UNIX-Introduktion

ÄG Datavetenskap

Tobias Olausson

Bartolomeus Jankowski

1 Introduktion

Denna guide är tänkt att hjälpa er bli bekväma med systemet som datorerna på chalmers använder. Vi som skrivit den har försökt plita ner de flesta bra saker som vi känner är bra att kunna. Vi hoppas att ni ska tycka den är till hjälp, och ni får gärna fråga om det är något ni undrar. Nu kör vi, full fart!

Hemkatalogen brukar ligga på `/chalmers/users/användarnamn/`. Det är denna katalog som användaren har till sitt förfogande. Användaren kan fritt skapa, ta bort och ändra rättigheter på filer och kataloger liggande i denna mapp. Hemkatalogen brukar betecknas med tildetecken `(~)` som fås genom att kombinera Alt-Gr och tangenten liggande till höger om Å. Det händer ofta att man vill komma till sin hemkatalog, detta uppnås enklast genom att skriva i prompten:

```
$ cd
```

eller

```
$ cd ~
```

Självklart fungerar också att skriva:

```
$ cd /home/ditt_användarnamn/
```

2 Filsystemet

Alla kataloger och filer i GNU/Linux ligger i en huvudkatalog som betecknas med forwardslash `(/)` och kallas för ”rotkatalogen” eller kortare ”roten”. För att förflytta sig mellan olika kataloger används commandot cd. I linux finns det förutom roten ytterliggare två specialkataloger: den nuvarande katalogen - betecknas med en punkt `(.)` och den ovanliggande katalogen som betecnas med två punkter `(..)`. Tex. för att komma till roten skriver man:

```
$ cd /
```

För att komma till `/home/` katalogen när man befinner síg i sin hemkatalog skriver man:

```
$ cd ..
```

detta ”förflyttar” användaren till den ovanliggande katalogen som är ju `/home/`. Man kan fråga sig vad `(.)` är bra för? Jo, det används tex. när man har en körbar fil (dvs ett program) i en mapp och vill köra den. I linux kör man ett program genom att skriva dess sökväg och namn i prompten. Så tex. för att köra filen `mitt_program` som ligger i katalogen `/home/kurser/labbar/lab1/` måste man skriva hela sökvägen och namnet

```
$ /home/kurser/labbar/lab1/mitt_program
```

men om man befinner sig i katalogen som innehåller filen i fråga behöver man bara skriva:

```
$ ./mitt_program
```

Som sagt kommadot cd med efterföljande sökvägen används för att komma tillr den katalogen som sökvägen pekar på. För att se sökvägen till den katalogen man just nu befinner sig i anwänds komandot pwd:

```
$ pwd
```

För att se innehållet i den nuvarande katalogen använder man kommandot ls, för att se en annan katalogs innehåll skriver man dess sökväg efteråt, tex.

```
$ ls ~/kurser/labbar/
```

för att se innehållet i `/home/användarnamn/kurser/labbar` Kommandot ls listar inte riktigt alla filer - filerna vars namn börjar med en punkt som oftast är systemfiler eller konfigurationsfiler, visas inte. En sådan gömd fil är tex `/.bashrc` som innehåller olika slags inställningar som användaren kan göra. För att visa alla filer i en katalog använder man växeln -a. För att visa lite mera info om filerna kan man använda växeln -l. Det går bra att slå ihop växlarna, tex:

```
$ ls -al ~/kurser/labbar/
```

För att kopiera en fil använder man kommandot cp:

```
$ cp /sökväg_1/filnamn_1 /sökväg2/filnamn_2
```

Kommandot kopierar filen med namnet `filnamn_1` som ligger i katalogen som pekas av `sökväg_1` till katalogen som pekas av `sökväg_2`, kopian ska ha det nya namnet `filnamn_2`. Det nya namnet är inte obligatoriskt, om inget anges behåller kopian `filnamn_1` som sitt namn.

