Linux – Friheden til systemadministration

Version 2.8.20060113 – 2020-12-31

Peter Toft

Hans Schou

Linux – Friheden til systemadministrationVersion 2.8.20060113 – 2020-12-31

af Peter Toft og Hans Schou

Administrer din egen Linux-server.

Indholdsfortegnelse

Tabelliste

Forord

1. Forord

Denne bog er skrevet af erfarne Linux-folk til dem, som vil i gang med at lære systemadministration under styresystemet Linux med henblik på server-funktioner.

I bogen *Linux – Friheden til at lære Unix* gennemgik vi opbygningen af et Unix- og dermed Linux-system og fik set nærmere på systemadministration for et enkeltbruger-system. I denne bog tager vi så næste skridt og ser på hvordan man kan bruge en Linux-maskine som server for hele netværket. Dog er web-server, databaser og PHP ikke inkluderet i denne bog – se i stedet *Linux – Friheden til egen webserver*. Tilsvarende er aspekter omkring sikkerhed lagt i bogen *Linux – Friheden til sikkerhed på internettet*.

I Kapitel 1 findes en gennemgang af de mange opsætningsfiler, som man finder i `/etc`.

I Kapitel 2 gennemgås betydningen af runlevels, systemadministration med linuxconf, køre programmer på givne tidspunkter med crontab samt opstart af Linux og NT på samme maskine.

I Kapitel 3 gennemgås flere aspekter af filsystemet, såsom installation af et Journaling filsystem, *ReiserFS*, vi ser på RAID og tuning af hastighed på dataoverførsel fra harddisk.

I Kapitel 4 er der en gennemgang af hvad Linux-kernen er og ikke er. Der er en gennemgang af proces-strukturen, proces-ID og `/proc`-filsystemet. Det vises hvordan man oversætter kernen og der er en gennemgang af moduler til kernen, samt en gennemgang af hvad der er kommet med i kerne 2.4-serien.

Videre til Kapitel 5, hvor systemadministratorens vigtigste værktøjer gennemgås, såsom **ping**, **telnet**, **ftp**. Der er også en gennemgang af post-håndtering med **sendmail**, **postfix**, **procmail**, kryptering af post med GnuPG, og endelig vises hvordan man kan læse nyhedsgrupper og sætte en news-server op.

Skal man lave et halvstort eller stort netværk, skal man bruge en navneserver, en DNS-maskine. I Kapitel 6 gennemgås hvordan IP-adresser bruges, og hvordan man sætter en navneserver op.

I Kapitel 7 er der vist hvordan man sætter mange server-funktionaliteter op, såsom DHCP-server (en dynamisk navneserver), FTP-server, NFS-server (til at dele filer over netværk mellem Unix/Linux-maskiner), NIS (til at dele adgangskoder elegant frem for at kopiere adgangskodefiler mellem maskinerne). Der er et afsnit om at få adgang til Linux-maskinen via modem-adgang (ring-ind-server), X-server, og AppleTalk-funktionalitet er også gennemgået.

Senere i Kapitel 10 beskrives hvordan du får Samba til at virke. Dette giver mulighed for at Windows-maskiner kan montere netdrev fra din Linux-maskine.

Endelig ser vi i Kapitel 11 nærmere på installation af programmer.

2. Linux-bøgerne

Bogen er en del af en serie, som kan findes på http://www.linuxbog.dk/

- *Linux – Friheden til at vælge installation* – Om at installere Linux.
- *Linux – Friheden til at lære Unix* – Om hvordan man bruger Linux' (og Unix') kommandolinjeværktøjer.
- *Linux – Friheden til at vælge grafisk brugergrænseflade* – Om alle de grafiske brugergrænseflader, der findes til Linux.
- *Linux – Friheden til at vælge programmer* – Om de programmer du kan få til Linux.
- *Linux – Friheden til systemadministration* – Om at administre sit eget linuxsystem.
- *Linux – Friheden til at programmere* – Programmering på Linux
- *Linux – Friheden til at programmere i C* – Om at programmere i sproget "C".
- *Linux – Friheden til at programmere i Java* – Om at programmere i sproget "Java".
- *Linux – Friheden til sikkerhed på internettet* – Om at sikre dit Linuxsystem mod indbrud fra internettet.
- *Linux – Friheden til egen webserver* – Om at sætte en webserver med databaser, CGI-programmer og andet godt op.
- *Linux – Friheden til at skrive dokumentation* – Om at skrive dokumentation (og andet) i SGML/DocBook, LaTeX eller andre formater.
- *Linux – Friheden til at vælge kontorprogrammer* – Kontorfunktioner på et Linux/KDE/OpenOffice.org-system.
- *Linux – Friheden til at vælge IT-løsning* – Om muligheder, fordele og ulemper ved at bruge Linux i sin IT-løsning.
- *Linux – Friheden til at vælge OpenOffice.org* – Om at bruge OpenOffice.org, både på Linux og på andre styresystemer.
- *Linux – Friheden til at vælge digital signatur* – Digital signatur på Linux.

3. Ophavsret

Denne bog er skrevet af Linux-brugere til Linux-brugere. Store dele af bogen er skrevet eller redigeret af enkelte forfattere, hvilket er nævnt i revisions-historien til bogen.

Bogen kan findes i opdateret form på http://www.linuxbog.dk/, mens prøve-udgaver kan findes på http://cvs.linuxbog.dk/.

Figur 1. ÅDL

Kommentarer, ris og ros og specielt fejl og mangler bedes sendt til linuxbog@sslug.dk (mailto:linuxbog@sslug.dk), men er du medlem af SSLUG kan du i stedet for med fordel skrive til sslug-bog@sslug.dk (mailto:sslug-bog@sslug.dk).

4. Om forfatterne og bogens historie

Bogen startede som et bogprojekt i sommeren 1998, hvor Peter Toft, Kenneth Geisshirt og Snebjørn Andersen fik skrevet "Linux – Friheden til at vælge". Bogen var i starten henvendt mod Linux-installation, at få lidt Unix-kendskab og noget om netværk. Bogen har altid været en fri "web-bog", der kunne læses enten i smådele eller som en helhed på http://www.sslug.dk.

Vi, forfatterne, har anvendt Linux i flere år og har stor erfaring med det. Vi er medlemmer af Skåne Sjælland Linux User Group (SSLUG), som er en uafhængig og non-profit svensk/dansk organisation, der søger at udbrede kendskabet til Linux. Det er gratis at være medlem af SSLUG, og organisationen findes på internettet på http://www.sslug.dk. Vi har skrevet denne bog fordi den kunne være til glæde for andre.

Den oprindelige bog "Linux – Friheden til at vælge" blev udgivet som hæfte på 96 sider fra IDG Forlag i februar 1999 med en Red Hat 5.2-cd-rom, hvor den i lang tid lå på bestsellerlisten for EDB-hæfter. I december 1999 udgav Forlaget Globe version 2.9 i bogform på over 300 sider og med tre cd-rom'er (Red Hat 6.1, SuSE 6.2 og StarOffice 5.1).

I juli 2000 var bogen vokset til ca. 350 sider og blev delt ud til flere bøger. Alle kan læses på www.linuxbog.dk (http://www.linuxbog.dk/). I februar 2001 er der 8 bøger på over 700 sider. I vinter 2001 blev version 1.6 af bogen udgivet på tryk fra forlaget IDG.

Fra version 1.0 af denne bog er det Peter Toft og Hans Schou, som vedligeholder bogen sammen med de mange, som skriver på dele af bogen.

Figur 2. Peter Toft

Peter Toft er civilingeniør og har en ph.d.-grad fra DTU. Han har kørt Linux dagligt siden 1994. Aktiv Linux-forfatter og foredragsholder om Linux i Danmark og Sverige. Tidligere bestyrelsesformand for SSLUG. Medorganisator ved de danske Linux-konferencer (Linux98, Open Networks 99, LinuxForum 2000-2003). Hjemmeside http://pto.linux.dk.

Figur 3. Hans Schou

Hans Schou stoppede brat på Københavns Teknikum i 1989 for at hellige sig undervisning i EDB samme sted. Han har anvendt Linux som bruger siden 1993 og har givet talrige foredrag om fri software og relaterede emner i Danmark, Skåne og Sverige. Fast SSLUG-fotograf siden august 1998. Hjemmeside http://schou.dk/.

5. Vi siger tak for hjælpen

Vi har haft stor glæde af mange SSLUG-medlemmers støtte, rettelser og forslag til forbedringer – bliv

ved med det. Specielt vil vi nævne:

- Forlaget IDG har bidraget med professionel korrekturlæsning til version 1.6 af bogen. Version 1.6 bliver netop udgivet som bog.
- Jacob Laursen har hjulpet meget med at få lavet splittet af "Linux – Friheden til at vælge" på en god måde.
- Lars Madsen og Michael Rasmussen for afsnittet om Samba.
- Claus Sørensen for afsnit om ringe-op-server.
- Henrik Størner for afsnittet om DHCP og afsnittet om at bygge RPM-pakker.
- Lars Lyngby Nielsen for afsnittet om "NT bootloader" og for igennem mange omgange at have rettet en stribe fejl og mangler.
- Jesper Laisen for en gennemgribende revision af hele bogen mht. sprog og specielt kommaer.
- Forlaget GLOBE for en stor sproglig revision.
- Ole Tange for en stor indsats med mange rettelser og for starten af afsnittet om DNS.
- Rolf Larsen (gentagne gange) og Steen Jensen for at have læst bogen fra ende til anden og afleveret en stribe rettelser til hele bogen.
- Afsnittet om TCP/IP og DNS er skrevet af Christian Rasmussen.
- Peter Frederiksen har skrevet afsnittet om RAID.
- Kristian Vilmann har lavet afsnittet om NIS og tilføjet en del om NFS.
- Jesper Krogh har skrevet afsnittet om leafnode.
- Niels Elgaard Larsen har skrevet afsnit om at få en stille maskine.
- Lars K. Schunk har lavet grundig sproglig korrektur – meget fornemt indsats.
- Til kapitlet om Linux-kernen er anvendt dele af rapporten "The Linux Way", skrevet at Hanne Munkholm, Jesper Andresen, Casper Troelsen og Eva D. Jensen.
- Kim Futtrup Petersen har skrevet afsnittet om Appletalk.
- Henrik Christian Grove har bl.a. skrevet afsnittene om CUPS og Big Sister.
- Andre bidragydere er: Allan Jacobsen, Allan Olesen, Anders Damkjær Møller, Anders Ebbesen, Anders Melchiorsen, Anders Pedersen, Anna Jonna Armannsdottir, Asbjørn Laurberg, Birger Langkjer, Brian Christensen, Brian Hemstedt, Carsten Nordstrøm Jensen, Christian Borup, Christian Treldal, Christoffer Hall-Frederiksen, Claus Boje, Claus Hindsgaul, Dan Windekilde, Danni Finne, Donald Axel, Emil S. Hansen, Esben Rugbjerg, Erik Andresen, Erik Søe Sørensen, Finn Dorph-Petersen, Flemming Bjerke, Flemming Mahler Larsen, Frank Damgaard, Gert Holtoft, Gitte Wange, Gunner Poulsen, Hanne Munkholm, Hans Christian Andersen, Henning Schou, Henrik Johansen, Henrik Lundquist, Ingvar Blychert, Jacob I. Christensen, Jacob Sparre Andersen, Jacob Thamsborg, Jakob Hilarius, Jakob Hilmer, Jan Hemmingsen, Jan Michael Due, Jan Vestergaard Larsen, Jens Axelsen, Jens Bygum, Jens Stavnstrup, Jesper Louis Andersen, Jesper Møller, Jesper Norskov Kristensen, Johan Myhre Andersen, Jon Loldrup, Jørgen I. Østergaard, Jørgen Ramskov, Kasper Hauge, Keld Simonsen, Kevin Ilchmann Jørgensen, Kim Hermansen, Kim Larsen, Klaus Hebsgaard, Knud Haugaard Sørensen, Kristian Støchkel, Kåre Løvgren, Lars Scheele Jensen, Magnus Østergaard, Martin Egholm Nielsen, Martin Lykke, Martin Stenderup, Martin René Pedersen, Michael Jacobsen, Michael Lerskov Munk Nielsen, Michael Rasmussen, Michael Skaarup, Mogens Kjær, Morten Christensen, Morten Jensen, Morten Olsen, Niels Damgaard, Niels Kristian Bech Jensen, Niels Jalling, Niels Rasmussen, Niels Sandmann, Olav Pedersen, Ole Tange, Owen Brotherwood, Palle Arentoft, Palle E. Nielsen, Peter Andersen, Peter B. Kvan, Peter Christensen, Peter Rasmussen, Peter Seidler, Poul Petersen, Poul-Erik Hansen, Rasmus Laursen, Rask Ingemann Lambertsen, René Seindal, Roy Sigurd Karlsbakk, Rune Tønnesen, Stefan Klukowski, Svend Erik Venstrup-Nielsen, Søren Sjørup, Thomas Mørch, Thomas Petersen, Thorbjørn Ravn Andersen, Tue Hansen.

Du kan i Appendiks A finde en liste over alle de revisioner, som bogen har været igennem.

Hvis du har ord du ikke forstår, så kan http://www.whatis.com være interessant. Her kan du slå mange computerord op, dog kun på engelsk.

6. Typografi

Vi vil afslutte indledningen med at nævne den anvendte typografi.

- Navne på filer og kataloger er skrevet som `foo.bar`
- Kommandoer, du udfører ved at taste, skrives som **help**
- Der er flere steder i bogen, hvor vi viser, hvad brugeren taster, og hvad Linux svarer. Det vil se ud som:

  ```
  [tyge@hven ~]$ Det her taster brugeren
  Det her svarer Linux
  ```

- Der er tilsvarende flere steder i bogen hvor vi viser hvad systemadministratoren (root) taster, og hvad Linux svarer. Det vil se ud som:

  ```
  hven# Dette taster systemadministratoren
  Dette svarer Linux.
  ```

  Det vigtige her er at kommandofortolkeren bruger nummertegnet (#) til at markere at man har systemadministratorrettigheder.

Kapitel 1. Hvor finder jeg opsætningsfilerne?

I dette afsnit vil vi beskrive, hvilke opsætningsfiler der er relevante at kende i et Linux-system. Det er normalt at gemme opsætningsfiler i `/etc` (det er meget muligt at katalog-navnet er en forkortelse af *editable text configurations*). Du kan have glæde af at læse de enkelte filer igennem, når du er blevet lidt erfaren i at styre Linux. Vi vil i dette afsnit give dig et overblik over de vigtigste opsætningsfiler, så du selv kan forstå, hvad der foregår.

Opsætningsfiler i `/etc` er vitale for dit Linux-system, så vær derfor forsigtig, når du retter i filerne. Vi vil foreslå, at du *altid* først laver en kopi af den fil, du vil ændre, dvs.

```
[root@linus /etc]# cp hosts hosts.orig
[root@linus /etc]# vi hosts
```

Hvis der så skulle gå noget galt, kan du altid fortryde og vende tilbage til den oprindelige udgave af filen med:

```
[root@linus /etc]# cp hosts.orig hosts
```

I eksemplet anvendte vi **vi** som editor, men du kan naturligvis selv vælge hvordan du vil redigere filerne.

1.1. /etc/passwd

Filen `/etc/passwd` indeholder en linje per bruger, som har login på maskinen. Brugeren "root" har også en linje her. Kun hvis man kører NIS (tidligere kendt som "yellow pages"), kan man lade ekstra brugere være styret af andre servere.

Et eksempel på en linje af adgangskodefilen, passwd, er:

```
tyge:x:500:501::/home/tyge:/bin/bash
```

Der står her, at brugeren "tyge" er bruger nummer 500 på maskinen (ikke, at der faktisk er 500). Brugeren "tyge" er med i gruppe nummer 501, hvilket er styret af `/etc/group`. Brugeren "tyge" har hjemmekatalog i `/home/tyge` og starter med **/bin/bash** som login-shell. Det lille x betyder, at man kører med skygge-adgangskoder (eng. shadow passwords), dvs. at adgangskoder står i `/etc/shadow`, og denne fil kan ikke læses af andre end systemadministratoren – dvs. skygge-adgangskoder er et ekstra niveau af sikring af adgangskoderne på systemet.

Hvis en person ikke skal have lov til at logge ind på maskinen, så kan du rette i `/etc/passwd` og indsætte en stjerne (dvs. *) foran den krypterede adgangskode.

