Kapitel 12

Blandade tips

Detta kapitel består av diverse tips som inte passar in i andra kapitel. Det innehåller både mjuka och hårda ämnen.

12.1 Utbildning

Många administratörer är till största delen självlärda, men det skadar inte att kombinera egna studier med mer formella kurser. Högskolepoäng är ofta ett bättre, och billigare, kvitto på kunskaper än många certifieringar.

Skaffa en bred och djup utbildning. Att ha en civilingenjörsexamen är inte i sig varken nödvändigt eller tillräckligt för att bli en bra systemadministratör. Men det är såklart ingen nackdel. Var dock inte rädd för att läsa andra kurser och ämnen, humaniora är mer relevant än vad många kan tro. Att vara admin innebär, bland annat, att hantera oerhörda mängder information, så kunskaper i språk (svenska, engelska, lingvistik eller liknande kurser) är alltid bra. Filosofi och idéhistoria tränar upp analytiskt tänkande och förmågan att se annorlunda perspektiv. Psykologi och annan beteendevetenskap är också bra, det är ju människor som använder datorerna.

Det är lite svårt att tipsa om specifika kurser i en bok, utbudet förändras. Men prova att skriva ord som ”linux” på *http://studera.nu/*. Nu (våren 2013) finns det bland annat bra kurser på Mittuniversitetet och Umeå Universitet. Sök

även efter ”TCP”, eller ”Nätverk” så kan du hitta något kul och användbart. Och som sagt: glöm inte humaniora och samhällsvetenskap.

12.2 Jobbkontroll

Ofta vill man ha tillbaka en prompt utan att behöva vänta på att programmet har kört klart. Det kan man göra genom att ange ett & efter kommandot, så körs det i bakgrunden medan du gör annat. Till exempel `gzip storafilen.txt &`. Ifall du glömmer & kan du trycka **ctrl-z** och skriva skalkommandot `bg` (som i background). Vill du istället köra det i förgrunden skriver du `fg`. Med kommandot `jobs` ser du vilka bakgrundsjobb du kör i det aktuella skalet. Vill du lägga ett specifikt jobb i förgrund eller bakgrund anger du vilket med `%n` där n är samma siffra som `jobs` anger inom [] först på raden. Att ange jobbnummer med % funkar även med `kill`.

%% betyder det senast startade jobbet.

Ett alternativ till %siffra är att ange % följt av inledande bokstav/bokstäver i kommandot. Det kan vara lättare.

Om man kör X och vill starta grafiska program från terminal gör man det lämpligast i bakgrunden, till exempel `firefox &`.

12.3 Screen

Ifall du kör program i terminal länge och vill kunna återansluta om förbindelsen går ner eller från en annan dator kan jag rekommendera `screen`. Om du bara skriver screen får du upp ett nytt skal, men i en subprocess till screen. Där kan du köra det program du vill ha och sen koppla bort (detach) med **ctrl-a, d**. För att återansluta skriver du `screen -r`. Är du inloggad från en annan maskin och inte har kopplat bort screen från den första datorn använder du `screen -rd`. Jag brukar köra IRC-klient (`irssi`) i screen på en maskin med stabil lina, så slipper man hoppa ut ur kanalen varje gång hemmabredbandet går ner. Programmet körs vidare hela tiden, du ser utdata som vanligt när du återansluter.

Det går att använda screen till massor andra saker, som att ha flera fönster i samma terminal, klipp-o-klistra med mera. Se `man screen` för mer information.

12.4 TAB

Att man kan komplettera namn på kommandon och filer med **TAB**-tangenten känner du nog redan till, men det används så ofta att det måste nämnas. I Bash är standardinställningen att en **TAB** kompletterar om det finns unik matchning, och en **TAB** till visar vilka alternativ som finns. Zsh är lite annorlunda, men det går att få den att uppföra sig som i Bash om man vill det. Jag använder även **TAB** efter kommandon som `mkdir`. Det skulle krävas avancerad AI för att den skulle kunna gissa rätt, men med en **TAB** ser man huruvida namnet redan finns.

