.4 top

**$ top** Visa processer i realtid

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man top**

Kommandot **top** visar de processer som ligger på topplistan vad gäller processorutnyttjande. Kommandot visar också en mängd information (mer än **ps**).

Avsluta kommandot genom att trycka på tangenten Q.

**top** visar också informationen i realtid i terminalen, så att du kan hålla koll på en process över tiden. Det är också mycket anpassningsbart med hjälp av en mängd alternativ. Se manual- eller info-sidan för mer information.

### 10.4.5 kill

**$ kill** Skicka stoppsignal till en process med id eller pid

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **util-linux**

**HJÄLP:** **help kill**

Man kan kommunicera med processer genom att använda olika kommandon och ange dess id-nummer eller jobbnummer som argument. Processens id-nummer, pid, är det unika löpnumret för processen, oavsett vilket skal det startades i. Jobbnumret, id, är ett löpnummer specifikt för det aktuella skalet.

För att avsluta en process, eller döda (*kill* på engelska) den som det kallas på UNIX-lingo, behöver du antingen processens id-nummer (pid) eller jobbnumret (id).

Kommandosyntax:

```
kill pid
```

Dödar en process, samt ger den tid att spara filer och avsluta ordnat.

```
kill %id
```

Samma som ovan, förutom att det använder ett id istället för ett pid. Du måste använda ett procent-tecken (%) för att använda id med kommandot.

```
kill -kill pid
```

Tvinga en process att dö utan att ge tid att spara filer och avsluta ordnat; använd endast när det verkligen är nödvändigt eftersom all data som programmet hade i minnet går förlorat. Är detsamma som **`kill -9 pid`**.

Det finns också många andra alternativ till **`kill`**, som **`kill -HUP`** (för *hangup* på engelska). Se manual- och infosidan för mer information.

**TIPS** Döda en process med namnet med hjälp av något av kommandona **`killall`** eller **`pkill`**, som båda ofta kan vara enklare att använda än **`kill`**. Med **`pkill`** (▷ sektion 10.4.7) kan du skriva in en del av ett namn för en process, medan **`killall`** (▷ sektion 10.4.6) kräver att du skriver in hela namnet.

## 10.4.6 killall

**`$ killall`** Skicka stoppsignal till en process med namn

**DEBIAN, UBUNTU:** **`psmisc`** **FEDORA:** **`psmisc`**

**HJÄLP:** **`man killall`**

Döda en process med dess namn istället för process-id (pid). Använd alternativet **`-v`** för att säga åt kommandot att rapportera om kommandot lyckades eller inte, använd alternativet **`-i`** för att använda kommandot i interaktivt läge, det vill säga att det frågar dig innan det dödar en process.

Exempel:

```
killall -iv mozilla
```

Dödar allt som heter ”mozilla”. Frågar innan det dödar och rapporterar om kommandot var framgångsrikt eller inte. Tyvärr kräver det att namnet stavas exakt rätt, annars fungerar det inte. Till exempel måste du skriva ”firefox-bin” för att döda webbläsaren Firefox.

**TIPS** Om du vill döda något som du inte vet det exakta namnet på kan du istället pröva kommandot **pkill**.

### 10.4.7 pkill

**$ pkill** Skicka stoppsignal till en process med reguljära uttryck

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man pkill**

Kommandot **pkill** används för att döda processer utifrån ett reguljärt uttryck. Använd alternativet **–u** och ett (eller flera om du vill) användarnamn för att begränsa resultatet till processer som ägs av den (eller de) angivna användaren.

Exempel:

```
pkill processnamn
```

”processnamn” i exemplet ovan behöver inte vara exakt skrivet. Till exempel fungerar ”firefox” lika bra som ”firefox-bin”.

Om du vill döda en process med namnet ”firefox” för användarna ”bertil” och ”berit” kan du skriva så här:

```
pkill –u bertil berit firefox
```

### 10.4.8 skill

**$ skill** Skicka stoppsignal med namn

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man pkill**

Kommandot **skill** används för att skicka en signal till ett kommando, en inloggad användare eller en terminal (tty).

**skill** kan användas för att stanna, återstarta eller döda processer genom att ange användarnamn, kommandonamn, process-id (eller genom att skicka en uppsättning olika signaler).

För att göra tolkningen tydlig accepterar kommandot ett antal alternativ (annars måste kommandot gissa vad du vill göra).

**-L** Visar alla signaler som kan skickas.

**-u** Ange ett användarnamn. Följs av användarnamnet eller en mellanslagsseparerad lista av användarnamn.

**-p** Ange ett process-id. Följs av ett pid.

**-c** Ange ett kommandonamn (på samma sätt som med killall). Följs av ett kommandonamn.

**-t** Ange en terminal. Följs av ett tty-nummer.

**-v** Ge utförlig information.

**-i** Interaktivt läge.

Användbara exempel:

```
skill -STOP krånglande_användare
```

Ovanstående kommando stannar alla processer för den angivna användaren. Skärmen hålls helt fryst tills du skriver

```
skill -CONT krånglande_användare
```

### 10.4.9 CTRL+C

Tangentbordskombinationen CTRL+C dödar (stoppar, avbryter) det som för närvarande körs i den aktuella terminalen.

### 10.4.10 CTRL+Z

CTRL+Z används för att pausa körningen av en process. Det kan användas för att tillfälligt flytta något till bakgrunden.

Om du till exempel håller på att redigera en textfil med **vim** eller **emacs** kan du trycka CTRL+Z för att återgå

till terminalen, utföra de handgrepp du vill göra, varpå du sedan kan skriva **fg** (med betydelsen förgrund) för att återgå till textfilen. Se ▷sektion 10.4 för mer information.

**TIPS** Om **fg** inte fungerar kan du skriva in **jobs** för att se alla processer i bakgrunden. Då kan du se vilket namn och nummer processen har och sedan använda **fg namn** eller **fg nummer**.

### 10.4.11 jobs

**$ jobs** Visar alla jobb som körs i skalet

**DEBIAN, UBUNTU:** **bash** **FEDORA:** **bash**

**HJÄLP:** **man bash**

Kommandot **jobs** är inbyggt i skalet Bash och visar de jobb som körs för tillfället, både jobb i bakgrunden och i förgrunden (läs mer i ▷sektion 10.4.12 och ▷sektion 10.4.13). Med jobb menas de processer som körs i den aktuella terminalen.

### 10.4.12 bg

**$ bg** Sätter en process i bakgrunden

**DEBIAN, UBUNTU:** **bash** **FEDORA:** **bash**

**HJÄLP:** **man bash**

Kommandot **bg** är inbyggt i skalet Bash och sätter en process i bakgrunden, så att det inte blockerar skalet du använder. Sätt en process i bakgrunden genom att först stanna processen och sedan återuppta den i bakgrunden med **bg**. Använd CTRL+Z för att stanna en process, skriv därefter in kommandot **bg** så återupptas processen i bakgrunden.

**TIPS** Du kan också direkt sätta en process i bakgrunden i samband med att du startar den. Sätt tecknet & (ampersand) efter kommandot när du startar det.

Kommandosyntax:

```
bg jobbnummer
```

eller

```
bg jobbnamn
```

Använd kommandot **fg** för återta en process till förgrunden.

### 10.4.13 fg

**$ fg** Hämtar en process till förgrunden

**DEBIAN, UBUNTU:** **bash** **FEDORA:** **bash**

**HJÄLP:** **man bash**

Flyttar en process till förgrunden så att du kan interagera med den. Efter att ha satt en process i bakgrunden med **bg** kan du återta den till förgrunden genom att helt enkelt skriva in kommandot **fg**.

Du kan sätta jobb i förgrunden genom att ange dess namn eller nummer (använd kommandot **jobs** för att ta reda på numret för ett jobb).

Kommandosyntax:

```
fg jobbnummer
```

eller

```
fg jobbnamn
```

### 10.4.14 nice

**$ nice** Kör program med ändrad prioritet

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man nice**

Sätter prioriteten för en process. **nice -20** är maximal prioritet (endast administratörsanvändare kan sätta negativ prioritet), **nice 20** är minimal prioritet. Du måste vara root-användare för att öka prioriteten för en process, men som vanliga användare kan du alltid sänka prioriteten på dina egna processer.

Exempel:

```
nice 5 make
```

Detta startar kompileringskommandot **make** och kör det med maximal prioritet.

### 10.4.15 renice

**$ renice** Ändra prioritet för en process

**DEBIAN, UBUNTU:** **bsdutils** **FEDORA:** **util-linux**

**HJÄLP:** **man renice**

Ändrar prioritet på ett befintligt kommando. Du kan använda alternativet **-u** för att ändra prioritet på alla processer för en viss användare och alternativet **-g** för att ändra prioritet för alla processer som hör till en viss grupp. Standardsättet är att ändra prioritet via processens id-nummer (pid).

Exempel:

```
renice +20 2222
```

Detta ändrar prioriteten för processen med pid ”2222” till ”+20” (minimal prioritet).

### 10.4.16 snice

**$ snice** Ändra prioritet för en process

**DEBIAN, UBUNTU:** **procps** **FEDORA:** **procps**

**HJÄLP:** **man snice**

Kommandot **snice** fungerar på liknande sätt som **skill**, med den skillnaden att det ändrar prioritet istället för avslutar processer. Dess funktioner liknar de hos **renice**.

För att kommandot ska kunna tolka det du vill göra rätt använder du ett eller flera alternativ. Du skriver helt enkelt **snice** följt av något av följande:

**-u** Ange ett användarnamn. Följs av användarnamnet eller en mellanslagsseparerad lista av användarnamn.

**-p** Ange ett process-id. Följs av ett pid.

**-c** Ange ett kommandonamn. Följs av ett kommandonamn.

**-t** Ange en terminal. Följs av ett tty-nummer.

**-v** Ge utförlig information.

**-i** Interaktivt läge.

Exempel:

```
snice -10 -u root
```

Detta höjer prioriteten på alla processer som ägs av root.

## 10.5 Kontrollera tjänster

### 10.5.1 Grundläggande koncept

UNIX-system använder skript för att kontrollera så kallade demoner (*daemons* på engelska), vilka tillhandahåller tjänster (*services* på engelska). Ett exempel på en sådan tjänst är utmatningen av ljud till datorns högtalarutgång. UNIX-system består av en stor mängd sådana demoner/tjänster.

En demon är en systemprocess som körs i bakgrunden och som tar hand om en viss uppgift eller väntar på en viss händelse och utför något när händelsen inträffar. Du kan inte interagera med en sådan process.

Namn på demoner slutar ofta på bokstaven d. Till exempel lyssnar **sshd** efter begäran om att ansluta med ett säkert säkert skal (*Secure Shell Deamon* på engelska) från en fjärrdator.

Demoner ansvarar ofta för kritiskt viktiga delar av systemet, exempelvis att hantera växlingsutrymmet och RAM-minnet, eller utför andra uppgifter.

### 10.5.2 service

**# service** Hantera tjänster

**DEBIAN, UBUNTU: sysvconfig FEDORA: initscripts**

**HJÄLP: man service**

Kommandot **service** är ett skalskript som finns installerat som standard bland annat på Mandrake/Mandriva- och Redhat/Fedora-system och som gör det möjligt att på olika sätt hantera tjänster.

**-s** Visar statusinformation om alla tillgängliga tjänster.

**-f** Använd alternativet **-f** tillsammans med namnet på en tjänst för att starta om den.

**-R** Startar om alla tjänster (detta startar om alla tjänster som körs, inklusive eventuell X-fönstermiljö som är igång).

Om du till exempel vill starta om sshd-demonen skriver du:

```
service -f sshd
```

### 10.5.3 /etc/init.d

Du kan också starta, starta om eller stoppa en tjänst genom att använda använda dess skalskript direkt i mappen `/etc/init.d`. Gå till mappen och skriv **`./skriptnamn`** där ”skriptnamn” är namnet på den tjänst du vill kontrollera. När du trycker på ENTER returnerar skriptet information om de alternativ det accepterar. Som standard brukar alternativen vara följande:

**start** Detta startar en tjänst (förutsatt att det inte redan är igång).

**stop** Detta stannar en tjänst (förutsatt att det är igång).

**restart** Omstart. Detta stoppar tjänsten och startar den på nytt.

**force-reload** Läser om konfigurationsfilerna utan att starta om tjänsten. Om tjänsten inte stödjer ominläsning av konfigurationsfilen så startas tjänsten om.

**reload** Fungerar endast med vissa skript. Läser om konfigurationsfilerna, utan att starta om själva tjänsten.

**status** Detta ger dig aktuell statusinformation om tjänsten, exempelvis om den är igång eller inte.

Exempel:

```
/etc/init.d/httpd restart
```

Detta startar om webbserverdemonen (vanligtvis Apache) på datorn.

# Kapitel 11

# Hantera användare

Här får du lära dig att lägga till användare och grupper på datorn, liksom olika sätt att få administratörsrättigheter på datorn.

## 11.1 root

Användaren root är administratörsanvändaren på datorn. Därför har root-användaren fullständig kontroll över systemet. Som root-användare kan du både förstöra systemet – och laga alla slags fel. I det flesta Linuxdistributioner är det root-användaren som används för att administrera funktioner på systemet.

## 11.2 su

| | |
|---|---|
| **`$ su`** | Byt till root-användaren |
| **DEBIAN, UBUNTU:** | **`login`** **FEDORA:** **`coreutils`** |
| **HJÄLP:** | **`man su`** |

Med kommandot **`su`** kan du byta till en annan användare i terminalfönstret (su står för ändra användare, *substitute user* på engelska).

Skriv helt enkelt följande:

```
su användarnamn
```

där användarnamn är namnet på den användare du vill byta till. När du trycker på ENTER ombeds du skriva in användarens lösenord.

Detta kommando kan vara praktiskt att använda när du är inloggad som en vanlig användare men tillfälligt önskar bli root-användaren för att administrera systemet. Med kommandot kan du göra det utan att logga ut från ditt vanliga användarkonto.

Om du vill byta till root-användaren kan du hoppa över användarnamnet och bara skriva följande:

```
su -
```

Det extra bindestrecket gör så användarens kompletta inloggningsmiljö laddas, inklusive användarens miljövariabler. (Läs mer om miljövariabler i ▷sektion 5.)

När du vill återgå till ditt vanliga användarkonto skriver du inte **su** igen, utan skriver **exit** eller trycker på tangenterna CTRL+D.

**TIPS** Observera att det anses vara säkrare att använda kommandot **sudo** istället för **su** när du tillfälligt behöver root-rättigheter på datorn. Se mer information nedan.

## 11.3 sudo

**$ sudo** Kör ett kommando som en annan användare (vanligtvis root)

**DEBIAN, UBUNTU:** **login** **FEDORA:** **sudo**

**HJÄLP:** **man su**

Kommandot **sudo** (förkortning av *super user do* på engelska, ungefär med betydelsen gör följande som administratörsanvändaren) används på vissa Linuxdistributioner som ersättning för ett speciellt root-lösenord. Istället för att logga in som root-användare och på så sätt få full tillgång till alla delar av systemet skriver du in **sudo** följt av det kommando du vill utföra.

Fördelarna med att använda **sudo** istället för root-användaren är bland annat att du som användare slipper komma ihåg ett extra lösenord, att du inte slentrianmässigt vänjer dig att logga in som root-användare – vilket i sig är en säkerhetsrisk – och att alla kommandon du kör tillsammans med sudo sparas i en loggfil. Om du använder **sudo** kan du spåra upp och rätta eventuella fel som du eller någon annan har orsakat på systemet.

Du använder **sudo** för att utföra ett visst kommando med root-rättigheter, trots att du är inloggad med ett vanligt användarkonto.

Kommandosyntax:

```
sudo kommandonamn
```

Exempel:

```
sudo rpm -U mittpaket.i386.rpm
```

Detta låter dig installera rpm-paketet utan att logga ut och sedan logga in som root-användaren.

Du kan också använda **sudo** för att köra kommandon som andra användare än root. Ange då alternativet **-u** på kommandoraden, samt användarnamnet du vill köra kommandot som.

Exempel:

```
sudo -u bertil /home/bertil/skalskript.sh
```

Detta kör ”skalskript.sh” i Bertils hemmapp som användaren ”bertil”.

**TIPS** Om du till **sudo** lägger till alternativet **-i** kommer kommandot köras med det skal som den angivna användaren har i `/etc/passwd`-filen. Dessutom laddar skalet in den angivna användarens miljövariabler.

Manual-sidan för **sudo** är mycket välskriven och uttömmande. Ta en titt i den för mer information om kommandot.

## 11.4 Användare och grupper

### 11.4.1 /etc/passwd, /etc/shadow och /etc/group

Vanligtvis sparas all information om användarkonton i filen `/etc/passwd`, användarlösenord i `/etc/shadow` och information om grupptillhörighet i filen `/etc/group`. Du kan visserligen lägga till en ny användare genom att redigera filerna för hand, men det är bättre att använda något av de inbyggda kommandona i systemet (se ▷sektion 11.4.4, ▷sektion 11.4.2 och ▷sektion 11.4.3).

Filen `/etc/passwd` innehåller information om varje användares inloggningsnamn, användar-id, grupp-id, användarnamn och användarens hemkatalog, liksom vilket skal som ska vara standardskal för användaren. Filen innehåller en rad per användaren med ovanstående information.

Namnet till trots innehåller inte `/etc/passwd` några lösenord. För att öka säkerheten på systemet har de flyttats till `/etc/shadow`, som inte innehåller lösenorden i klartext. Istället sparas lösenorden krypterade i `/etc/shadow`. Filen är dessutom inte läsbar för vanliga användare, endast root-användaren kan öppna den.

Gruppfilen `/etc/group` innehåller information om vilka användare som hör till vilka grupper på systemet. Om du vill lägga till en användaren till en grupp är det den här filen du ska redigera. Filen innehåller information om gruppnamnet, eventuellt grupplösenord, grupp-id samt en kommaseparerad lista med de användare som hör till gruppen.

### 11.4.2 adduser/useradd

| | |
|---|---|
| **# adduser** | Lägger till en användare till systemet |
| **DEBIAN, UBUNTU:** | **adduser** **FEDORA:** **shadow-utils** |
| **HJÄLP:** | **man adduser, man useradd** |

Använd kommandot **adduser** för att lägga till ett användarkonto på systemet. Kommandot tar automatiskt hand om att redigera och skapa alla nödvändiga filer och mappar för att användaren sedan ska kunna logga in på sitt nya konto.

**OBS** För Fedora-användare är **adduser** ett alias för kommandot **useradd**. Läs mer om kommandot genom att skriva **man useradd**.

För att lägga till en användare skriver du helt enkelt kommandots namn följt av det önskade användarnamnet för den nya användaren.

Kommandosyntax:

```
adduser nytt_användarnamn
```

Detta startar ett interaktivt skript som frågar dig om önskat lösenord för användaren, användarens fullständiga namn samt diverse kontaktinformation. Därefter redigerar skriptet filerna `/etc/passwd`, `/etc/shadow` och `/etc/group` och skapar en ny hemmapp för den nya användaren.