1.2. /etc/shadow

Formatet af `/etc/shadow`-filen følger formatet for `/etc/passwd`-filen med en linje per bruger. Et eksempel er:

```
tyge:$1$/hbYueDa$46ggNKFugoDABWOJZ3xvz0:10784:0:99999:7:::
```

For brugeren "tyge" står der de krypterede adgangskoder. For Red Hat er det længere end for SuSE, idet Red Hat anvender en bedre krypteringsmåde. Derefter vises antal dage siden 1/1-1970 for sidste ændring af adgangskoderne. Næste felt viser antal dage siden sidste ændring af adgangskode. Derefter antal dage mellem at der sendes en advarsel til brugeren om kontoudløb og det faktiske kontoudløb (her 99999 dage dvs. det anvendes ikke). Sidst anvendte felt (7) er antal dage siden 1/1-1970 til kontoen bliver lukket. Endelig er der et felt som ikke anvendes.

Hvis din maskine ikke kører skygge-adgangskoder, og vil du dette, så kør **/usr/sbin/pwconv** som brugeren root. Tilsvarende findes **/usr/sbin/pwunconv** til at få adgangskoderne gemt i selve passwd-filen og ikke shadow-filen.

1.3. /etc/group

Hver person er med i én eller flere grupper, og det bruger man til at udskille, hvem der har adgang til hvilke dataområder. Kommandoen **chmod g+w fil** vil f.eks. gøre, at andre brugere, som er i samme gruppe som ejeren af filen, kan skrive i `fil`.

Hver linje i `/etc/group` svarer til en gruppe. Et eksempel på en gruppe kan være:

```
web:x:1000:tyge,niels,svend
```

Dvs. gruppen "web" har gruppe-ID nummer 1000, og "x" betyder, at en eventuel adgangkode for gruppen (anvendes ved skift til ny gruppe med **newgrp**) findes i shadow-filen `/etc/gshadow`. Hvis "x" blev fjernet, anvendes gruppe-adgangskoder ikke. Til sidst på linjen står der de login-navne (fra passwd-filen) som er medlemmer af gruppen.

1.4. /etc/gshadow

Filen `/etc/gshadow` følger nøje `/etc/group` med en linje pr. gruppe. Ideen er ligesom med `/etc/shadow` at gemme gruppe-information for brugerne.

1.5. /etc/fstab

Filen `/etc/fstab` indeholder information om de disk-partitioner, som skal eller kan monteres ved systemopstart. Et eksempel kan være:

```
/dev/hdb2   /            ext2    defaults        1 1
/dev/hda1   /dosc        vfat    defaults        0 0
/dev/hdb3   /home        ext2    defaults        1 2
/dev/hdb1   swap         swap    defaults        0 0
/dev/fd0    /mnt/floppy  vfat    noauto,user,rw  0 0
/dev/cdrom  /mnt/cdrom   iso9660 noauto,user,ro  0 0
none        /proc        proc    defaults        0 0
none        /dev/pts     devpts  mode=0622       0 0
```

De to sidste linjer er noget Red Hat automatisk selv har sat op. Det ser vi bort fra. Derefter kan vi se, at der mindst er to harddiske i maskinen – hda og hdb. Den sekundære harddisk på første IDE-controller, dvs. hdb har mindst tre partitioner. Swap-partitionen er sat op på den første partition `/dev/hdb1`, og de to næste partitioner på hdb, dvs. `/dev/hdb2` og `/dev/hdb3`, er af typen ext2 (Linux-type), og de monteres i rækkefølge som vist yderst til højre, dvs. `/dev/hdb2` (roden) før `/dev/hdb3` (brugerens hjemmekatalog i `/home`). De to sidste linjer viser, at en almindelig bruger (user) kan montere cd-rom'er og disketter, men at det ikke gøres automatisk (noauto) under opstart. cd-rom'en vil efter **mount /mnt/cdrom** kunne findes under `/mnt/cdrom`. Brug i øvrigt **man mount**.

Det bør nok nævnes, at brugere med scsi-diske bør bruge UUID i stedet for device-navne til at identificere devices. Ellers opfører hele filsystemet sig underligt, hvis én af diskene på scsi-controlleren går ned. Se **man fstab**.

1.6. /etc/hosts

Hosts-filen anvendes ofte på maskiner, som ikke har navneserver (eng. DNS-server eller name server) til rådighed. Man skriver en IP-adresse, det fulde netværksnavn og et eventuelt alias. Et eksempel kan være:

```
127.0.0.1       localhost.localdomain localhost
192.168.0.1     linus.intranet       linus
192.168.0.2     alan.intranet        alan
192.168.0.3     richard.intranet     richard
192.168.0.4     eric.intranet        eric
```

IP-adressen 127.0.0.1 er en speciel "loopback"-adresse, som ikke anvendes til andet end, at Linux-kernen hurtigt kan kommunikere med sig selv. De fire maskiner linus, alan, richard og eric har fået IP-adresser i 192.168.0.*, som er et lukket net – dvs. de adresser er garanteret ikke-eksisterende på internettet.

Skal du selv sætte et hjemmedomæne op, hvor maskinerne ikke skal være kendte på internettet, og du vil være sikker på at dit hjemmedomænenavn ikke er kendt på internettet, så brug f.eks. domænenavnet

".hjemme" eller som her ".intranet", så linus-maskinen bliver til "linus.intranet" – så får du aldrig overlap med en anden maskine. Skal maskinerne være kendt på internettet, så skal du ansøge om et domæne, sætte DNS op, og erstatte ".intranet" med ".MITDANSKEDOMAENE.dk", eller hvad du nu vælger.

Uden DNS (navneserver – se Afsnit 6.9) bruger systemet automatisk hosts-filen ved ssh, telnet, ftp, ping osv.

1.7. /etc/resolv.conf

Resolver-filen viser, hvilken maskine der er navneserver, så man kan slå en vilkårlig adresse op på internettet. Et eksempel kan være

```
search intranet
nameserver 129.142.6.64
nameserver 129.142.6.65
```

Første linje (search) viser domæne-navnet for netværket. De to næste linjer er to navneservere (danpost-maskiner). Laver du internet-opkobling til din egen ISP, skal du have nogle andre IP-adresser end de viste.

NB: Brug *IKKE* ekstra nuller til at udfylde de cifre, der ikke bruges. I det ovenstående må du ikke skrive

```
search intranet
nameserver 129.142.006.064
nameserver 129.142.006.065
```

1.8. /etc/inetd.conf

Denne fil er vigtig for din netværks-sikkerhed, idet den viser en stribe af de services, som Linux-maskinen tilbyder netværket. Du vil f.eks. finde disse linjer:

```
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
```

som viser, at du modtager ftp- og telnet-login. Hvis du kan undvære disse services, så udkommentér dem ved at sætte et # foran dem. Næste gang du genstarter maskinen (eller inet-dæmonen), kan man ikke logge på via **telnet**. Prøv **man inetd** for at læse mere.

1.9. /etc/issue.net og /etc/issue

`/etc/issue` og `/etc/issue.net` kan være en god måde for en cracker at finde ud af hvilket system og hvilken version af Linux man kører. `/etc/issue` vises på skærmen lige op over login-prompten og `/etc/issue.net` vises blandt andet når man logger på systemet via **telnet**. Derfor fjern al tekst i begge filer og sæt havelåger foran dette i `/etc/rc.local` (SuSE `/etc/rc.d/boot.local`):

```
# This will overwrite /etc/issue at every boot. So, make any changes you
# want to make to /etc/issue here or you will lose them when you reboot.
# echo "" > /etc/issue
# echo "$R" >> /etc/issue
# echo "Kernel $(uname -r) on $a $SMP$(uname -m)" >> /etc/issue
# cp -f /etc/issue /etc/issue.net
# echo >> /etc/issue
fi
```

1.10. /etc/rc.d/

I mappen `/etc/rc.d/` findes styringen af hvilke programmer, der automatisk startes op, når maskinen startes. Der er lidt forskel mellem Red Hat og SuSE, men begge har samme struktur, kaldes SysV (udtales system fem), i modsætning til BSD.

1.10.1. Red Hat-opstartsfiler

Red Hat har følgende indhold af `/etc/rc.d/`

```
init.d  rc.local    rc0.d  rc2.d  rc4.d  rc6.d
rc      rc.sysinit  rc1.d  rc3.d  rc5.d
```

hvor `rc0.d` op til `rc6.d` er mapper med links for hvert af de syv runlevels. Links laves til `init.d`, der indeholder script-filer, som starter programmerne. Se i øvrigt Afsnit 2.5.

Filen `rc.sysinit` ændrer man oftest ikke, idet den styrer opstart af tastatur, tjek af filsystemer, montering af swap, samt andet basalt.

I filen `rc.local` (SuSE `/etc/rc.d/boot.local`) kan man selv tilføje kommandoer, som skal startes efter de almindelige programmer.

Alle de ovenstående filer kan læses af alle på et Red Hat-system, mens SuSE har valgt, at kun "root" kan læse de filer.

1.10.2. SuSE-opstartsfiler

SuSE kan have følgende indhold af `/etc/rc.d/`

```
README      cron        kbd         random      rcS.d       skeleton
apmd        dummy       kerneld     rc          reboot      ssh
at          gpm         lpd         rc0.d       route       syslog
autofs      halt        network     rc1.d       routed      xdm
boot        halt.local  nfs         rc2.d       rpc
boot.d      inetd       nfsserver   rc3.d       rwhod
boot.local  init.d      pcmcia      rc4.d       sendmail
boot.setup  inn         pcnfsd      rc5.d       serial
cdb         ipfwadm     powerfail   rc6.d       single
```

Som det ses, er der mapper rcX.d (hvor X går fra 0 til 6), med links for hvert af de syv runlevels, ligesom Red Hat. I modsætning til Red Hat er der i SuSE intet `init.d`-katalog, og script-filerne, der linkes til, findes direkte i `/etc/rc.d/`. Indholdet af hver af opstartsfilerne, f.eks. `autofs`, er kommandoer til at starte, stoppe, tjekke og genstarte den pågældende service. Man skriver således **/etc/rc.d/autofs stop** for at stoppe autofs og tilsvarende **/etc/rc.d/autofs start** for at starte den igen.

1.11. /etc/sysconfig/

I Red Hat er mappen `/etc/sysconfig/` vigtig. Se f.eks.
http://www.comptechdoc.org/os/linux/howlinuxworks/linux_hlsysconfig.html
(http://www.comptechdoc.org/os/linux/howlinuxworks/linux_hlsysconfig.html) for en nærmere gennemgang af de enkelte filtræer i /etc/sysconfig.

```
apmd   console  keyboard  network          sendmail   static-routes
clock  init     mouse     network-scripts  soundcard
```

For Red Hat vil alle filer i dette katalog indeholde opsætningsparametre og er koblet til filerne i `/etc/rc.d/init.d`. F.eks. vil `mouse` indeholde musetype.

Et skridt længere nede i filtræet finder du mappen `/etc/sysconfig/network-scripts`, som f.eks. kan indeholde:

```
chat-ppp0   ifcfg-ppp0   ifdown-ppp    ifup-plip    network-functions
chat-ppp1   ifcfg-ppp1   ifdown-sl     ifup-post
chat-ppp2   ifcfg-ppp2   ifup          ifup-ppp
ifcfg-eth0  ifdown       ifup-aliases  ifup-routes
ifcfg-lo    ifdown-post  ifup-ipx      ifup-sl
```

Her er alle script-filer til at starte PPP-forbindelser og netkort op og tilsvarende lukke ned. Har du en meget underlig opkobling til internettet, ender du med at skulle rette i filer her.

1.11.1. /etc/sysconfig/clock

Hvis man har Windows installeret på samme maskine kan man ikke køre med GMT som tiden i BIOS. Windows kan ikke dette. Det er ellers smart da Linux så automatisk styrer sommertid perfekt.

I filen `/etc/sysconfig/clock` sætter man tidsstyringen op. Der er tre parametre:

- `zone="Europe/Copenhagen"` – sætter tidszonen til dansk zone.
- `utc=false` – UTC sættes til "true", hvis du vil have BIOS-tiden til at opfattes som UTC-tid (GMT-tid). Med "true" fungerer håndtering af sommertid perfekt. Hvis du har Windows installeret, så skal du køre med "false", da Windows ellers vil køre med en eller to timers forskydelse af tiden. Windows kan ikke køre med "utc=true".
- `arc=false` – Dette anvendes kun på Linux til Alpha. Se mere på http://www.comptechdoc.org/os/linux/howlinuxworks/linux_hlsysconfig.html (http://www.comptechdoc.org/os/linux/howlinuxworks/linux_hlsysconfig.html)

1.12. /etc/crontab

Crontab-systemet og filstyring er forklaret i Afsnit 2.7.

1.13. /etc/sendmail.cf

Sendmail er det gode gamle postprogram til Linux, som desværre tit viser sig at have sikkerhedsfejl. Opsætningsfilen til sendmail er notorisk svær, og du bør ikke rette ret meget i din. Du kan dog med fordel rette feltet DS til

```
DSsmtp.ISP.dk
```

hvis du sender post til din internet-udbyder, og denne har en hurtig "mailforwarder" – ofte kaldet en SMTP-maskine, i eksemplet kaldet "smtp.ISP.dk".

1.14. /etc/aliases

Hvis du har brug for at have nem adgang til at lave post-aliaser, så breve til <web@sslug.dk> automatisk bliver videresendt (eng. forwarded) til <tyge@sslug.dk>, mens f.eks. alle breve til "root" også skal ende hos "tyge", så kan du i `/etc/aliases` nemt anføre dette.

```
web: tyge
root: tyge
```

Efter du har redigeret filen, skal du køre **newaliases**-kommandoen.

1.15. /etc/printcap

For hver printer vil programmet **printtool** i Red Hat tilføje et afsnit til `/etc/printcap` med definition af printeren, såsom

```
##PRINTTOOL3## LOCAL POSTSCRIPT 600x600 a4 {} PostScript Default 1
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/lp/filter:
```

Her er "sd" spool-directory. Printeren sidder på `/dev/lp0`, og der printes med filter som vist med "if".

Det skal nævnes at flere Linux-distributioner er ved at gå væk fra denne måde at sætte printere op på. Dels er CUPS-projektet i fuld gang og LPRng vinder også indpas (bl.a. i Red Hat 7.0).

1.16. /etc/conf.modules

Denne fil indeholder opsætning og parametre til de moduler, som kan læses ind og ud af systemet uden at genstarte. Der er mere forklaring i Afsnit 4.7.

1.17. /etc/securetty

I `/etc/securetty` anføres alle de terminaler, hvorfra root-brugeren kan logge ind. Filen ser ofte således ud

```
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
```

`tty1-tty8` svarer til de terminal-vinduer som er på selve maskinen. Det er således ikke tilladt for root at logge ind via telnet på maskinen, og det bør *ikke* tillades.

1.18. /etc/exports

Hvis din maskine skal køre NFS, dvs. kunne være disk-server for andre Unix-maskiner, skal du i `/etc/exports` skrive, hvilke mapper du vil lade andre montere. For at det virker, skal du have "portmap" og "nfs" kørende.

Desuden kan du anvende **man exports** for at lære syntaks af `/etc/exports`. Som eksempel viser vi hvordan du lader alle maskiner læse (ikke skrive) fra dit cd-rom-drev, som du her har monteret fast på `/mnt/cdrom`. Derudover lader vi maskiner i domænet kongeh.dk montere hele `/usr/local/` med både læse- og skrive-rettigheder, så alle maskinerne kan lægge nyt programmel ind her, og programmellet anvendes så på alle klient-maskinerne. Husk, at du måske også skal tænke over sikkerheden, når du distribuerer data over netværket. Giv kun adgang til data til de personer, som skal have den adgang.

```
/mnt/cdrom      *(r)
/usr/local      *.kongeh.dk(rw)
```

1.19. /etc/X11/XF86Config - /etc/XF86Config

I Red Hat er den basale X Window-opsætningsfil `XF86Config` lagt i `/etc/X11/XF86Config`. I SuSE er den i `/etc/XF86Config` for XFree86 version 3.x og i `/etc/X11/XF86Config` for version 4.x. Syntaksen findes ved at skrive: **man XF86Config**.

1.20. Versionsoplysninger

1.20.1. Finde versionsoplysninger for et Debian-system

Hvis du er logget ind på et Debian-system og har brug for at finde ud af hvilken udgave af Debian det er, så kan du se det med kommandoen:

```
[tyge@hven ~]$ cat /etc/debian_version
```

Har man brug for at finde oplysninger om versionnummeret for en enkelt pakke kan man bruge kommandoen **dpkg --status pakkenavn**. Dette giver en række status-oplysninger, blandt andet et felt der hedder "Version" som indenholder versionsnummeret for den installerede pakke.

