Kapitel 3

Använda GNU/Linux

Jag beskriver här grunderna för hur du använder GNU/Linux i terminalen. Jag tar inte upp de grafiska verktygen eftersom: de varierar mellan distributioner, de förändras oftare mellan releaser, de är svårare att fjärranvända och de är långsammare och mer begränsade. Nästan allting i ett GNU/Linux-system är filer, och nästan all konfiguration sker via textfiler. En editor är det vanligaste verktyget du använder vid administration, jag rekommenderar Vim eller Emacs. De är beskrivna i kapitel 4.

Jag börjar istället med att berätta om katalogstrukturen i ett vanligt system.

3.1 Kataloger

I UNIX är nästan allt en fil. Det finns bara ett katalogträd, inga enheter som C: och D:, utan nya enheter monteras i trädet. Vad katalogerna i roten används till känner du kanske redan till, men repetition kan vara bra.

/bin
Här finns vanligt använda och viktiga kommandon. Du bör kunna nästan alla om du kör `ls` här.

/boot
Här finns kärnan och bootladdaren Grub.

/dev
Devices (enheter), som hårddiskar och terminaler. Numera skapas de ofta dynamiskt.

/etc
Konfigurationsfiler för massor av saker. De flesta som ren text.

/home
Användarnas hemkataloger, alla utom root.

/lib och **/lib64**
Biblioteksfiler (32- och 64-bitars). Till exempel libc ligger här.

/lost+found
Den ska vara tom, utom om du har en trasig disk.

/media och **/mnt**
Monteringspunkter för tillfälliga saker.

/opt
Katalog för saker som inte följer med systemet.

/proc
Virtuella filer som ger information från kärnan.

/root
Hemkatalog för användaren root.

/sbin
Viktiga program som vanliga användare sällan behöver, och inte alltid kan, köra.

/sys
Virtuella filer för att ge information till kärnan.

/tmp
Tillfällig lagring.

/usr
Startpunkt för ett nytt delträd med /usr/bin, /usr/sbin och så vidare. Många program finns här. Filer under /usr ska normalt sett bara ändras när man uppgraderar.

/usr/local
Här hamnar ofta program man själv installerar. Den har underkataloger som bin, lib, sbin osv.

/var
Filer som skapas, växer och försvinner. Till exempel loggfiler, inkommande och utgående e-post och databaser för DNS och RPM.

3.2 cd

Du förflyttar dig i katalogträdet med kommandot `cd`. Med `cd ..` kommer du ett steg närmare roten. Med `cd /` kommer du direkt till roten. Du kan använda både relativa och

absoluta sökvägar: `cd /home/daniel` är en absolut sökväg och `cd ../daniel` är en relativ (som funkar om du till exempel står i /home/ao). Men för att komma till en användares hemkatalog är det enklast att använda tecknet tilde: `cd ~daniel`. För att komma till din egen hemkatalog skriver du enbart `cd`, det funkar oavsett var du står.

Ett annat tips med cd är `cd -` med det kan du hoppa fram och tillbaka mellan de senaste två katalogerna. Exempel på användning:

```
[ao@a ffs]$ cd /etc/httpd/conf
[ao@a conf]$ sudo vi httpd.conf
[sudo] password for ao:
[ao@a conf]$ sudo /etc/init.d/httpd reload
[ao@a conf]$ cd /var/log/httpd/
[ao@a httpd]$ tail -f access_log
[ao@a httpd]$ cd -
/etc/httpd/conf
[ao@a conf]$ sudo vi httpd.conf
[ao@a conf]$ sudo /etc/init.d/httpd reload
[ao@a conf]$ cd -
/var/log/httpd
[ao@a httpd]$ tail -f access_log
[ao@a httpd]$ cd -
/etc/httpd/conf
[ao@a conf]$
```

Du kan alltså växla mellan att redigera httpd.conf och läsa loggfilen, utan att behöva ange hela sökvägen. Eller växla mellan vilka två kataloger som helst. Var du hamnar skrivs ut. Om du däremot går till en ny katalog blir det den som du kommer tillbaka till med `cd -` (man kan göra liknande med pushd och popd om du är van att använda en stack).