Som standard skapar **adduser** även en ny grupp för varje ny användare. Gruppen får samma namn som den nya användaren.

### 11.4.3 addgroup/groupadd

**# addgroup** Lägger till en grupp till systemet

**DEBIAN, UBUNTU:** **adduser** **FEDORA:** **–**

**HJÄLP:** **man addgroup, man groupadd**

Använd kommandot **addgroup** för att lägga till användare till systemet.

Kommandosyntax:

```
addgroup nytt_gruppnamn
```

**OBS** Fedora-användare använder istället kommandot **groupadd**. Läs mer om kommandot genom att skriva **man groupadd**.

### 11.4.4 passwd

**# passwd** Ändrar användarlösenord

**DEBIAN, UBUNTU:** **passwd** **FEDORA:** **passwd**

**HJÄLP:** **man passwd**

Använd kommandot **passwd** för att ändra en användares lösenord. Om du vill ändra något annat lösenord än ditt eget måste du vara root-användaren.

Skriv helt enkelt in kommandonamnet följt av det användarnamn du vill byta lösenord för:

```
passwd användarnamn
```

Detta ändrar lösenordsinformationen för den angivna användare i filen `/etc/shadow`.

# Kapitel 12

# Verktyg för att hantera text

Det här kapitlet är det längsta i hela boken. Det finns ett enkelt skäl till det: det mesta av tiden du ägnar åt datorn i terminalfönstret handlar om att interagera med text – läsa eller redigera text.

Detta kapitel går genom några av de vanligaste textredigeringsverktygen som du kan använda i terminalfönstret, liksom verktyg som du kan använda för att söka filer och inuti filer. Du får också lära dig hur du kan ändra textformat så att dina filer går bra att hantera på Windowsdatorer.

## 12.1 Textredigeringsverktyg

Vilket textredigeringsverktyg som är det bästa är ett kärt trätoämne bland Linux- och UNIX-fantaster. I det här kapitlet recenserar vi inte verktygen, utan berättar om några av de vanligaste. Du får (och bör) själv lista ut vilket verktyg som passar dig bäst. Välj den textredigerare som gör dig mest effektiv, så kan du fokusera på innehållet istället för att krångla med programmet.

### 12.1.1 vi

**$ vi** Världens bästa textredigerare

**DEBIAN, UBUNTU:** **vim** **FEDORA:** **vim-minimal**

**HJÄLP:** **man vi**

En mycket vanligt förekommande textredigerare på alla slags UNIX-system. För att använda det behöver du lära dig ganska många tangentbordsgenvägar, men det är mycket kraftfullt samtidigt som det är litet; **vi** tar mycket litet systemresurser i anspråk. En sak som många tycker är krångligt med **vi** är att man måste hoppa två lägen för att få något gjort: redigeringsläget och kommandoläget. En modernare variant av **vi** är **vim**, som i många Linuxdistributioner används istället för **vi**.

På `http://www.dsv.su.se/projektx/sv_editvim` hittar du en snabbintroduktion på svenska till **vi**/**vim**.

### 12.1.2 emacs

**$ emacs** Världens bästa textredigerare

**DEBIAN, UBUNTU:** **emacs22-gtk** **FEDORA:** **emacs**

**HJÄLP:** **man emacs**

Programmet **emacs** är mycket mer än en textredigerare, om man vill kan man använda det som en fullfjädrad programutvecklingsmiljö. Det kan ta tid att sätta sig in i programmets funktioner för textredigering, men liksom vi är det också mycket kraftfullt. Med **emacs** kan du faktiskt göra i princip vad som helst – surfa på nätet, tjatta, spela spel med mycket mera. Men det går förstås utmärkt att bara skriva och redigera text med programmet.

På `http://www.dsv.su.se/projektx/sv_editemacs` hittar du en snabbintroduktion på svenska till **emacs**.

### 12.1.3 nano

**$ nano** Världens bästa textredigerare

**DEBIAN, UBUNTU:** **nano** **FEDORA:** **nano**

**HJÄLP:** **man nano**

**nano** är ett litet och i sammanhanget mycket enkelt program som kan fungera för dig som tycker att något av ovanstående alternativ är svårgreppbara. En fördel med **nano** är att du i nederdelen av fönstret ser vilka tangentbordsgenvägar du kan använda för att göra vanliga handgrepp, exempelvis avsluta programmet.

### 12.1.4 gedit, kwrite och andra

Om du tillhör de användare som utforskar kommandoraden via ett grafiskt gränssnitt och virtuella terminaler kan du utan att skämmas använda någon av de mer lättanvända grafiska textredigerarna som finns. I GNOME är **gedit** populärt, i KDE används ofta **kwrite**. Båda använder tangentbordsgenvägar som är snarlika dem i redigeringsverktygen på Windows och Mac OS.

Det finns utöver dessa en uppsjö textredigerare. Sök på nätet med en sökmotor för tips och inspiration.

## 12.2 Textvisningsverktyg

### 12.2.1 head

**$ head** Visar de första raderna i en fil

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man head**

Kommandot **head** visar de 10 första raderna av en fil, om inga alternativ anges.

Kommandosyntax:

```
head textfil
```

Skriv **head -n x** för att visa ”x” antal rader istället för 10.

Exempel:

```
head -n 20 loggfil.txt
```

Detta visar de 20 första raderna i textfilen ”loggfil.txt”.

### 12.2.2 tail

**$ tail** Visar de sista raderna i en fil

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man tail**

Fungerar enligt samma princip som **head**, men visar istället de sista raderna i filen.

Om du inte anger några alternativ visar **tail** de 10 sista raderna av en fil.

Skriv **tail -n x** för att visa ”x” antal rader istället för 10.

Exempel:

```
tail -n 20 loggfil.txt
```

Detta visar de 20 sista raderna i textfilen ”loggfil.txt”.

Om du vill se en kontinuerligt uppdaterad version av filen kan du testa **tail -f**, vilket gör att du direkt ser om filens innehåll ändras. För att avsluta programmet i det kontinuerliga läget måste du trycka CTRL+C.

### 12.2.3 less

**$ less** Visar textfiler på skärmen

**DEBIAN, UBUNTU:** **less** **FEDORA:** **less**

**HJÄLP:** **man less**

Kommandot **less** visar text, i all enkelhet. Du kan rulla upp och ned i en textfil med piltangenterna, eller hoppa till filens början med HOME eller slut med END. Programmet har också flera alternativ som beskrivs i manual-sidan för kommandot.

Kommandosyntax:

```
less filnamn.txt
```

När less redan är igång och du har flera öppna filer samtidigt kan du använda [:] [N] och [:] [P] (skriv ett kolon följt av tecknet) för att flytta till nästa respektive föregående fil.

**OBS** Kommandot **less** är en förbättrad variant av **more**. Det senare alternativet är en mer begränsad föregångare till **less**. Som den enklaste beskrivningen om kommandot brukar lyda: *less is more*.

### 12.2.4 cat

**$ cat** Sätt ihop textfiler och visa dem på skärmen

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man cat**

Kommandot **cat** sätter samman (konkatenerar, *concatenate* på engelska) flera filer till en enda fil. Kommandot fungerar också med enstaka filer.

Några användbara alternativ:

**-b** Numrera alla rader som inte är tomma.

**-n** Numrera alla rader.

**TIPS** Pröva också att använda **nl** för att numrera rader (det kan göra mer komplexa radnumreringar). Mer info finns i ▷sektion 12.4.12.

Exempel:

```
cat del1.txt del2.txt del3.txt > helatexten.txt
```

Detta konkatenerar ”del1.txt”, ”del2.txt” och ”del3.txt” till en ny fil med namnet ”helatexten.txt”.

### 12.2.5 tac

**$ tac** Sätter ihop filer och visar rader i bakvänd ordning

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man tac**

Fungerar enligt samma princip som **cat**, men visar raderna i textfilen eller textfilerna i omvänd ordning, **tac** är **cat** skrivet baklänges.

```
tac del1.txt del2.txt del3.txt > helatexten.txt
```

Detta konkatenerar (sätter samman, *concatenate* på engelska) ”del1.txt”, ”del2.txt” och ”del3.txt” till en ny fil med namnet ”helatexten.txt”, men raderna kommer i omvänd ordning.

### 12.2.6 z*-kommandon

Till början på namnet till många kommandon kan bokstaven z fogas. Dessa varianter av kommandona kan läsa och bearbeta filer inuti gzip-komprimerade filer.

Några av exemplen är **zcat**, **zless**, **zmore**, **zgrep**, **zcmp** och **zdiff**. Se ▷sektion 17.2.1 för mer information.

### 12.2.7 bz*-kommandon

Det finns också några kommandot som bokstäverna bz fogas. Dessa varianter av kommandona kan läsa och bearbeta filer inuti bzip2-komprimerade filer.

Dessa verktyg är **bzcat**, **bzless** och **bzgrep**. Se ▷sektion 17.2.2.

## 12.3 Textinformationsverktyg

### 12.3.1 wc

**$ wc** Visar statistik över rader, ord och byte för filer

**DEBIAN, UBUNTU:** `coreutils` **FEDORA:** `coreutils`

**HJÄLP:** `man wc`

Kommandot **wc** räknar hur många ord ett dokument innehåller (**wc** är en förkortning av *word count* på engelska). Du kan också använda verktyget för att räkna antal rader eller antalet byte i en fil.

Använd följande alternativ för att styra vad som räknas:

**-w** Räknar ord.

**-l** Räknar linjer.

**-c** Räknar antal byte.

Utan alternativ räknar kommandot alla tre.
Kommandosyntax:

```
wc -alternativ fil.txt
```

### 12.3.2 cmp

**$ cmp** Jämför två filer, byte för byte

**DEBIAN, UBUNTU:** **diff** **FEDORA:** **diffutils**

**HJÄLP:** **man cmp**

Kommandot **cmp** (kortform av jämför, *compare* på engelska), används för att kolla om två filer skiljer sig åt. Kommandot fungerar med alla slags filer, även binära filer som bilder. Funktionen liknar den för kommandot **diff**, med den skillnaden att den jämför filen på binär nivå istället för enbart text.

### 12.3.3 diff

**$ diff** Jämför två filer, rad för rad

**DEBIAN, UBUNTU:** **diff** **FEDORA:** **diffutils**

**HJÄLP:** **man diff**

Jämför två textfiler och talar om vad som skiljer dem åt (kortform för skillnad, *difference* på engelska).

Kan användas för att skapa en patch-fil. En patch-fil innehåller endast information om skillnaderna mellan två filer och kan användas med programmet **patch** för att uppdatera en gammal fil till en ny version.

Exempel:

```
diff fil1.txt fil2.txt
```

Detta gör att diff för varje rad som finns i den andra filen men inte i den första skriver ut ett >. Varje rad som finns i den första filen men inte i den andra inleds med tecknet <.

### 12.3.4 sdiff

**$ sdiff** Sätter ihop och visar två filer sida vid sida

DEBIAN, UBUNTU: **diff** FEDORA: **diffutils**

HJÄLP: **man sdiff**

Istället för att visa skillnaden i form av en diff-rapport skriver **sdiff** ut filerna i två kolumner, sida vid sida.

### 12.3.5 diff3

**$ diff3** Sätter ihop och visar tre filer sida vid sida

DEBIAN, UBUNTU: **diff** FEDORA: **diffutils**

HJÄLP: **man diff3**

Fungerar på samma sätt som **diff**, men med tre filer.

### 12.3.6 comm

**$ comm** Jämför två filer, rad för rad

DEBIAN, UBUNTU: **coreutils** FEDORA: **coreutils**

HJÄLP: **man comm**

Jämför två filer, rad för rad, och skriver ut de rader som är unika för fil 1 i en första kolumn, de rader som är unika för fil 2 i en andra kolumn, samt rader som är gemensamma för båda filerna i en tredje kolumn.

Använd **comm** med alternativen **-1**, **-2** eller **-3** för att dölja visningen av en viss kolumn.

Kommandosyntax:

```
comm fil1 fil2
```

### 12.3.7 look

**$ look** Visa rader som börjar med en viss text

DEBIAN, UBUNTU: **bsdmainutils** FEDORA: **util-linux**

HJÄLP: **man look**

Visar en lista med ord, från systemets ordbok, som innehåller en viss sträng – det är användbart för att hitta ord som börjar med särskilda tecken.

Ge strängen som ett argument. Det spelar ingen roll om du använder gemenena eller versala tecken.

```
look sträng
```

## 12.4 Textmanipuleringsverktyg

### 12.4.1 sort

**$ sort** Sorterar rader i textfiler

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man sort**

Du kan använda kommandot **sort** för att sortera text i alfabetisk (eller annan) ordning. Du kan också köras på flera textfiler för att sortera dem alfabetiskt liksom för att kombinera flera filer till en.

Använd alternativet **-r** för att sortera i omvänd ordning, använd alternativet **-g** för att sortera i numerisk ordning, det vill säga genom att sortera i enlighet med hela numret, inte bara första siffran i numret.

Exempel:

```
cat inköpslista.txt | sort
```

Detta kör kommandot **cat** på filen ”inköpslista.txt”, och sedan kommandot **sort** för att sortera och visa resultatet i alfabetisk ordning.

```
sort -r inköpslista.txt
```

Ovanstående kör kommandot **sort** på filen och sorterar innehållet i omvänd alfabetisk ordning.

Avancerade alternativ med **sort**:

Här är några lite krångliga men likafullt användbara alternativ till kommandot. Använd alternativet **-t** för att tala om att ett visst tecken används för att separera innehållet i filen i kolumner. Med alternativet **-k** kan du sedan ange vilken kolumn du vill sortera efter. Den första kolumnen innan

första förekomsten av separatortecknet är kolumn 1, nästa kolumn 2 och så vidare.

Ett exempel:

```
sort -t : -k 4 -k 1 -g /etc/passwd | more
```

Detta sorterar innehållet i filen `/etc/passwd` genom att använda tecknet kolon, ':', som separator. Sorteringen görs via den fjärde kolumnen (sektionen gid i filen) och – om flera rader har samma gid – sedan sortera innehållet från den första sorteringen utifrån första kolumnen (namnet). Alternativet **`-g`** används för att kommandot ska sortera utifrån hela nummer och inte bara första siffran, annars hade exempelvis ”4000” kommit före ”50”.

### 12.4.2 join

**`$ join`** Slår ihop rader med delvis lika innehåll

**DEBIAN, UBUNTU:** **`coreutils`** **FEDORA:** **`coreutils`**

**HJÄLP:** **`man join`**

Sätter samman och visar rader från två olika filer under förutsättning att en del av båda rader innehåller samma tecken. Kommandot visar endast rader som har delvis samma tecken.

Kommandosyntax:

```
join fil1 fil2
```

### 12.4.3 cut

**`$ cut`** Klipper ut och visar önskade delar av rader

**DEBIAN, UBUNTU:** **`coreutils`** **FEDORA:** **`coreutils`**

**HJÄLP:** **`man cut`**

Visar valda delar av rader (i en textfil) – eller annorlunda uttryckt, tar bort vissa delar av en rad. Du ta bort saker utifrån förekomst av tabulator-tecken, kommatecken eller något annat valfritt särskiljande tecken.

Alternativ:

**-d** Låter dig ange valfritt särskiljande tecken, exempelvis ':' som ofta används i exempelvis `/etc/passwd`.

Se exempel nedan.

**-f** Detta alternativ gör så att **cut** delar upp texten i kolumner, separerade med ett särskiljande tecken. Om du till exempel vill se en lista med användare på datorn kan du ange att du vill se första kolumnen i filen `/etc/passwd`.

```
cut -d ':' -f 1 /etc/passwd
```

Detta visar dig alla användarnamn i `/etc/passwd`.

**, (kommatecken)** Används för att särskilja nummer, så att du kan klippa ut enskilda kolumner.

Exempel:

```
cut -d ':' -f 1,7 /etc/passwd
```

Detta visar användarnamnet och det standardskal (kolumn 7) som är inställt för respektive användare i `/etc/passwd`.

**– (bindestreck)** Används för att ange spann, exempelvis från rad x till rad y, eller från kolumn x till kolumn y (med kolumn menas här teckenplats i en rad).

```
cut -c 1-50 fil1.txt
```

Detta visar tecken de första 50 tecknen räknat från vänster för varje rad, eventuell överskjutande delar ignoreras.

### 12.4.4 aspell

**$ aspell** Interaktiv stavningskontroll

**DEBIAN, UBUNTU:** **aspell** **FEDORA:** **aspell**

**HJÄLP:** **info aspell, man aspell**

Kör kommandot **aspell** med alternativet **-c** (kontrollera, *check* på engelska) för att stavningskontrollera en fil interaktivt. Kommandona varnar när det har hittat något som verkar felstavat, varpå du kan rätta ordet och/eller fortsätta. Exempel:

```
aspell -c fil.txt
```

Detta kör kommandot **aspell** interaktivt på filen ”fil.txt”.

Med alternativet **-l** kan du ange ett annat språk än standardspråket på datorn.

Exempel:

```
aspell -l sv -c svenska.txt
```

Ovanstående stavningskontrollerar filen ”svenska.txt” med hjälp av en svensk ordlista.

**OBS** Den svenska ordlistan till **aspell** heter i Debian, Ubuntu och Fedora `aspell-sv`. Installera det med hjälp av pakethanteringssystemet (se kapitel 7) om svenska inte redan är standardspråk på din dator.

### 12.4.5 chcase

**$ chcase** Byt gemener mot versaler (och vice versa) i filnamn

**DEBIAN, UBUNTU:** ▶ **ladda ned** **FEDORA:** ▶ **ladda ned**

**HJÄLP:** **chcase --help**

Exempel:

```
chcase *.JPG
```

Detta ändrar versaler i alla filer med filändelsen ”.JPG” till gemener.

Testa **chcase -e** för att få se en lista med exempel på hur du kan använda kommandot. Kör **chcase --help** för en översikt över olika alternativ till kommandot.

Kommandot **chcase** är ett Perl-skript. Du kan ladda ned det från `http://www.primaledge.ca/chcase.html`, hemsidan för skriptet.

### 12.4.6 fmt

**$ fmt** Enkel textformaterare

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man fmt**

Kommandot **fmt** är en enkel textformaterare. Använd kommandot med alternativet **-u** för att ”städa” textfilen så att mellanrummet mellan ord alltid är ett mellanslag och mellanrummet mellan meningar är två mellanslag (vilket är amerikansk skrivmaskinsstandard, svensk är ett mellanslag).

Exempel:

```
fmt -u essay-us.txt
```

Detta justerar mellanslagen i filen så att det används ett mellanslag mellan ord och två mellan meningar.

### 12.4.7 paste

**$ paste** Slår ihop textrader

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man paste**

Sätter ihop rader från två filer, antingen sida vid sida, separerade med exempelvis tabulator (men du kan också välja något annat tecken), eller ”seriellt”, det vill säga med raderna från första filen i rad (separerade med valfri särskiljare), följt av raderna från andra filen (också med samma särskiljare).