1.20.2. Finde versionsoplysninger for et Red Hat-system

Hvis du er logget ind på et Red Hat-system og har brug for at finde ud af hvilken udgave af Red Hat det er, så kan du se det med kommandoen:

```
[tyge@hven ~]$ cat /etc/redhat-release
```

Har man brug for at finde oplysninger om versionnummeret for en enkelt pakke kan man bruge kommandoen **rpm --query pakkenavn**. Dette giver pakkens navn og versionsnummer. Hvis der er installeret flere udgaver af pakken, vil alle de forskellige udgavers versionsnumre blive vist. Med **rpm --query --info pakkenavn** kan man få flere oplysninger om pakken.

1.21. Videre læsning

Der er en masse godt om opsætningsfilerne i `/etc` at læse på http://ctdp.tripod.com/os/linux/howlinuxworks/index.html.

En anden god kilde er O'Reilly-bogen *Linux Device Drivers* af Alessandro Rubini & Jonathan Corbet, som kan findes på http://www.xml.com/ldd/chapter/book/.

Kapitel 2. Systemadministration

I dette kapitel ser vi nærmere på nogle af de værktøjer og opgaver som systemadministratoren kommer ud for.

2.1. Tilføje brugere

Ud over de grafiske programmer til at håndtere brugere, så kan det være praktisk at kunne addere brugere til systemet via en konsol. Skal man lave en konto til brugeren Peter Toft med brugernavn "pto", hjemmekatalog `/home/pto` og denne bruger skal være medlem af gruppen `linuxbog`, så køres følgende:

```
[root@linus ~]# /usr/sbin/useradd -c 'Peter Toft \
 pto@sslug.dk' -d /home/pto -g linuxbog -m -n pto
```

Her er tegnet **** anvendt til at kunne fortsætte kommandoen på næste linje.

Peter Tofts e-post-adresse er også tilføjet for at have flere informationer gemt i `/etc/passwd`. Man kunne også have tilføjet telefon-nummer. *HUSK* at give brugeren en adgangskode:

```
[root@linus ~]# passwd pto
```

2.2. Erstatning af tekst i flere filer

Ofte kommer man ud for at skulle erstatte tekst i filer. Nogle gange er det nemmest at starte en emacs- eller vi-editor op og blot rette filen til. Andre gange skal man lave samme rettelse i mange filer, og så er det meget nyttigt at kunne en smule shell og Perl. Antag f.eks. at man i alle .txt-filer skal erstatte "Windows" med "Linux". Det kan gøres med følgende lille script som kan udføres direkte i kommandofortolkeren.

```
[tyge@hven ~]$ for I in *.txt
 do
   perl -pi.bak -e 's/Windows/Linux/g' $I
 done
```

Denne operation efterlader en .bak-fil for hver fil du laver søg-og-erstat i (bare for en sikkerheds skyld).

2.3. Har du glemt din adgangskode?

Hvis du har dummet dig meget og glemt din root-adgangskode, så kan du få sat en ny ind.

I så fald booter du med "linux init=/bin/sh" . Så starter den en shell som proces nr. 1. Herefter køres:

```
[root@linus ~]# mount -n -o remount,rw /      # re-mount / read-write
[root@linus ~]# passwd                          # skift adgangskode
[root@linus ~]# mount -n -o remount,ro /      # remount / read-only
[root@linus ~]# exec /sbin/init 3              # start init i runlevel 3
```

Dette lyder jo som et stort sikkerhedshul, og det er det også, hvis man har fysisk adgang til maskinen. Du kan dog undgå dette ved at tilføje linjen "password=MIN_ADGANGSKODE" til din `/etc/lilo.conf`, men så skal du kunne huske root-adgangskoden og den adgangskode, du har sat i `/etc/lilo.conf`.

2.4. Opstart af Linux

2.4.1. Samtidig installation af Windows og Linux

Windows og Linux kan sagtens være installeret på samme computer. Ved at bruge et opstartsprogram kan man, når man starter computeren, vælge om den skal starte i Linux eller i Windows. Det er samtidig også muligt at have flere forskellige udgaver af Linux og/eller Windows installeret på samme maskine. Man kan dog kun køre én ad gangen (med mindre man bruger VMware).

Når man starter en computer, der kun har Windows installeret indlæses der først et lille startprogram fra *Master Boot Record* (MBR), der ligger først på den første harddisk (`/dev/hda`) sammen med beskrivelsen af hvordan harddisken er opdelt (partitionstabellen). Startprogrammet fra MBR indlæser et egentligt startprogram fra den første del af den første harddisk (`/dev/hda1`) og overfører kontrollen til det. Hvis computeren kører Windows 95 eller 98 vil DOS blive startet, og DOS vil så automatisk starte Windows. Kører computeren Windows NT eller 2000 giver startprogrammet et valg mellem forskellige opsætninger af styresystemet.

Microsoft anbefaler, at C-drevet under Windows NT er formateret som et FAT-filsystem. Det giver flere muligheder for fejlfinding end hvis man bruger det nyere NTFS-filsystem. En praktisk sideeffekt af at bruge et FAT-filsystem er at Linux ikke har problemer med at læse fra og skrive til C-drevet. En anden fordel er at hvis man får en maskine hvor Windows fylder hele harddisken, så kan man bruge DOS-programmet **fips.exe** til at frigøre en del af Windows-partitionen til brug for Linux. Det er derimod ikke muligt at omstrukturere NTFS-filsystemer og det vil derfor være nødvendigt at slette og geninstallere Windows, hvis man har en harddisk der er fyldt op af et enkelt NTFS-filsystem.

Når man starter en maskine, der kun har Linux installeret, startes *LInux LOader* (LILO), der ligesom Windows' startprogram ligger i MBR. Der findes også en række andre startprogrammer til Linux, men dem vil vi ignorere i denne omgang. LILO giver brugeren mulighed for at vælge hvilken udgave af kernen, der skal startes. Typisk vil man kun have én udgave liggende, men det giver let mulighed for at afprøve en nyoversat kerne. Hvis den ikke virker kan man bare genstarte, og vælge den gamle og velfungerende kerne i stedet for. Der er mere om LILO i bogen "Linux - Friheden til at vælge installation".

Hvis det er Windows' startprogram der ligger i MBR, skal Windows-opsætningen ændres så det også kan starte Linux. Hvis det er Linux' startprogram (LILO) der ligger i MBR skal dets opsætning tilsvarende ændres, så det kan starte Windows. De følgende afsnit vil gennemgå begge muligheder.

Bemærk at når Windows (gen)installeres, bliver MBR harddisken overskrevet! Hvis LILO er installeret i MBR vil den derfor blive slettet, og man kan således ikke starte Linux. Det er derfor meget vigtigt at lave en startdiskette (eng. boot disk) til Linux, før man går i gang med at installere Windows på en maskine der allerede er Linux på. Det kan også være fornuftigt at lave en startdiskette før man begynder at eksperimentere med LILO. Så kan man altid redde sig ud af en uheldig opsætning af LILO.

Startdisketten kan fremstilles efter installationen med Linux-kommandoen **mkbootdisk**. Husk at have en formateret diskette parat. Det er vigtigt, at disketten er 100% fri for fysiske fejl!

Eksempel:

```
[root@linus /root]# fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
[root@linus /root]# mkbootdisk `uname --release`
Insert a disk in /dev/fd0. Any information on the disk will be lost.
Press <Enter> to continue or ^C to abort:
```

Det er også en smart idé at lave en startdiskette til Windows før du går igang med LILO-eksperimenter. I Windows 95 og 98 kan man køre **format a: /s** på kommandolinjen. I Windows NT og 2000 skal disketten formateres, da det medfører at de nødvendige systemfiler bliver kopieret ud på disketten.

2.4.1.1. Linux styrer opstarten

Den letteste løsning er at lade Linux styre opstarten af maskinen. Det ser også mest blæret ud :-)

Under installationen af Red Hat gøres dette ved at vælge at lægge LILO i MBR. Under Red Hat-installationen vil et eventuelt eksisterende Windows-system automatisk blive tilføjet som en valgmulighed i LILO.

Har du allerede installeret Linux, kan et Windows-system føjes til LILO's menu ved at føje følgende linjer til `/etc/lilo.conf`:

```
other=/dev/hda1
        label=dos
        table=/dev/hda
```

Husk, at hver gang opsætningen af LILO ændres skal du køre **/sbin/lilo** som root. Det vil så opdatere informationerne, der ligger i MBR.

Når LILO starter får du en oversigt over mulighederne. Typisk kan du vælge mellem 'linux' og 'dos'. Når du vælger 'dos', kalder LILO Windows' "OS Loader". Med ældre udgaver af LILO er det nødvendigt at trykke på tabulatortasten for at blive præsenteret for valgmulighederne.

2.4.1.2. Windows styrer opstarten

Dette afsnit gælder ikke for Windows 95 og Windows 98. Hvis du bruger et af disse to systemer skal du lade Linux styre opstarten som beskrevet i forrige afsnit. Opsætningen af Windows NT's og 2000's startprogram, "OS Loader" findes i filen `C:\Boot.ini`.

Når Windows styrer opstarten skal LILO installeres i *boot*-sektoren på rod- eller `/boot`-filsystemet. For at kunne starte Linux fra Windows skal *boot*-sektoren kopieres over i en fil, overføres til C-drevet og føjes til `C:\Boot.ini`.

Først skal bootsektoren gemmes som en fil, for eksempel `bootsect.lnx`. I `/etc/lilo.conf` er der en linje "boot=...". Den fortæller hvor LILO's *boot*-sektor findes.

Eksempel:

```
[root@linus /root]# grep "^boot" /etc/lilo.conf
boot=/dev/hda3
```

Vi kan se at i eksemplet er LILO placeret i *boot*-sektoren på `/dev/hda3`. Vi kan nu kopiere *boot*-sektoren ud i filen `bootsect.lnx` med kommandoen:

```
[root@linus /root]#  dd if=/dev/hda3
of=/bootsect.lnx bs=512 count=1
```

(kopiér den første blok à 512 bytes fra `/dev/hda3` til `bootsect.lnx`)

`bootsect.lnx` skal nu kopieres til Windows' C-drev og den følgende linje skal føjes til `c:\Boot.ini`:

```
C:\bootsect.lnx="Linux - Red Hat"
```

Denne linjer betyder at NT's startprogram kalder LILO - i filen `C:\bootsect.lnx` - hvis du vælger Linux i NT's opstartsmenu. Dette trick kan også laves med flere *boot*-sektorer.

Hvis C-drevet er i FAT-format, kan du montere det fra Linux (det er de to første linjer nedenfor). De to sidste linjer kopierer *boot*-sektorfilen til Windows-drevet, og føjer den til "OS Loaderens" opsætningsfil. Bemærk at >> *skal* bruges, da > alene vil overskrive filen og ikke føje til den:

```
[root@linus /root]#  mkdir /dos
[root@linus /root]#  mount -t vfat /dev/hda1 /dos
[root@linus /root]#  mv /bootsect.lnx /dos/bootsect.lnx
[root@linus /root]#  echo 'C:\\bootsect.lnx="Linux - Red Hat 7.0"' >> /dos/BOOT.INI
```

Dette virker ikke hvis dit Windows-drev er NTFS-formateret. Har du adgang til en FTP-server, kan du kopiere `bootsect.lnx` til en FTP-server, genstarte maskinen i Windows og hente filen igen. Et alternativ er at kopiere filen ud på en diskette med brug af **mcopy**, som vist her:

```
[root@linus /root]# mcopy /bootsect.lnx a:
```

Genstart derefter din maskine.

Eksempel på en komplet `C:\Boot.ini`:

```
[boot loader]
timeout=5
default=multi(0)disk(0)rdisk(0)partition(3)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(3)\winnt="NT Workstation 4.00 SP3"
multi(0)disk(0)rdisk(0)partition(3)\winnt="NT Workstation 4.00 SP3 [VGA
mode]" /basevideo /sos
C:\="Windows 95"
```

Denne fil er Windows' startprograms opsætningsfil. Filen skal nu tilrettes, så Linux også bliver en valgmulighed.

Skriv i DOS/NT:

```
C:
CD \
ATTRIB -s -r BOOT.INI
EDIT BOOT.INI
```

Tilføj følgende linje:

```
C:\bootsect.lnx="Linux - Red Hat 7.0"
```

Sæt derefter filens attributter igen:

```
ATTRIB +s +r BOOT.INI
```

Nu kan Windows' "OS Loader" kalde LILO og dermed starte Linux. Så du *kan* opnå valgfrihed - også med Windows! Husk, at hvis du ændrer opsætningen af LILO, skal du huske at fremstille en ny `bootsect.lnx`-fil og kopiere den over på Windows-filsystemet.

Hvis du vil vide meget mere om MBR-placering, opstart og diskopdeling, kan det anbefales at læse HOWTO-guiden om LILO. På Red Hat Linux-cd-rom'en hedder filen `/doc/HOWTO/mini/LILO`.

2.5. Hvad er et "runlevel"?

Et runlevel er den tilstand, som systemet kører i. Computeren kan køre i tilstande såsom "opstarter systemet", "enkeltbrugersystem", "flerbrugersystem", "genstarter systemet" og "slukker computeren". Hvis man f.eks. vil kopiere `/home`-kataloget over på en større harddisk fordi ens brugere har for mange MP3-filer liggende, så er det nødvendigt at sikre, at der ikke er nogen brugere, der har filer åbne, mens man kopierer, idet der så kan opstå en fejl i kopieringen. Det kan man sikre sig ved at skifte til enkeltbruger-tilstand, der svarer lidt til fejlsikret tilstand i Windows. Når man er færdig med systemarbejdet, hopper man tilbage til "flerbrugersystem" og folk har igen adgang til computeren, og vil ikke bemærke at deres filer pludselig ligger på et andet drev.

Computeren kører en masse services for brugerne. Det kan f.eks. være netværksforbindelse (network) og forskellige netværksrelaterede services som f.eks. e-post (sendmail/routed/named/snmpd/portmap), adgang til filsystemer på andre computere (nfs), mulighed for at udskrive til printer (lpd) og lydkort (sound). Men mens Linux er ved at starte, har man ikke brug for at have adgang til printeren, så printer-servicen startes først når systemet går ind i "flerbrugersystem"-tilstanden. Når man er ved at genstarte eller slukke computeren, slukker man igen for printer-servicen.

Når computeren omstilles fra én tilstand til en anden tilstand, udføres en række scripts, der bestemmer hvilke services der skal tændes og slukkes. Hvilke scripts der kører ved skift til f.eks. runlevel 3 (flerbrugersystem uden grafisk login), bestemmes af, hvilke scripts der er placeret i kataloget `/etc/rc.d/rc3.d` (Corel, Debian og nu også Red Hat har tilsvarende `/etc/rc3.d`).

```
[tyge@hven ~]$ ls /etc/rc.d/rc3.d
K30mcserv  K40snmpd   K55routed  K75gated  K80random   K95nfsfs
K96pcmcia  K97network S01kerneld S30syslog S40atd      S40crond
```

```
S40portmap S50inet    S55named   S60lpd    S75keytable S80sendmail
S85sound   S99local
```

Disse filer svarer til services der skal startes (dem der starter med S) eller dræbes (starter med K for Kill) når systemets tilstand ændres til tilstand 3 (Flerbrugersystem). Tallet bestemmer rækkefølgen; `S01kerneld` startes altså før `S30syslog`, mens `K96pcmcia`-servicen dræbes efter `K30mcserv`. Hver fil i dette katalog er i virkeligheden et link til et script i `/etc/rc.d/init.d/`, og "S80sendmail" betyder i virkeligheden at "/etc/rc.d/init.d/sendmail start" udføres, hvilket starter sendmail-dæmonen op.

Det er ikke svært at finde ud af, hvilket runlevel din computer er i. Nedenfor er vist hvordan.

```
[root@linus /root]# /sbin/runlevel
N 3
```

3-tallet viser dig, at din pc er i runlevel 3. Det er muligt at skifte runlevel uden at lukke computeren ned, men N'et viser, at der intet tidligere runlevel er, dvs. computeren har været i runlevel 3, siden den blev tændt. Du skifter runlevel med kommandoen `telinit`. Du bør altid synkronisere dine harddiske (tømme filsystemernes buffere) inden du skifter runlevel, dvs. kør **sync** et par gange, lige inden du skifter runlevel med **telinit**.