För att flytta en fil används kommandot mv. Detta är också det ända sättet att döpa om en fil. mv används på ett motsvarande sätt som cp. Tex:

```
$ mv fil_1 fil2
```

döper om filen `fil_1` till `fil_2`. För att skapa en katalog används kommandot mkdir, tex:

```
$ mkdir katalognamnet
```

För att ta bort ta bort en fil används kommandot rm, tex:

```
$ rm filnamn
```

Detta funkar inte på kataloger, då måste man ange att man vill ta bort saker rekursivt dvs. själva katalogen och filer/kataloger den innehåller.

```
$ rm -r ~/kurser
```

I GNU/Linux kan man bestämma for varje fil om denna ska kunna läsas, tas bort/ändras och köras. För varje fil kan detta bästämmas för tre olika grupper av användare: filens ägare, filens grupp, och övriga användare som varken är filens ägare eller tillhör filens gupp. För att se de nuvarande rättigheterna använder man kommandot ls tillsammans med växeln -l. Anta att det finns en fil som heter prog i den nuvarande katalogen:

```
$ ls -l prog
-rwxrw-r-- 1 brtk dv09 2772 2009-08-12 18:40 prog
```

Första tecknet (-) betyder att det är vanlig fil, skulle det stå (d) skulle det betyda att filen var en katalog. Sedan följer tre triplar med bokstäver och (-). Varje triplet visar vilka rättigheter som gäller för olika typer av användare: filens ägare, filens grupp och övriga användare. (r) betyder att man kan läsa filen, (w) betyder att man kan ändra eller ta bort filen och (x) betyder att man får lov att köra filen. Om det står (-) istället för en bokstav, betyder det att man inte har den rättigheten som den motsvarande bokstaven representerar.

Vidare står det lite mer användbar info: filens ägare - brtk, filens grupp - dv09, datum när filen skapades och filens namn.

I det här fallet ser man att ägaren kan läsa, ändra och köra filen, gruppen kan bara läsa och ändra filen och de övriga får endast läsa filen och varken ändra den eller köra den.

För att ändra rättigheterna på en fil använder man kommandot chmod. Lättast är nog att använda oktal notation för rättigheterna man vill ändra till:

```
$ chmod XYZ filnamet
```

X, Y och Z är tre tal på oktal form (talen mellan 0 och 7) som sepresenterar rettigheterna. X representerar den första triplen - ägarens rättigheter, Y representerar gruppens och Z de övrigas. Hur räknar man ut X, Y och Z? Det är enkelt: man ska bara komma ihåg hur mycket varje bokstav är värd. (r) är värd 4, (w) är värd 2 och (x) är värd 1. När man bestämt sig vilka rättigheter filen ska ha summerar man bara de bokstäverna som ska vara med. I ovanstående exemplet är X=7 eftersom (r), (w) och (x) är med alltså 4+2+1. Y=6 eftersom bara (r) och (w) är med, alltså 4+2 och till sist Z=4 eftersom bara (r) är med. Anta nu att det finns en fil med namnet `prog_2` och man vill ge den likadana rättigheter som för filen prog. Då skulle man kunna använda chmod på följande sätt:

```
$ chmod 764 prog_2
```

Man kan även ändra rättigheter på följande sätt:

```
$ chmod ugo+rx prog_3
```

Exemplet ovan ger user (u), group (g) och other (o) rättighet att läsa och exekvera programmet.

Ibland vill man ändra filens ägare eller grupp. Detta kan man göra med kommandot chown. För att låta användaren Kalle äga prog skriver man följande:

```
$ chown Kalle prog
```

Man måste äga filen för att kunna ändra dess rättigheter och ägare. Nu kan Kalle ändra filens grupp till dv06:

```
$ chown :dv06 prog
```

Om man vill se hur mycket utrmme man har på hå hårddisken kan man använda kommandot df:

```
$ df -Ph
```

Vill man se hur mycket utrymme en fil tar använder man kommandot du:

```
$ du -h filnamn
```

Vill man se alla filer skriver man bara

```
$ du -ha
```

3 Konfiguration

Det finns många sätt att styra hur systemet fungerar och beter sig på chalmers datorer, och på UNIX-system i allmänhet. Några som är bra att ha går vi igenom i detta avsnittet. Det första som kan vara bra att veta är hur man ändrar sitt lösenord. Normalt på UNIX-system så kan man skriva passwd för att ändra, men det går inte på chalmers, eftersom systemet inte ligger lokalt på den dator du sitter vid, utan centralt på en server. För att ändra lösenord får man surfa in på `https://cdks.chalmers.se/` och logga in. Om man vill komma åt NOMAD (chalmers wlan), måste man logga in på CDKS och skapa sig ett /net-lösenord. Mer info om hur det funkar finns på CDKS.