Det enklaste sättet att förstå hur det här kommandot fungerar är att skapa två egna testfiler, exempelvis en första med namnet ”A.txt” med innehållet

```
A
B
C
D
E
```

och en andra med namnet ”1.txt” med innehållet

```
1
2
3
4
5
```

Använd sedan dessa exempelfiler när du prövar exemplen nedan – då blir de hjärnvrickande förklaringarna enklare att begripa.

För att se en lista med rader sida vid sida, där första raden från första filen följs av den första raden från den andra filen, allt separerat med tabulator, skriver du så här:

```
paste A.txt 1.txt
```

Du kan också visa en lista i seriellt läge, det vill säga första raden från första filen, följt av särskiljare, följt av andra raden från första filen och så vidare tills första filen är slut. Därefter kommer första raden från andra filen, följt av särskiljare, följt av andra raden från från andra filen och så vidare tills andra filen är slut. Så här gör du det:

```
paste --serial A.txt 1.txt
```

### 12.4.8 expand

**$ expand** Konverterar tab-tecken till mellanslag

**DEBIAN, UBUNTU:** `coreutils` **FEDORA:** `coreutils`

**HJÄLP:** `man expand`

Detta kommando konverterar tabulator-tecken till mellanslag och visar resultatet. Använd alternativet **`-t`** för att ange antal mellanslag som ska användas istället för tabulatortecknet.

Kommandosyntax:

```
expand fil.txt
```

### 12.4.9 unexpand

**$ unexpand** Konverterar mellanslag till tab-tecken

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man unexpand**

Konverterar mellanslag till tabulator-tecken och visar resultatet.

Kommandosyntax:

```
unexpand fil.txt
```

### 12.4.10 uniq

**$ uniq** Varna för eller ta bort dubblett-rader

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man uniq**

Kommandot **uniq** eliminerar dubbletter i en fil (exempelvis en adresslista).

Alternativ du kan använda med **uniq**:

**-c** Räkna antalet förekomster av dubbletter.

**-u** Visa bara unika förekomster (ej dubbletter).

**-d** Visa bara dubbletter.

Exempel:

```
uniq -cd adresslista.txt
```

Detta visar endast dubbletter i ”adresslista.txt” och räknar antal gånger som en adress förkommer.

### 12.4.11 tr

**$ tr** Översätter tecken till andra tecken, eller tar bort tecken

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man tr**

Kommandot **tr** (kortform av översätt, *translate* på engelska) används för att söka och ersätta alla förekomster av tecken i en textfil, eller att ta bort mellanslag.

Exempel:

```
tr '3' '5' < fil.txt > ny_fil.txt
```

Detta skickar med hjälp av mindre än-tecknet innehållet i filen ”fil.txt” till **tr**, som ersätter alla förekomster av siffran 3 med 5 och med hjälp av större än-tecknet sparar resultatet i ”ny_fil.txt”.

Du kan också göra andra saker med kommandot, som följande:

```
tr 'A-Z' 'a-z' < fil.txt > ny_fil.txt
```

Detta skickar med hjälp av mindre än-tecknet innehållet i filen ”fil.txt” till **tr**, som konverterar alla versaler till gemener och med hjälp av större än-tecknet sparar resultatet i ”ny_fil.txt”.

**TIPS** Du kan också göra sök- och ersätt-operationer med ett enradigt Perl-skript, läs mer om det på ▷sektion 12.4.14.

### 12.4.12 nl

**$ nl** Lägger till radnummer till filer

**DEBIAN, UBUNTU:** `coreutils` **FEDORA:** `coreutils`

**HJÄLP:** `man nl`

Kommandot **nl** (kortform av numrera rader, *number lines* på engelska) används för att addera nummer till alla rader samt för att indentera texten i en fil.

Det här kommandot accepterar lite mer avancerade numreringsalternativ, läs om detta i info-sidan för kommandot. Dessa alternativ handlar om anpassningar av numreringen, inklusive olika format för olika sektioner, sidor, sidfötter och så vidare.

Använd **nl** på ett av två sätt.

```
nl fil.txt
```

Ovanstående lägger till radnummer till varje rad i ”fil.txt”.

Du kan också använda **nl** för att lägga till radnummer till utdata från ett annat kommando, som i följand exempel:

```
grep telefon kontaktlista.txt | nl
```

Ovanstående visar och numrerar alla rader med texten ”telefon” ur filen ”kontaktlista.txt”.

**TIPS** Du kan också använda kommandot **cat** med alternativet **-n** för att numrera alla rader i en fil, eller med alternativet **-b** för att numrera alla icke-tomma rader. Läs mer i ▷sektion 12.2.4.

### 12.4.13 sed

**$ sed** Icke-interaktiv textredigerare

**DEBIAN, UBUNTU:** **sed** **FEDORA:** **sed**

**HJÄLP:** **info sed, man sed**

**sed** är precis som **emacs**, **gedit** eller **vi** ett helt vanligt textredigeringsverktyg – så när som på att det fungerar helt icke-interaktivt. Du kan med andra ord inte se texten du redigerar. Detta är inte så allvarligt som det låter, **sed** är till exempel en rackare på att söka och ersätta text.

Kommandosyntax för sök- och ersätt-funktionaliteten:

```
sed 's/sökord/ersättning/g' fil
```

där **s** indikerar att det handlar om en ersättning (*substitution* på engelska), ”sökord” är ordet du söker efter och ”ersättning” är det som ska ersätta ”sökord” i filen. **g** säger åt **sed** att byta ut alla förekomster av sökordet.

Istället för ett enkelt sökord kan du använda reguljära uttryck för att matcha mer komplicerade mönster. Se kapitel 20 för mer information om reguljära uttryck.

Ett praktiskt och enkelt exempel är följande:

```
sed 's/Berit/Margareta/g' brev44.txt > brev45.txt
```

Detta byter ut alla förekomster av ”Berit” mot ”Margareta” i ”brev44.txt”, och sparar med hjälp av större än-tecknet resultatet i ”brev45.txt”.

**TIPS** Kommandot **sed** är riktigt mångsidigt. Titta på `http://sed.sourceforge.net/sed1line.txt` för en (lång) lista med smarta enradskonverteringar som använder kommandot.

### 12.4.14 Sök och ersättning med Perl

**$ perl** Skriptspråk

**DEBIAN, UBUNTU:** **perl-base** **FEDORA:** **perl**

**HJÄLP:** **man perl**

Du kan använda följande enradsskript skrivet i Perl för att söka och ersätta text i en fil.

Kommandosyntax:

```
$ perl -pi -e "s/gammal_text/ny_text/g" fil(er)
```

Med ovanstående byts alla förekomster i fil(er) av ”gammal_text” ut mot ”ny_text”. Du utföra sök- och ersättningsoperationen på en eller flera filer, samtidigt.

Exempel:

```
$ perl -pi -e "s/vänster/höger/g" körskola-1967.txt
```

I exemplet ovan byts alla förekomster i filen ”körskola-1967.txt” av ”vänster” ut mot ”höger”.

## 12.5 Textkonverterings- och textfilterverktyg

Eftersom UNIX- och Microsoft-system använder två olika standarder för att indikera ny rad i textfiler kommer du antagligen från tid till annan behöva konvertera från det ena formatet till det andra. Många moderna textredigerare kan göra konverteringen åt dig, men inte alla. Nedan finns ett antal fristående verktyg som kan hjälpa dig att konvertera filer mellan UNIX- och Dos-/Windows-radbrytning.

Ibland är problemen värre än så – exempelvis om du har fått ett Microsoft Word-dokument skickat till dig, men du inte har möjlighet att över huvud taget läsa det eftersom du saknar lämplig programvara (som exempelvis OpenOffice.org). Även här finns hjälp att få.

Skillnaden mellan Dos-/Windows-filer och UNIX-filer är att ny rad indikeras med ett retur-tecken följt av ett ny rad-tecken (\r \n) i Windows, medan man i UNIX helt enkelt använder enbart ett ny rad-tecken (\n).

### 12.5.1 dos2unix

| | |
|---|---|
| **$ dos2unix** | Konverterar textfiler från DOS- till UNIX-format |
| **DEBIAN, UBUNTU:** | **tofrodos** **FEDORA:** **dos2unix** |
| **HJÄLP:** | **man dos2unix** |

Kommandot **dos2unix** konverterar ny rad-tecknen från Windows- till UNIX-typ.

Kommandosyntax:

```
dos2unix fil.txt
```

### 12.5.2 unix2dos

| | |
|---|---|
| **$ unix2dos** | Konverterar textfiler från UNIX- till DOS-format |
| **DEBIAN, UBUNTU:** | **tofrodos** **FEDORA:** **unix2dos** |
| **HJÄLP:** | **man unix2dos** |

Kommandot **unix2dos** konverterar ny rad-tecknen från Windows- till UNIX-typ.

Kommandosyntax:

```
unix2dos fil.txt
```

### 12.5.3 antiword

| | |
|---|---|
| **$ antiword** | Visa text och bilder i Word-dokument |
| **DEBIAN, UBUNTU:** | **antiword** **FEDORA:** **antiword** |
| **HJÄLP:** | **man antiword** |

Filtret **antiword** konverterar Microsoft Word-filer till rena textfiler (ASCII-filer).

Kommandosyntax

```
antiword fil.doc
```

Om ett installationspaket inte finns tillgängligt i din distribution kan du ladda ned **antiword** från hemsidan http://www.winfield.demon.nl/.

### 12.5.4 recode

**$ recode** Konverterar filer mellan olika teckenkodningar och -format

**DEBIAN, UBUNTU:** **recode** **FEDORA:** **recode**

**HJÄLP:** **info recode, man recode, recode -l**

Konverterar textfiler mellan olika format, inklusive HTML och dussintals andra format.

Använd **recode -l** för att se en komplett lista över möjliga konverteringar. Du kan också använda kommandot för att konvertera text till och från Windows- och UNIX-format, så att radbrytningarna ser ut som de ska.

**OBS** Som standard skriver **recode** över filen du kör kommandot på med resultatet. För att istället skapa en ny fil med det modifierade innehållet kan du använda < och >.

Exempel:

```
recode ..pc fil.txt
```

Kodar om från UNIX- till Windows-format.

```
recode ..pc/ fil.txt
```

Kodar om från Windows- till UNIX-format.

```
recode ..pc < fil.txt > ny_fil.txt
```

Kodar om från UNIX- till Windows-format, sparar resultatet i en ny fil utan att skriva över ursprungsfilen.

**TIPS** En introduktion till programmet (på engelska) finns här: http://recode.progiciels-bpi.ca/manual/Tutorial.html#Tutorial

### 12.5.5 enscript

**$ enscript** Konverterar textfiler till Postscript, HTML med flera

**DEBIAN, UBUNTU:** **enscript** **FEDORA:** **enscript**

**HJÄLP:** **man enscript**

Kommandot **enscript** konverterar textfiler till olika format: postscript, html och rtf. Kommandot har en stor mängd alternativ som kan användas för att anpassa resultatet, bland annat i kolumner, med färg med mera. Programmet är mycket användbart för att skriva ut text på papper – bra för att granska program- eller kanske html-kod – eller för att visa kod snyggt på webbsidor.

Använd alternativet **-w** för att ange vilket format du vill konvertera till, alternativet **-o** för att ange den nya filens namn, alternativet **-E** och **--color** för att slå på färgmärkning av koden och alternativet **--toc** för att skapa en innehållsförteckning.

Exempel:

```
enscript -w html fil.txt -o ny_fil.html
```

Detta tar innehållet i ”fil.txt” och skapar en ny html-fil med samma innehåll.

```
enscript -E --color -w html -o foo.html *.h *.c
```

Ta alla filer som slutar med ”.h” och ”.c” (C källkods- och header-filer) och visa dem i en fil som heter ”foo.html” med färgmärkning av koden.

Kör **enscript --help-highlight** för att se en lista med alla format som kan användas för färgmärkning av kod.

Läs mer om alla möjliga alternativ i den välskrivna manual-sidan för **enscript**.

### 12.5.6 figlet

**$ figlet** Konverterar text till ASCII-art

**DEBIAN, UBUNTU:** **figlet** **FEDORA:** **-**

**HJÄLP:** **man figlet**

Du kan använda för att skapa ”ASCII art”, det vill säga bilder uppbyggda enbart av bokstäver och siffror. Ett exempel på en sådan bild finns i förordet till den här boken.

**OBS** Programmet **figlet** är inte tillgängligt i Fedora, men Fedora-användare kan istället testa **cowsay**, som är ett precis lika onödigt program som **figlet**.

## 12.6 Söka och hitta text i filer

### 12.6.1 grep

---

**$ grep** Visar rader som matchar ett sökmönster

---

**DEBIAN, UBUNTU:** **grep** **FEDORA:** **grep**

---

**HJÄLP:** **man grep**

---

Söker efter text inuti filer. Sökmönstret kan vara text eller ett reguljärt uttryck (se kapitel 20).

Kommandosyntax:

```
grep sträng fil.txt
```

Några användbara alternativ:

**–v** Detta alternativ används för att visa rader som *inte* innehåller strängen du söker efter – ett slags inverterad sökning med andra ord.

**–n** Visar radnummer.

**–w** Gör att **grep** endast hittar förekomster av strängen som hela ord, inte där strängen ingår som en del av ett längre ord.

**–A x eller –B x** Visar x antal rader före och/eller efter platsen där strängen du söker efter förekommer. Använd **–A** för att ange antal rader efter (*after* på engelska), använd **–B** för att ange antal rader före (*before* på engelska).

**–r eller rgrep** Sök i filer rekursivt (öppnar även filer som ligger i mappar under mappen du anger). Är detsamma som **rgrep**.

Kommandot **grep** använder reguljära uttryck, vilka till en början kanske är lite krångliga att förstå sig på – men de är mycket kraftfulla. Om du tar dig tiden att sätta dig in i hur de fungerar får du snabbt igen investeringen.

I detta exempel söker **grep** i filen ”rpmlist.txt” för allt som börjar med ”rpm”:

```
grep rpm rpmlist.txt
```

Du kan också använda grep så här, för att söka i utdata från ett annat kommando:

```
rpm -qa | grep ogg
```

Det första kommandot visar alla rpm-paket som är installerade på din dator, det andra söker reda på och visar alla rpm-paket som innehåller strängen ”ogg”.

### 12.6.2 rgrep

**$ rgrep** Samma som **grep**, men söker i filer rekursivt

**DEBIAN, UBUNTU:** **grep** **FEDORA:** **–**

**HJÄLP:** **man grep**

Kommandot **rgrep** är en rekursiv variant av **grep** som söker i alla filer i den aktuella mappen, liksom i mappar som finns i den mappen och så vidare . . . Kommandot visar alla filer som innehåller träffar samt alla rader i filerna som matchar söksträngen. Använder samma syntax som **grep**.

**OBS** Kommandot **rgrep** finns inte i Fedora. Det går dock lika bra använda **grep -r** för att uppnå samma sak.

### 12.6.3 fgrep

**$ fgrep** Samma som **grep**, men utan att expandera reguljära uttryck

**DEBIAN, UBUNTU:** **grep** **FEDORA:** **grep**

**HJÄLP:** **man grep**

Den här versionen av **grep** använder sig av alternativet **-F** tillsammans med **grep**. Den söker efter bokstavligt

exakta träffar till söksträngen, utan att expandera (tolka) exempelvis reguljära uttryck som används som sökstring.

Exempel:

```
fgrep 'a$*b?' fil.txt
```

Istället för att tolka ”a$*b?” som ett reguljärt uttryck söker **fgrep** efter förekomster av strängen i ”fil.txt”.

# Kapitel 13

# Verktyg för att konvertera mediafiler

Kommandoraden har väldigt många bra verktyg för att konvertera och hantera mediafiler, avsaknaden av grafiskt gränssnitt till trots.

Med verktygen i det här kapitlet kan du konvertera, redigera och förbättra filer som är avsedda att användas i ett grafiskt gränssnitt. Det faktum att du inte kan titta på filerna direkt i kommandoraden är här inget problem, det kan du göra med lämpligt grafiskt program istället. Fördelen med dessa verktyg är att de är mycket snabbare att använda än sina grafiskt uppbyggda motsvarigheter. Dels kan de användas med jokertecken (▷sektion 3.4) för att hantera flera filer åt gången, dels kan de användas för att utföra flera operationer i ett enda steg, med hjälp rör eller kommandosubstitution (se kapitel 19).

## 13.1 Verktyg för Postscript- och PDF-filer

Postscript är ett programmeringsspråk som bland annat används för att beskriva utseendet på filer som förberetts

för utskrift på en skrivare och i många grafiska program. Postscript används även för typsnitt av slagen Type 1 och OTF (*Open Type Format*).

PDF (*Portable Document Format*) är en förenklad variant av Postscript och har blivit standardformat för filöverföring och -hantering i grafik- och tryckeribranschen. Om du ska skicka filer till ett tryckeri är det oftast bästa att konvertera dem till PDF först. PDF är också ett utmärkt format för bild- eller ordbehandlingsdokument som ska läggas upp på nätet med helt bibehållet utseende.

Med Linux följer ett stort antal kommandoradsverktyg för att hantera både Postscript- och PDF-filer, inklusive verktyg som på andra plattformar kostar en hel del pengar.

### 13.1.1 ghostscript

**$ ghostscript** Tolk för Postscript och PDF

DEBIAN, UBUNTU: **ghostscript** FEDORA: **ghostscript**

HJÄLP: **man ghostscript**

I Linux används ofta Ghostscript för att konvertera sidor som skrivs ut till exakt det format som din skrivare kräver, utan att du märker att det körs. Men det går också bra att använda Ghostscript från kommandoraden för att titta på filer eller själv konvertera filer mellan olika format och för att göra sammanslagningar eller andra ändringar av Postscript- och PDF-filer.

Ghostscript används i terminalfönstret oftast med kommandot **gs**.

Kommandosyntax:

```
gs alternativ fil(er)
```

I normalläget körs Ghostscript i interaktivt läge, det vill säga att det startar och sedan väntar på att du gör ytterligare inmatningar. Om du använder ett grafiskt användargränssnitt visar Ghostscript dig filen du jobbar med i ett fönster. För att stänga av det interaktiva läget använder du alternativen **-dNOPAUSE** och **-dBATCH**. Det är viktigt att du använder alternativet **-q** (för tyst, *quiet* på engelska) för att stänga av felmeddelanden från Ghostscript som annars kan leta sig in i själva filen. För att konvertera till PDF-format anger du

den virtuella ”skrivarenheten” **-sDEVICE=pdfwrite** (du kan använda många andra slags format, se manual-sidan för mer information). Med alternativet **-sOutputFile** anger du den nya filens namn.

Gör så här för att konvertera en Postscript-fil till PDF:

```
gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite
-sOutputFile=ny_fil.pdf fil.ps
```

Detta konverterar ”fil.ps” till PDF-filen ”ny_fil.pdf” med hjälp av alternativet **-sDEVICE=pdfwrite**.

Du kan också använda Ghostscript för att slå ihop flera filer (både Postscript och PDF) till en ny fil.