Runlevel 3 er det mest almindelige for servere, mens grafiske arbejdsstationer og kontormaskiner typisk vil anvende runlevel 5. Skemaet nedenfor viser betydningen af de forskellige runlevels.

Tabel 2-1. Runlevels

Runlevel	Betydning
0	Lukker systemet ned (**`shutdown -h now`**)
1	Enkeltbrugertilstand (eng. »single-user mode«)
2	Flerbrugertilstand typisk ret mange netværksservices startet
3	Flerbrugertilstand uden grafisk login
4	Bruges ikke
5	Flerbrugertilstand med grafisk login
6	Genstart systemet (**`reboot`**)

Som det fremgår af skemaet, er runlevel 6 det samme som at genstarte systemet. Dvs. at når systemadministratoren (root) genstarter systemet ved at bruge kommandoen **reboot** (genstart), skiftes der til runlevel 6.

Enkeltbrugertilstand er et meget nyttigt runlevel. Hvis det sker, at du kommer til at konfigurere et eller andet forkert en dag - ja, det vil ske på et eller andet tidspunkt! - og din pc låser under opstarten, kan du starte den op i enkeltbrugertilstand og foretage diverse rettelser.

I filen `/etc/inittab` angives, hvilket runlevel der er det forvalgte runlevel. Ønsker man f.eks. at maskinen starter i runlevel 5 og ikke 3, så skal man ændre linjen

```
id:3:initdefault:
```

til

```
id:5:initdefault:
```

Se også Afsnit 1.10 for detaljer om forskelle mellem de forskellige Linux-distributioner.

2.6. Nedlukning af Linux

En Linux-maskine er, som du nok allerede har forstået, et meget stort system med mange muligheder. Du skal også være meget opmærksom på, at man (ligesom med en Windows-maskine) ikke bare må slukke for en Linux-maskine - den skal helst lukkes pænt ned.

Som vi viste i Afsnit 2.5, er runlevel 0 det samme som at lukke ned. En generel, men lidt besværlig måde at lukke en Unix-maskine på, er at skifte til root og skrive **`init 0`** eller **`telinit 0`**

```
[tyge@hven ~]$ su -
Passwd: hemlig
[root@linus /root]# init 0
```

En hurtig nedlukning fås ved at trykke "Ctrl-Alt-Delete", når du er i en tekstkonsol - f.eks. efter du trykker "Ctrl-Alt-F1".

Du skal dog en gang for alle lave en ændring, så "Ctrl-Alt-Delete" kommer til at medføre at Linux lukkes ned. Ændres intet, genstarter systemet. For at få maskinen til at stoppe kaldes **/sbin/halt**, og du skal så ændre i `/etc/inittab`

```
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

så der kommer til at stå

```
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -h now
```

Du kan selvfølgelig være fræk, så maskinen slet ikke kan stoppes fra tastaturet med "Ctrl-Alt-Delete", f.eks. kan du ændre `/etc/inittab` til

```
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:echo "No reboot for you my friend!\n"
```

Dette er uhyre praktisk, hvis din maskine står et sted, hvor der er mange pilfingre, f.eks. EDB-lokalet på en skole!

2.7. Crontab

Crontab er et smart system til automatisk at køre programmer på givne tidspunkter, såsom daglig backup. Det er en standarddel af et Unix-system og er derfor altid med i Linux. Man kan få kørt vilkårlige programmer på vilkårlige tidspunkter, f.eks. hvert minut, hver time, hver dag, ugedag eller månedsdag.

Crontab-filerne omfatter `/etc/crontab`, `/etc/cron.daily`, `/etc/cron.hourly`, `/etc/cron.monthly` og `/etc/cron.weekly`, samt cron-filer til systemets brugere i `/var/spool/cron/tabs`.

Syntaksen for crontab-filerne er den samme: En kommando pr. linje og i rækkefølge fra venstre mod højre, minut (0-59), time (0-23), månedsdag (0-31), måned (0-12), ugedag (0-7, hvor 0 og 7 begge er søndag) og endelig den kommando du vil have udført. Hvert felt adskilles af et mellemrum, og hvis man vil have, at programmet køres alle dage, så indsæt en "*" for månedsdag og måned. En kommando kan udføres f.eks. hvert 5. minut med "*/5 * * * *" og til et tids-interval bruges "-" som "1-5" for mandag til fredag. Flere tidspunkter adskilles med ",".

Ofte udløser en crontab-kommando en email til root. Det kan være uheldigt med mange emails til root, som blot fortæller, at systemet opfører sig normalt. Hvis linjen afsluttes med `>/dev/null 2>&1` sendes meddelelsen i stedet ud i intetheden.

Vi giver nu et eksempel, hvor vi hver morgen kl. 7 fra mandag til fredag sender brugeren "root" et brev med oplysninger om hvor meget belastning der er på maskinen.

```
0 7 * * 1-5 uptime | mail root
```

Brug **man -a crontab** til at læse mere om crontab. Skift til næste man-fil ved at trykke 'q'. Kommandoen **man 1 crontab** fortæller om administration af crontab-filer og **man 5 crontab** om indholdet i crontab-filerne.

2.7.1. /etc/crontab

Systemadministrator kan skrive kommandoer direkte i filen `/etc/crontab`. For at opretholde adskillelsen imellem systemet og brugerne, bruger administrator normalt også en fil i `/var/spool/cron/tabs`.

I /etc/crontab er defineret, hvordan der køres jobs hver time fra `/etc/cron.hourly` til tilsvarende hver dag, uge og måned via `/etc/cron.daily`, `/etc/cron.monthly` og `/etc/cron.weekly`.

Skriver administrator kommandoer direkte i `/etc/crontab`, skal brugernavnet, som kommandoen skal udføres under, angives efter de 5 tidspunkts-kolonner og før selve kommandoen.

2.7.2. /etc/cron.daily, hourly, monthly, weekly

I de 4 biblioteker under `/etc` kan systemadministrator placere scripts, som skal udføres på givne tidspunkter. Afviklingstidspunktet styres i `/etc/crontab`, så selve scriptet skal ikke indeholde oplysninger om tidspunkt, og da der er en fil til hver opgave, må scriptet brede sig over flere linjer.

2.7.3. Brugernes crontab-filer

Den almindelige bruger har ikke adgang til `/etc/crontab`. Kommandoer, de vil have kørt på et givet tidspunkt, lægges i filer med brugerens navn i `/var/spool/cron/tabs`. Ofte bruger systemadministrator også denne fil.

Med **crontab -e** kan brugeren skrive direkte i filen, men bl.a. for at kunne bruge sin normale editor i stedet for at være tvunget til at bruge **vi**, er det normalt at skrive en kladde i en almindelig tekst-fil (oftest med navnet `.crontab` i sit hjemmekatalog), og køre kommandoen **crontab ~/.crontab**.

I stedet for de 5 tidsangivelser kan linjen starte med '@reboot', hvorefter den udføres når systemet genstartes. Denne mulighed virker ikke for kommandoer, som er skrevet direkte i `/etc/crontab`.

2.7.4. Kør ikke-gentagne jobs med at

Som det er forklaret i de forrige afsnit så vil **crontab** primært anvendes til at køre automatiske scripts eller egentlige programmer på fastlagte tidspunkter gentaget hver time, dag, uge eller måned.

Nogle gange har man desuden brug for at køre programmer en enkelt gang, men man har måske ikke mulighed for selv at starte programmet på det tidspunkt man ønsker. Det kunne være at man lige ville starte en ekstra backup midt i den kommende nat. Linux-maskinen har ligesom andre UNIX-maskiner mulighed for at køre programmer når man selv ønsker dette. Vil man køre en kompleks serie af kommandoer bør man skrive kommandoerne ind i en fil, men ellers er den almindelige måde at indskrive kommandoerne interaktivt. Hvis vi gerne vil vide hvem som er logget ind på maskinen kl. 03:00 og hvor meget hukommelse der er brugt på maskinen på det tidspunkt, dvs. man vil gerne køre **who > ~/login_info.txt** kl. 3 om natten og næste morgen kan man lige gennemse filen `~/login_info.txt`. Derudover vil man gerne lige køre **free**, der viser hvor meget af den fysiske hukommelse samt swap-filen der er brugt. Dette gøres ved at køre kommandoen **free**.

Kommandoerne indskrives efter at man skriver **at** og tidspunktet. Antallet af kommandoer, som indskrives på de efterfølgende linjer en for en, er ikke begrænset. Kommandoerne afsluttes med **Ctrl-d** på en linje for sig.

```
[tyge@hven ~]$ at 3:00
at> who > ~/login_info.txt
at> free >> ~/login_info.txt
at> Ctrl-d
```

Næste morgen vil filen **~/login_info.txt** indeholde følgende:

```
 3:00am  up 23:03,  8 users,  load average: 0.00, 0.04, 0.08
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
pto      :0       console          Sat 2pm   ?     0.00s   ?     -
pto      pts/0    -                Sat 2pm 14:03m  0.00s   ?     -
pto      pts/2    -                 2:17pm  2:57m  3.45s  0.56s  /bin/bash
pto      pts/3    -                 1:21pm  8:18m  0.10s  0.10s  /bin/bash
pto      pts/4    -                 2:26pm  8:06m  0.52s  0.07s  bash
pto      pts/5    -                10:36pm  1.00s  0.27s  0.27s  /bin/bash
pto      pts/6    -                10:37pm  2:07   1.61s  1.51s  pine
             total       used       free     shared    buffers     cached
Mem:        190896     185820       5076          0      37244      76432
-/+ buffers/cache:      72144     118752
Swap:       329324      19936     309388
```

De sidste fire linjer kommer fra **free** og de første kommer fra **who**. Eksemplet illustrerer at det er nemt at anvende **at**, og det skal nævnes at hvis kommandoerne der udføres bringer tekst til stdout (standard output - dvs. tekst til skærm), så vil dette blive sendt som epost til brugeren.

Dette kan udnyttes sammen med en af de andre gode ting ved at. Hvis man gerne vil undgå at glemme den kaffe man lige har sat over, kan man få **at** til at sende sig en besked om 10 minutter med:

```
[tyge@hven ~]$ at now+10min
at> echo Kaffen er klar.
```

Man kan selvfølgelig også finde på mere seriøse anvendelser.

Har man brug for at se hvilke kommandoer man har i **at**-køen kan **atq** anvendes. Vil man slette jobs, da gøres dette med **atrm**. Endelig kan det nævnes at Linux-maskiner også har en **batch**-funktion, der fungerer som **at**, men er designet til at køre jobs, når der er ledig CPU-plads på maskinen. Det er efterhånden de færreste som anvender dette.

2.8. Sikkerhedskopiering

En vigtig del af systemadministration er sikkerhedskopiering, så man i tilfælde af nedbrud, harddiskcrash, brand eller andre uheld kan fortsætte uden større tab af data.

Sikring af ens data kan ske på flere måder, hvilket selvfølgelig er afhængigt af den grad af sikkerhed man ønsker, samt mængden af data, som skal sikres.

2.8.1. Metoder for sikkerhedskopiering

Grundlæggende kan man dele sikkerhedskopiering op i lokal- og fjernkopiering, hvilket bestemmes af det omfang af sikring som ønskes samt de tekniske muligheder som forelægger eller investering i samme.

2.8.1.1. Lokal sikkerhedskopiering

Lokal sikkerhedskopiering sker ved at man gemmer kopier af sine data på samme sted, som dataene er på. Her benyttes oftest disketter, bånd, brændbare cd'er eller udskiftelige harddiske som lagringsmedier.

2.8.1.2. Sikkerhedskopiering via internettet

Fjernkopiering sker ved, at man aftaler med andre, at de krypteret spejler ens data samtidig med, at man gør det samme med deres. Det kræver selvfølgelig en tro på dem, som man udveksler data med, selvom dataene er krypterede.

Fjernkopiering er blevet mere aktuel efter at højhastighedsforbindelser (ADSL mv.) er kommet ned i et prisleje, hvor også private og mindre virksomheder kan være med.

Man nøjes med at kopiere de forskelle, som er sket siden sidst, så man belaster forbindelsen mindst muligt. Hvis man ikke stoler blindt på den, som giver plads til ens sikkerhedskopi, kan man altid kryptere dataene med GnuPG eller lignende krypteringsværktøjer.

2.8.1.3. Hvad skal kopieres?

Generelt skal man gemme de data, som man ikke kan undvære, og som er umulige at fremskaffe på anden vis f.eks. via installation.

Dette kan være en af baggrundene for at dele et Linux-system op i flere partitioner, da visse værktøjer netop sikkerhedskopierer efter partitioner. De vigtigste kataloger er `/home` (brugerdata), `/var` (serverdata) og `/etc` (opsætningsdata), som således også bør have deres egne partitioner.

2.8.2. Værktøjer til sikkerhedskopiering

Valget af værktøj til sikkerhedskopiering er afhængigt af en lang række faktorer så som lagringsmedie, type af adgang til systemet, graden af automation og omfanget af data.

I de næste afsnit vil forskellige værktøjer kort blive gennemgået, hvor deres fordele og ulemper vil blive belyst. Listen er på ingen måde udtømmende, men kan benyttes som et meget hurtigt overblik over de muligheder, som findes til sikkerhedskopiering i et Linux-miljø.

2.8.2.1. Sikkerhedskopiering med tar

Den mest benyttede kommando til sikkerhedskopiering er **tar**, som er en forkortelse for Tape ARchive. **tar** samler blot filer og deres data sammen i en enkelt fil, som så kan gemmes på et medie til sikkerhedskopiering f.eks. et bånd.

Hvis du vil tage en sikkerhedskopi på disketter af alle filerne i kataloget `/home` gøres det således:

```
[root@linus /root]# tar -cMf /dev/fd0H1440 /home
```

Hvor **-cMf** står for henholdsvis oprette (**c**reate), på flere medier (**m**ultiple) og viser hvilke filer (**f**iles), som indgår i sikkerhedskopieringen.

Når der ikke er mere plads på den første diskette, spørger programmet automatisk efter den næste.

Hver gang man har foretaget en sikkerhedskopiering, skal man huske at kontrollere den, så man er sikker på, at man kan genskabe dataene fra sikkerhedskopien. Det gøres således:

```
[root@linus /root]# tar -compare -verbose -f /dev/fd0H1440
/home
/home/alle
/home/alle/annoncetekst.txt
....
```

I stedet for de lange tilvalg kan benyttes de kortere udgaver, hvilket kan ses af man-siden for **tar**.

Skal man genskabe dataene igen, så er det ligetil:

```
[root@linus /root]# tar -extract -same-permissions -verbose -file /dev/fd0H1440
/home
/home/alle
/home/alle/annoncetekst.txt
....
```

Se mere ved hjælp af man-siden (**man tar**) eller i afsnittet om sikkerhedskopiering i "The Linux System Administrators' Guide": http://www.tldp.org/LDP/sag/html/.

2.8.2.2. Sikkerhedskopiering med dump og restore

Hvis man benytter Linux-filsystemet ext2, kan man benytte kommandoen **dump** til at gemme sikkerhedskopier helt eller delvist af hele filsystemet. De genskabes igen med kommandoen **restore**.

Som standard gemmer den kun de forskelle, som er opstået siden sidste sikkerhedskopiering, men den kan sættes til at tage en kopiering af det hele.

Første gang man gemmer (samt med betryggende mellemrum) tages en sikkerhedskopi af alle filerne i en partition. Lad os sige at `/home` ligger på partition `/dev/hda6`. Så ser det sådan ud:

```
[root@linus /root]# dump -u0 -f /dev/st0 /home
```

Genskabelsen af data sker ved, at man først genskaber filsystemet på den partition, som skal genskabes:

```
[root@linus /root]# mke2fs /dev/hda6
[root@linus /root]# mount /dev/hda6 /mnt
[root@linus /root]# cd /mnt
```

Så den bliver klar til at genskabe de data, som ligger i sikkerhedskopien:

```
[root@linus /root]# restore rf /dev/st0
```

Efter genskabelsen skal `/home` monteres igen:

```
[root@linus /root]# mount /mnt/home
```

Se mere om **dump** og **restore** ved at bruge deres man-sider.

2.8.2.3. Sikkerhedskopiering med afio

Hvis man gerne vil have adgang til filerne individuelt efter sikkerhedskopieringen, kan man benytte værktøjet **afio**, som blandt andet bliver benyttet af KBackup.

Læs mere med **man afio**.

2.8.2.4. Sikkerhedskopiering med KBackup

KBackup en alsidig applikation til håndtering af sikkerhedskopiering bygget oven på andre værktøjer blandt andet **tar** og **afio**.

Den menubaserede grænseflade er meget intuitiv at bruge, så man nemt får sat en god rutine op for sikkerhedskopiering af de data, man selv ønsker.

Flere informationer kan findes på KBackups hjemmeside: http://kbackup.sourceforge.net

2.8.2.5. Sikkerhedskopiering med Arkeia

Arkeia er et meget professionelt grafisk værktøj til håndtering af sikkerhedskopiering. Det er desværre closed sourced, men gratis (også for virksomheder) at bruge til en server og to klienter.

Hent vejledningen og programmet fra deres hjemmeside: http://www.arkeia.com/ og installér det.

2.8.2.6. Microsoft Windows-værktøjer via Samba

Ligger ens data på en Samba-server, kan man i de fleste tilfælde bruge de Microsoft Windows-værktøjer, som følger med båndstation eller anden hardware til sikkerhedskopiering. Man skal så blot huske at oprette en sti til lagringsmediet i opsætningsfilen til Samba (`/etc/smb.conf`).

2.9. DPMS - aktiv strømstyring

Dette afsnit handler om hvordan man kan få maskinen til at kunne slukke skærmen automatisk og generelt prøve at spare strøm. For at få strømstyring til at virke under X skal man tilføje `Option "DPMS"` til filen `/etc/X11/XF86Config-4`

```
Section "Monitor"
        Identifier   "Monitor0"
        VendorName   "Monitor Vendor"
        ModelName    "Monitor Model"
        HorizSync   30.0-64.0
        VertRefresh 50.0-110.0
        Option "DPMS"
...
```

I brugerens `.tcshrc`/`.bashrc` eller `/etc/bashrc` tilføjes `xset dpms 120 200 0`. Dermed får man skærmen til at blanke efter 120 sekunder (standby) og slukke (suspend) efter 200 - det sidste nul skulle få maskinen til ikke at slukke.

Du skal også have oversat din kerne, så den understøtter "Power Management".