3.3 Filrättigheter

Varje fil har en ägare och grupp:

```
[ao@a GLAN]$ ls -l /etc/passwd
-rw-r--r--. 1 root root 1863 26 dec 00.35 /etc/passwd
[ao@a GLAN]$ ls -l ~/.bashrc
-rw-r--r--. 1 ao ao 154 23 nov 18.36 /home/ao/.bashrc
[ao@a GLAN]$ ls -l /var/spool/mail/ao
```

```
-rw-rw----. 1 ao mail 0 31 jul 04.19 /var/spool/mail/ao
[ao@a GLAN]$
```

Filen passwd ägs av root och har gruppen root. Min .bashrc ägs av ao:ao. Båda dessa är skrivbara av ägaren och läsbara för grupp och andra. Däremot ägs min mailbox av ao, men den är även skrivbar av gruppen mail och är inte läsbar för andra. Mailservern (eller en viss del av den för Postfix) kör som gruppen mail, och kan därför fylla på min inlåda med nya brev. Och andra ska inte kunna läsa mina mejl.

3.3.1 ugo, rwx

Filers rättigheter, även kallade behörigheter, listas i ordningen ugo: user, group, others. User är ägaren (kallas även owner men u är vanligare förkortning). Alla användare listas i `/etc/passwd`. Grupper definieras i `/etc/group`, och på många system finns personliga grupper med samma namn som ägaren och ingen annan som medlem. Men man kan även ha egna grupper såsom students, admins eller web med en eller flera medlemmar. Others är just övriga, inte alla utan de som varken är user eller group.

Vad man får göra med en fil eller katalog anges i ordningen rwx – Read, Write och eXecute. För en fil betyder r att man får läsa filens innehåll, w att man får skriva till filen och x att man får köra/exekvera filen. Dessa tre rättigheter upprepas tre gånger: för user, group och others. Det första strecket i fillistningen anger att det är en vanlig fil, för kataloger står det ett ’d’ där.

För att byta rättigheter används kommandot `chmod`. Som argument anger man antingen tre (eller fyra) oktala tal eller symboliska rättigheter. Exempel på den oktala varianten är

```
chmod 755 fil
chmod 600 fil
chmod 705 fil
chmod 644 fil.
```

Jag illustrerar siffrorna i tabell 3.1 och förklarar efter den.

För varje kolumn summerar du de rättigheter du vill ha. 755 betyder alltså -rwxr-xr-x och 644 -rw-r--r-- i samma ordningsföljd (ugo) som vanligt. Just 755 och 644 är bland de vanligaste, 755 för körbara filer och 644 för de som enbart

Tabell 3.1: chmod

	u	g	o
r	4	4	4
w	2	2	2
x	1	1	1

ska läsas. Ger du en fil 600 kan bara ägaren läsa och skriva till den, 660 så kan ägaren och gruppen både läsa och skriva men alla andra inget alls. Med 705 så får ägaren göra allt, gruppen inget och alla andra läsa och exekvera. Att ge 0 till gruppen kan vara användbart till exempel i skolor där studenterna tillhör en grupp och lärarna en annan, då kan studenterna inte läsa varandras programmeringsuppgifter om inte ägaren byter rättighet.

Ett skript i bash, Perl, Python eller liknande måste ha både r och x satt för att kunna köras. Däremot räcker det för ett kompilerat program med x för att köra, men oftast har man även r.

3.3.2 chown och chgrp

Det är enbart ägaren till en fil, och root, som kan byta rättigheter med `chmod`. Aldrig grupp eller andra.

Användaren root kan även byta ägare till en fil, med `chown user fil` (i vissa äldre UNIX-varianter kunde vanliga användare skänka bort filer, men det är inte lämpligt). Alla som äger en fil kan byta grupp med `chgrp grupp fil` om de är medlemmar i gruppen de byter till. Vill root både byta ägare och grupp på en gång kan han/hon använda `chown user:grupp fil`, med kolon mellan ägare och grupp. Till exempel
`chown ao:students hello.c`.

3.3.3 Katalogers rättigheter

För kataloger betyder r att man får lista filnamnen, och x att man får gå in i katalogen med cd. Denna distinktion används sällan, i praktiken är det 755 och 700 som är de vanligaste rättigheterna för en katalog. Det händer dock att

man glömmer att ge kataloger x-biten. I vissa fall kan 711 vara bra, då kan andra komma åt filer och underkataloger om, och endast om, de vet namnen.