Exempel:

```
gs -q -dNOPAUSE -dBATCH -sDEVICE=pdfwrite
-sOutputFile=sammanslagen.pdf fil1.pdf fil2.pdf
```

Om du tycker detta såg krångligt ut kan du istället pröva **pdfjoin** (▷ sektion 13.1.6) eller **pdftk** (▷ sektion 13.1.8), som båda klarar av att slå ihop PDF-filer.

Se även Ghostscripts specialkommando för PDF, **ps2pdf**, för ytterligare PDF-alternativ som du kan använda med Ghostscript.

### 13.1.2 ps2pdf

**$ ps2pdf** Konvertera från Postscript till PDF

**DEBIAN, UBUNTU:** **ghostscript** **FEDORA:** **ghostscript**

**HJÄLP:** **man ghostscript**

Ghostscript använder sig av Postscript-syntax för alternativen, vilket kan kännas krångligt. För att göra det lättare att använda Ghostscript för PDF-filer finns skriptet **ps2pdf**.

Med det kan du snabbt konvertera en Postscript-fil till PDF.

Kommandosyntax:

```
ps2pdf alternativ fil.ps ny_fil.pdf
```

Liksom med **gs** kan du använda ett stort antal alternativ som styr PDF-filens storlek, bland annat om du önskar in-

kludera typsnitt i filen eller om du vill komprimera bilderna i filen så att den tar mindre plats.

Med kommandot kan du också anpassa den nya PDF-filen till olika ändamål. En fil som ska laddas ned via nätet behöver kanske inte högupplösta bilder, medan en fil som ska skickas till ett tryckeri behöver bilder av hög kvalitet. Det finns ett antal fördefinierade inställningar som du kan använda med Ghostscript. Använd alternativet **`-dPDFSETTINGS=`** tillsammans med något av följande:

**`/screen`** Lämpligt för filer som ska skickas med e-post eller laddas ned via nätet. Motsvarar Adobe Destillers ”Screen Optimized”-läge.

**`/ebook`** Lämpligt för skärmläsare för e-böcker, med aningen bättre kvalitet på bilderna än med **`/screen`**. Motsvarar Adobe Destillers ”eBook”-läge.

**`/printer`** Lämpligt för utskrift på laserskrivare. Motsvarar Adobe Destillers ”Print Optimized”-läge.

**`/prepress`** Lämpligt för filer som skickas till ett tryckeri. Motsvarar Adobe Destillers ”Prepress Optimized”-läge.

**`/default`** Standardläget, som fungerar för de flesta ändamål men kan ge onödigt stora filer.

Namnet till trots kan du även konvertera befintliga PDF-filer, inte bara Postscript-filer, med **`ps2pdf`**. Gör så här för att anpassa en PDF-fil till någon av dessa inställningar, exempelvis för att krympa en fil så att den blir lättare att skicka med e-post.

```
ps2pdf -dPDFSETTINGS=/screen klumpig.pdf smidig.pdf
```

Detta tar filen ”klumpig.pdf”, applicerar inställningarna för ”/screen”-läget, och sparar den krympta filen som ”smidig.pdf”.

**TIPS** Om du blir nyfiken på Ghostscripts fulla PDF-kapacitet (det är i princip jämförbart med Adobe Distiller, så när som på att gränssnittet inte är grafiskt) rekommenderas manualsidan samt `http://pages.cs.wisc.edu/~ghost/doc/cvs/Use.htm`

### 13.1.3 pdfinfo

**$ pdfinfo** Visar information om en PDF

**DEBIAN, UBUNTU:** **poppler-utils** **FEDORA:** **poppler-utils**

**HJÄLP:** **man pdfinfo**

Med kommandot **pdfinfo** kan du snabbt se information om en PDF-fil som den som skapade filen fyllde i, exempelvis titel, ämne, författarnamn och skapelsedatum. Du ser också vilka eventuella säkerhetsinställningar som har gjorts för filen och vilken version av PDF-formatet som används.

Kommandosyntax:

```
pdfinfo fil.pdf
```

Detta visar all tillgänglig information om filen.

**OBS** Om PDF-filen är lösenordsskyddad kan du ange filägarens lösenord med alternativet **-opw** (det låser upp allt, inklusive kopieringsskyddet) eller ett användarlösenord för PDF-filen med alternativet **-upw**.

**TIPS** Testa också att använda kommandot **pdffonts** för att se information om typsnitten som används i en PDF-fil. Detta visar all tillgänglig information om typsnitten i filen, inklusive typsnittsnamn och typ av typsnitt (Postscript, Truetype, OTF eller något annat).

### 13.1.4 pdftotext

**$ pdftotext** Sparar texten från en PDF som en textfil

**DEBIAN, UBUNTU:** **poppler-utils** **FEDORA:** **poppler-utils**

**HJÄLP:** **man pdftotext**

Med kommandot **pdftotext** kan du extrahera text från PDF-filer och spara dem i en vanlig text-fil. Detta kan vara smidigt om du behöver jobba vidare med innehållet i en PDF.

Kommandosyntax:

```
pdftotext fil.pdf
```

Detta kommando gör att textinnehållet i ”fil.pdf” sparas i en ny fil med namnet ”fil.txt” i den aktuella mappen.

Du kan använda alternativen **-f** och **-l** (för första, *first* på engelska, respektive sista, *last* på engelska) för att avgränsa kommandot till valda sidor. Du kan också ange önskat filnamn för textfilen som ett argument efter PDF-filen.

Exempel:

```
pdftotext -f 3 -l 7 fil.pdf texten.txt
```

Detta kommando gör att textinnehållet på sidorna 3–7 i filen ”fil.pdf” sparas i en ny fil med namnet ”texten.txt” i den aktuella mappen.

**OBS** Om PDF-filen är lösenordsskyddad kan du ange filägarens lösenord med alternativet **-opw** (det låser upp allt, inklusive kopieringsskyddet) eller ett användarlösenord för PDF-filen med alternativet **-upw**.

### 13.1.5 pdfimages

**$ pdfimages** Sparar bilderna i en PDF till hårddisken

**DEBIAN, UBUNTU:** **poppler-utils** **FEDORA:** **poppler-utils**

**HJÄLP:** **man pdftotext**

Med kommandot **pdfimages** kan du extrahera bilder från PDF-filer och spara dem på hårddisken.

Kommandosyntax:

```
pdfimages fil.pdf mappnamn
```

Detta kommando gör att alla bilder i ”fil.pdf” sparas i mappen ”mappnamn”.

Använd alternativen **-f** och **-l** (för första, *first* på engelska, respektive sista, *last* på engelska) för att avgränsa kommandot till valda sidor.

Exempel:

```
pdfimages -f 8 -l 12 fil.pdf ~/bilder
```

Detta sparar alla bilder på sidorna 8–12 i filen ”fil.pdf” i mappen ”bilder” i hemmappen.

**OBS** Om PDF-filen är lösenordsskyddad kan du ange filägarens lösenord med alternativet **`-opw`** (det låser upp allt, inklusive kopieringsskyddet) eller ett användarlösenord för PDF-filen med alternativet **`-upw`**.

### 13.1.6 pdfjoin

**`$ pdfjoin`** Slår ihop flera PDF-filer till en ny PDF-fil

**DEBIAN, UBUNTU:** **`pdfjam`** **FEDORA:** **`pdfjam`**

**HJÄLP:** **`man pdfjoin`**

Använd kommandot **`pdfjoin`** för att slå ihop flera separata PDF-filer till en. Ange de filer som du vill slå ihop som argument, i den ordning du vill foga samma dem.

Kommandosyntax:

```
pdfjoin fil1.pdf fil2.pdf
```

Detta skapar en ny fil med namnet ”fil2-joined.pdf” (programmet tar automatiskt namnet på den sista filen och lägger till ”-joined”. Du kan också använda **`-outfile`** för att ange ett valfritt nytt filnamn för den sammanslagna filen.

Om du vill slå ihop ett eller flera dokument som har olika fysiska storlekar kan du tvinga dem att anpassa sig till önskat pappersformat (funktionen är praktisk om du exempelvis vill anpassa ett dokument från den amerikanska papperstandarden letter till ISO-standarden A4 som används i Europa). Ange pappersformatet för den nya filen med alternativet **`--paper`** och be programmet att skala alla sidor till det önskade formatet med **`--fitpaper false`**.

Exempel:

```
pdfjoin fil1.pdf fil2.pdf --fitpaper false
--paper a4paper --outfile ~/nyfil-a4.pdf
```

Detta skalar (upp eller ned, beroende på ursprungsstorleken) sidorna i filerna till exakt A4-format och sparar den sammanslagna filen med det nya namnet i den angivna mappen.

Kommandot har fler alternativ, exempelvis för att beskära sidinnehållet (kan vara bra om du vill ta bort skärmärken

som syns i hörnen av en tryckförberedd PDF). Se manualsidan och kör **pdfjoin --help** för mer information.

**OBS** Observera att **pdfjoin** kräver paketet **texlive-base-bin** för att fungera. Detta paket är en del av typsättningssystemet TEX.

### 13.1.7 pdfnup

**$ pdfnup** Sätter ihop flera sidor ur en PDF på varje ark

**DEBIAN, UBUNTU:** **pdfjam** **FEDORA:** **pdfjam**

**HJÄLP:** **man pdfnup**

Använd kommandot **pdfnup** för att skapa ett nytt PDF-dokument med ett valfritt antal av ursprungsfilernas sidor monterade på samma sida i den nya filen. Funktionen kan användas exempelvis för att skapa en A5-broschyr, med två sidor från ursprungsfilen på varje ark.

Kommandosyntax:

```
pdfnup fil.pdf --nup XxY --outfile ny_fil.pdf
```

Använd alternativet **--nup** för att ange hur många rader (”X” ovan) och kolumner (”Y”) av sidor du vill visa på varje ark. Argumentet ”2x1” skapar en ny fil med två sidor bredvid varandra på varje ark (två kolumner, en rad), argumentet ”3x3” skapar en ny fil med 9 sidor på varje ark (tre kolumner, tre rader).

Exempel:

```
pdfnup fil.pdf --nup 2x1 --outfile dubbel.pdf
```

Detta skapar en fil med namnet ”dubbel.pdf” och som har två sidor från ”fil.pdf” monterade bredvid varandra.

Du kan dessutom använda en mängd extraalternativ med kommandot. Se manual-sidan och kör **pdfnup --help** för mer information.

**OBS** Observera att **pdfnup** liksom **pdfjoin** kräver paketet **texlive-base-bin** för att fungera. Detta paket är en del av typsättningssystemet TEX.

### 13.1.8 pdftk

**$ pdftk** Multiverktyg för PDF-filer

**DEBIAN, UBUNTU:** **pdftk** **FEDORA:** **pdftk**

**HJÄLP:** **man pdftk**

Kommandot **pdftk** är något av en schweizisk armékniv för PDF-filer. Verktyget klarar bland annat av att slå ihop, dela upp, vattenmärka, rotera, kryptera/avkryptera och reparera PDF-filer.

Kommandosyntax:

```
pdftk pdf_fil(er) alternativ ny_fil
```

där alternativ är en av flera möjliga operationer, till exempel **cat** (slå ihop, kortform av *concatenate* på engelska), **burst** (bryt isär) eller **background** (vattenmärk med bakgrundsbild). Använd **output** för att ange namnet på den nya filen.

Exempel:

```
pdftk fil1.pdf fil2.pdf cat output sammanslagen.pdf
```

Där ”fil1.pdf” och ”fil2.pdf” slås ihop till en ny fil med namnet ”sammanslagen.pdf”.

Man kan också använda jokertecken för att slå ihop flera filer:

Exempel:

```
pdftk *.pdf cat output sammanslagen.pdf
```

Där alla alla PDF-filer i den akutella mappen slås ihop till en ny fil.

För att göra komplexa operationer mer överskådliga kan man använda ”genvägar” till filnamn. Det kan vara smidigt om man önskar skjuta in en sida från en PDF mitt i en befintlig fil, eller kombinera olika sidor från olika filer i valfri ordning i en ny PDF.

Exempel:

```
pdftk A=fil1.pdf B=fil2.pdf cat A1-7 B5 A8
output ny.pdf
```

Där en ny PDF med namnet ”ny.pdf” börjar med sidorna 1–7 från ”fil1.pdf”, sedan fortsätter med sidan 5 från ”fil2.pdf” och avslutas med sidan 8 från ”fil1.pdf”.

Om en PDF krånglar kan du testa att reparera den med verktyget, helt enkelt bara genom att köra filen genom programmet.

Exempel:

```
pdftk trasig.pdf output fixad.pdf
```

Detta försöker laga filen ”trasig.pdf” och sparar den nya filen som ”fixad.pdf”.

Kommandot har en välskriven manual-sida. Läs den om du är intresserad av att utnyttja verktyget till fullo.

## 13.2 Verktyg för att redigera bilder

### 13.2.1 convert

**$ convert** Multiverktyg för punktbaserade bilder

**DEBIAN, UBUNTU:** **imagemagick** **FEDORA:** **imagemagick**

**HJÄLP:** **man convert**

Programpaketet ImageMagick innehåller ett antal mycket kraftfulla verktyg för att redigera punktbaserade bilder (*bitmap* på engelska) från kommandoraden. Att kunna ändra färgläge eller format på en eller flera bilder direkt i terminalfönstret kan vara mycket smidigt, särskilt om du behöver åtgärda filer på en annan dator via **ssh** (▷ sektion 15.4.1), exempelvis en webbserver. Med de följande kommandona kan du göra jobbet utan att först behöva ladda ned bilderna till din lokala dator, fixa dem en och en i GIMP och sedan ladda upp dem igen.

Kommandot **convert** är ett mycket mångsidigt verktyg i ImageMagick-paketet.

Kommandsyntax:

```
convert bild alternativ ny_bild
```

**OBS** Om du vet med dig att du inte vill behålla originalbilden kan du istället för **convert** använda **mogrify**. Det senare kommandot klarar av exakt samma alternativ och bildredigeringsoperationer som **convert**, men skriver över ursprungsfilen med resultatet.

Programmet stöder ett mycket stort antal filformat, inklusive PSD (formatet som används som standard av Adobe Photoshop). Ange ett önskat filformat med alternativet **-format**. Kör **convert -list format** för att se en lista med alla filformat som stöds.

Exempel för att konvertera alla JPG-bilder i en mapp till TIFF-bilder:

```
convert -format tiff *.jpg
```

Detta konverterar alla JPG-bilder till TIFF-format.

Du kan ändra storleken på en bild med alternativet **-resize**:

```
convert stor.png -resize 30% liten.png
```

Detta krymper bilden ”stor.png” med 30 procent och sparar resultat som ”liten.png”.

Med alternativet **-geometry** kan du anpassa bredden och/eller höjden till ett givet antal bildpunkter:

```
convert -geometry 600x480 bil.png
```

Detta gör att bilden ”bil.png” blir maximalt 600 bildpunkter bred och maximalt 480 bildpunkter hög.

Du kan också lägga på flera operationer på en och samma gång, exempelvis både konvertera till ett annat format samt ändra bilderna till gråskala.

Exempel:

```
convert -format tiff -type grayscale *.jpg
```

Skriv in **mogrify -help** för en komplett lista med olika åtgärder du kan utföra på dina bilder.

### 13.2.2 jhead

**$ jhead** Läsa/redigera EXIF-information i JPG-bilder

**DEBIAN, UBUNTU:** **libjpeg-progs** **FEDORA:** **libjpeg-progs**

**HJÄLP:** **man jhead**

Med verktyget **jhead** kan du läsa och redigera EXIF-data i dina JPG-bilder, exempelvis för att ändra tidsstämpeln, ta bort eller lägga till en tumnagelbild (som sparas inuti bildfilen) eller ändra metadata som bildkommentarer.

Kommandosyntax:

```
jhead alternativ fil
```

Om EXIF-datan innehåller information om bildorientering, kan du rotera digitalbilder i JPG-format utan att kvaliteten försämras. Detta är smidigt om du har en digitalkamera med inbyggd sensor för att känna av och spara kamerans orientering när bilden togs.

Normalt sett måste en JPG-bild öppnas och sparas om för att kunna ändras på något sätt. Men med hjälp av **jhead** kan du göra rotationen utan att bildens komprimering först måste avkodas, genom att skyffla runt bildinformationen utan att först avkoda den och därefter uppdatera EXIF-fältet (för dessa operationer tar **jhead** hjälp av **jpegtran**, som också ingår i samma paket).

Använd alternativet **-autorot** för att rotera JPG-bilder utan kvalitetsförlust.

Exempel:

```
jhead -autorot *.jpg
```

Detta roterar alla JPG-bilder i den aktuella mappen, så att de alltid visas med rätt orientering.

### 13.2.3 pstoedit

**$ pstoedit** Konverterar Postscript/PDF till andra vektorformat

**DEBIAN, UBUNTU:** **pstoedit** **FEDORA:** **imagemagick**

**HJÄLP:** **man pstoedit**

Om du använder Inkscape eller någon annan SVG-redigerare och vill kunna öppna filer exempelvis från Adobe Illustrator, kan **pstoedit** komma till undsättning. Verktyget klarar av att konvertera från Postscript, PDF, Adobe Illustrator (.ai) och EPS till en stor mängd olika vektorformat.

Kommandsyntax:

```
pstoedit vektorbild ny_vektorbild
```

där ”vektorbild” kan vara i något av formaten Postscript, PDF, Illustrator eller EPS. Filändelsen du använder för ”ny_vektorbild” styr vilket format filen får. Bland formaten du kan välja finns WMF/EMF, PDF och SVG.

Exempel:

```
pstoedit logga.ai logga.svg
```

Detta konverterar Adobe Illustrator-filen ”logga.ai” till SVG-format och sparar den nya filen som ”logga.svg”.

**OBS** Kommandot använder Ghostscript i konverteringsprocessen. Utan Ghostscript installerat fungerar inte kommandot.

### 13.2.4 epstopdf

---

**$ epstopdf** Konverterar från EPS till PDF

---

**DEBIAN, UBUNTU:** **epstopdf** **FEDORA:** **epstopdf**

---

**HJÄLP:** **man epstopdf**

---

Med kommandot **epstopdf** kan du konvertera en EPS-fil till PDF-format.

Kommandosyntax:

```
epstopdf fil.eps
```

Detta skapar en PDF med namnet ”fil.pdf” i den aktuella mappen.

**OBS** Kommandot använder Ghostscript för konverteringen. Utan Ghostscript installerat fungerar inte kommandot.

# Kapitel 14

# Matematiska verktyg

Mycket på kommandoraden handlar om text – men inte allt. Datorer är ju också sällsynt bra på att räkna. Här lär du dig om några bra matematiska verktyg.

## 14.1 units

**$ units** Konverterar mellan olika mått-, vikt- och volym-enheter

**DEBIAN, UBUNTU:** **units** **FEDORA:** **units**

**HJÄLP:** **man units**

Kommandot **units** konverterar mellan olika mått-, vikt- och volym-enheter, till exempel från centimeter till tum eller från liter till gallon.