```
[*] Power Management support
[ ]   ACPI support
```

```
<*>   Advanced Power Management BIOS support
[ ]     Ignore USER SUSPEND
[*]     Enable PM at boot time
[ ]     Make CPU Idle calls when idle
[*]     Enable console blanking using APM
[*]     RTC stores time in GMT
[ ]     Allow interrupts during APM BIOS calls
[*]     Use real mode APM BIOS call to power off
```

Her kan der være forskelle alt efter maskintype.

2.10. Måling af CPU-temperatur

Med Pentium II, III, AMD K6 II og lignende nyere processorer kan man direkte måle temperatur på CPU'en samt holde styr på blæseren på CPU'en. Det er noget som CPU'en skal have indbygget at det virker via en I2C kanal.

Det første der skal gøres er at installere `lm_sensors`. På Red Hat 7.2 gøres dette med **rpm -ivh lm_sensors*.rpm** hvis man står i cd-rom'ens RPMS-katalog.

Dernæst skal man detektere maskinen ved at køre **/usr/sbin/sensors-detect**. Programmet er selvforklarende. Det første der gøres er at indlæse de kerne-moduler der skal til at lave målingers f.eks. `i2c-piix4`. Programmet prøver at gætte dette selv og man skal oftest bare trykke return et par gange.

```
...
 I will now generate the commands needed to load the I2C modules.
 Sometimes, a chip is available both through the ISA bus and an I2C bus.
 ISA bus access is faster, but you need to load an additional driver module
 for it. If you have the choice, do you want to use the ISA bus or the
 I2C/SMBus (ISA/smbus)?

WARNING! If you have some things built into your kernel, the
below list will contain too many modules. Skip the appropriate ones!
To load everything that is needed, add this to some /etc/rc* file:

#----cut here----
# I2C adapter drivers
modprobe i2c-matroxfb
modprobe i2c-isa
# I2C chip drivers
modprobe eeprom
modprobe w83781d
#----cut here----

To make the sensors modules behave correctly, add these lines to either
/etc/modules.conf or /etc/conf.modules:
```

```
#----cut here----
# I2C module options
alias char-major-89 i2c-dev
#----cut here----
```

Som det ses skal giver **sensors-detect** et par linjer som skal indsættes i f.eks. `/etc/rc.local` (SuSE `/etc/rc.d/boot.local`) og tilsvarende til `/etc/modules.conf`

Når maskinen har været genstartet kan man med kommandoen **sensors** detektere temperatur osv. Dette kan se således ud.

```
w83781d-i2c-0-2d
Adapter: SMBus PIIX4 adapter at 5000
Algorithm: Non-I2C SMBus adapter
VCore 1:   +1.98 V  (min =  +1.80 V, max =  +2.20 V)
VCore 2:   +1.47 V  (min =  +1.80 V, max =  +2.20 V)       ALARM
+3.3V:     +3.42 V  (min =  +2.97 V, max =  +3.63 V)
+5V:       +4.97 V  (min =  +4.50 V, max =  +5.48 V)
+12V:     +11.70 V  (min = +10.79 V, max = +13.11 V)
-12V:     -11.40 V  (min = -10.78 V, max = -13.18 V)
-5V:       -4.94 V  (min =  -4.50 V, max =  -5.48 V)
fan1:     4066 RPM  (min = 3000 RPM, div = 2)
fan2:        0 RPM  (min = 3000 RPM, div = 2)              ALARM
fan3:        0 RPM  (min = 3000 RPM, div = 2)              ALARM
temp1:    +31.0°C   (limit = +60°C, hysteresis = +50°C)
temp2:    +208.0°C   (limit = +60°C, hysteresis = +50°C)
temp3:    +25.0°C   (limit = +60°C, hysteresis = +50°C)
vid:      +2.00 V
alarms:   Chassis intrusion detection
beep_enable:
          Sound alarm disabled
```

Man kan således se at CPU-blæseren kører 4066 omdrejninger per minut og der er tre CPU-temperaturer. Desuden er der 3 falske alarmer. Derfor skal der foretages justeringer i filen `/etc/sensors.conf` i sektionen om den chip som blev fundet tidligere. Chippen w83781d deler data med flere andre chips.

```
chip "lm78-*" "lm78-j-*" "lm79-*" "w83781d-*"

    label in0 "VCore 1"
    label in1 "VCore 2"
    label in2 "+3.3V"
    label in3 "+5V"
    label in4 "+12V"
    label in5 "-12V"
    label in6 "-5V"

    set in0_min 2.8*0.95
    set in0_max 2.8*1.05
    set in1_min 2.8*0.95
    set in1_max 2.8*1.05
```

```
set in2_min 3.3 * 0.95
set in2_max 3.3 * 1.10
set in3_min 5.0 * 0.95
set in3_max 5.0 * 1.05
set in4_min 12 * 0.95
set in4_max 12 * 1.05
set in5_min -12 * 0.93
set in5_max -12 * 1.07
set in6_min -5 * 0.95
set in6_max -5 * 1.05

ignore in1
set fan2_min 0
set fan3_min 0
```

Hver af chippens interne måle-punkter (f.eks. in0) tildeles et navn som f.eks."VCore 1". Hvert måle-punkt skal holde sig indenfor et vist område, der som regel sættes til at være nominalværdien plus/minus 5%. Tolerancerne kan selvfølgelig tilpasses maskinen. Især måle-punkterne in0 og in1 bør konfigureres og tilpasses processoren. Man kan genstarte maskinen og kigge i dens BIOS og kigge især efter om der måles "VCore". Hvis der kun måles en værdi, kan sensors indstilles som ovenfor, til at ignorere måle-punktet in1.

Ændringer i filen `/etc/sensors.conf` må indlæses i modulerne, ved at køre følgende som root

```
[root@linus /root]# sensors -s
```

Resultatet kan straks undersøges med kommandoen **sensors w83781d-i2c-0-2d** eller bare med kommandoen **sensors** hvis man ønsker resultater fra alle konfigurerede chips.

Der er en masse forklarende kommentar i filen `/etc/sensors.conf`, så når man har bestemt hvilke chips (der kan nemlig være flere) der er i ens maskine, kan man fordybe sig i hvordan den skal konfigureres.

2.10.1. Fejlfinding

Overvågning af computerens maskinel, skal bruge passende moduler til maskinen samt i2c bus understøttelse.

Når sensors modulerne er installeret i kernen, dannes filerne: `/proc/sys/dev/sensors/*` `/proc/bus/i2c*` . Hvis disse filer ikke er at finde, er sensors modulerne ikke installeret i kernen.

Hvis i2c bus modulerne eller hardware modulerne findes på computeren kan de findes med kommandoerne

```
[root@linus /root]# ls -l /lib/modules/`uname -r`/kernel/drivers/i2c/
```

```
[root@linus /root]# ls -l /lib/modules/`uname -r`/kernel/drivers/sensors/
```

Ovennævnte placering gælder for en specialbygget kerne i Debian unstable, men kan formodes at gælde for alle 2.4 kerner.

Hvis maskinovervågningen giver forkerte oplysninger, kan der være flere årsager:

- Opsætnings-programmet **sensors-detect** er ikke perfekt og det kan installere forkerte moduler. Det kan medføre udlæsning af forkerte oplysninger om maskinens tilstand. Der er eksempler på udlæsninger af for høj eller for lav CPU temperatur, blot fordi der var blevet installeret et modul der ikke passede til maskinen.
- Nogle bundkort er sådan indrettet, at modulerne skal konfigureres med særlige parametre for at fungere korrekt. Opsætnings-programmet **sensors-detect** kan finde de fleste af disse parametre men ikke alle. I dette tilfælde kan man forsøge, at finde de korrekte parametre og derpå kan man indsende en fejl-rapport med de parametre man har fundet. Medfølgende med lm-sensors, er dokumentation om hvordan disse parametre bør være. Samme dokumentation findes på webadressen: http://www2.lm-sensors.nu/~lm78/docs.html
- Der er fejl i selve modulets kilde-tekst. Hak den og indsend en fejl-rapport og en eventuel rettelse.

2.10.2. Indsamling af oplysninger til fejlrapport

Hvis opsætnings-programmet danner en fejlagtig opsætning kan man overveje at indsende en fejl-rapport direkte til udviklings-holdet for lm-sensors på web adressen: http://www2.lm-sensors.nu/~lm78/ Fejl-rapporten kan indeholde oplysninger om PCI bussen og en dump udskrift af i2c bussen. Der kan også laven en dump udskrift af ISA bussen men der advares kraftigt imod at gøre det, idet områder i hukommelsen kan blive overskrevet, som kan indeholde oplysninger der er vitale for maskinens stabilitet. Hvis man absolut skal dumpe ISA bussen, bør man genstarte maskinen i enkeltbrugertilstand fra diskette eller cd-rom. På den måde er der mindst risiko for at miste oplysninger eller endda et helt filsystem.

Oplysninger om PCI-bussen i en maskine med et Asus bundkort med Intel TX chipsæt og Realtek 8139 netkort, kan se således ud:

```
[tyge@hven /]$ /bin/lspci -n
00:00.0 Class 0600: 8086:7100 (rev 01)
00:01.0 Class 0601: 8086:7110 (rev 01)
00:01.1 Class 0101: 8086:7111 (rev 01)
00:01.2 Class 0c03: 8086:7112 (rev 01)
00:01.3 Class 0680: 8086:7113 (rev 01)
00:09.0 Class 0200: 10ec:8139 (rev 10)
00:0c.0 Class 0100: 9004:5078 (rev 03)
```

Modulet i2c-dev skal være installeret før man kan dumpe i2c bussen. En dump udskrift af i2c bussen kan se således ud:

```
[root@linus /root]# /usr/sbin/i2cdump 0 0x18
```

```
Warning: no size specified (using byte-data access)
  WARNING! This program can confuse your I2C bus, cause data loss and worse!
  I will probe file /dev/i2c-0, address 0x18, mode byte
  You have five seconds to reconsider and press CTRL-C!

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: 20 00 04 3c 20 00 04 3c 20 00 04 3c 20 00 04 3c
10: 20 00 04 3c 20 00 04 3c 20 00 04 3c 20 00 04 3c
...
```

Parametrene til i2cdump kan man få ved at kigge på filnavnene i mappen `/proc/sys/dev/sensors/` hvor der tilfældigvis findes mappen `max1617-i2c-0-18`.

Desuden bør man medtage alle øvrige oplysninger der kan være af nytte ved opklaringen af problemet.

2.11. Log-filer

En Linux-installation indeholder forskellige logfiler der registrerer forskellige ting der sker med systemet. Den vigtigste logfil er `/var/log/messages` der sammen med andre fortæller systemets tilstand.

2.11.1. /var/log/messages

Tekstfilen `/var/log/messages` indeholder beskeder fra kernen og andre programmer der rapporterer til syslogd. Herunder er vist noget af det man kan se i logfilen:

```
Jun 25 10:45:18 figus syslogd 1.4.1: restart (remote reception).
Jun 25 10:45:22 figus sshd: sshd startup succeeded
Jun 25 11:01:35 figus ntpdate[4053]: step time server 129.240.64.3 offset 26.235235 sec
Jun 25 14:05:42 figus su(pam_unix)[4454]: session opened for user root by (uid=506)
Jun 25 14:08:45 figus sshd[4521]: Failed password for root from 192.168.1.2 port 1050 ssh2
```

Logfilen indeholder mange informationer fra mange programmer. Disse kan være lidt svære at overskue, så det kræver noget træning at hvad der er som det skal være, og hvad der ikke er. I ovenstående eksempel er det maskinen med maskinnavnet `figus` der ses i logfilen. Programmerne syslogd, sshd, ntpdate og su har skrevet i logfilen. Kun den sidste linje ser lidt mistænkelig ud, nogen prøvede at logge ind som root fra IP-nummer 192.168.1.2. Mistænkelig er den dog ikke, for ham der prøvede at logge ind havde ret til det, han havde bare glemt at root ikke kunne logge ind direkte, men skal først logge ind som en almindelig bruger. Så det kræver lidt træning at overskue denne logfil.

De typiske værktøjer man bruger til at se var på `/var/log/messages` er: less, tail og grep. Med **less** kan man med fordel taste 'G' for at gå til bunden af filen, og 'F' for at følge hvad der kommer af nye beskeder.

```
[root@linus /root]# less /var/log/messages
[root@linus /root]# tail -f /var/log/messages
[root@linus /root]# grep restart /var/log/messages | less
```

Se også "Friheden til sikkerhed på internettet" for mere information om `/var/log/messages`.

2.11.1.1. syslog remote

Der kan være flere fordele ved at logge syslog kommandoer til en anden server. En kunne være at samlet holde øje med flere servere, en anden kunne være for at sikre sig at en cracker ikke går ind og sletter `/var/log/messages` på den maskine der bliver kompromitteret

Først går man til den maskine der skal modtage syslog-beskeder fra andre maskiner. Her ændres syslog til at starte med kommandoen '-r' (remote). Dette kan enten gøre i `/etc/sysconfig/syslog`, eller `/etc/init.d/syslog`, afhængigt af hvilken distribution man kører.

```
# Filnavn: /etc/sysconfig/syslog
SYSLOGD_options="-m 0 -r"
```

På klienter der skal logge til syslog-serveren kan der så rettes i `/etc/syslog.conf`. Vil man logge alt fra klienten over til serveren `minlog.domain.dk` skrives:

```
*.*  @minlog.domain.dk
```

Er det kun mail-log der skal sendes over kan man skrive:

```
mail.*  @minlog.domain.dk
```

Ovenstående ændringer kræver at **/etc/init.d/syslog** bliver genstartet.

2.11.2. dmesg

dmesg er ikke en logfil, men et program der udskriver beskeder fra kernens ring-buffer. Her står al den information kernen skriver under boot, og hvad der efterfølgende sker af fejl i kernen. Vil man vide noget om hvilke IDE-harddiske der er i systemet, kan man for eksempel bruge kommandoen **dmesg | grep hd** og få et output der minder om:

```
[root@linus /root]# dmesg | grep hd
ide0: BM-DMA at 0xfcf0-0xfcf7, BIOS settings: hda:pio, hdb:DMA
hda: FUJITSU MHM2200AT, ATA DISK drive
hdb: TOSHIBA DVD-ROM SD-C2502, ATAPI CD/DVD-ROM drive
hda: 39070080 sectors (20004 MB) w/2048KiB Cache, chs=2584/240/63, UDMA(33)
hdb: ATAPI 24X DVD-ROM drive, 128kB Cache, UDMA(33)
```

2.11.3. /var/log/mail/info

`/var/log/mail/info` er logfilen for ind- og udgående mail. Her står hvem der har modtaget filer, og hvem der er sendt filer til. Typisk er det ved fejl at man kigger i denne fil, og i mere sjældne tilfælde bruges den til at dokumenterer at en mail blevet sendt eller modtaget. Filen ses typisk med **less**, **tail** eller **grep**. I det viste eksempel herunder, er det **fetchmail** der henter mail fra en ISP, hvilket man ikke kan se, og afleverer denne mail til localhost. Afsenderen er <chlor@sslug.dk>, og modtageren er <chlor@localhost.w0.dk>.