3.3.4 Symboliska rättigheter

Att ange rättigheter med oktala tal kallas även att ge absoluta rättigheter. Det andra sättet kan kallas symboliska eller relativa rättigheter. Då anger man först för vem/vilka man vill ändra med u, g, o eller a (där a betyder alla). Sen skriver man +, - eller = följt av r, w eller x. Till exempel `chmod o-r fil` (gör den icke läsbar för andra),
`chmod u+x fil` (gör den exekverbar för ägaren),
`chmod a=x fil` (x för alla) eller `chmod g-w fil` (ta bort skriv för grupp). Ändringar med minus eller plus görs oavsett tidigare rättigheter, och de ändrar inte för andra än de som anges. Därför funkar till exempel `chmod o-xw *` för att göra både filer och kataloger oläsbara för others utan att förändra execute och annat på kataloger och filer. Med oktala rättigheter ändrar man alltid alla rättigheter. För mer detaljer se manualen med `man chmod`.

3.3.5 suid, sgid och sticky

Jag nämnde att det ibland anges fyra rättigheter. `chmod 4755` kallas suid eller setuid. Det betyder set user id och är ganska farligt. Med den biten satt kommer programmet exekveras som filens ägare, oavsett vem som kör den. Till exempel `/usr/bin/passwd` ägs av root och har suid, så den kommer exekveras som root även om en vanlig användare kör det (vilket möjliggör att den kan ändra i `/etc/shadow`). Program med suid är ofta en säkerhetsrisk, och du bör inte använda det själv om du inte är helt säker på vad du gör. Motsvarigheten för grupper är sgid, set group id, där programmen körs som den grupp som filen tillhör. Det anger du med `chmod 2755`. Inte heller det bör du använda i onödan, även om det är lite säkrare, i de fall där det är en ”ofarlig” grupp. I `ls -l` visas suid som -rwsr-xr-x och sgid med ett s i grupp-positionen -rwxr-sr-x.

Sticky bit , som sätts med en etta i första positionen, `chmod 1777`, har numera enbart betydelse för kataloger. Där betyder det att enbart ägaren till en fil får radera den. Normalt sett får den som har skrivrättigheter till en katalog

även ta bort filer som ligger där, det förhindras med sticky bit och används för kataloger som /tmp och /var/tmp där alla får skriva. För länge sedan betydde sticky bit på ett program att det skulle ligga kvar i RAM, men det är bara på riktigt gammal UNIX, i Linux har det aldrig varit så.

Även sgid har annan betydelse för kataloger än för filer. `chmod 2775 katalog` betyder att alla nyskapade filer och underkataloger kommer få samma grupp som föräldern. Det är mycket praktiskt för kataloger som ska användas av flera personer.

3.3.6 umask

Vilka rättigheter som nyskapade filer och kataloger får bestäms av användarens umask. Den kan ses som komplementet till rättigheterna, eller som 777 minus umask. Vanliga värden på `umask` är 022 som kommer ge 755 för kataloger och 644 för filer, eller 002 som även ger skrivrättigheter till gruppen (775 och 664). Vill man vara mera hemlighetsfull kan man ha 007 så blir det 770 och 660, det vill säga inga rättigheter alls för alla andra. Du sätter umasken så här:
`umask 022`

Det gäller enbart för det fönster det sätts i, så vill du ha det mer permanent får du lägga till raden med `umask` i filen `~/.bashrc`

3.4 Hitta filer

Det finns flera olika sätt att hitta filer om man vet namnet men inte platsen. Snabbast är `locate`, t.ex. `locate passwd`. Nackdelen med den är att databasen bara uppdateras en gång per dygn, så du kan inte hitta nya filer. Vill du söka efter både stora och små bokstäver lägger du till `-i` (ignorecase).

Ofta händer det att man glömmer vad filen man laddade ner eller skapade senast hette, men vet vilken katalog den bör ligga i. Då kan man köra `ls -ltr` (long, time, reverse), så får man de nyaste filerna längst ner i listan. Hoppa över `r` om du vill ha dem först i listan.

För att hitta var ett program ligger kan du använda `whereis`, t.ex. `whereis ifconfig`. Detta kommando söker

i en förutbestämd lista, så det kommer säga /sbin/ifconfig även om du inte har /sbin i din normala sökväg ($PATH). Med `whereis` listas även man-sidor.

Om du däremot enbart vill se filer i din $PATH kan du köra `which`, t.ex. `which python`. Det visar enbart en fil, första träffen, så du ser om det är /usr/local/bin/python eller /usr/bin/python som skulle köras. Ett relaterat kommando är `type`, som även visar huruvida programmet är inbyggt i skalet. Jämför `which echo` och `type echo` där den senare ger rätt svar.