12.5 ESC_

Jag har läst ganska många NIX-böcker, men bara en har tipsat om Escape-Underscore. Men efter jag lärt mig det använder jag det ofta. Trycker man **ESC** följt av understreck fyller skalet i sista ”ordet” från föregående rad. Det är ofta som man behandlar samma ”sak” direkt efter varandra. Exempel:

```
vi artikeln_med_det_långa_namnet.tex
pdflatex <ESC>_
```

eller

```
mkdir /tmp/lagringsplats
cd <ESC>_
```

12.6 ctrl-r

Med **ctrl-r** söker man bakåt i sin kommandohistorik. Jag skriver ofta `stamplaut logwatch, forum` på en dator jag är inloggad på. Snabbare än att bläddra med pil upp är att trycka **<ctrl-r><wa>** vilket oftast ger rätt rad direkt. Vad på raden som ger bäst träff kan får man välja själv, blir det fel rad som matchas trycker man **ctrl-r** igen.

12.7 Tre tider och touch

UNIX och Linux sparar tre tider för varje fil eller katalog. De är mtime, ctime och atime. Dessa sparas i en datastruktur som heter inod, där även rättigheterna finns. Men inte filnamnet, det sparas enbart i katalogen tillsammans med en pekare till filens inod.

- mtime (modification time) är filens modifieringstid, alltså senaste tidpunkten för ändring av innehållet i filen. Till exempel om du lägger till en rad med en editor.
- ctime (change time) är inodens ändringstid, metadata som ägare och rättigheter. Till exempel om du kör `chmod` ändras ctime men inte mtime.
- atime (access time) är när filen senast användes (öppnades). För att snabba upp läsning kan man montera filsystem med alternativet noatime. Då saknas det värdet helt, och inget behöver skrivas till filsystemet vid varje läsning. Det passar bra för till exempel /var/www/.

Märk väl att *NIX normalt sett *inte* sparar vilken tid som en fil skapades. Har du redigerat en textfil efter den skapats går det inte att se annat än tiden för senaste ändringen. I Ext4 har man lagt till date-created, men det är inte bara filsystemet som behöver ändras utan även systemanrop och många bibliotek som glibc. Så tillsvidare är det de tre ovan som kan användas.

Med kommandot `touch` kan du uppdatera mtime och atime för en fil, eller skapa en ny tom fil. Med `touch -m` uppdateras enbart mtime och med `touch -a` enbart atime.

Alla tiderna sparas i ”epoch-formatet”, alltså antalet sekunder sen 1970-01-01. Det är ett 32-bitars tal, så år 2038 blir det problem. Vill du se alla tre tiderna för en fil kan du köra `stat filnamn`. Kommandot `ls -l` visar mtime, och det är oftast den som är mest relevant att veta. Med `ls -lc` kan du se ctime och med `ls -lu` atime.

I nyare kärnor är upplösningen en nanosekund (för de filsystem som stöder det), på riktigt gamla är det en sekund.

12.8 grep av processer

Om du kör `ps aux|grep imap` så kommer ibland, men inte alltid, även processen grep visas. Vissa använder `|grep -v grep` för att undvika detta, men en snyggare lösning är att använda ett enkelt reguljärt uttryck. Skriv `ps aux|grep '[i]map'` så kommer grep-processen, som körs samtidigt, inte matcha sig själv. Speciellt viktigt är detta vid `grep -c`. Jag använder detta knep i stycke 9.2. Alla skal behöver inte citattecknen, men de skadar aldrig. Knepet funkar med alla utom riktigt gamla grep. Jag lärde mig det här i Aeleen Frischs bok *Essential System Administration*.

Du kan förstås även använda `pgrep` och `pkill`.

12.9 Klipp och klistra

Om du kör X11 kan du klistra in det senast markerade med mittenknappen/hjulet på musen. Eller genom att trycka både höger och vänster musknapp samtidigt.

Är det i `vi` du jobbar så trycker du **p** för att få senaste raderna från **dd** eller **yy**. Du kan även ha 26 olika namngivna buffertar.

Om du kör GNOME eller KDE fungerar det även med **ctrl-c**, **ctrl-x** och **ctrl-v**.

Det är ganska praktiskt att kunna ha olika strängar tillgängliga på olika sätt, men det kan vara lite förvirrande ibland.

Om du har skrivit en lång och bra rad vid prompten, och vill spara i en fil, så kan du göra så här. Skriv, sist på raden. `">fil`, tryck **ctrl-a** och skriv `echo "`. Byt ut till enkla citattecken om din rad innehåller dubbla citattecken eller dollarexpansion. Detta tips funkar även i konsolen.