```
Jun 26 07:44:44 figus postfix/smtpd[8079]: connect from localhost.localdomain[127.0.0.1]
Jun 26 07:44:44 figus postfix/smtpd[8079]: 09BEC1326AE: client=localhost.localdomain[127.0.
Jun 26 07:44:44 figus postfix/cleanup[8080]: 09BEC1326AE: message-id=<20020626054426.20973.
Jun 26 07:44:44 figus postfix/qmgr[1665]: 09BEC1326AE: from=<chlor@sslug.dk>, size=804, nrc
  (queue active)Jun 26 07:44:44 figus postfix/local[8082]: 09BEC1326AE: to=<chlor@localhost
  relay=local, delay=0, status=sent ("|/usr/bin/procmail -Y -a $DOMAIN")
Jun 26 07:44:47 figus postfix/smtpd[8079]: disconnect from localhost.localdomain[127.0.0.1]
```

2.12. Kontrol af diskforbrug med quota

Hvis man driver en server med mange brugere, sker det fra tid til anden at en bruger bevidst eller ubevidst kommer til at fylde disken op med filer. For at et sådant uheld ikke skal kunne påvirke de andre brugere på systemet, så kan man installere quota. Quota gør det muligt at sætte en disk forbrugs-grænse for den enkelte bruger eller evt. for en gruppe.

Systemet er så fleksibelt, at man yderligere kan sætte en ekstra-pulje op, så brugeren i en fast tidsperiode (kaldet grace-perioden) godt må "svine" med pladsen bare der ryddes op igen; f.eks. kan man have 10 MB diskplads, men kan få lov til i en periode på to timer ekstra at bruge yderligere 5 MB.

Både den almindelige kvote og den ekstra diskkvote virker ved at man pludselig ikke kan gemme filer, når man er oppe på maksimum. Brugeren må derefter selv slette filer.

Derfor er det vigtigt at systemadministratoren forklarer brugeren hvilke grænser, der er sat og hvad man gør for at følge eget forbrug - og endelig hvordan man rydder op.

Installation: Der skal være quota-understøttelse i Linux-kernen. Normalt vil en Linux-distribution have quota understøttet, ellers må du oversætte kernen på ny og da svare ja til "Quota Support" under "Filesystems" (CONFIG_quota=y).

I quota-tools pakken findes de værktøjer der skal være installeret for at du kan benytte diskkvoterne. Enten kan du benytte hvad der hører til din distribution, (fx. quota-3.01pre9-3.i386.rpm hos Red Hat 7.2) eller også hente og installere pakken http://prdownloads.sourceforge.net/linuxquota/quota-3.03.tar.gz

Quota sættes på de enkelte partitioner, dvs. `/etc/fstab` skal redigeres, så man i `options`-feltet tilføjer `usrquota` til hver partition man ønsker at køre quota på.

F.eks. ændres linjen i `/etc/fstab` fra

```
/dev/hda2   /home   ext2    defaults 1 1
```

til

```
/dev/hda2   /home   ext2    defaults,usrquota 1 1
```

Til at starte quota skal der nu indekseres og quota-programmet skal startes. Kør følgende som root

```
[root@linus /root]# /bin/mount -v -o remount /home  <-- for at gen-montere den ændrede partition
[root@linus /root]# /sbin/quotacheck -a -m <--- for at lave et index over brugernes forbrug
[root@linus /root]# /sbin/quotaon -uv /home <--- start quota
```

I den sidste kommando kan **/home** skiftes ud med **-a**, i så fald gælder kommandoen alle filsystemer.

mount køres kun en gang for alle, for at få systemet til at genindlæse `/etc/fstab`.

quotacheck køres for at danne et indeks over brugernes diskforbrug. Indekset gemmes på ældre systemer i `/home/quota.user` og i `/home/aquota.user` på nyere. **quotacheck** skal køres igen hvis man har rettet fejl i filsystemet med fsck - typisk under opstart efter strømudfald. Det klares da ofte automatisk af ens initscripts ellers kan følgende indsættes, inden quotaon kaldes, i `rc.sysinit`:

```
if [ -x /sbin/quotacheck ]; then
  echo "Undersøger kvoter på lokale filsystemer"  /sbin/quotacheck -v -a
fi
```

quotaon skal køres hver gang maskinen startes op, og kan automatiseres i med et par linjer i `rc.sysinit` eller `rc.local` (SuSE `/etc/rc.d/boot.local`):

```
  if [ -x /sbin/quotaon ]; then
    echo "Starter kvotestyring for lokale filsystemer"
    /sbin/quotaon  -a
  fi
```

Nu kan man redigere grænsen for en bruger - hvis man har mange brugere, så opret en test-bruger "quotauser" og lave følgende som skabelon og sidenhen kan man kopiere test-brugerens quota-opsætning til alle andre brugere.

Grænser redigeres ved at køre edquota for brugeren. Grænser for diskforbrug anføres i kbytes, dvs. 10000 betyder 10MB. Hvis quota ikke sættes op er standard-indstillingen at der ingen grænse er (dette anføres som et nul i opsætningsfilen).

```
[root@linus /root]# /usr/sbin/edquota -u quotauser
Disk quotas for user tyge (uid 500):
Filesystem      blocks       soft       hard     inodes     soft   hard
/dev/hda2       695879          0          0       6741        0      0
```

Vi ser at brugeren tyge med uid 500 på filsystemet /dev/hda2, (hvor `/home` mountes ifølge `/etc/fstab`), i øjeblikket benytter 695879 filblokke (679MB) og 6741 filer, samt at der ikke er sat grænser for hvor meget brugeren må benytte.

Ret nu tallene til det brugeren må. Det er vi-editoren som kommer frem, hvis du ikke har sat EDITOR variablen.

- "soft" betyder grænse, som brugeren normalt forventes at holde sig under. Systemet tillader dog at denne grænse overstiges i kortere tid ("grace"-perioden).
- "hard" betyder grænse, som brugeren aldrig kan overstige.

Til højre for de første soft/hard står der inodes og igen soft/hard - dette er antal filer man kan lave. Dette bruges kun sjældent. Hvis man ønsker at brugeren fast kan bruge 10 MB og 5 MB ekstra i grace-perioden, samt et ubegrænset antal filer, skrives

```
Disk quotas for user tyge (uid 500):
Filesystem      blocks       soft       hard     inodes     soft   hard
/dev/hda2       695879       10000      15000     6741        0      0
```

Afslut med at skrive **:wq!**

Hvis man derefter vil sætte "grace"-perioden hvor ekstra diskforbrug tolereres, så kan dette indstilles ved at skrive

```
[root@linus /root]# /usr/sbin/edquota -t
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem      Block grace period      Inode grace period
/dev/hda2        7 days                    7days
```

For illustrations skyld kan vi sætte en grace-periode helt ned til to timer ved at rette teksten til

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem      Block grace period      Inode grace period
/dev/hda2        2hours                    2hours
```

Afslut med at skrive **:wq!**

Tanken med at tildele en test-bruger "quotauser" er at man relativt nemt kan tildele samme quota til alle almindelige brugere ved at køre

```
[root@linus /root]# edquota -p quotauser `awk -F: '$3 > 499 {print $1} /etc/passwd`
```

Tricket er her at man finder alle almindelige brugere ved bruger-nummer på 500 eller større.

Andre interessante kommandoer:

- **repquota -a** <-- viser status på diskforbrug
- **warnquota** <-- bruges til at alarmere brugere som er over kvote

Til warnquota, der sender mail til brugere over kvota, hører opsætningsfilerne /etc/quotatab og /etc/warnquota.conf. Disse benyttes især til at gøre meddelelserne til brugerne mere læse venlige. quota-tools understøtter i øvrigt også GNU gettekst, så der er mulighed for at oversætte værktøjerne til brugernes eget sprog (make pot && kbabel pot.po).

I /proc/sys/fs systemet findes dquot-nr og dquot-max, som viser antallet af cachede kvota forekomster, antallet af frie disk kvota forekomster samt øvre grænse for antallet af disk kvota forekomster.

Med quota er det altså muligt at begrænse antallet af filer og deres samlede størrelse pr. bruger. Systemet kan desuden konfigureres til at begrænse antallet af filer og deres samlede størrelse pr. gruppe.

Med quotastats kan man få lidt information om hvorledes selve kvota systemet har det.

```
[tyge@hven ~]$ /usr/sbin/quotastats
Number of dquot lookups: 101289
Number of dquot drops: 101271
Number of still active inodes with quota : 18
Number of dquot reads: 93
Number of dquot writes: 2077
Number of quotafile syncs: 134518740
Number of dquot cache hits: 7391
Number of allocated dquots: 90
Number of free dquots: 2036
Number of in use dquot entries (user/group): -1946
```

Flere informationer om quota kan findes på

- http://www.linuxdoc.org/HOWTO/mini/Quota.html
- http://www.gnuskole.dk/bog/quota.html
- http://www.mvw.cistron.nl/quota.html
- http://www.sf.net/projects/linuxquota/

Den medfølgende dokumentation er lidt spredt, men kan samles til HTML og PostScript med følgende liner:

```
for f in $(file *|grep troff|cut -d: -f-1)
do
  man2html $f >doc/$f.html
done
cd doc
sgml2html quotadoc.sgml
groff -Thtml -ms quotas.ms >quotas.html
ln -s quotas-1.eps quotas-1.ps
```

2.13. Hvordan man får sin linuxmaskine til at holde op med at bruge harddisken

Jeg har en gammel bærbar i mit soveværelse. Den bliver brugt som printserver og gateway til et trådløst (802.11) netværk. Den skal være stille; især når jeg sover.

Jeg har også en computer på arbejdet. Den er bl.a. FTP- og webserver. Mine kollegaer bryder sig ikke om at den larmer, når jeg ikke er der.

Det er acceptabelt og stort set uundgåeligt at en maskine larmer når man bruger selv den, for eksempel skriver ud til printeren, retter i dokumenter osv, men så er man jo også vågen.

Målet er at få den til at være stille når den ikke laver noget, eller kun laver simple ting (rutning, FTP af små filer osv.).

Det, man kan gøre noget ved, er larmen fra harddisken.

Først skal man have sat spin-down tiden på harddisken (eller harddiskene hvis der er mere end en). På nogle maskiner kan man gøre det i BIOS'en.

Fra Linux kan man gøre det med **hdparm**-programmet. Følgende kommando sætter den første IDE-harddisk (`/dev/hda`) til at lukke ned efter et minut:

```
[root@linus ~]# hdparm -S 12 /dev/hda
```

For at køre kommandoen hver gang computeren startes, kan man komme den i en opstartsfil, se Afsnit 2.5.

2.13.1. Problematiske programmer

Her er nogle af de programmer, der kan finde på at starte disken uden at det er nødvendigt.

- *syslog* Skriver som default "-- MARK --" i log filerne hvert 20. minut.

  Løsning Start syslogd med **-m 0** optionen. Rettes i `/etc/init.d/syslog`

- *modprobe* Kan finde på at lede i `/lib/modules/kernerver/` efter et ikke-eksisterende modul hvert minut.

  Løsning 1 Se i `/var/log/ksymoops/20020702.log` (brug den rigtige dato). F.eks. fandt jeg: `20020702 161032 start /sbin/modprobe -s -k -- char-major-6 safemode=1`

  Det var fordi jeg kørte **lpd** dæmonen uden en driver for parallelporten.

  Løsning 2 Oversæt en kerne med alt hvad du skal bruge og lad være med at køre modprobe (modutils).

- *ntp* Skriver som default i `/var/lib/ntp/ntp.drift` og `/var/log/ntpstats/loopstats` med få minutters mellemrum.

  Løsning 1 Man kan minimere problemet ved at sætte tiden mellem opdateringer af uret til det maksimale.

  I `/etc/ntp.conf` sæt for hver server: **server clock.server.com minpoll 16 maxpoll 17**

  Løsning 2 Det er som regel ikke nødvendigt at bruge ntp. Brug **ntpdate** i stedet.

- *CUPS* Skriver et SSL-certifikat i `/etc/cups/ssl/server.crt` hvert 5. minut. Den læser `/etc/groups` hver gang, så det er ikke nok at lave `/etc/cups/certs` til et link på en RAM-disk.

  Løsning 1 Hent kildeteksten. I `scheduler/main.c` ret de 300 i **if ((time(NULL) - RootCertTime) >= 300)** til noget meget større, eller fjern helt if-sætningen.

  Løsning 2 Fra og med CUPS version 1.1.16 kan man undgå skrivningen af certifikater at at indsætte kommandoen: "RootCertDuration 0" i filen `/etc/cups/cupsd.conf`

- *cron* Kører en masse jobs som ikke er nødvendige.

  Løsning Fjern alt hvad du ikke behøver fra */etc/cron.d*, `/etc/crontab` og `/etc/cron.daily`. Se også om mail senere.

- *ftpd* FTP-servere tilgår typisk `/etc/{passwd,hosts,group}`, hjemmekataloger osv.

Løsning: muddleftp og tmpfs Lav en tmpfs partition, og brug muddleftpd. Sæt muddleftp op med `/etc/muddleftpd.com` på flg. måde:

Sæt "logfile" og "scratch file" til filer i roden tmpfs partitionen. Sæt "rdnstimeout 0" i sektionen "main" for at slå "reverse DNS lookup" fra. Sæt "uid" og "gid" i sektionen "localusers" for at undgå brug af `/etc/passwd` og `/etc/group`. Sæt "authmethod internal" og "internal_passfile /fileontmpfs/" i sektionen "localusers". Brug mudpasswd til at lave password filen. Sæt root directory til noget på tmpfs partitionen. Password filen skal kopieres til tmpfs filsystemet hver gang muddleftp startes. Det gøres i `/etc/init.d/muddelftpd` (Debian):

```
case "$1" in
    start)
    cp -a /home/bruger/mpw /tfs/
...
```

`muddleftpd.conf` findes under kataloget `eksempler` til denne bog.

- *browsere* Netscape 4.7 skriver sit cache-indeks, selv når den ikke bliver brugt. Den gratis version af Opera læser fra disken ca. hvert 5. minut for at udskifte reklamen.

  Løsning Luk din browser, når du ikke bruger den.

  Eller brug browsere, der ikke skriver på disken, når de ikke bruges. Som erstatning for Opera på små computere kan "Dillo" anbefales.

- *mail* MTA'er (sendmail, exim, postfix, etc) køres ofte ca. hvert 15. minut fra `/etc/cron.d/MTA`, hvor `MTA` er navnet på din MTA

  Tjek `/etc/init.d/MTA` for at se om en MTA, der tjekker spool-filerne, startes med et fast interval. Det gøres som regel med en **-q time** (fx **-q 1h22m**) option.

  Løsning Det er sjældent nødvendigt at tjekke spool-filerne med faste intervaller. Brugere kan sende og modtage post uden dette.

  Fjern `/etc/cron.d/MTA` filen og **-q** parameteren i `/etc/init.d/MTA`.

- *updatedb* kører **find** mm. hver dag for at resultaterne af **locate** kommandoen er up-to-date.

  Løsning 1 Afinstallér **updatedb**.

  Løsning 2 Fjern `/etc/cron.daily/find`

Hvis man har brug for **locate** kommandoen kan man altid manuelt køre **updatedb** som root.

2.13.1.1. Andre vink

Montér filsystemer med "noatime" optionen Hvis man monterer et filsystem med "noatime", bliver inode-access-tiden ikke opdateret. Det betyder at hvis en fil eller et katalog, der er cached, bliver læst, så startes disken ikke for at skrive filen eller katalogets access-tid tilbage.

Skriv følgende i din **/etc/fstab** fil:

```
/dev/hda3 /home ext2 defaults,noatime 0 0
```

2.13.1.2. Hvordan man finder de skyldige programmer

Nogen gange starter disken bare, uden at man ved hvorfor. Det kan være ret frustrerende.

Som en del af DIKU Linux kerne kurset (http://www.diku.dk/teaching/2002f/336/) blev en patch til 2.4.18 kernen udviklet.

