Kapitel 17

RAID, backup och rsync

17.1 RAID

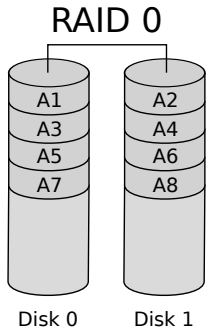
RAID betyder ”redundant array of independent disks”, I:et stod från början för inexpensive. Det finns flera nivåer av RAID.

RAID-0 är ”striping”, ett sätt att slå ihop två eller flera diskar till en större enhet. Vartannat block på varannan disk, se bild 17.1. RAID-0 bör man inte använda för viktig data, går den ena disken sönder så får man många trasiga filer.

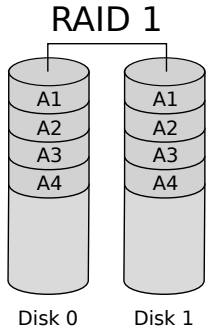
RAID-1 är spegling, där all information sparas dubbelt. Se bild 17.2. Det är ett bra sätt att få högre tillgänglighet och dataintegritet. Vilken som helst av båda diskarna kan gå sönder, och du har ändå allting kvar. Att skriva går lite långsammare än med en disk, men läsning kan gå snabbare.

Nivåerna 2–4 används sällan. Nivå 5 och 6 är vanligare. Där fördelas extrainformationen på alla diskarna, så att allt går att återskapa om en (nivå 5) eller två diskar (nivå 6) går sönder. Se bild 17.3 För RAID-5 behövs minst 3 diskar, och för RAID-6 minst 4.

Det finns även kombinationer av flera RAID-nivåer, som 1+0 (även kallad 10) där man har striping av speglade diskar.



Figur 17.1: *RAID-0*

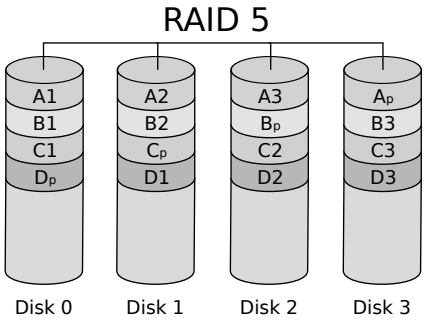


Figur 17.2: *RAID-1*

Så här kan man skapa en RAID-1 i Linux:

```
mdadm --create /dev/md0 --level=1 \
--raid-devices=2 /dev/sd[ab]2
```

RAID är bra – men det är ingen ersättning för backup. Mer än en disk kan gå sönder, hela datorn eller huset kan brinna upp. Användare, program eller inkräktare kan radera filer, av misstag eller medvetet. Därför bör du alltid ha bra backup, helst på flera ställen och för flera dagar bakåt.



Figur 17.3: *RAID-5*

17.2 tar

Ett gammalt, men fortfarande ofta använt, sätt för säkerhetskopiering är `tar`. Det stod ursprungligen för ”tape archiver”, men används numera oftast till filer. De vanligaste argumenten är x=extract, c=create, t=list. Andra vanliga flaggor är v=verbose och f=file. Nyare versioner av `tar` kan även komprimera arkiven, flaggan z för gzip och j för bzip2. Exempel:

```
tar xvf fil.tar
tar xzvf fil.tar.gz
tar cvf etc.tar /etc
cd /
tar cf /tmp/home.tar home
tar cjf /tmp/home.tar.bz2 home
tar czvf /home/backup/var_www.tgz var/www
cd /home/backup
tar tzf var_www.tgz|less
tar cf - /home/ao/*.txt|ssh b tar xf -
```

Det sista exemplet kopierar alla textfiler till dator b över SSH. Ett ensamt minustecken betyder STDIN eller STDOUT.

17.3 rsync

Ett mycket bra program är `rsync`. Det använder en algoritm som utvecklades av Andrew Tridgell och beskrivs i hans doktorsavhandling från 1999. Finessen med `rsync` är att enbart förändringar överförs. Ändras några få byte behöver man inte föra över hela filen. Det är mycket lämpligt vid backup.

Manualsidan `man rsync` är detaljerad och lång. Men de vanligast använda argumenten är `rsync -a` och `rsync -av`, där `-a` betyder ”archive mode”, samma som `-rlptgoD` och `-v` betyder verbose. Exempel:

```
rsync -a * einstein:/home/backup
rsync -av * einstein:/home/backup
```

Rsync använder numera SSH som transportprotokoll automatiskt. För backup kan det vara bra att ha nycklar utan lösenord, se 16.4 om du inte har satt upp det tidigare.

Om du vill ha daglig backup från en dator till en annan så är det lämpligt att ha ett katalogträd för varje dag, och använda hårda länkar för alla oförändrade filer. Det sparar i de flesta fall mycket diskutrymme. Ett program som gör så är rsnapshot *http://www.rsnapshot.org/* Men det går att skriva egna skript med liknande funktion. På nästa sida ser du ett exempel.

```
#!/bin/bash
cd /home/backup
datum=$(date "+%y%m%d")
#skapa katalog ifall du lägger till ny server.

for server in zeus apollo poseidon athena
do
  senast1=$(ls $server/|tail -1)
  senast2=$(ls $server/|tail -2|head -1)
  aldst=$(ls $server/|head -1)
#Kommentera bort rm i det antal dagar du vill spara.
#    rm -rf $server/$aldst
  rsync -avHS --numeric-ids -e ssh \
  --link-dest=$PWD/$server/$senast1 \
  --link-dest=$PWD/$server/$senast2 \
  --exclude-from=ex_$server \
  $server:/ $server/$datum/ >> /dev/null 2>&1
#$server:/ $server/$datum/>>/var/log/b/$server 2>&1
  exit=$?
  if [ $exit -ne 0 ] && [ $exit -ne 24 ] ; then
    mail -s "rsyncfel $server" root
  fi
#  echo $datum >> /var/log/backup/$server
done
```

Första tiden som du kör det bör du ha en # framför `rm -rf`, t.ex. i 30 dagar, beroende på diskutrymme. Sen tar du bort # och det äldsta trädet raderas varje natt. Du bör även slå på loggningen i början, som är bortkommenterad i detta skript, och hålla lite koll på vad som händer varje dag. Däremot när du vet att det fungerar så tar loggfilerna bara onödig plats.

Jag använder `--link-dest` till de två senaste katalogerna. Filen som listar vilka kataloger som ska exkluderas heter för datorn zeus `ex_zeus` och ligger i samma katalog som backupkatalogerna hamnar. Exempel på innehåll i en sådan fil är:

```
/proc
/sys
/dev
/home/*/nobackup
```

De tre första katalogerna bör man inte ta backup på, och den sista raden ger användarna möjlighet att ha en katalog `~/nobackup/` med stora filer som inte behöver kopieras.

Du bör hålla koll på att hårddiskarna på backupservern inte blir fulla och att backup verkligen fungerar. Skriptet ovan mejlar root vid de flesta fel (felkod 24 är att filer försvunnit under tiden som skriptet körs, det gav så många falsklarm att jag inte vill ha de mejlen).

Tänk även på att data på backupservern är lika känslig som på datorerna i sig. Och att root på backup kan bli root på de andra. Så ha denna dator på ett skyddat ställe, både fysiskt och via nätet.