Ett annat tips är att använda kommentartecknet vid prompten. T.ex. om du har skrivit `mount /dev/sdc1 /mnt/c1` och kommer på att katalogen `c1` inte finns. Då kan du trycka **ctrl-a**, skriva #, trycka enter och därefter skapa katalogen. Sen behövs bara pil upp och ta bort #. Det är praktiskt att kunna spara rader i historiken utan att behöva köra dem.

12.10 tail och head

Kommandot `tail` har jag redan visat på några ställen i boken. Utan andra argument än ett filnamn visar det filens sista tio rader. Så här `tail /etc/passwd`. Vill du se ett annat antal rader anger du en siffra:

```
tail -2 /var/log/httpd/access_log
```

Vill du se början av en fil heter kommandot `head`. Även det kan ta minus antal som argument.

För `tail` är `-f` ett användbart argument för filer som växer. Då ser du allt nytt rad för rad. Till exempel:

```
tail -f /var/log/httpd/access_log
```

Det funkar även med flera filer:

```
tail -f /var/log/httpd/*
```

Då får du först de tio första raderna av alla, därefter alla nya rader från varje fil.

12.11 watch

Ett annat användbart kommando är `watch`. Det kör ett kommando upprepade gånger och visar senaste körningen i realtid. Till exempel `watch w` kommer köra `w` varannan sekund och låter dig se hur belastningen varierar och vilka som är inloggade. Med flaggan `-d` markeras ändringar, t.ex. så visar:

```
watch -n 30 -d ls -l
```

inverterad bakgrund i en halv minut om tider eller storlek ändras på filer i aktuell katalog. Överst i fönstret ser du intervall, kommando och tid för senaste körning. Du avslutar watch med **ctrl-c**.

12.12 Hårdvara

Den här boken handlar mer om mjukvara än hårdvara, men lite vill jag nämna om datorer för serverdrift.

Ofta kan man använda vanliga konsumentdatorer även som servrar. Till exempel för att testa eller för mindre viktiga uppgifter. Fördelar med det är att de är billiga – du kanske redan har dem. Datorer som känns för långsamma för kontorsanvändning kan fungera bra som servrar för intranät eller backup. Men för mer seriös drift bör man ha riktiga serverdatorer. De är mer stabila och avsedda för drift dygnet runt. De har ofta ECC-minne, IPMI (vad det är berättar jag i 16.5) och dyrare processorer med mera cache.

På större företag har man oftast servrarna i rack. De är 19 tum breda och höjdenheten heter U, som i unit, ett U är 1,75 tum (ca 45 mm). Mindre servrar är 1 U, större t.ex. 2 U eller 4 U. Rackservrar är avsedda för kontinuerlig drift, och låter en hel del jämfört med vanliga kontorsdatorer. I en serverhall kan det finnas många rack, varje rack är ofta 42 U eller 44 U högt och kan alltså rymma ca 40 1U-servrar och ett par switchar. Man bör även ha en UPS för att slippa nertid vid strömavbrott. Större hallar har även dieselmotor med generator för att klara många timmar utan ström. Även kylning är viktigt.

När du väljer att köpa en server är det ofta RAM du ska prioritera, mycket arbetsminne betyder ofta mer för prestandan än snabb CPU. Men det beror så klart på vad den ska göra. Om du inte ska lagra mycket data kan SSD var bättre än traditionella hårddiskar. Det går även att blanda, SSD för snabb åtkomst och snurrdisk för långtidslagring. Ska man ha det allra snabbaste kan det bli dyrt, men tänk på att din nya server ändå blir omodern inom ett år (kanske vid nästa reboot), så satsa hellre på mellansegmentet om du inte behöver det allra senaste.

12.13 bc

På sidan 34 nämnde jag kommandot `bc`. Den använder jag ofta som miniräknare, och här får du några tips för att göra den trevligare. Jag brukar använda följande alias:

```
alias bc='bc -q ~/.bcoptions'
```

Med det slipper jag den inledande texten, och får möjlighet att ange variabler i filen `~/.bcoptions`. I den anger jag scale=3 för att alltid få tre decimaler, och vissa konstanter som jag ofta använder när jag räknar, som hyra och lön.