Denne patch får kernen til at logge alle filoperationer. Det logger PID, tid, programnavn, filnavn osv.

Få patchen her (http://www.agol.dk/quietlinux/noisylog.patch).

Dette brugerprogram (http://www.agol.dk/quietlinux/do_noisy_logging.c) slår logging til og fra. Oversæt det med **cc -o do_noisy_logging -I /usr/src/linux/include do_noisy_logging.c**.

Programmet køres som **./do_noisy_logging 0** eller **./do_noisy_logging 1** for at slå logging fra hhv. til.

2.14. Kickstart

Kickstart er en smart måde at installere mange maskiner ens. Man kan ud fra en maskine gemme opsætningen i en fil og få KickStart systemet under Red Hat og Mandrake til at installere den næste maskine med samme opsætning. Her er et par steder hvor man kan finde mere dokumentation om det:

Mark's Kickstart Examples http://www.linuxgazette.com/issue43/nielsen.kickstart.html

Red Hat Linux 7.1: The Official Red Hat Linux Customization Guide http://ftp.linux.cz/pub/linux/redhat-cz/7.1/disks/disk2/doc/RH-DOC/rhl-cg-en-7.1/s1-kickstart2-options.html

Red Hat Linux KickStart HOWTO http://mirrors.sunsite.dk/ldp/HOWTO/KickStart-HOWTO.html

2.15. Sætte tiden

Skal du sætte tiden på din maskine så kan du gøre dette med **date -s 13:45:00**, men det kan også betale sig at se på NTP-systemet i Afsnit 5.7, idet det giver en meget bedre kontrol over at alle maskiner viser ens tid.

2.16. Oprette enheder

Hvis man skal oprette en enhed (det kan for eksempel være hvis det er lykkedes for en fjollet systemadministrator at slette `/dev/null`) skal man bruge kommandoen **mknod**. Hvis vi tager eksemplet med `/dev/null`, så kan vi på et fungerende system se at det er en tegn-enhed (eng.: character device) med primærnummeret 1 og sekundærnummeret 3:

```
[tyge@hven ~]$ ls -l /dev/null
crw-rw-rw-    1 root     root       1,   3 jan 21 21:22 /dev/null
```

Det er c'et i første søjle der fortæller os at det er en tegn-enhed. Alternativet er at det er en blok-enhed, hvilket markeres med et b i første søjle. Vi kan nu ovre på systemet, hvor `/dev/null` er forsvundet genoprette enheden:

```
[root@saltholm ~]# mknod /dev/null c 1 3
[root@saltholm ~]# ls -l /dev/null
crw-r--r--    1 root     root       1,   3 jan 21 21:25 /dev/null
```

Nu er enheden oprettet, men ikke med de helt rigtige rettigheder. Det kan vi rette op på med **chmod**:

```
[root@saltholm ~]# chmod go+w /dev/null
[root@saltholm ~]# ls -l /dev/null
crw-rw-rw-    1 root     root       1,   3 jan 21 21:26 /dev/null
```

2.17. Tips og tricks

Der er meget mere i systemadministration, end dette korte kapitel har præsenteret. Til Linux findes der især to dokumenter, som er gode at læse, nemlig "System Administrator's Guide" (SAG) og "Network Administrator's Guide" (NAG) - begge kan hentes fra http://sunsite.dk/ldp og følger med Red Hat-distributionen. De er en del af "Linux Documentation Project" (LDP). Red Hat's hjemmeside

indeholder endvidere en række gode tricks og nyttige oplysninger. De bedste bøger om Unix systemadministration er uden tvivl Æleen Frischs "Essential System Administration" udgivet af O'Reilly, og "UNIX System Administration Handbook" (Evi Nemeth, Garth Snyder, Scott Seebass og Trent R. Hein) udgivet af Prentice Hall PTR.

Vil du vide mere om sikkerhedskopiering under Linux, så kig i "The Linux System Administrators' Guide": http://www.linuxdoc.org/LDP/sag/index.html.

Kapitel 3. Filsystemer

Linux bruger som standard et filsystem kaldet *ext2* (Extended file system version 2), men kan også læse fra og skrive til mange andre filsystemer, f.eks. FAT-baserede filsystemer, der bruges af DOS og Windows.

Generelt formateres et filsystem under Linux ved at benytte kommandoen **mkfs** (MaKe File System). Har du en uformateret partition (her `/dev/hda5`), som du ønsker at formatere, sker formateringen ved at skrive (som root):

```
[root@linus /root]# mkfs /dev/hda5
```

Programmet **mkfs** finder selv ud af, hvilket filsystem du har valgt til den ønskede partition, idet denne oplysning findes i partitionstabellen.

Når din computer går ned (ja, det sker, f.eks. når strømmen går), kan du komme ud for at dine filsystemer er i en fejlbehæftet tilstand. Det kan dog ofte reddes af programmet **fsck** (File System CheckK). Faktisk køres dette program altid ved opstart for at sikre at alle filsystemer er i en god tilstand.

Vi du køre filtjek manuelt på en ext2-partition, så kan du direkte bruge **e2fsck**:

```
[root@linus /root]# /sbin/e2fsck /dev/hda5
```

3.1. Defekte blokke på harddisken

Hvis man har en harddisk med defekte blokke på så bliver man nødt til at finde dem og få filsystemet til at gå uden om. I det følgende gives en vejledning i at gøre dette.

3.1.1. Starte maskinen uden om harddisken(e)

Hvis du har en Red Hat-cd-rom, kan du starte maskinen fra den. Det kan kræve en ændring i opsætningen af din BIOS-opsætning, som vi dog ikke vil komme ind på her. Når der kommer en LILO-promt frem skriver du: "linux rescue".

Redningsproceduren (eng. rescue) vil bede om at få sprog og tastatur valgt – eksakt som det gøres ved en egentlig installation. Bare rolig – det er ikke en installation vi starter på. Når redningsproceduren beder om at montere rod-filsystemet under `/mnt/sysimage` skal der bare svares "no" (nej) hvis man ikke tør mounte harddisken, og "yes" hvis man kører rescue-CD'en for at redde data på harddisken eller fx. vil editere en system-konfigurationsfil, inden man genstarter systemet. Man svarer et eller andet, og derefter får man en kommandofortolker (et kommando-linie prompt. Ingen grafiske user-interface ting her).

Hvis du har en rednings-CD eller et "bootable business card" fra fx. lnx-bbc.org (Linux - Bootable Business Card projektet) (http://lnx-bbc.org/) som er en CD-ROM i visitkortstørrelse, kan du også boote fra fra den. Når systemet er startet, får du besked om, hvad systemets brugernavn (login-name eller bare login) og password (adgangskode) er. På rescue CD fra http://www.sysresccd.org/ (http://www.sysresccd.org/) er der faktisk slet ikke nogen adgangskode, man er logget på med det samme.

3.1.2. Hvilke filsystemer skal tjekkes?

En rednings-CD (rescue CD) kan give dig en liste over partitioner og filsystemer på den harddisk, som er direkte koblet på systemet:

```
fdisk -l /dev/hda

# hvis det er en IDE eller FAST IDE disk, eller hvis det er SCSI:

fdisk -l /dev/sda
```

Den kommando vil give dig en liste med de tilgængelige filsystemer. Den kan for eksempel se sådan ud:

```
   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1        12     96358+   b  Win95 FAT32
/dev/hda2            13        63    409657+  83  NTFS
/dev/hda3            64       127    514080   8e  Linux LVM
/dev/hda4           128      3722  28876837+   5  Extended
/dev/hda5           179      1198   8193118+  83  Linux
/dev/hda6          3239      3279    329301   82  Linux swap
```

Nu kan du så mounte disse partitioner og arbejde på dem.

Hvis du er interesseret i at se filerne på et Linux filsystem, så kan du mounte det som ext2, også selv om det er ext3. Forskellen er, at hvis du kører fsck (File System Consistency checK) så er det hurtigere, hvis man har ext3 og en gyldig "journal-file", som ofte ligger udenfor partitionen og oftest heldigvis ikke er beskadiget.

Hvis du kører en rescue-CD, som automatisk mounter dine disk-partitioner for dig (som Knoppix kan gøre) så vil du typisk være interesseret i de filsystemer, der er monterede som `/mnt/`-et-eller-andet-tal, og som er af typen "ext2". Vi kan foreksempel være interesserede i filsystemet på partitionen `/dev/hda2`.

Du kan bruge kommandoen **mount** til at få oplysninger om hvilke filsystemer der er monterede (eng.: mounted):

```
[root@lnx-bbc ~]# mount
```

```
/dev/sda2 on / type ext2 (rw) []
none on /proc type proc (rw)
/dev/sda3 on /boot type ext2 (rw) []
/dev/vg1/lv_home on /home type ext2 (rw)
/dev/vg1/lv_usr on /usr type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/hdb1 on /backup type ext2 (rw) []
```

Og hvis det ikke giver nok oplysninger, kan du bruge kommandoen **fdisk** til at spørge til partitioneringen af dine diske. IDE-diske (det typiske i skrivebordsmaskiner) har i Linux navnene `/dev/hd`-et-eller-andet-bogstav. Et linux-systems anden IDE-disk hedder således `/dev/hdb` og du kan spørge til dens opdeling med kommandoen:

```
[root@lnx-bbc ~]# fdisk -l /dev/hdb

Disk /dev/hdb: 16 heads, 63 sectors, 59560 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1     59560  30018208+  83  Linux
```

3.1.3. Finde og mærke dårlige blokke

Kør nu kommandoen **badblocks -o /tmp/blocks /dev/hda5** hvor `hda5` erstattes af den partition man ønsker at tjekke for defekte blokke, og `/tmp/blocks` er et arbitrært valgt filnavn.

Ældre udgaver af programmet **badblocks** skal have at vide hvor mange blokke der er på den partition der skal tjekkes. Den oplysning kan man få ved at bede **fdisk** om en liste med alle partitionerne på den relevante disk. Den disk partitionen `/dev/hda2` ligger på hedder `/dev/hda` (man stryger altså bare nummeret):

```
[root@lnx-bbc ~]# fdisk -l /dev/hda
...
   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1        70    529168+  82  Linux Swap
/dev/hda2   *        71       487   3152520   83  Linux
...
```

Det er tallet `3152520` vi er ude efter. Det føjes til som et sidste argument til **badblocks**:

```
[root@lnx-bbc ~]# badblocks -o /tmp/blocks /dev/hda 3152520
```

Derefter *kan* man læse filen med **cat /tmp/blocks** for at se om der blev fundet nogle defekte blokke. Hvis der ikke blev fundet nogen, så behøver man ikke at fortsætte.

Hvis der blev fundet defekte blokke skal du dernæst køre kommandoen **e2fsck -l /tmp/blocks /dev/hda5**. Den indsætter listen over de defekte blokke fra `/tmp/blocks` i filsystemets liste med defekte

blokke. Til sidst taster du "Ctrl-d" for at komme ud af kommandofortolkeren og genstarte maskinen. Husk at tage din cd-rom ud.

Herefter vil maskinen ikke længere prøve at bruge de defekte blokke **badblocks** fandt på harddisken. Du kan finde en mere information om de kommandoerne med **man badblocks** og **man e2fsck**.

Se også http://www.redhat.com/mailing-lists/ext3-users/msg02454.html for mere information om emnet.

3.2. Hjælp min partitionstabel er væk

Hvis man er særlig uheldig, så kan man miste sin partitionstabel ved et Windows-crash. Her er et par erfaringer. Skriv partitionstabellen (anvend `fdisk /dev/hda`, hvis din harddisk er `/dev/hda`) ud på papir og gem den sammen med udskrift af `/etc/fstab`. Hvis du har disse informationer så kan man typisk genskabe indhold af harddisken.

Hvis man ikke har styr på hvor på harddisken man havde de enkelte partitioner, så kan de programmer på http://inet.uni2.dk/~svolaf/utilities.htm være af *STOR* gavn. Med `findext2.exe` er det muligt at find Linux-partitioner af ext2/ext3-typen. Programmet kræver en DOS-diskette med en fortolker og det tager typisk et par timer at søge harddisken igennem for mulige partitioner.

Nedenfor er vist hvad programmet kan give af informationer. Programmet er kørt 29/6-2003. Dagen efter at partitionstabellen blev slettet. Tre linjer indikerer hvilke partitioner som er "rigtige" – bemærk datoer for hvornår de blev monteret og tjeket sidste gang – dagen før.

Konklusionen blev at der var tre partitioner, som skulle reddes ved at indtaste følgende grænser manuelt i **fdisk**.

```
Linuxpartition 1 : start 474  - slut 920
Linuxpartition 2 : start 951  - slut 2739
Linuxpartition 3 : start 2740 - slut 4865

Findext2, version 1.6
Copyright Svend Olaf Mikkelsen, 2002.

Searches for ext2/ext3 superblocks. False positives might be found.
The partition location can be wrong. Data MB may be used megabytes,
especially for superblock 0. Mount/Write/Check are date/time if
valid. Note that cylinders are numbered from 0. Linux fdisk
numbers from 1.

OS:  DOS 7.10  WINDOWS 4.10

Disk: 1   Cylinders: 4865   Heads: 255   Sectors: 63   MB: 38162
```

```
Start cylinder: 0   End cylinder: 4864

-PCyl N ID -----LBA -----Num ---MB -Start CHS- --End CHS-- BS  CHS

    0 - 83    48191 14329917  6997    2 254 60  894 253 59 B0   OK
      Data MB    1857               Actual end sector: 54
      Mount 2002.07.16  19:55:28    Superblock 0  Block 0
      Write 2002.07.16  19:55:28    Superblock LBA 48193
      Check 2002.07.16  16:36:30    Superblock CHS 2/254/62
      Blocks:  KB: 4    First: 0    Per group: 32768
      Ext3
      Searched        100   0  1

    0 - 83       63 14329917  6997    0   1  1  891 254 63 B5   OK
      Data MB    142                Actual end sector: 58
      Mount                    0     Superblock 9  Block 294912
      Write 2002.07.16  16:36:36    Superblock LBA 2359359
      Check 2002.07.16  16:36:30    Superblock CHS 146/220/10
      Blocks:  KB: 4    First: 0    Per group: 32768
      Ext3
      Searched        200   0  1
      Searched        300   0  1
      Searched        400   0  1

    0 - 83  7620209  7164927  3498  474  85 45  920  84 44 B0   OK
      Data MB    2396               Actual end sector: 37
      Mount 2003.06.28  14:24:18    Superblock 0  Block 0
      Write 2003.06.28  14:24:18    Superblock LBA 7620211
      Check 2003.04.12  16:09:18    Superblock CHS 474/85/47
      Blocks:  KB: 4    First: 0    Per group: 32768
      Ext3

    0 - 83  7614873  7164927  3498  474   1  1  919 254 63 B    OK
      Data MB    87                 Actual end sector: 56
      Mount                    0     Superblock 1  Block 32768
      Write 2003.04.12  16:09:21    Superblock LBA 7877017
      Check 2003.04.12  16:09:18    Superblock CHS 490/82/2
      Blocks:  KB: 4    First: 0    Per group: 32768
      Ext3
      Searched        500   0  1
      Searched        600   0  1
      Searched        700   0  1
      Searched        800   0  1
      Searched        900   0  1

    0 - 83 15277878 28740222 14033  951   1  1 2739 254 63 B0   OK
      Data MB    13873              Actual end sector: 57
      Mount 2003.06.28  14:24:33    Superblock 0  Block 0
      Write 2003.06.28  23:05:59    Superblock LBA 15277880
      Check 2002.07.17  00:43:41    Superblock CHS 951/1/3
      Blocks:  KB: 4    First: 0    Per group: 32768
      Ext3
      Searched       1000   0  1
```

```
  Searched       1100   0  1
  Searched       1200   0  1
  Searched       1300   0  1
  Searched       1400   0  1
  Searched       1500   0  1

0 - 83 25415665  8388740  4096 1582  13 17 2104  57 54 B    OK
  Data MB    4096
  Mount          -1535052965     Superblock 30168  Block-1241421144
  Write 2020.12.31  90:03:06     Superblock LBA 25415665
  Check 1997.05.24  00:00:14     Superblock CHS 1582/13/17
  Blocks:  KB: 1    First: 0     Per group:-1024238801
  Ext3
  Searched       1600   0  1
  Searched       1700   0  1
  Searched       1800   0  1
  Searched       1900   0  1
  Searched       2000   0  1
  Searched       2100   0  1
  Searched       2200   0  1
  Searched       2300   0  1
  Searched       2400   0  1
  Searched       2500   0  1
  Searched       2600   0  1
  Searched       2700   0  1

0 - 83 44018163 34147192 16673 2740   1  1 4865 144 58 B0   OK?
  Data MB    6096
  Mount 2003.06.28  14:24:34     Superblock 0  Block 0
  Write 2003.06.28  23:05:59     Superblock LBA 44018165
  Check 2002.09.22  09:51:38     Superblock CHS 2740/1/3
  Blocks:  KB: 4    First: 0     Per group: 32768
  Ext3
  Searched       2800   0  1
  Searched       2900   0  1
  Searched       3000   0  1
  Searched       3100   0  1
  Searched       3200   0  1
  Searched       3300   0  1
  Searched       3400   0  1
  Searched       3500   0  1
  Searched       3600   0  1
  Searched       3700   0  1
  Searched       3800   0  1

0 - 83 61836487     1952     0 3849  36 35 3849  67 33 B0   OK
  Data MB    1
  Mount 2002.09.10  20:31:45     Superblock 0  Block 1
  Write 2002.09.10  20:31:45     Superblock LBA 61836489
  Check 2002.09.10  20:31:45     Superblock CHS 3849/36/37
  Blocks:  KB: 1    First: 1     Per group: 8192

0 - 83 61843559     1632     0 3849 148 51 3849 174 44 B0   OK
```

```
Data MB    1
Mount 2002.09.10  20:31:50     Superblock 0  Block 1
Write 2002.09.10  20:31:51     Superblock LBA 61843561
Check 2002.09.10  20:31:50     Superblock CHS 3849/148/53
Blocks:  KB: 1    First: 1     Per group: 8192
Searched       3900   0  1
Searched       4000   0  1
Searched       4100   0  1
Searched       4200   0  1
Searched       4300   0  1
Searched       4400   0  1
Searched       4500   0  1
Searched       4600   0  1
Searched       4700   0  1
Searched       4800   0  1
```