Kör gärna programmet med alternativet **--verbose**, så visar programmet exakt vilka beräkningar det har gjort.

Du kan göra programmet interaktivt genom att helt enkelt skriva kommandonamnet.

Exempel:

```
units --verbose
```

Då får du först en fråga om vad du vill konverterar från, därefter en fråga vad du vill konvertera till, varpå

programmet ger dig svaret. Tryck på tangenterna CTRL+D för att avsluta programmet.

Du kan också ange det du vill konvertera från och till direkt som argument, då avslutas programmet direkt efter att det har gett dig svaret.

Exempel:

```
units --verbose 1km feet
```

Detta berättar hur många fot 1 kilometer är (svaret är 3 280,8399 fot).

## 14.2 numgrep

**$ numgrep** Visar rader som matchar ett sökmönster av siffror

**DEBIAN, UBUNTU: num-utils FEDORA: -**

**HJÄLP: man numgrep**

Kommandot **numgrep** fungerar ungefär på samma sätt som **grep**, men med enbart siffror. Med kommandot kan du söka inuti filer efter siffror. Kommandot är en del av paketet `num-utils`.

Användandet av kommandot följer vissa speciella regler. Kapsla alltid in sökuttrycken mellan / (snedstreck). Använd kommatecken för att separera uttryck och .. (två punkter) för att ange spann.

Kör kommandot rätt upp och ned för att låta det ta emot inmatning direkt från standard input:

```
numgrep
```

Du kan också ange en fil som indata.

Exempel:

```
numgrep /1..1000/ fil.txt
```

Ovanstående söker efter nummer mellan 1 och 1 000 i ”fil.txt”.

Läs mer om kommandot manual-sidan för **numgrep**.

## 14.3 bc

**$ bc** Interaktiv programmerbar miniräknare

**DEBIAN, UBUNTU: bc FEDORA: bc**

**HJÄLP: man bc**

Du kan använda kommandot **bc** både för snabba uträkningar och för extremt avancerade beräkningar.

I sin allra enklaste form startar du kommandot med kommandonamnet:

```
bc
```

Det första du bör göra innan du skriver in en beräkning är att ange med hur många decimalers noggrannhet du vill ha svaret. Detta gör du genom att skriva **scale=x** där x är valfri siffra, och trycka på ENTER. Därefter kan du skriva in din räkneoperation, exempelvis ”2/54”. Tryck på ENTER så får du svaret.

Skriv **quit** för att avsluta programmet.

Om du inte vill köra kommandot interaktivt utan direkt återgå till prompten efter att ha kört kommandot kan du använda **echo** tillsammans med rörtecknet för att skicka din operation till **bc**.

Exempel:

```
echo 'scale=5;2/54' | bc
```

Detta ger dig svaret direkt, med 5 decimalers noggrannhet.

# Kapitel 15

# Nätverkskommandon

Det här kapitlet ger dig information om olika verktyg som du kan använda med datorer som är uppkopplade på lokala nätverk och internet.

Kapitlet visar hur du kan konfigurera datorn i nätverket liksom föra över filer till, få information om och jobba med andra datorer i nätverket.

## 15.1 Nätverksstatistik- och information

### 15.1.1 netstat

| | |
|---|---|
| **# netstat** | Visar nätverksinformation av många olika slag |
| **DEBIAN, UBUNTU:** | **net-tools** **FEDORA:** **net-tools** |
| **HJÄLP:** | **man netstat** |

Kommandot **netstat** (kortform för *network statistics* på engelska) visar en mängd information om nätverksaktiviteterna på datorn, bland annat routing-tabellen, aktiva nätverksförbindelser, med mera. Kommandot har många alternativ, varav några är följande: alternativet **-r** ber kommandot visa routing-tabellen i kärnan, alternativet **-i** visar information om konfigurerade nätverksgränssnitt på datorn och alternativet **-t** visar aktiva TCP-förbindelser.

För att snabba upp kommandot kan du också lägga till alternativet **-n**, vilket ber kommandot använda IP-adresser i resultatet istället för värdnamn (vilka kan ta tid att slå upp i DNS-servrar).

Exempel:

```
netstat -nr
```

Ovanstående visar routing-tabellen i kärnan, använder IP-nummer istället för värdnamn.

```
netstat -i
```

Detta visar alla konfigurerade nätverksgränssnitt. I resultatet finns ett antal kolumner. Kolumnerna RX-OK och TX-OK visar hur många paket som har tagits emot respektive skickats framgångsrikt, varianterna RX-ERR och TX-ERR visar antal paket som kommit fram skadade, varianterna RX-DRP och TTX-DRP visar tappade paket.

```
netstat -ta
```

Detta visar aktiva och passiva socklar/kontakter (*sockets* på engelska), det vill säga en lista med alla servrar som antingen väntar på en förbindelse eller som har en aktiv förbindelse med en annan dator på nätet.

### 15.1.2 hostname

**# hostname** Visar och ställer in värdnamnet på datorn

DEBIAN, UBUNTU: **hostname** FEDORA: **net-tools**

HJÄLP: **man hostname**

Kommandot **hostname** talar om vad datorn du är inloggad på har för värdnamn.

### 15.1.3 ping

**$ ping** Testar förbindelsen till en dator i nätverket

DEBIAN, UBUNTU: **tcpdump** FEDORA: **iputils**

HJÄLP: **man ping**

Kommandot **ping** skickar frågor i form av små paket (*echo-request* på engelska) till en dator i nätverket. Om datorn är igång och är konfigurerad att svara på sådana frågor skickar det ett svar (*echo-response* på engelska) tillbaka. Kommandot är mycket användbart för att testa om nätverksförbindelsen till en dator fungerar eller inte.

Ange ett värdnamn eller ett IP-nummer som argument till kommandot.

Exempel:

```
ping www.dn.se
```

För att avbryta pingandet trycker du på CTRL+C, annars fortsätter det i all evighet.

### 15.1.4 traceroute

**$ traceroute** Spårar vägen som paket tar till en dator på nätverket

**DEBIAN, UBUNTU:** **traceroute** **FEDORA:** **traceroute**

**HJÄLP:** **man traceroute**

Kommandot **traceroute** visar vilken väg ett paket tar för att nå sitt mål. Det försöker visa alla värdar som ett paket passerar till en given destination (som kan vara ett IP-nummer eller ett värdnamn).

Exempel:

```
traceroute www.dn.se
```

### 15.1.5 tcpdump

**# tcpdump** Fångar och visar all nätverkstrafik

**DEBIAN, UBUNTU:** **tcpdump** **FEDORA:** **tcpdump**

**HJÄLP:** **man tcpdump**

Kommandot **tcpdump** fångar in alla paket som passerar ett nätverksgränssnitt (exempelvis ditt nätverkskort) och visar dem för dig. Du kan också spara paket för senare granskning.

Genom att köra kommandot rätt upp och ned ser du all TCP-trafik som passerar datorn. Kommandot är avancerat

och har otaliga alternativ. Användningsområdet är främst felsökning av program som kommunicerar på nätverket, eller av nätverket i sig.

För att köra kommandot måste du vara root-användaren.

### 15.1.6 nmap

**$ nmap** Utforskar och säkerhetstestar portar för datorer i nätverk

**DEBIAN, UBUNTU:** **nmap** **FEDORA:** **nmap**

**HJÄLP:** **man nmap**

Nätverksverktyget **nmap** är ett verktyg för att undersöka och säkerhetstesta datorer i ett nätverk. Kommandot används för kontrollera vilka portar som är öppna på fjärrdatorer och lokala datorer, men kan också ge information om vilket operativsystem och vilka programvaruversioner för tjänster som används. Kommandot kan användas för att kontrollera enstaka maskiner eller flera maskiner på en och samma gång.

Skriv in kommandot följt av värdnamnet eller IP-numret på den dator du vill kontrollera. Exempel:

```
nmap 192.168.0.3
```

Detta frågar datorn med IP-nummer ”192.168.0.3” på det lokala nätverket vilka portar som är öppna.

Kommandot är mycket kraftfullt, fullständig dokumentation finns i manual-sidan för kommandot liksom på `http://nmap.org`, hemsidan för **nmap**.

Du kommer åt fler av kommandots funktioner om du använder det som root-användaren.

### 15.1.7 findsmb

**$ findsmb** Visa SMB-kapabla datorer på nätverket

**DEBIAN, UBUNTU:** **smbclient** **FEDORA:** **samba-client**

**HJÄLP:** **man findsmb**

Med **findsmb** kan du snabbt hitta datorer på nätverket som svarar på på SMB-frågor (exempelvis Windowsdatorer som har utdelade mappar).

Kommandosyntax:

```
findsmb
```

Detta söker så brett som möjligt för att hitta alla tillgängliga maskiner. Du kan också ange ett visst subnät för att avgränsa sökresultatet.

## 15.2 Nätverkskonfiguration

### 15.2.1 ifconfig

**# ifconfig** Ställer in ett nätverksgränssnitt

**DEBIAN, UBUNTU: net-tools FEDORA: net-tools**

**HJÄLP: man ifconfig**

Kommandot **ifconfig** används för att konfigurera nätverksgränssnitt på datorn, eller för att visa den aktuella konfigurationen.

Använd kommandot för diagnos av aktuella inställningar genom att skriva kommandonamnet:

```
ifconfig
```

Detta visar all tillgänglig information om alla nätverksenheter som är igång på datorn.

Om du snabbt vill konfigurera ett nätverksgränssnitt från kommandoraden kan följande instruktion i tre steg hjälpa dig:

1. Konfigurera enheten ”eth0” att använda IP-nummer ”192.168.0.11”, med nätmasken ”255.255.255.0” samt sätta broadcast till ”192.168.0.255”:

   ```
   ifconfig eth0 192.168.0.11 netmask
   255.255.255.0 broadcast 192.168.0.255
   ```

2. Aktivera gränssnittet ”eth0”:

   ```
   ifconfig eth0 up
   ```

3. Använda kommandot **route** (se ▷ sektion 15.2.4) för att ställa in datorn att använda routern ”192.168.0.1” för all IP-trafik.

```
route add default gw 192.168.0.1
```

Ovanstående aktiverar nätverksgränssnittet ”eth0” med det angivna IP-numret och övriga nödvändiga uppgifter, så att du kan komma ut på nätet.

```
ifconfig eth0 down
```

Detta stänger av gränssnittet ”eth0” (förutsatt att det finns och är igång). Enheten kommer inte fungera förrän du kör kommandot igen med argumentet **up**.

Det finns många alternativ till kommandot. Ta som vanligt en titt i manual-sidan för att lära dig mer.

### 15.2.2 ifup

**# ifup** Aktiverar ett nätverksgränssnitt

**DEBIAN, UBUNTU: ifupdown FEDORA: initscripts**

**HJÄLP: man ifup**

Använd **ifup** för att aktivera nätverksgränssnitt som har definierats i filen `/etc/network/interfaces`, den fil som används för att ställa in nätverket i de flesta Linuxdistributionerna. När datorn startar använder den inställningarna i filen `/etc/network/interfaces` för att aktivera önskade gränssnitt (gränssnitt som ska starta automatiskt markeras i filen med texten ”auto”). Om du är nyfiken på mer information om hur du ställer in filen, se dess manualsida med **man interfaces**.

**OBS** På Fedora och Fedora-baserade distributioner sparas inställningarna i filer i mappen `/etc/sysconfig/network-scripts` istället.

Med kommandona **ifup** och **ifdown** kan du själv slå på och av gränssnitt definierade i filen.

Slå på ett avstängt gränssnitt enligt följande:

```
ifup eth0
```

Detta slår på ”eth0” under förutsättning att det finns och för tillfället är inaktivt.

### 15.2.3 ifdown

**# ifdown** Inaktiverar ett nätverksgränssnitt

**DEBIAN, UBUNTU: ifupdown FEDORA: initscripts**

**HJÄLP: man ifdown**

Läs också texten för kommandot **ifup** (▷ sektion 15.2.2) ovan för en introduktion till hur **ifdown** hör ihop med filen `/etc/network/interfaces`.

Använd **ifup** för att avaktivera nätverksgränssnitt som har definierats i filen `/etc/network/interfaces`.

Exempel:

```
ifdown eth0
```

Detta inaktiverar ”eth0” om det är aktivt.

### 15.2.4 route

**# route** Visar och ändrar routing-tabellen

**DEBIAN, UBUNTU: net-tools FEDORA: net-tools**

**HJÄLP: man route**

Kommandot **route** används för att visa och modifiera kärnans IP routing-tabellen.

Om du vill se routing-tabellen skriver du bara kommandots namn och trycker enter:

```
route
```

Ett vanligt användningsområde för kommandot är att ställa in den standard-gateway som du använder för att komma ut på nätet.

Exempel:

```
route add default gw 192.168.0.1
```

Detta använder kommandot **route** för att ställa in datorn att använda routern ”192.168.0.1” för all IP-trafik.

## 15.3 Internetverktyg

### 15.3.1 host

**$ host** Slår upp DNS-uppgifter

DEBIAN, UBUNTU: **bind9-host** FEDORA: **bind-utils**

HJÄLP: **man host**

Med **host** kan du slå upp IP-numret för ett domännamn, eller domännamnet för ett IP-nummer, med hjälp av DNS (förkortning av domännamnssystemet, *Domain Name System* på engelska).

Kommandosyntax:

```
host ip-adress
```

eller

```
host domännamn
```

### 15.3.2 dig

**$ dig** Slår upp DNS-uppgifter

DEBIAN, UBUNTU: **dnsutils** FEDORA: **bind-utils**

HJÄLP: **man dig**

Kommandot **dig** är mer kraftfullt än **host**. När du skriver kommandonamnet följt av ett värdnamn får du reda på diverse information om värden.

Exempel:

```
dig www.amazon.com
```

Ger dig information om ”www.amazon.com”.

För att ta reda på värdnamnet för en viss IP-adress (så kallad omvänd uppslagning, *reverse lookup* på engelska anger du alternativet **-x**.

```
dig -x 66.35.250.150
```

Detta slår upp IP-numret ”66.35.250.150” och ger adressen till värden som svar (om det finns en värd med det sökta

IP-numret, vill säga). Om 66.35.250.150 exempelvis hade råkat peka på `slashdot.org` skulle du få reda på det.

Du kan använda en stor mängd alternativ med **dig** (nästan för många, till och med). Se manual-sidan för information.

### 15.3.3 whois

**$ whois** Slår upp uppgifter i whois-databaser

DEBIAN, UBUNTU: **whois** FEDORA: **jwhois**

HJÄLP: **man whois**

Kommandot **whois** används för att slå upp information i whois-databaser. Observera att informationen i dessa databaser ofta är begränsad, eftersom den kan användas som startpunkt för illasinnade som söker information för att orsaka skada.

## 15.4 Verktyg för fjärradministration

### 15.4.1 ssh

**$ ssh** Loggar in på fjärrdatorer säkert

DEBIAN, UBUNTU: **openssh-client** FEDORA: **openssh-clients**

HJÄLP: **man ssh**

**ssh** står för säkert skal (*secure shell* på engelska) och låter dig logga in med krypterad förbindelse på maskiner som kör demonen **sshd**, förutsatt att du har ett användarkonto på maskinen, förstås. När du väl är inloggad får du tillgång till ett skal precis som om du vore inloggad på maskinen lokalt. Du kan utföra kommandon, som att kopiera filer eller starta om datorn, precis som om du satt framför din egen dator.

OBS **ssh** är ett mycket säkrare alternativ för fjärranslutning än **telnet**. Du bör alltid använda **ssh**, om möjligt.

Kommandosyntax:

```
ssh värdnamn
```

Ovanstående ansluter dig till datorn med ditt aktuella användarnamn, varpå du ombeds att skriva in ditt lösenord på datorn. Du kan använda ett IP-nummer istället för ett värdnamn.

Du kan också manuellt ange det användarnamn du vill ansluta med:

Exempel:

```
ssh pelle@192.168.0.34
```

Detta ansluter till maskinen med ”192.168.0.34” med användarnamnet ”pelle”. Du behöver förstås känna till lösenordet för användaren du försöker ansluta med.

### 15.4.2 screen

**$ screen** Hanterar virtuella skärmar

**DEBIAN, UBUNTU:** **screen** **FEDORA:** **screen**

**HJÄLP:** **man screen**

**screen** är ett väldigt användbart program som du kan använda för att växla mellan flera virtuella terminaler i ett och samma terminalfönster. **screen** är är en kommandoradsbaserad fönsterhanterare. Om du har tillgång till virtuella terminaler är nyttan med **screen** liten, men det är fantastiskt användbart när du loggar in på datorer via **ssh** och liknande fjärranslutningsverktyg (se ▷sektion 15.4). Programmet använder sig av tangentbordsgenvägar. Om du skriver:

```
screen
```

på kommandoraden öppnas en virtuell terminal. Du kan skapa en ny genom att trycka på CTRL+A och sedan på CTRL+C.

Använd CTRL+A CTRL+N för att hoppa till nästa virtuella terminal. Använd CTRL+A CTRL+P för att gå till föregående virtuella terminal. Pröva också att trycka ned tangentkombinationerna två gånger, CTRL+A CTRL+A, för att växla mellan de två senast använda terminalerna.

Du kan även pausa **screen** om du behöver logga ut från terminalen. Tryck CTRL+A CTRL+Z innan du loggar ut.

Efter att du har loggat in kan du återuppta sessionen med **screen -r**.

**screen** har också diverse andra förmågor som du kan pröva. Dokumentationen och handledningarna till programmet är välskrivna – läs manual-sidan och sök information på nätet. CTRL+A ? visar en kort översikt över tillgängliga tangentbordsgenvägar.

## 15.5 Internetprogram

### 15.5.1 ftp

**$ ftp** Överför filer mellan datorer

**DEBIAN, UBUNTU:** **netkit-ftp** **FEDORA:** **ftp**

**HJÄLP:** **man ftp**

Programmet **ftp** gör det möjligt att ansluta till FTP-servrar på nätet, för att ladda ned och/eller upp filer mellan din dator och en fjärrdator. Du startar programmet genom att skriva kommandot följt av den server du vill ansluta till.

Kommandosyntax:

```
ftp -alternativ server
```

Använd alternativet **-p** (för passivt läge, av *passive* på engelska) om din dator finns bakom en brandvägg som hindrar ingående anslutningar. Testa alternativet om **ftp** krånglar.

När du väl har startat programmet byts din vanliga kommandoprompt till **ftp**-programmets egen prompt. Följande är några av de vanligaste kommandona du kan köra:

**dir** Visar innehållet i mappar på fjärrdatorn (motsvarar **ls** i Bash). Om du inte anger ett mappnamn visar kommandot innehållet i den mapp du står i.

**cd** Byter till den mapp på fjärrdatorn som du anger som argument.

**put** Använd det här kommandot för att ladda upp en fil från din lokala dator till fjärrdatorn. Ange som argument namnet på den fil som du vill föra över, exempelvis så här: **put /home/bertil/dokument/fil.txt**

eller kort och gott **put fil.txt** om ”fil.txt” ligger i den mapp du stod i när du startade **ftp** i skalet.