När ni använder terminaler kommer det finnas ett skal som tolkar det ni skriver in i terminalen, det vanligaste (som är standard) heter bash (bourne again shell). Man kan ställa in en del saker i bash, så som alias av kommandon, hur texten innan kommandon ser ut, etc. Sådana saker styrs med fördel av att lägga dessa kommandon i en fil som heter .bashrc i eran hemkatalog. Här kommer några exempel:

```
alias ll='ls -hl'
alias sshome='ssh user@home.se'
PATH=$PATH:~/bin/:~/.cabal/bin/
PS1='[\u@\h \w]\$ '
export SVN_EDITOR=vim
```

De första två raderna är alias för kommandon, så att när jag skriver ”ll” så körs istället ”ls -hl” som gör att man får en trevligare listning av filer i katalogen. Det tredje direktivet sätter PATH till PATH och två ytterligare kataloger. PATH är en så kallad miljövariabel som bash tittar på när du vill starta program. Alla kataloger i PATH söks igenom efter ett program med det namn du skriver, och detta används även vid tab-completion. Det fjärde direktivet sätter en annan miljövariabel, PS1, som styr hur prompten ser ut. I exemplet ovan skrivs username, host och katalog ut. I mitt fall kan det se ut så här:

```
[tobsi@wobsi ~/ag_data/2009/RePe/unix]$
```

Mer info om hur man sätter PS1-variabeln kan man hitta i manualbladet för bash. Sista direktivet exporterar en miljövariabel som görs tillgänglig för hela system. Skillnaden mellan PATH/PS1-raden och export-raden är att de tidigare sätter variabler som blir tillgängliga endast för bash, medan export

sätter en ”global” miljövariabel. När man kodar java kan det vara läge att sätta en variabel som heter CLASSPATH som styr var java letar efter filer som har .class som filändelse.

Det finns en hel drös med programvara installerad på chalmers system, och för det mesta räcker dessa utmärkt till allt man vill göra. Dock finns det ibland fall då du behöver en specifik version av ett program (till exempel en kompilator), och då har chalmers ett system för detta som heter vcs (version control system). Det gör både så att du kan välja versioner av program, och du kan ställa in det för att få tag på program som inte finns standard, och därmed inte är supportade. För att få access till program som inte är supportade (unsup), skriv följande i en terminal:

```
echo unsup >> $HOME/.vcs4/pathsetup
```

Mer info om hur man väljer versioner och dylikt finns på: `http://www.chalmers.se/its/EN/computer-workplace/linux/various-linux-questions`

4 Processer

Om man vill köra ett program som finns i den katalogen man befinner sig i skriver man

```
$ ./mitt_program
```

Ibalnd vill man ta reda på hur lång tid ett program eller kommando tar att köra. Kommandot time visar den tiden:

```
$ time ./mitt_program
```

Ibland vill man avbryta ett program som körs, kanske för att man inte har tid att vänta tills det har terminerat eller för att programmet har hängt sig. Kortkommandot ctrl+c försöker avbryta det programmet som körs i konsollen just nu. Ibland vill man inte avbryta progarmmet utan bara stoppa det. Kortkommandot ctrl-z gör det. Anta att man kör bildvisningsprogrammet Eye Of GNOME, för att visa bilden min_bild.jpg, man startar det genom kommandot eog.

```
$ eog min_bild.jpg
```

Det som händer nu är att eog har ”tagit kontrollen” över konsolen. För att återta kontrollen kan man trycka ctrl+c för att avbryta eller ctrl+d för att stoppa programmet. Har man valt att stoppa, kan man inte längre använda eog, det har stannat men inte avslutats dock. Nu kan man välja att låta

programmet återta kontrollen över konsollen eller köra det i bakgrunden. Kmmandot fg gör att det senast stoppade programmet satartar igen och återatr kontrollen över konsollen.

```
$ fg
```

För att fortsätta att köra progarmmet i bakgrunden och samtidigt kunna använda konsollen, kör man kommandot bg istället:

```
$ bg
```

Man kan låta prgrammet köras i bakgrunden direkt när man startar det, detta genom att skriva ampersandtecken efter programnamnet:

```
$ eog &
```

Ovanstående har samma effekt som att köra eog och trycka sedan ctrl+z samt bg.