Det program som är mest flexibelt för att hitta filer är `find`. Det har ganska avancerad syntax, men grunden är find katalog -vad -gör. Till exempel `find . -name passwd -print` för att börja söka i aktuell katalog och underkataloger efter alla filer som heter passwd. Vill du söka efter en del av ett namn kan du använda jokertecken, till exempel så här `find . -name '*pass*' -print`. Glöm inte att ha enkla citattecken, utan dem kommer skalet att expandera eventuellt filnamn i katalogen du står. På GNU-find är `-print` standard och kan hoppas över, men inte så på t.ex. äldre Solaris. Vill du ha en detaljerad fillista byter du ”action” till `-ls`, alltså `find . -name passwd -ls`. Det går även att radera filer med `-delete` och utföra ett kommando för var och en med `-exec`. För exec betyder {} aktuellt filnamn och kommandolistan måste avslutas med ett semikolon. Båda dessa tecken bör skyddas från att expanderas av skalet. Exempelvis: `find /etc -name passwd -exec cat '{}' \;`

Lättare och bättre blir det dock med `xargs`. Det är ett kommando som tar emot en lista av filnamn, till exempel från find, och sen kör valfritt kommando med denna lista som argument. Föregående exempel blir så här med xargs. `find /etc -name passwd | xargs cat`. Om du vill radera alla filer rekursivt, med utgångspunkt där du står, men behålla alla kataloger så kan du köra denna rad.
`find . -type f|xargs rm`. Ett problem är dock att den blir förvirrad av filnamn med mellanslag. Det löser man genom en nullterminerad lista, via argument till både find och xargs. Så här: `find . -type f -print0 | xargs -0 rm`
Eller med ett annat exempel, visa alla filer som heter något.txt: `find . -iname '*.txt' -print0|xargs -0 cat`
Växeln `-iname` betyder ”ignore case”, alltså hitta såväl filen andreas.txt som kalle.TXT.

Andra vanliga användningar av find är att hitta filer som har modifierats inom en viss tid. Det kan du göra så här: `find . -mtime 3`, det kommandot kommer visa alla filer som har förändrats för exakt tre dagar sen. Mer användbart är `find . -mtime -3` som visar alla filer som modifierats för mindre än tre dagar sen, eller `find . -mtime +3` för att hitta alla som är äldre än tre dagar. Vill du ha minuter istället för dagar så finns det `-mmin` med motsvarande syntax. (Se avsnitt 12.7 för mer information om filers olika tider.)

Kommandot find kan mycket mera, se `man find` för fler exempel.

3.5 root och sudo

Användaren root kan göra allting, det kontot måste alltid finnas och har user id 0. När man kör som root har man ofta en # i prompten. Det är lätt, och allvarligt, att göra misstag som root.

Bättre än att använda `su -` eller att logga in som root, är att använda `sudo`. Med `sudo` måste du tänka före varje kommando, och skriva in ditt lösenord om det har gått mer än fem minuter. Och alla kommandon loggas så man kan se i efterhand vem som har gjort vad.

Vilka som får köra `sudo` bestäms i filen `/etc/sudoers`. Den ska man alltid redigera med kommandot `visudo` så man får kontroll av syntax och låsfil så inte två ändrar samtidigt. På de flesta distributioner körs som standard editorn `vi` vid `visudo`. Vanligaste ändringen består av att leta reda på raden med:

```
root    ALL=(ALL)       ALL
```

och på den raden trycka **yy** följt av **p** och skriva rätt användarnamn. Spara sen filen med **:wq**. Den nya raden ska alltså se ut så här för användaren ao:

```
ao    ALL=(ALL)       ALL
```

På Ubuntu körs ofta editorn `pico` även om kommandot heter `visudo`, men det går att ändra. Mer om editorer kommer i nästa kapitel.

3.6 Lästips

Den här boken är som sagt inte en lista över alla kommandon. Utöver tidigare nämnda *Effektivare Linux* så finns det många bra engelska böcker. Till exempel *Fedora Linux Toolbox* av Christopher Negus och *Ubuntu Linux Toolbox*, av Christopher Negus och Francois Caen. De är utgivna på Wiley. Det finns även en *BSD UNIX Toolbox* av samma författare, den är bra om man vill lära sig mera om BSD-systemen.

För systemadministration av Linux/UNIX i allmänhet är den bästa boken:
UNIX and Linux System Administration Handbook av Nemeth et al. Fjärde upplagan kom 2010 på Prentice Hall.

En lite äldre men fortfarande läsvärd bok är *Essential System Administration* av Aeleen Frisch. Senaste upplagan på O'Reilly är från 2002.