**get** Använd **get** för att ladda ned en fil från fjärrdatorn till din lokala dator. Ange namnet på den fil du vill föra över som argument.

**mput** Med **mput** kan du ladda upp flera filer från din lokala dator till fjärrdatorn. Du ange ett eller flera filnamn som argument, liksom använda jokertecken som * och ? (jokertecknen tolkas på samma sätt som i skalet, se ▷sektion 3.4 för mer information).

**mget** Använd detta för att ladda ned flera filer från fjärrdatorn till din lokala dator. Liksom med ovanstående fungerar detta med flera filnamn som argument eller med jokertecken (se ▷sektion 3.4).

**mkdir** Använd detta kommando för att skapa en ny mapp på fjärrdatorn. Ange den nya mappens namn som argument.

**help** Visar en lista med alla tillgängliga kommandon.

**bye** Avslutar anslutningen och programmet (du kan också använda CTRL+D).

Läs om alla tillgängliga alternativ och kommandon i manual-sidan.

### 15.5.2 sftp

**$ sftp** Överför filer säkert mellan datorer

DEBIAN, UBUNTU: **openssh-client** FEDORA: **openssh-clients**

HJÄLP: **man sftp**

Kommandot **sftp** (säker ftp, *secure ftp* på engelska) har samma funktioner som **ftp** (▷sektion 15.5.1), men använder sig av krypteringen i **ssh** för anslutningen. Det är därför mycket säkrare än den vanliga **ftp**-programmet.

Starta programmet genom att ange kommandot följt av den server du vill ansluta till.

```
sftp användare@fjärrdator:/sökväg/till/mapp
```

**sftp** använder sig därefter av samma FTP-kommandon som **ftp**, exempelvis **help** för att se en hjälptext, **put** för att skicka filer till en server, **get** för att ladda ned från en server och många andra.

Se avsnittet om **ftp** (▷sektion 15.5.1) och manualsidan för mer information.

### 15.5.3 scp

**$ scp** Kopierar filer till/från fjärrdatorer säkert

**DEBIAN, UBUNTU:** **openssh-client** **FEDORA:** **openssh-clients**

**HJÄLP:** **man scp**

Med kommandot **scp** (säker kopiering, *secure copy* på engelska) kan du kopiera filer till fjärrdatorer. Kommandot använder sig av **ssh** för att kryptera anslutningen, vilket gör filöverföringen säker.

Kommandosyntax:

```
scp kopiera_från kopiera_till
```

Sökvägarna kan vara både lokala mappar eller mappar på en fjärrmaskin. Sökvägen på fjärrmaskiner skrivs vanligvis på formen ”värdnamn:/sökväg/till/rätt/mapp”.

Exempel:

```
scp index.html 192.168.0.25:/var/www/hemsida
```

I ovanstående exempel kopieras filen ”index.html” (som ligger i mappen du står i för tillfället) till fjärrdatorn med IP-nummer 192.168.0.25. Filen placeras i mappen ”/var/www/hemsida”.

**OBS** Om du utesluter mappstrukturen väljer kommandot automatiskt att använda hemmappen på fjärrdatorn.

Med alternativet **-r** kan du kopiera filer rekursivt, det vill säga för att föra över hela mappar och allt dess innehåll.

```
scp -r bilder 192.168.0.25:/var/www/hemsida
```

Detta kopierar mappen ”bilder”, samt allt innehåll i den mappen, inklusive andra mappar, till ”/var/www/hemsida” på datorn med IP-nummer 192.168.0.25.

Du kan också använda kommandot för att kopiera från en fjärrdator till en annan:

```
scp -r gammal.se:/dokument ny.se:/dokument
```

Om vi antar att du för närvarande är inloggad som användaren ”bertil” behöver du ha ett användarkonto med namnet ”bertil” både på ”gammal.se” och ”ny.se” i exemplet ovan. Annars kan du ange valfria andra användarnamn för respektive värd.

Exempel:

```
scp -r maja@gammal.se:/filer berra@ny.se:/filer
```

Om du vill kopiera från en fjärrdator till din lokala dator går det också bra – vänd bara på steken.

Exempel:

```
scp gammal.se:/dokument/* .
```

Detta kopierar alla filer i mappen ”dokument” på ”gammal.se” till den mapp du står i för tillfället.

**OBS** När du kopierar filer till eller från en fjärrmaskin måste du använda ett kolon (:) efter värdnamnet. Annars kommer kopieringen att misslyckas.

### 15.5.4 rsync

**$ rsync** Kopierar filer säkert och platsbesparande

**DEBIAN, UBUNTU:** **rsync** **FEDORA:** **rsync**

**HJÄLP:** **man rsync**

**rsync** är ett flexibelt verktyg för att kopiera filer från en lokal maskin till en annan lokal maskin, från en lokal maskin till en fjärrmaskin (och vice versa), liksom till och från rsync-servrar. I Debian, Ubuntu och Fedora är **rsync** inställt så att anslutning till fjärrdatorer görs krypterat med hjälp av **ssh**.

Tack vare avancerade jämförelseberäkningar kopierar endast **rsync** nya och ändrade filer, och bland de ändrade filerna kopierar det enbart skillnaderna mellan filerna (inte hela filer). Det gör att **rsync** sparar både tid och bandbredd

– vilket gör det till ett utmärkt verktyg för säkerhetskopiering över nätet.

Med **rsync** kan du också utforma avancerade exklusionskriterier (som med **tar**). Kommandot har en välskriven manual-sida, se den för ytterligare information.

Kommandosyntax:

```
rsync -alternativ kopiera_från kopiera_till
```

Exempel:

```
rsync -avz /home/bertil berra@server.se:/backup
```

Detta säkerhetskopierar allt innehåll i ”/home/bertil” rekursivt (med alternativet **-a** för arkivera, *archive* på engelska) från till mappen ”/backup” på datorn ”server.se”, med användarnamnet ”berra”. Alternativet **-v** (*verbose* på engelska, ungefär utförlig) gör att kommandot ger mer information än annars om hur överföringen fortskrider, alternativet **-z** gör att kommmandot komprimerar filerna under överföringen, vilket sparar ytterligare tid.

Nästa gång kommandot körs går det betydligt snabbare än första, eftersom endast skillnaderna mellan två olika versioner förs över till destinationen.

En uppsättning bra exempel på skript för säkerhetskopiering finns på `http://www.samba.org/rsync/examples.html`.

### 15.5.5 wget

**$ wget** Laddar ned filer från nätet

**DEBIAN, UBUNTU:** **wget** **FEDORA:** **wget**

**HJÄLP:** **man wget**

Med **wget** kan du ladda ned enstaka filer från nätet direkt via kommandoraden. Du kan också tanka ned hela sajter – HTML-filer, bilder, stilmallar och andra filer.

Kommandosyntax:

```
wget url_till_fil
```

Detta laddar helt enkelt ned den HTML-fil som finns på den angivna webbadressen.

Kommandot accepterar olika alternativ:

**-m** Laddar ned en komplett webbsajt från den angivna webbadressen. Detta inkluderar HTML-sidor, bilder och alla andra filer som finns på den angivna webbadressen. Hämtar även undersidor, men inte sidor som länkas till andra webbsidor än den angivna. **-m** står för spegel (*mirror* på engelska).

**-nc** Som standard skriver inte **wget** över en fil om den redan finns, utan lägger till ett löpnummer till den senaste nedladdade versionen. Du kan stänga av detta beteende, det vill säga helt hindra **wget** från att ladda ned filer som redan finns, med alternativet **nc**. Läs manual-sidan för kommandot för ytterligare information om alternativet.

**-c** Använd alternativet **-c** för att fortsätta ladda ned en fil som bara delvis har laddats ned tidigare. **-c** står för fortsätt (*continue* på engelska).

**wget** kan också ladda ned flera filer med hjälp av vanliga jokertecken (samma som i Bash), som * och ?. Använd **wget** som vanligt, men sätt enkla apostrofer kring adressen för att hindra Bash från att försöka expandera jokertecknen först.

Kommandot har flera användbara finesser och alternativ. Du hittar information om alla dessa i kommandots manualsida. Några exempel på avancerade användningsområden:

```
wget --spider --force-html -i bookmarks.html
```

Detta säger åt **wget** att läsa igenom filen ”bookmarks.html” och kollar att alla länkar fungerar som de ska.

```
wget -r -p --convert-links http://www.gnu.org
```

Detta laddar ned hemsidan `www.gnu.org` rekursivt (med alternativet **-r**, det vill säga att **wget** också hämtar hem sidor i undermappar, samt laddar ned alla bilder, stilmallar och andra filer som behövs för att visa sidan korrekt (alternativet **-p**). Dessutom konverterar kommandot alla länkar i sidorna så att de pekar på lokala filer (alternativet **--convert-links**. Detta gör det möjligt att surfa på den

nedladdade versionen av hemsidan utan att var uppkopplad på nätet.

```
wget -r -l1 --no-parent -A *.png http://www.fsf.org
```

Detta laddar ned rekursivt (**-r**) från ”www.fsf.org” till ett djup av 1 nivå (**-l1**. Kommandot ignorerar referenser till föräldramappen (**--no-parent**) och laddar också ned alla filer som slutar på ”.png” (**-A *.png**).

Om du också hade velat ladda ned alla PDF-filer hade du kunnat ange **-A *.pdf**.

### 15.5.6 curl

**$ curl** Laddar ned filer från nätet

**DEBIAN, UBUNTU:** **curl** **FEDORA:** **curl**

**HJÄLP:** **man curl**

**curl** är ett alternativ till **wget**, med ungefär samma funktionalitet. Kommandot använder en mängd smarta tricks för att göra olika saker och kan ladda ned från dict-, ldap-, ftp-, http- och gopher-servrar (med mycket mera). Se manualsidan för mer information.

Du använder **curl** på ungefär samma sätt som **wget**. Du kan även ange ett användarnamn och ett lösenord för lösenordsskyddade sidor, med hjälp av alternativet **-u**.

Exempel:

```
curl -u namn:lösenord http://sajt.se/fil.txt
```

Du kan också ladda upp filer med alternativet **-T**.

Exempel:

```
curl -T fil.txt ftp://ftp.uploadsite.com
```

Du kan fortsätta en avbruten nedladdning med alternativet **-C**. För att be kommandot att automatiskt hitta och fortsätta från platsen för avbrottet genom att titta i den delvis nedladdade filen lägger du till ett minus-tecken till alternativet.

Exempel:

```
curl -C - -o file http://www.site.com
```

### 15.5.7 lynx

**$ lynx** Webbläsare för kommandoraden

**DEBIAN, UBUNTU:** **lynx** **FEDORA:** **lynx**

**HJÄLP:** **man lynx**

Det går om inte utmärkt så i alla fall bra att surfa på nätet utan ett grafiskt gränssnitt, tack vare programmet **lynx**. Programmet är en fullfjädrad webbläsare, förutom att den enbart visar text, inte bilder och annan grafik.

Kommandosyntax:

```
lynx
```

Du kan också starta **lynx** med valfri hemsida.
Exempel:

```
lynx http://www.hme.se
```

Du flyttar dig mellan länkarna på sidan med hjälp av TAB- och piltangenterna på tangentbordet. Öppna en länk genom att flytta dig till den och sedan trycka ENTER. Avsluta programmet genom att trycka på Q och sedan bekräfta genom att trycka på Y.

# Kapitel 16

# Säkerhet

Det här kapitlet ger dig en grundläggande introduktion till filrättigheter och säkerhet i GNU/Linux.

## 16.1 Filrättigheter

### 16.1.1 Kort introduktion

Varje fil i ett Linuxsystem har en *ägare* (*owner* på engelska) och tillhör en *grupp* (*group* på engelska). Dessa två utgör två olika *användarklasser*. Dessutom finns en tredje användarklass som innefattar alla *andra* användare (*others* på engelska).

| Bokstav | Betydelse |
|---|---|
| `u` | Ägare, *user* på engelska |
| `g` | Grupp, *group* på engelska |
| `o` | Andra, *other* på engelska |

Tabell 16.2: Användarklasser i GNU/Linux.

För var och en av de tre användarklasserna finns det tre möjliga rättighetsinställningar – *läs-* (*read* på engelska),

*skriv-* (*write* på engelska) och *kör*-rättigheter (*execute* på engelska).

| Bokstav | Betydelse |
|---|---|
| `r` | Läs, *read* på engelska |
| `w` | Skriv, *write* på engelska |
| `x` | Kör, *execute* på engelska |

Tabell 16.4: Bokstavskoder för rättigheter i GNU/Linux.

Använd kommandot **`ls -l`** för att se filrättigheterna för de respektive användarklasserna för filerna i mappen du står i, eller ange ett specifikt filnamn som argument för att se filrättigheterna enbart för den filen.

Exempel:

```
ls -l fil.txt
```

Du ser då ett resultat liknande detta:

```
-rw-r--r-- 1 bertil bertil 481842 2008-02-29 12:29 fil.txt
-rwxr-xr-x 1 bertil bertil 4096   2008-02-29 12:29 skript.sh
drwxr-xr-x 7 bertil bertil 4096   2007-09-09 11:45 dokument
```

Den första kolumnen innehåller rättighetsinformationen, där `r` står för läs (*read* på engelska), `w` står för skriv (*write* på engelska) och `x` står för kör (*execute* på engelska).

Den allra första positionen i den första kolumnen anger om det rör sig om en fil eller en mapp. `-` betyder att det rör sig om en vanlig fil, `d` att det är en mapp (*directory* på engelska) och `l` att det är en symbolisk länk. De tre följande positionerna anger rättigheterna för ägaren, de tre därpå följande rättigheterna för gruppen och de tre sista rättigheterna för alla andra.

För ”fil.txt” är rättigheterna sålunda följande: ägaren har rätt att läsa och skriva (`rw`) filen, gruppen har rätt att läsa (`r`) filen och alla andra har rätt att läsa (`r`) den.

För mappen ”dokument” gäller att ägaren har rätt att läsa, skriva och köra (`rwx`) innehållet i mappen, gruppen har rätt att läsa och köra (`r-x`) innehållet och alla andra har rätt att köra (`x`) innehållet. För en mapp är köra detsamma som att man har rätt att öppna den.

### 16.1.2 chmod

**# chmod** Ändrar rättigheterna till en fil

**DEBIAN, UBUNTU:** `coreutils` **FEDORA:** `coreutils`

**HJÄLP:** `man chmod`

För att ändra rättigheterna för en fil eller en mapp använder du kommandot **chmod** (kortform för ändra läge, *change mode* på engelska). Det finns två sätt att ange rättigheterna för en fil med kommandot – med bokstäver eller siffror. Bokstavsmetoden är vanligtvis enklast att komma igång med.

#### Bokstavsmetoden

Använd **+** eller **–** (plus- eller minus-tecken) tillsammans med bokstaven för en användarklass för att lägga till eller dra ifrån rättigheter för den användarklassen. Du kan också använda **=** (likamed-tecken) för att ändra befintliga rättigheter för en klass.

| Tecken | Betydelse |
|---|---|
| + | Lägger till rättighet(er) |
| – | Tar bort rättighet(er) |
| = | Sätter rättighet(er) och slänger de tidigare |

Tabell 16.6: Ändra rättigheter i GNU/Linux med bokstavsmetoden.

Du kan använda följande bokstavskoder för användarklasserna: **u** med betydelsen användare (*user* på engelska), **g** med betydelsen grupp (*group* på engelska) och **o** med betydelsen andra (*other* på engelska). Bokstaven **a**, med betydelsen alla (*all* på engelska), fungerar som en genväg för samtliga tre.

Se följande exempel:

```
chmod u+rw fil.txt
```

Detta ger användaren läs- och skrivrättigheter.

```
chmod o-rwx fil.txt
```

Detta tar bort läs-, skriv- och kör-rättigheter från klassen andra användare.

```
chmod a+r fil.txt
```

Detta ger alla läs-rättigheter för filen.

```
chmod a=rx fil.txt
```

Detta ger alla läs- och kör-rättigheter (om någon klass har läs-rättigheter tas de samtidigt bort).

### Siffermetoden

Istället för bokstäver kan du använda siffror för att sätta filrättigheterna. Följande siffror används för var och en av de olika klasserna:

**4** läs-rättigheter

**2** skriv-rättigheter

**1** kör-rättigheter

Siffrorna kan adderas så att du i ett slag kan sätta läs-, skriv- och kör-rättigheter för var och en av de olika användarklasserna.

**6** läs- och skriv-rättigheter (4+2)

**5** läs- och kör-rättigheter (4+1)

**7** läs-, skriv- och kör-rättigheter (4+2+1)

Exempel:

```
chmod 777 fil.txt
```

Detta ger alla användarklasser rätt att läsa, skriva och köra ”fil.txt”. Första siffran anger filrättigheterna för ägaren, den andra för gruppen och den sista för alla andra.

```
chmod 521 fil.txt
```

Detta ger ägaren läs- och kör-rättigheter, gruppen skriv-rättigheter och alla andra kör-rättigheter. Detta exempel är i praktiken inte särskilt användbart, men det visar tydligt hur enkelt och flexibelt filrättigheterna kan sättas.

### 16.1.3 chown

**# chown** Ändrar ägare och grupp för en fil

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man chown**

**chown** ändrar ägare och/eller grupp för en fil (därav namnet, som är en kortform av byt ägare, *change owner* på engelska). Kommandot kan endast användas av root-användaren.

Använd alternativet **-R** för att ändra rekursivt, det vill säga alla filer i den angivna mappen, inklusive alla andra filer och mappar i mappen.

Kommandosyntax:

```
chown ägare:grupp fil
```

Exempel:

```
chown bertil:webbutvecklare index.html
```

Ovanstående ändrar ägaren till filen ”index.html” till ”bertil”, samt sätter gruppen till ”webbutvecklare”.

### 16.1.4 sticky bit

Utöver de grundläggande inställningarna för läs-, skriv- och kör-rättigheter finns det tre extra lägen som du kan slå på och av. En av dem är något som kallas *sticky bit*.

Om flera personer behöver tillgång till en mapp är det en bra idé att låta alla dessa personer bli medlemmar i en och samma grupp. Då kan du göra mappen läs- och skrivbar för gruppen – vips så har alla rätt att lägga till filer i mappen. Men var och en av medlemmarna i gruppen får samtidigt rätt att radera filer i mappen. En viss användare kan radera en annan användares fil(er).

För att förhindra detta kan du sätta den så kallade *sticky bit* (klistriga biten) på mappen. Då kan användare endast radera sina egna filer, trots att mappen är skrivbar av alla medlemmar i gruppen. De enda som kan ändra, radera eller flytta en given fil är ägaren till filen, ägaren till mappen som filerna ligger i och root-användaren.

Ett exempel på en mapp som i Linux alltid brukar ha *sticky bit* påslagen är mappen `/tmp`. Mappen används av olika program för att spara temporära filer, och *sticky bit* förhindrar att en användare raderar någon annan användarens temporära filer.