Ibland går det tyvärr inte att avbryta ett program med ctrl+c - programmet svarar helt enkelt inte. I det fallet om man vet programmet idnummet (pid) kör man kommandot kill på följande sätt

```
$ kill -9 pid
```

Om man vet programmets namn är det oftast lättare att köra killall:

```
$ killall -9 mitt_programm
```

Flaggan -9 tvingar programmet att avsluta, om man utelämnar den ber systemet istället programmet att avsluta snällt, vilket ofta är att föredra, men inte alltid funkar.

För att ta reda på ett program id kan man köra kommandon top eller ps. Top ger en dynamisk bild av processer som körs på systemet, ps skriver däremot ut en ”snapshot” av processer som körs just då:

```
$ top
```

Man kan trycka på (h) för att få lite kort hjälp med top eller avbryta med (q) eller ctrl+c.

Kommandot `ps` skriver ut programmen som körs i den nuvarande konsollen. Om man vill ha alla processer i systemet använder man växeln aux

```
$ ps aux
```

Man kan även lista processer som en viss användare har startat:

```
$ ps aux | grep användarnamn
```

Vill man se alla användare och vilka program de kör kriver man helt enkelt `w`

```
$ w
```

5 Programmering

Mycket av erat arbete vid datorerna kommer bestå av programmering, och det finns flera olika miljöer att arbeta med på chalmers datorer.

Här kommer några praktiska editorer, nano är den absolut lättaste av dessa, medan ViM och emacs kräver mer kunskap, men i gengäld är betydligt mer avancerade. Betänk att många gånger kan det vara lättare att jobba i en IDE (som har direkt stöd för kompilering av kod etc), men det är generellt sett mödan värt att lära sig hantera minst en av de vanliga texteditorerna. Många tycker även att gedit (som följer med GNOME) är smidig och trevlig, speciellt för personer som tidigare varit vana vid windows.

5.1 nano (Nano's ANOther text editor)

nano är en väldigt liten och smidig editor som följer med nästan alla UNIX-baserade system. När nano startas (tex: nano file.txt) så kan man direkt skriva i filen, och längst ner syns en liten menyrad med alternativ. ^G betyder ctrl-g, etc. Nedan följer de vanligaste kommandona (även om om står i menyraden):

^X Avsluta

^O Spara

^R Öppna

^W Sök

^K Klipp ut

^U Klistra in

5.2 ViM (Vi IMproved)

ViM är en mer avancerad editor som lämpar sig bra för programmering. När man använder ViM så finns det olika kommandon som fungerar olika beroende på vilket ”mode” man är i, då det finns flera olika. För mer info om ViM än den vi visar nedan, skriv :help i Normal Mode, detta är rekommenderat då ViMs egen hjälp är betydligt mer ingående än vår.

Följande gäller i Normal Mode (ViM startar i Normal Mode)

dd klipp ut en rad

D klipp ut resten av aktuell rad

yy kopiera en rad

p klistra in under aktiv rad

P klistra in ovanför aktiv rad

i byt till Insert Mode

u ångra

v byt till visual mode

Följande gäller också (: aktiverar Command Mode)

:w spara fil

:q avsluta

:q! avsluta utan att spara ändringar

:wq spara och avsluta

:e öppna fil i ny buffer

:vs splitta fönster vertikalt

:<siffra> gå till motsvarande rad

Vi rekommenderar att ni läser manualen för ViM för vidare hjälp, :help aktiverar.

5.3 emacs

Emacs skrevs från början av Richard Stallman, initiativtagare till GNU-projektet, samt grundare av Free Software Foundation. Emacs är den antagligen mest utvecklade editorn för UNIX-system och det finns allt ifrån IRC-klienter till webbläsare till spel skrivna för denna editor. Emacs har till skillnad från ViM bara ett egentligt mode, och de flesta kommandon utförs i två steg. C-x betyder ctrl-x, C-x s betyder ctrl-x s, C-x C-s betyder ctrl-x ctrl-s. M-x betyder alt-x etc.