Att *sticky bit* är satt på mappen indikeras av att sista positionen i första kolumnen är ett `t`.

```
drwxrwxrwt 13 root root 4096 2008-02-29 15:07 /tmp
```

### Sätta *sticky bit* på en mapp

Gör så här för att sätta *sticky bit* på en mapp med bokstavsmetoden:

```
chmod +t mappens_namn
```

där **`+t`** har betydelsen slå på *sticky bit*.
Eller skriv följande för att använda siffermetoden:

```
chmod 1775 mappens_namn
```

där första siffran, 1, har betydelsen slå på *sticky bit*. Övriga inställningar för ägare, grupp och andra spelar här ingen roll, de kan vara vad som helst annat än just ”775”.

### Ta bort *sticky bit* på en mapp

Gör så här för att ta bort *sticky bit* på en mapp:

```
chmod -t mappens_namn
```

där **`-t`** har betydelsen ta bort *sticky bit*.
Eller skriv följande:

```
chmod 0700 mappens_namn
```

där första siffran, 0, har betydelsen slå av *sticky bit*. Övriga inställningar för ägare, grupp och andra spelar här ingen roll, de kan vara vad som helst annat än just ”700” som i exemplet.

### 16.1.5 suid

Vanligtvis ägs en process av den användare som startade den. Om du till exempel startar kommandot **top** visas du som ägare till processen `top` om du kör **ps u**.

Om du slår på *suid*, sätt användar-id (*set user id* på engelska) för en körbar fil innebär det att processen alltid ägs av ägaren till filen, oavsett vilken användare som startade processen. Det innebär en potentiell säkerhetsrisk att slå på läget, särskilt för körbara filer som ägs av root, så var försiktig.

Använd **chmod** för att slå på *suid*.

Exempel:

```
chmod u+s skript.sh
```

Ovanstående ger alla rätt att köra ”skript.sh” med de rättigheter som ägaren till skriptet har.

**OBS** Undvik så långt det går att slå på *suid*-läget, eftersom det kan innebära säkerhetsrisker för ditt system.

### 16.1.6 chattr

**# chattr** Ändrar filattribut i filsystemet ext2/ext3

**DEBIAN, UBUNTU: e2fsprogs FEDORA: e2fsprogs**

**HJÄLP: man chattr**

Med **chattr** (betydelse ändra attribut, *change attributes* på engelska) kan du ställa in vissa egenskaper, eller attribut, för filer på filsystemsnivå (kommandot är bara till nytta om du har formaterat partitionen med något av filsystemen ext2 eller ext3). Ett användbart attribut är flaggan *immutable* (ungefär oföränderlig på svenska), vilket gör att en fil inte kan uppdateras. Det kan vara bra att använda för att skydda filer från att ändras exempelvis i samband med systemuppdateringar eller -uppgraderingar.

Använd kommandonamnet tillsammans med **+** för att lägga till attribut eller **–** för att ta bort attribut, följt av själva attributet.

Exempel:

```
chattr +i /sbin/lilo.conf
```

Detta lägger till flaggan *immutable* på filen ”lilo.conf”. När detta är gjort kommer filen inte att kunna redigeras. Du måste vara root-användaren för att kunna lägga till den här flaggan.

För att ta bort attributet använder du istället ett minustecken:

```
chattr -i /sbin/lilo.conf
```

Detta gör det åter möjligt att redigera filen.

Förutom alternativet **i** finns det några andra attribut som **chattr** kan ändra. Här är några av dem:

**a** För en fil med attributet *a*, med betydelsen ”endast lägg till” (*append only* på engelska), kan inget innehåll raderas, endast läggas till. I en mapp med detta attribut kan du lägga till filer, men inte döpa om eller radera befintliga filer. Attributet kan endast sättas av root-användaren.

**s** När en fil med attributet **s**, med betydelsen ”säker radering” (*secure deletion* på engelska), raderas skrivs innehållet över med nollor på hårddisken (som med kommandot **shred**). Detta fungerar med ext2- och ext3-filsystem.

### 16.1.7 lsattr

**# lsattr** Visar filattribut i filsystemet ext2/ext3

**DEBIAN, UBUNTU:** **e2fsprogs** **FEDORA:** **e2fsprogs**

**HJÄLP:** **man lsattr**

Med **lsattr** (betydelse visa attribut, *list attributes* på engelska) kan du kontrollera vilka attribut som är satta på filer och mappar med **chattr**.

Använd alternativet **-R** för att visa resultatet rekursivt, använd alternativet **-d** tillsammans med ett mappnamn för att visa attribut satta på mappen istället för att visa attribut satta på filer som ligger i mappen.

Kommandosyntax:

```
lsattr
```

Detta visar attribut för filer i den aktuella mappen.

Du kan också ange en viss fil eller en viss mapp.
Exempel:

```
lsattr mapp_eller_fil
```

## 16.2 Säkerhetsverktyg

### 16.2.1 md5sum

**$ md5sum** Beräknar och kontrollerar MD5-checksummor

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man md5sum**

Med kommandot **md5sum** kan du skapa en 128-bitars checksumma av slaget md5 för en fil. Du kan använda kommandot för att verifiera att en kopia av en fil är precis lika dan som originalet, exempelvis efter att du har kopierat den från en dator till en annan över nätet. Om någon enstaka bit har ändrats i filen blir checksummorna för ursprungsfilen och den nya kopian olika, om allt har gått bra i filöverföringen blir checksummorna helt identiska.

Kommandosyntax:

```
md5sum fil
```

På många ställen där du kan ladda ned skivavbildningar (.iso-filer) anges en checksumma. När du är klar med nedladdningen kan du själv köra kommandot för att se att du får samma checksumma. Om du får det är filen helt intakt.

Exempel:

```
md5sum ubuntu-8.10-desktop-i386.iso
```

### 16.2.2 sha1sum

**$ sha1sum** Beräknar och kontrollerar SHA1-checksummor

**DEBIAN, UBUNTU:** **coreutils** **FEDORA:** **coreutils**

**HJÄLP:** **man sha1sum**

Kommandot **sha1sum** fungerar på samma sätt som **md5sum**, men använder en annan metod för att beräkna checksumman (160-bitar).

Kommandosyntax:

```
sha1sum fil
```

### 16.2.3 pwgen

**$ pwgen** Slumpar fram lösenord

**DEBIAN, UBUNTU:** **pwgen** **FEDORA:** **pwgen**

**HJÄLP:** **man pwgen**

Med kommandot **pwgen** kan du slumpa fram ett säkert lösenord av valfri längd. Du kan använda kommandot för att skapa ett lösenord till en ny användare på datorn, eller om du själv inte kan komma på ett säkert lösenord när du behöver det.

Exempel:

```
pwgen 12
```

Detta skapar ett tolv tecken långt lösenord.

### 16.2.4 mkpasswd

**$ mkpasswd** Skapar och krypterar lösenord

**DEBIAN, UBUNTU:** **mkpasswd** **FEDORA:** **expect**

**HJÄLP:** **man mkpasswd**

Kommandot **mkpasswd** är ett alternativ till **pwgen** med betydligt fler funktioner än **pwgen**, bland annat möjlighet att kryptera lösenord med olika krypteringsmetoder (DES, MD5, SHA-256 eller SHA-512) och ett krypteringssalt ().

Kommandosyntax:

```
mkpasswd lösenord salt
```

Du kan också använda det för att slumpa fram ett nytt lösenord.

# Kapitel 17

# Arkivera och komprimera filer

I det här kapitlet får du lära dig hur du kan arkivera filer med några lättanvända och populära kompressionsverktyg.

Du kan ha stor nytta av kunskaperna i det här kapitlet exempelvis för att minska storleken på filer, packa upp paket du har laddat ned från nätet, eller skapa arkiv av hela mappar för att kunna skicka dem med e-post.

## 17.1 tar

| | |
|---|---|
| **$ tar** | Arkiverar filer till tar-bollar |
| **DEBIAN, UBUNTU:** | **tar** **FEDORA:** **tar** |
| **HJÄLP:** | **man tar** |

Med programmet **tar** kan du samla ihop mappar och filer till en arkivfil. Historiskt har kommandot använts för att spara säkerhetskopior på magnetband (namnet kommer av ordet bandarkiv, *tape archive* på engelska. Sådana arkivfiler kallas ofta tar-bollar. Nu för tiden komprimeras ofta tarbollar med hjälp av ett komprimeringsverktyg, exempelvis **gzip** eller **bzip2**, för att spara plats och göra det enklare att flytta dem över nätverk. För att göra det lättare för an-

vändaren att skapa och packa upp komprimerade tar-bollar kan **tar** automatiskt ta hjälp av komprimeringsprogram.

Alternativ tillgängliga i **tar**:

**-c** Skapa ett arkiv.

**-x** Extrahera (packa upp) från ett arkiv.

**-v** Visa mer detaljerad utdata, inklusive detaljinformation om vilka filer som behandlas för tillfället.

**-f** Anger namnet på den fil som ska skapas/packas upp från. Ska alltid anges som det sista alternativet.

**-z** Komprimera filen med **gzip**.

**-j** Komprimera filen med **bzip2**.

**-p** Ändra inte filrättigheter och datumstämplar.

**--exclude=filspecifikation** Hindrar de angivna filerna från att inkluderas i arkivet (filspecifikationen kan innehålla jokertecken som matchar ett visst mönster, eller ett filnamn).

**-t** Visa innehållet i ett arkiv, utan att packa upp det.

Exempel:

```
tar -cvpf namn_på_arkiv.tar filer_att_inkludera
```

Detta skapar ett tar-arkiv, utan att komprimera det, med filrättigheter och -tidstämplar intakta.

```
tar -cvpzf namn_på_arkiv.tar.gz filer_att_inkludera
```

Detta skapar ett tar-arkiv och komprimerar det med **gzip**, med filrättigheter och -tidstämplar intakta.

```
tar -zxvpf mitt_arkiv.tar.gz
```

Detta packar upp filerna (och berättar utförligt vilka filer som behandlas) från ett komprimerat tar-arkiv.

## 17.2 Kompressionsverktyg

Det finns två huvudprogram för kompression i GNU/Linux, **bzip2** och **gzip**. Vanligtvis används **tar** för att först sätta ihop en mängd filer till en enda tar-boll, sedan används något av kompressionsverktygen för att minska storleken på tar-bollen.

Detta är helt i linje med UNIX-filosofin att låta varje verktyg göra en enda sak, men göra det väl. För att förenkla handhavandet har **tar** dock alternativ som automatiskt kallar på något av kompressionsverktygen för att packa ihop och komprimera i ett moment.

### 17.2.1 gzip

**$ gzip** Komprimerar eller packar upp filer i gzip-format

**DEBIAN, UBUNTU:** **gzip** **FEDORA:** **gzip**

**HJÄLP:** **man gzip**

**gzip** är det mest använda kompressionsformatet på UNIX-liknande operativsystem. Du kan förstås använda **gzip** med valfri fil, inte bara tar-bollar.

```
gzip fil.tar
```

Detta komprimerar ett tar-arkiv med **gzip**, vanligtvis med filändelsen .gz.

```
gunzip fil.tar.gz
```

Detta avkomprimerar ”fil.tar.gz” till ”fil.tar” och lämnar filen i den aktuella mappen.

**TIPS** Både **gzip** och **bzip2** (se ▷sektion 17.2.2) följer det med olika verktyg som du kan använda för att arbeta med komprimerade filer, exempelvis verktyg för att visa filer i arkiv utan att packa upp dem, köra **less** på dem eller använda en specialvariant av **grep** för att söka inuti filer i ett arkiv, med mera.

För **gzip** har de tillgängliga verktygen prefixet ’z’: **zcat**, **zless**, **zgrep**.

### 17.2.2 bzip2

**$ bzip2** Komprimerar eller packar upp filer i bzip2-format

**DEBIAN, UBUNTU:** **bzip2** **FEDORA:** **bzip2**

**HJÄLP:** **man bzip2**

**bzip2** är en nyare kompressionsprogram som erbjuder bättre kompression samtidigt som det kräver mindre processortid.

```
bzip2 fil.tar
```

Detta komprimerar ”fil.tar” med **bzip2**-kompressionsprogrammet. **bzip2** kan komprimera valfri annan filtyp lika väl som en tar-boll.

```
bunzip2 fil.tar.bz2
```

Detta avkomprimerar ”fil.tar.bz2” till ”fil.tar” och lämnar filen i den aktuella mappen.

**TIPS** Liksom för **gzip** följer det med olika verktyg med **bzip2** som du kan använda för att arbeta med komprimerade filer. För **bzip2** har de tillgängliga verktygen prefixet ’bz’: **bzcat**, **bzless**, **bzgrep**.

## 17.3 zip

**$ zip** Komprimerar filer till zip-arkiv

**DEBIAN, UBUNTU:** **zip** **FEDORA:** **zip**

**HJÄLP:** **man zip**

I Windowsvärlden är zip-arkiv mycket vanliga. Zip-arkiv är komprimerade för att spara plats och förenkla hanteringen. Med kommandot **zip** kan du enkelt skapa sådana arkiv i Linux. Kommandot **unzip** gör det möjligt att packa upp zip-arkiv.

Exempel:

```
zip arkiv.zip *
```

Detta skapar ”arkiv.zip” med alla filer i den aktuella mappen.

```
zip -r arkiv.zip dokument
```

Detta skapar ”arkiv.zip” med allt innehåll, inklusive alla filer och mappar, i mappen ”dokument”.

Kommandot har en mängd alternativ och en utförlig manual-sida.

## 17.4 unzip

---

**$ unzip** Packar upp zip-arkiv

---

**DEBIAN, UBUNTU:** **unzip** **FEDORA:** **unzip**

---

**HJÄLP:** **man unzip**

---

Kommandot **unzip** packar upp arkiv som har skapats med **zip** (se ▷ sektion 17.3).

Exempel:

```
unzip arkiv.zip
```

Detta packar helt enkelt upp innehållet i ”arkiv.zip”, och återskapar mapp- och filstrukturen för innehållet i den aktuella mappen.

Kommandot har en mängd alternativ och en utförlig manual-sida.

# Kapitel 18

# Schemalägga kommandon för körning

Det här kapitlet visar hur du kan ställa in datorn att utföra olika uppgifter automatiskt. På så sätt kan du utnyttja datorns resurser bättre och du slipper starta kommandon manuellt.

Genom att schemalägga kommandon kan du låta datorn jobba när du inte är närvarande – vilket är praktiskt för exempelvis automatisk säkerhetskopiering, indexering av din hårddisk, eller något annat som tar mycket datorresurser (processorkraft, bandbredd) i anspråk. Du kan både schemalägga enstaka automatiska kommandon med **`at`** och regelbundet återkommande kommandon med **`cron`**. Det finns också en variant av det senare kommandot som lämpar sig för bärbara och andra datorer som inte alltid kan förväntas vara igång, **`anacron`**.

## 18.1 Engångshändelser

### 18.1.1 at

**$ at** Schemalägger jobb för körning vid ett visst tillfälle

**DEBIAN, UBUNTU:** **at** **FEDORA:** **at**

**HJÄLP:** **man at**

**at** kör ett kommando vid ett tillfälle (en viss dag, vid ett visst klockslag).

Du kör kommandot helt enkelt genom att skriva kommandonamnet följt av den tidpunkt du önskar köra kommandot på. Därefter skriver du in det kommando du vill köra.

Exempel

```
at 21:30
```

Skriv därefter in det eller de kommandon du vill köra klockan 21:30. Tryck på CTRL+D när du är klar. Tangentbordsgenvägen CTRL+D har betydelsen ”slut-på-filen” (*end-of-file (EOF)* på engelska) som används för att ange att du är har skrivt klart kommandon (du har nått slutet, därav begreppet ”EOF-kommandot”).

Du kan också pröva följande:

```
at now + 1 hour
```

Ovanstående kör det eller de kommandon du anger ett givet antal timmar/minuter/sekunder i framtiden, räknat från det aktuella klockslaget. Tryck på CTRL+D när du är klar.

Du kan också använda **at** icke-interaktivt, genom att direkt i anslutning till **at**-kommandot mata in kommandot du önskar köra. Använd alternativet **-f** för att specificera kommandot du vill köra.

Exempel:

```
at -f skript.sh now + 1 hour
```

Detta kör ”skript.sh” en timme räknat från den aktuella tiden.

### 18.1.2 atq

**$ atq** Visa användarens egna köade jobb

**DEBIAN, UBUNTU:** **at** **FEDORA:** **at**

**HJÄLP:** **man atq**

Visar jobb i at-kön för den aktuella användaren. Om du kör kommandot som root ser du alla schemalagda kommandon i at-kön. För varje jobb visas ett unikt jobbnummer.

Kommandosyntax:

```
atq
```

Kommandot accepterar inga alternativ.

### 18.1.3 atrm

**$ atrm** Tar bort schemalagda jobb

**DEBIAN, UBUNTU:** **at** **FEDORA:** **at**

**HJÄLP:** **man atrm**

Detta tar bort jobb från at-kön.

Kommandosyntax:

```
atrm jobbnummer
```

Du kan se jobbnumret för det önskade kommandot genom att först köra **atq**.

## 18.2 Återkommande händelser

Med cron-funktionen (cron är kortform av klocka, *chronograph* på engelska) kan du schemalägga ett kommando så att det körs regelbundet – exempelvis varje minut, varje timme, en gång per dag, en gång i veckan eller en gång per månad. Cron-demonen körs i bakgrunden och startar kommandon på angivna klockslag.

### 18.2.1 crontab

**$ crontab** Hantera schemaläggning av återkommande jobb

**DEBIAN, UBUNTU:** **cron** **FEDORA:** **vixie-cron**

**HJÄLP:** **man crontab**

Kommandot **crontab** rekommenderas för att redigera cron-schemaläggningen som sedan läses in av cron-demonen. Med **crontab** kan du visa, lägga till och ta bort kommandon för körning vid regelbundet återkommande tidpunkter.

Kör **crontab** med något av följande alternativ:

**-e** Redigera cron-filen.

**-l** Visa innehållet i cron-filen.

**-u användarnamn** Ange en användare för att visa eller redigera den användarens cron-inställningar.

Om du kör **crontab -e** startas standardredigeraren för text med innehållet för den aktuella användarens crontab-fil.

**OBS** Cron-filerna sparas i `/var/spool/cron/crontabs`, en för varje användare, men dessa filer bör aldrig redigeras manuellt. Använd *alltid* **crontab -e** för att redigera filen.

**Minut** Ange minut, tillåtna värden: 0–59 eller * (asterisk).

**Timme** Ange timme, tillåtna värden: 0–23 eller * (asterisk).

**Dag i månaden** Ange dag i månaden, tillåtna värden: 1–31 eller * (asterisk).

**Månad** Ange månad under året, tillåtna värden: 1–12, eller månadens namn på engelska, eller * (asterisk).

**Dag i veckan** Dag i veckan, tillåtna värden: 0–7 (0 och 7 är söndag), eller förkortningar av namn på engelska (tre bokstäver), eller * (asterisk).