Här är ett gäng kommandon som är användbara i emacs:

C-x s spara

C-x c avsluta

M-e Gå till slutet av mening

M-a Gå till början av mening

C-_ Ångra

C-g Avbryt kommando (obs, INTE esc)

Vi rekommenderar användare av emacs att läsa tutorialen som följer med, den kan man hitta genom att starta emacs utan argument, och klicka på "Emacs Tutorial"

6 Övrigt

6.1 Arkiv

Många gånger ska labbar packas på något sätt. Det rekommenderade programmet för sådant är tar, men för kompabilitet visar vi även hur zip/unzip och unrar fungerar.

6.1.1 tar

c - create

x - extract

j - bzip2 - komprimerar bättre än gzip men är långsammare

z - gzip

v - verbose - skriver ut info på skärmen

f - file

Exempel på användning:

`tar cvjf foo.tar.bz2 foo/`
: Skapar ett arkiv med bzip-komprimering från katalogen foo

`tar cvzf foo.tar.gz foo/`
: Skapar ett arkiv med gzip-komprimering från katalogen foo

`tar xvjf foo.tar.bz2`
: Packar upp arkivet bzip-komprimerade arkivet foo.tar.bz2

6.1.2 unrar/unzip

`unrar x foo.rar`
: Packar upp arkivet foo.rar

`unzip foo.zip`
: Packar upp foo.zip

6.2 SSH

Denna sektionen tar upp programmen SSH (Secure SHell) och SCP (Secure CoPy) som med fördel används för kommunikation mellan en personlig dator och ditt konto på chalmers. All ssh/scp-trafik är krypterad. Om du har en dator med windows, finns en gratisklient för SSH som heter PuTTY, som även kan användas för telnet.

6.2.1 SSH - Shell Access

`ssh remote1.studat.chalmers.se -l username`

`ssh username@remote1.studat.chalmers.se`
: Ansluter till remote1 med användarnamnet username, dessa två är ekvivalenta

`ssh username@remote1.studat.chalmers.se -X`
: Som ovan, fast med X11-forwarding. Grafiska program visas på din dator men exekveras på remote1.

6.2.2 SCP - Krypterad Filöverföring

`scp username@remote1.studat.chalmers.se:katalog/fil.txt .`
Kopierar fil.txt från remote1 till aktuell katalog

`scp fil.txt username@remote1.studat.chalmers.se:katalog/`
Kopierar fil.txt till katalogen ”katalog/” på remote1

6.3 Sökning

Många gånger vill man söka, kanske i filer, kanske i resultatet av något annat kommando, eller varför inte i ett manualblad?

6.3.1 grep

grep (Global Regular Expression and Print) är ett program som söker efter mönster i strömmar, filer och dylikt. Exempel på användning:

`grep apple fruits.txt`
Sök efter 'apple' i texten om frukter

`grep -w apple fruitlist.txt`
Visar bara matchande hela ord (pineapple matchar ej tex)

`grep -i apple fruitlist.txt`
Söker skiftlägesokänsligt efter äpplen i fruktlistan.

Exemplen ovan är tagna från `http://en.wikipedia.org/wiki/Grep`

6.3.2 man, less etc

I program som man och less kan man enkelt söka genom att skriva / och sedan det man vill söka efter. För att hoppa till nästa matchning trycker man n. För att hoppa till föregående matchning trycker man N.

6.3.3 locate

Om man vill veta var en viss fil kan man använa locate, det letar genom filsystemet och skriver ut de filer och kataloger som matchar mönstret.

6.4 Diverse

Ibland vill man bara skriva ut en fil till terminalen, det gör man lättast med kommandot cat. cat används egentligen för att sätta ihop filer efter varandra, men med en fil, så skriver det bara ut filen:

```
cat /proc/cpuinfo
```

skriver ut lite information om processorn. Om man vill ha reda på vilket program som passar till att göra någon speciell sak, kan man använda whereis eller apropos, exempelvis:

```
apropos pdf
```

skriver ut allt som systemet vet är relaterat till pdf