**Kommando** Kommandot som ska köras av skalet.

Det finns också en uppsättning kortkommandon med olika betydelser, se manual-sidan för **crontab** för mer information om dessa.

Tecknet * (asterisk) betyder ”när som helst” och fungerar här som ett jokertecken. Om du till exempel på dag-platsen skriver in en asterisk betyder det varje dag.

Det går bra att ange listor och spann för varje plats, exempelvis ”1,2,5,9” eller ”0-4,8-12”.

Du kan också ange stegvisa uppräkningar i anslutning till tidsspann, exempelvis ”0-19/2”, vilket betyder varannan timme från midnatt till 19:00. ”*/2” betyder varannan timme dygnet runt.

För att skriva en komplett rad med **crontab** matar du in information om var och en av de 6 positionerna i tabellen. Den sista positionen ska innehålla kommandot du vill utföra.

Exempel:

```
5 4 * * sun /home/bertil/.backupskript.sh
```

Ovanstående kör skriptet ”.backupskript.sh” i klockan 04:05 varje söndag.

### 18.2.2 anacron

**$ anacron** Hantera flexibel schemaläggning av återkommande jobb

**DEBIAN, UBUNTU:** **anacron** **FEDORA:** **anacron**

**HJÄLP:** **man anacron**

Det finns en speciell variant av cron som är lämplig för datorer som inte alltid är påslagna och igång, exempelvis bärbara datorer. Kommandon som borde ha körts när datorn var avslagen körs så fort det är möjligt med anacron.

Anacron fungerar ungefär som cron, men med mindre precision. Du kan ange körning av kommandon dagligen, veckovis eller månadsvis, men inte på timme- eller minutnivå.

Till skillnad från cron används inte något specialkommando à la **crontab** för att redigera anacrontab-inställningarna. Istället måste du redigera filen `/etc/anacrontab` för hand.

I `/etc/anacrontab` kan du mata in fyra värden per kommando:

**Period** Antal dagar som ska gå mellan två körningar av kommandot.

**Väntetid** Antal extra minuter som anacron väntar innan jobbet körs. Detta anges för att förhindra att en stor mängd kommandon körs igång på en och samma gång vid systemstart.

**Jobb-idenfierare** Godtyckligt namn/id, kan innehålla valfria tecken (dock får fältet inte lämnas tomt).

**Kommando** Kommandot som ska köras av skalet.

Exempel:

```
1 20 extrakopia /home/bertil/.backupskript.sh
```

Ovanstående kör skriptet ”.backupskript.sh” varje dag 20 minuter efter att **anacron** startade.

# Kapitel 19

# Skicka in- och utdata mellan program (avancerat)

Detta kapitel förklarar hur du kan skicka resultatet av ett kommando som argument till ett annat. På så sätt kan du ”klistra ihop” olika kommandon till nya som utför de uppgifter du önskar.

## 19.1 Koncept och definitioner

Alla tre följande definitioner kallas filströmmar (*File Streams* på engelska) och förmedlar information som antingen tas emot från en plats eller skickas någonstans. I ett UNIX-system är de tre filströmmarna tangentbordet (*standard input*), text som skrivs på skärmen (*standard output*) och felmeddelanden som skrivs på skärmen (*standard error*).

**Standard input** Standard input är indata från användare, normalt sett från tangentbordet som är standardinmatningsenhet på UNIX-system.

**Standard output** Standard output är utdata från program – resultatet av ett kommando – och som skrivs ut på skärmen, exklusive felmeddelanden (se standard error nedan).

**Standard error** Standard error är felmeddelanden från program. Denna utdata skickas också till skärmen och syns oftast blandad med standard output. Skillnaden mellan standard output och standard error är att standard error inte samlas ihop (buffras) och visas i ett sjok, utan direkt när ett fel uppträder. Standard error visas också bara när något går fel (då visar det uppgifter om vad som gick galet).

## 19.2 Användning

**> (större än)** Större-än-tecknet används för att skicka information någonstans (exempel till en textfil).

Exempel (med kommandot **`cat`**, som sätter ihop – konkatenerar – filer):

```
cat fil1 fil2 > fil1_och_2.txt
```

Kommandot **`cat`** sätter ihop de två filerna till en stor sammanslagen fil med hjälp av större-än-tecknet. Observera att kommandot skriver över en eventuellt redan befintlig fil.

**< (mindre än)** Mindre-än-tecknet infogar information från någonstans (exempelvis en textfil), som om du skrev den själv direkt på kommandoraden. Ofta används den med kommandon som är utformade att endast ta emot information från standard input.

Ett exempel med kommandot **`tr`**:

```
tr A-Z a-z < filnamn.txt > nytt_filnamn.txt
```

Exemplet ovan infogar innehållet i ”filnamn.txt” som argument till **`tr`** (▷ sektion 12.4.11) och skickar utdatat till ”nytt_filnamn.txt”.

**>> (dubbla större än)** Lägger till information till slutet av en fil. Om filen inte finns skapas den.

**<< (dubbla mindre än)** Används ibland med kommandon som tar emot information från standard input (tangentbordet). Detta används allra mest för skalskript där tecknen används för att indikera start på en inmatning. Skriv helt enkelt "<< kodord" (där kodord är vilket godtyckligt ord som helst) på slutet av raden för att indikera start på inmatning. Kommandot läser in din inmatning fram tills dess att du skriver "kodord" igen. Då läser kommandot in hela inmatningen.

Pröva genom att skriva **`cat << KLAR`** i ett terminalfönster. Detta startar programmet **`cat`**. Nu kan du börja skriva in valfri text – skriv några rader djupsinnigtheter eller nonsenstext. När du är klar med inmatningen skriver du **`KLAR`** på en egen rad. Som du ser avslutar detta programmet, utan att du behöver trycka på CTRL+D.

**2>** Skickar vidare felmeddelanden. Om du till exempel vill slippa se alla felmeddelanden för ett kommando kan du skicka dem till **`/dev/null`**. (**`/dev/null`** är datorns motsvarighet till ett svart hål).

Exempel:

```
make en_fil 2> /dev/null
```

Detta kör kommandot **`make`** (kommando för att kompiliera källkod till en körbar fil) på filen "en_fil" och skickar alla felmeddelanden till **`/dev/null`**.

**| (rörtecken)** Rör-tecknet gör det möjligt att skicka utdata från ett kommandot som argument till ett annat.

Exempel:

```
cat fil1.txt fil2.txt | less
```

Detta sätter ihop två filer med **`cat`** och skickar resultatet till kommandot **`less`**.

Du kan också få samma resultat genom att använda dig av kommandosubstitution. Se följande avsnitt för två exempel.

**`tee`** Skickar utdata från ett program både till en fil och till standard output. Tänk på **`tee`** som en T-korsning.

Exempel:

```
ls /home/mitt_användarnamn | tee mina_mappar.txt
```

Detta visar alla filer i hemmappen på skärmen samtidigt som utdatat skickas till filen ”mina_mappar.txt”.

**&>** Skickar standard output och standard error till en specifik riktning.

Exempel:

```
make &> /dev/null
```

Detta skickar både standard error och standard output till **/dev/null**, så att du inte ser någon utdata från kommandot **make** över huvud taget.

## 19.3 Kommandosubstitution

Kommandosubstitution kan skrivas på två skilda sätt.

### 19.3.1 Metod 1: grava accenter

Använd grava accenter (*back-quote* på engelska). Skriv helt enkelt:

```
kommando1 `kommando2 -alternativ`
```

Detta kör först kommando2 och dess resultat matas in som argument till kommando1.

Exempel:

```
less `cat fil1.txt fil2.txt`
```

### 19.3.2 Metod 2: dollartecken

Skriv helt enkelt:

```
kommando1 $(kommando2)
```

Precis som föregående exempel kör detta först kommando2 och dess resultat matas in som argument till kommando1.

Exempel:

```
less $(cat fil1.txt fil2.txt)
```

Båda dessa varianter av kommandosubstitution ger samma resultat som exemplet med rör-tecken ovan.

## 19.4 Utföra fler än ett kommando åt gången

Du kan utföra kommandon i sekvens, där det andra kommandot endast utförs om körningen av det första lyckades.

Skriv så här för att göra det:

```
kommando1 && kommando2
```

Kommando2 utförs om kommando1 avslutades korrekt (om kommando1 misslyckas körs inte kommando2). Detta kallas för ett logiskt OCH-villkor (*AND* på engelska).

Du kan också utföra det andra kommandot endast om det första misslyckas.

Skriv följande för att göra detta:

```
kommando1 || kommando2
```

Kommando2 utförs om körningen av kommando1 inte avslutas korrekt (och om kommando1 lyckas körs inte kommando2). Detta kallas för ett logiskt ELLER-villkor (*OR* på engelska).

Det är också möjligt att utföra kommandon i sekvens oavsett hur körningen av de enskilda kommandona går. Gör så här:

```
kommando1; kommando2
```

Kommando2 körs så snart kommando1 har avslutats (oavsett hur det gick) om de separeras med ett semikolon.

**TIPS** Du kan fortsätta använda ;, || eller && för att lägga till valfritt antal kommandon på kommandoraden.

## 19.5 xargs

**$ xargs** Ställer samman och kör (avancerade) kommandon

**DEBIAN, UBUNTU:** **findutils** **FEDORA:** **findutils**

**HJÄLP:** **man xargs**

Kommandot **xargs** är mycket kraftfullt och kan användas för att bygga kommandon som garanterat inte ställer till prestandaproblem. Om du till exempel vill köra kommandot **ls** i en mapp som innehåller en extrem stor mängd filer kan antalet filer bli större än vad datorn klarar av (till exempel på grund av otillräckligt internminne). Med **xargs** behöver du inte oroa dig; kommandot ser till att köra det önskade kommandot på en hanterbar mängd filer åt gången och fortsätter automatiskt tills allt är klart. Du kan alltså tänka på **xargs** som ett extremsäkert alternativ till kommandosubstitution för att bygga avancerade kommandon. Med kommandot kan du i vissa fall utvinna mer prestanda ur datorn än annars – till exempel kan du styra att ett kommando delas upp så att det körs parallellt på flera processorer.

```
find /tmp -name test -type f | xargs /bin/rm -f
```

Kommandot **find** används i exemplet ovan för att visa alla filer i mappen /tmp som heter test i en lista, varpå resultatet skickas till xargs, som kör kommandot **rm -f** på listan för att radera dem.

En nackdel med **find**-kommandot i exemplet ovan är att det inte fungerar med filnamn som innehåller mellanslag. (Till exempel tolkas filen ”min fil” av **find** som två filer, ”min” och ”fil”.) Alternativet **-print0** gör att **find** separerar filer i resultatet med ett specialtecken som inte kan förekomma i filnamn istället för mellanslag. För att **xargs** ska veta att specialtecknet separerar filnamn behöver du lägga till alternativet **-0** till kommandot.

Exempel:

```
find /tmp -name test -type f -print0 | xargs -0 /bin/rm -f
```

**xargs** accepterar olika alternativ, bland dem:

**-nx** Säger åt **xargs** att sätta samman max x kommandon per kommandorad.

**-lx** Säger åt **xargs** att utföra kommandot på x rader indata åt gången.

**-p** Säger åt **xargs** att fråga innan strängen körs.

**-t** (t för *tell* på engelska), visa varje kommando med **echo** innan det körs.

# Kapitel 20

# Jokertecken och reguljära uttryck (avancerat)

Jokertecken och reguljära uttryck kan användas bland annat för avancerade sök- och ersätt-operationer i programmeringsspråk, men också med vanliga kommandon i skalet. Läs det här kapitlet för en introduktion.

## 20.1 Skalets jokertecken

Jokertecken är användbara för en uppsjö saker på ett GNU/Linux-system. Kommandon kan använda jokertecken (till exempel * och ?) för att utföra åtgärder på flera än en fil åt gången, eller för att hitta delar av en fras i en textfil. Det finns två huvudsakliga användningsområden för jokertecken – det första är mönstermatchning som ofta används i skalet för att hantera flera filer samtidigt, det andra är för att konstruera reguljära uttryck som ofta kommer till användning för att söka och manipulera text.

Ibland behöver man använda tecken som * som vanlig text, utan att de tolkas som jokertecken. Då kan man skydda tecknet från att tolkas (expanderas, med ett annat ord) genom att sätta ett bakstreck framför det, så här: *.

Nedanstående sektioner är delvis baserade på manual- och info-sidorna för kommandot **grep**. Det finns också mer information att hämta om du kollar **man glob**. Läs dem om du vill fördjupa dig i ämnet.

Jokertecken (kallas ibland *globbing patterns* på engelska, ungefär matchningsmönster) används med olika kommandoradsverktyg för att hantera flera filer på en och samma gång. Jokertecken fungerar med de allra flesta vanliga kommandona, exempelvis **mv**, **cp**, **rm** och många andra.

De vanligaste jokertecknen är följande:

**? (frågetecken)** Jokertecknet ? representerar ett enda tecken, vilket som helst. Se ▷sektion 3.4 för exempel.

*** (asterisk)** Detta är det vanligast förkommande jokertecknet, med betydelsen matcha allt. Se ▷sektion 3.4 för exempel.

**[ ] (hakparenteser)** Hakparenteser används för att ange spann. Om du skriver ”m[aou]m” kan det stå för såväl mam, mum, mom. Om du skriver m[a-d]m kan det står för allt som börjar och slutar med m och som har något tecken mellan a och d emellan: mam, mdm, mcm, mdm. Detta anger ett ”eller”-förhållande mellan alternativen (endast ett av alternativen måste stämma för att det ska betraktas som en träff).

Exempel:

```
ls dokument-[A-Z].doc
```

Detta visar alla filer i den aktuella mappen vars namn börjar med ”dokument-” följt av en versal och därefter ”.doc”.

**{ } (måsvingar)** Måsvinge-parenteser anger ett eller flera mönster, där flera mönster separeras med kommatecken.

Nedanstående är ett giltigt exempel:

```
cp *.{doc,pdf} ~
```

Detta kopierar allt som slutar med .doc eller .pdf till användarens hemmapp. Observera att mellanslag inte är tillåtna efter kommatecknen eller någon annan stans mellan måsvingarna.

**[! ] (hakparenteser och utropstecken)** Genom att inkludera ett utropstecken först i hakparentesen vänder du på steken så att sökningen *inte* matchar innehållet i parentesen. Detta motsvarar ett logiskt inte-villkor.

Exempel:

```
rm minfil[!9]
```

Detta tar bort minfil1, minfil2 et cetera men inte minfil9 och andra varianter som innehåller 9 i filnamnet.

**\ (bakstreck)** Används som *escape*-tecken, det vill säga för att skydda ett följande specialtecken (ofta ett jokertecken) så att det inte tolkas. Till exempel kan du använda '\\' för att matcha bakstreck som faktiskt förekommer i en text, eller '\(\)' för att matcha paranteser.

## 20.2 Reguljära uttryck

Reguljära uttryck är en typ av mönstermatchning för text som används för att söka efter och hantera text. Du kan använda reguljära uttryck med en stor mängd kommandon, bland annat **`grep`**. Dessa vanliga reguljära uttryck är mer flexibla och kraftfulla än de enkla jokertecken och paranteser som används i skalet Bash och som beskrivs i föregående avsnitt.

**. (punkt)** Matchar ett godtyckligt tecken. Motsvarar ? bland standard-jokertecknen. "m.a" matchar sålunda mpa och mea men inte ma eller mppa.

**.* (punkt och asterisk)** Används för att matcha en godtycklig sträng. Motsvarar * bland skalets jokertecken.

*** (asterisk)** Används här för att matcha noll eller flera förekomster av det föregående tecknet. "n*" matchar n, nn, nnn, nnnnn och så vidare men inte na eller någon n tillsammans med något annat tecken.

**^ (taktecken)** Används för att indikera början på en rad. ”^a” betyder alltså sök efter en rad som börjar med ett a.

**$ (dollartecken)** Används för att indikera slutet på en rad. ”a$” betyder alltså sök efter en rad som slutar med ett a.

Exempel:

```
cat fil.txt | grep '^s.*n$'
```

Detta söker i filen fil.txt efter rader som börjar med s och slutar med n, med godtyckligt antal tecken däremellan, och visar träffarna på skärmen.

**[ ] (hakparenteser)** Hakparenteser används för att ange spann. Om du skriver ”m[aou]m” kan det stå för såväl mam, mum, mom. Om du skriver m[a-d]m kan det står för allt som börjar och slutar med m och som har något tecken mellan a och d emellan: mam, mdm, mcm, mdm. Denna typ av tecken anger ett ”eller”-förhållande mellan alternativen (endast ett av alternativen måste stämma för att det ska betraktas som en träff).

**| (rörtecken)** Detta tecken har den logiska innebörden ELLER. Med detta tecken kan du söka efter en förekomst av något ELLER något annat (kanske i sin tur två olika reguljära uttryck). För att skalet inte ska uppfatta tecknet som ett rör (se kapitel 19) bör du använda ett bakstreck före tecknet eller ”kapsla in” hela uttrycket med citattecken.

**[^ ] (hakparenteser och taktecken)** Om du börjar strängen med ^ betyder det att sökningen matchar vad som helst som inte finns innanför hakparenteserna. Detta motsvarar ett logiskt inte-villkor.

Exempel:

```
grep [^9] /etc/passwd
```

**\ (bakstreck)** Används som ”escape”-tecken, det vill säga för att skydda ett följande specialtecken (ofta ett

jokertecken) så att det inte tolkas. Till exempel söker '\\' efter bakstreck i texten.

# Kapitel 21

# GNU Free Documentation License



## 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

formaters or for automatic translation to a variety of formats suitable for input to text formaters. 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 in 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 Warranty 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 © 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.

# Kapitel 22

# Revisionshistorik

## 22.1 Den svenska översättningen

**Revision sv-1.1** 12 februari 2010
Andra utgåvan, första tryckningen. Korrektur samt utökning med kapitel om verktyg för PDF- och grafikmanipulering. Faktagranskning av Andreas Önnebring.

**Revision sv-1.0.2** 6 februari 2009
Tredje tryckningen. Rättat korrekturfel, förbättrat sakregistret.

**Revision sv-1.0.1** 4 juni 2008
Andra tryckningen. Rättat korrekturfel, förbättrat sakregistret.

**Revision sv-1.0** 6 april 2008
Första utgåvan av *Effektivare Linux – kom igång med kommandoraden*, utgiven av HME Publishing. Översättning, bearbetning och utökning av revision 1.2 av *GNU/Linux Command-Line Tools Summary* (se nedan). Faktagranskning av Mikael Tillenius och Martin Adolfsson.

## 22.2 Den engelska förlagan

Förlagan till den här boken heter *GNU/Linux Command-Line Tools Summary* och är skriven av Gareth Anderson, med bidrag av flera personer, bland andra Chris Karakas. En kom-

plett lista med bidragsgivarna till originalet finns på http://tldp.org/LDP/GNU-Linux-Tools-Summary/html/contributors.html.

# Index