Langt hurtigere er `findpart.exe` som også kan give partitionsgrænserne direkte, men den viser ikke hvornår de enkelte partitioner blev anvendt sidst.

```
Findpart, version 4.33 - for Windows 95/98/ME/NT/2000/XP.
Copyright Svend Olaf Mikkelsen, 2003.

Searches for partitions type 01, 04, 06, 07, 0B, 0C, 0E, 82, 83,
plus Fdisk F6 and Lilo sectors. Information based on bootsectors
is marked B. If the disk is larger than supported by BIOS, the
supported part of the disk is examined. Disks are numbered from 1.

OS:  Windows 4.10

Disk: 1   Cylinders: 4865   Heads: 255   Sectors: 63   MB: 38162

-PCyl N ID -----Rel -----Num ---MB -Start CHS- --End CHS-- BS  CHS
    0 - 0B       63  7616512  3719    0   1  1  474  28  1 B    OK
    Fdisk F6 sector                   3   0  1
    Fdisk F6 sector                   5   1  1
  474 1 83       63 70550487 34448  474   1  1 1023 254 63 NB   NB?
  920 1 82       63   497952   243  920   1  1  950 254 63       OK
  920 2 05  7663005 62887545 30706  951   0  1 1023 254 63 474  NB?
  951 1 83       63 28740222 14033  951   1  1 2739*254 63 OK 3 OK
  951 2 05 36403290 34147260 16673 2740#  0  1 1023 254 63 474  NB?
 2740 1 83       63 34147197 16673 2740#  1  1 1023 254 63 NB   NB?
 2740 - 83       63 34147192 16673 2740   1  1 4865 144 58 B0 3 OK?
    Fdisk F6 sector               4407   0  1
    Fdisk F6 sector               4407   1  1
```

Det skal bemærkes at de værktøjer som er nævnt ovenfor regner start og slut sektorer fra 0, hvor **fdisk** regner fra 1 og frem. Derfor skal man trække en fra hvis man anfører de grænser i `/etc/fstab`.

Det kan nævnes at den rigtige partitionstabel var

```
Disk /dev/hda: 40.0 Gb, 40020664320 byte
```

```
255 hoveder, 63 sektorer/spor, 4865 cylindre
Enheder = cylindre af 16065 * 512 = 8225280 byte

    Enhed Opstart Start       Slut   Blokke   Id  System
/dev/hda1   *         1       474   3807373+   c  Win95 FAT32 (LBA)
/dev/hda2           475      4866  35275275    5  Udvidet
/dev/hda5           475       921   3590496   83  Linux
/dev/hda6           922       951    240943+  82  Linux swap
/dev/hda7           952      2740  14370111   83  Linux
/dev/hda8          2741      4866  17073598+  83  Linux
```

og den tilsvarende `/etc/fstab` var

```
/dev/hda1 /mnt/windows vfat iocharset=iso8859-15,codepage=850,umask=0 0 0
/dev/hda5 / ext3 defaults 1 1
/dev/hda6 swap swap defaults 0 0
/dev/hda7 /home ext3 defaults 1 2
/dev/hda8 /storage ext3 defaults 1 2
```

Endelig skal det nævnes at bl.a. Mandrake har en mulighed for at boote op på første cd-rom og vælge *rescue* og genskabe boot-information, hvis det er muligt. Dette er ikke altid muligt...

Andre relevante links er http://www.stud.uni-hannover.de/user/76201/gpart/ som har Linuxprogrammet **gpart** som kan nogenlunde det samme.

3.3. ReiserFS

ReiserFS er et af de nyere filsystemer til Linux, og har kun været i de officielle kerner siden version 2.4.1.

ReiserFS er en anden type filsystem end den der kendes fra Linux' ext2, og DOS/Windows' FAT-drev, idet ReiserFS er et journaliserende filsystem. Det betyder, at alle filer til en hver tid er opdaterede. Det betyder, at hvis din computer går ned pga. en strømafbrydelse, vil du ikke miste data. Endvidere betyder det, at de partitioner som er formateret ved hjælp af ReiserFS, ikke skal tjekkes ved opstart. Hvis du har meget store harddiske vil det betyde at du får en meget kortere opstartstid. Dette kan være vigtigt, hvis din computer er server med f.eks. 100 Gb harddisk (ikke ualmindeligt i den virkelige serververden).

Tidligere versioner af ReiserFS ville ikke virke med Lilo-boot fra en ReiserFS-partition. Det kunne løses ved at lægge `/boot` på en ext2-partition, og hvis `/boot` er en lille selvstændig partition, kan `/`-partitionen være ReiserFS-formateret. I de nyeste versioner af ReiserFS (november 2000) bør denne begrænsning være væk, blot man monterer boot-partitionen med et ekstra flag, "`notail`":

```
/dev/hda2 on / type reiserfs (rw,notail)
```

ReiserFS følger med Mandrake (fra version 7.1) og SuSE (fra 6.4) og under disse distributioner er det let at vælge formatering med ReiserFS i stedet for ext2 under installationen. Men selv under andre distributioner er det ikke svært. Du skal sikre dig, at kernen understøtter ReiserFS som filsystem. Måske skal du omkonfigurere din kerne (se også Afsnit 4.6). Det kan godt betale sig at få ReiserFS oversat som et kernemodul. Modulsystemet vil typisk selv finde ud af, at modulet skal bruges, dvs. når du begynder at bruge kommandoer som involverer en ReiserFS-baseret partition, vil modulet automatisk blive indlæst.

Først og fremmest skal du have en partition, som er tom. Du kan oprette en partition til formålet. Her kan du bruge programmer som **fdisk** og **cfdisk**. Når du har en partition klar, skal du formatere den. I eksemplet benytter vi partitionen `/dev/hda5`, som vi antager allerede er oprettet.

```
[root@linus root]# mkreiserfs /dev/hda5
<-----------MKREISERFS, 1999----------->
ReiserFS version 3.5.18
Block size 4096 bytes
Block count 325576
First 16 blocks skipped
Super block is in 16
Bitmap blocks are :
   17, 32768, 65536, 98304, 131072, 163840, 196608, 229376, 262144, 294912
Journal size 8192 (blocks 18-8210 of device 0x3:0x5)
Root block 8211
Used 8221 blocks
ATTENTION: ALL DATA WILL BE LOST ON '/dev/hda5'! (y/n) y
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..

ReiserFS core development sponsored by SuSE Labs (suse.com)

Journaling sponsored by MP3.com.

Item handlers sponsored by Ecila.com

To learn about the programmers and ReiserFS, please go to
http://www.devlinux.com/namesys

Have fun.
[root@linus ~]#
```

Du er nu klar til at montere din partition. Følgende vil montere partitionen `/dev/hda5` som `/home`.

```
[root@linus ~]# mount -t reiserfs /dev/hda5 /home
Checking ReiserFS transaction log (device 03:35)  ...
Relayed 0 transaction in 0 seconds
ReiserFS version 3.5.18
```

Naturligvis kan du indsætte din nye partition i filen `/etc/fstab` med typen *reiserfs*. Derved vil partitionen blive monteret under opstart.

En glimrende oversigtsartikel, der sammenligner det gamle ext2, det nyere ext3, ReiserFS og endelige XFS kan læses på http://www.linuxgazette.com/issue68/dellomodarme.html. En sammenligning kan findes på http://oregonstate.edu/~kveton/fs/ (http://oregonstate.edu/~kveton/fs/). Se også http://www.namesys.com/benchmarks/benchmark-results.html.

3.4. Migrering fra ext2 til ext3

Hvis du ønsker at flytte eksisterende ext2-partitioner over til ext3, så kan man på http://www.symonds.net/~rajesh/howto/ext3/index.html finde en vejledning til dette. Det er egentlig nemt. Først skal man sikre sig at ens Linux distribution er klar til ext3 - det er alle nyere Linux-distributioner i dag. Derefter skriver man som root **tune2fs -j /dev/hda1** hvis det er `/dev/hda1` der skal flyttes til ext3. Dette gentages for hver af de ext2-partitioner, der skal flyttes. Endelig skal med i `/etc/fstab` rette alle steder med ext2 til ext3 og genstarte maskinen.

Denne øvelse kræver dog at din Linux-kerne er oversat med ext2 og ext3-understøttelse. Dette er tilfældet med de nyere Linux-distributioner, men pas på her.

3.5. Software-RAID

3.5.1. Hvad er RAID?

RAID er en metode til at slå flere diske sammen til én *volume* (dvs. én logisk disk); det kan gøres for at opnå et større brugerdrev eller for at højne sikkerheden ved redundans.

Derfor er der flere måder at kombinere diske til et RAID, og man har derfor vedtaget nogle RAID-levels.

3.5.2. Forskel mellem hardware- og software-RAID

RAID kan laves hardware- eller software-mæssigt. Det sikreste er via hardware eller med andre ord en RAID-controller.

3.5.2.1. Hardware-RAID

Ved at lade hardwaren styre diskene og "narre" styresystemet til at tro at der er tale om et eller flere logiske drev, behøver man ikke at bekymre sig så meget om opsætning af Linux, da hardwareproducenterne som regel har en brugervenlig opsætningsmenu installeret i kortets BIOS.

Ulempen er at disse kort som regel er dyre at anskaffe og ofte benytter sig af de noget dyrere scsi-diske. Med andre ord: En RAID-controller er ofte "bare" en avanceret scsi-controller.

3.5.2.2. Software-RAID

Her lader man et stykke software styre RAID, hvilket sparer en RAID-controller og det er muligt at blande scsi-diske med IDE eller kun benytte sig af IDE.

Ulempen er at man så selv skal sætte RAID op. På Linux er det lidt besværligt.

3.5.3. Opsætning af spejlede diske.

Et disk-spejl er hvis man ønsker 2 diske som en logisk enhed, hvor begge diske indeholder samme data. Sikkerheden er her at hvis en disk står af, så kører den anden videre.

Hvis man ønsker yderligere sikkerhed, kan man benytte en ekstra reservedisk som vil kunne erstatte en nedbrudt disk i spejlet. På den måde undværer man kun sikkerheden i den tid som det tager RAID-softwaren at opbygge et nyt spejl på reservedisken.

3.5.3.1. Fremgangsmåde

Start i enkeltbrugertilstand - (**linux s** ved opstart eller **init 1**).

Opret partitionen på begge diske med **fdisk**. Filsystemet skal være "Linux RAID autodetect". Skriv enhedens nummer ned.

Opret en `/etc/raidtab`-fil, med indhold som kunne se således ud:

```
raiddev               /dev/md0
raid-level                  1
nr-raid-disks               2
chunk-size                  8
persistent-superblock       1
nr-spare-disks              0
```

```
    device            /dev/sdc1
    raid-disk                 0
    device            /dev/sdd1
    raid-disk                 1
```

Opret RAID-enheden med kommandoen **mkraid /dev/md0**.

Opret filsystemet til din logiske enhed `/dev/md0` med kommandoen **mkfs /dev/md0**.

Montér det logiske drev med f.eks. **mount /dev/md0 /spejl**

Opret en linje i `/etc/fstab` svarende til det nye spejl.

```
/dev/sda1  /           ext2    defaults        1 1
/dev/md0   /Spejl      ext2    defaults        1 2
/dev/sda5  /home       ext2    defaults        1 2
/dev/cdrom /mnt/cdrom  iso9660 noauto,owner,ro 0 0
/dev/sdb5  /vol2       ext2    defaults        1 2
/dev/sdb1  swap        swap    defaults        0 0
/dev/fd0   /mnt/floppy ext2    noauto,owner    0 0
none       /proc       proc    defaults        0 0
none       /dev/pts    devpts  gid=5,mode=620  0 0
```

Har du tidligere haft et RAID-system sat op på diskene, skal du bruge **mkraid --force /dev/md0** til at initialisere diskene igen, også selvom de er slettede med fdisk. Husk dog at tage backup (f.eks. **tar pcfz data.tar.gz /mount/data**, før du anvender **mkraid --force /dev/md0**, da RAID-diskene slettes derved.

3.5.4. Opsætning af linear RAID

Betegnelsen *Linear RAID* betyder, at man ønsker at slå flere diske sammen til én logisk enhed og udnytte den samlede disk kapacitet. Det giver ikke større data-sikkerhed - kun større samlet plads. Et eksempel på en `/etc/raidtab` hvor 2 diske som slås sammen er følgende:

```
raiddev               /dev/md0
raid-level            linear
nr-raid-disks         2
chunk-size            8
persistent-superblock 1
nr-spare-disks        0
    device            /dev/sdc1
    raid-disk         0
    device            /dev/sdd1
    raid-disk         1
```

3.5.5. raidhotadd og raidhotremove

Det er muligt at tilføje og fjerne diske fra RAID-systemet medens systemet kører. Det kan være en god ting, hvis filsystemet af en eller anden grund er gået i stykker på den ene af diskene. Det kan ske uden at disken rent fysisk har fejlet.

Fjern disk

F.eks. fjern disken `/dev/hdd1` fra RAID-systemet:

```
[root@linus /etc]# raidhotremove -a /dev/md0 /dev/hdd1
```

Nu kan man arbejde på disken med **fdisk**.

Tilføj disk

Disken som er blevet repartitioneret, kan tages i brug således:

```
[root@linus /etc]# raidhotadd -a /dev/md0 /dev/hdd1
```

RAID-systemet vil automatisk blive opdateret, og et evt. spejl vil blive genopbygget medens produktionen kører, performance-nedgangen er lille selv ved IDE-diske.

3.6. Harddisk-tuning

Har du brug for at få ekstra "tryk" på din IDE-harddisk, så kan du bruge **hdparm** til at tune hastigheden. Her er givet de to vigtigste parametre (bruge DMA og lookahead), som nærmest alle nyere diske understøtter. Der kan tunes endnu mere - have fun! :-)

Når det virker fint, kan og bør du lægge kommandoerne ind i filer, der køres når systemet startes op. I Red Hat f.eks. i bunden af `/etc/rc.local` (SuSE `/etc/rc.d/boot.local`).

```
[root@linus /root]# /sbin/hdparm -tT /dev/hda

/dev/hda:
 Timing buffer-cache reads:   128 MB in  3.21 seconds = 39.88 MB/sec
 Timing buffered disk reads:  64 MB in 10.14 seconds =  6.31 MB/sec
```

Den første og sidste kommando måler hastigheden på harddisken. Der kommer to tal. Det første "buffer-cache" er reelt et mål for processor- og bushastighed, mens det andet tal angiver I/O fra disken.

For de fleste diske er den standardopsætning du har fin, men får du væsentligt mindre på det andet tal (se ovenfor), så skal du i gang med at lege :-)

Start med at notere hvordan dine parametre er nu (og som virker):

```
[root@linus /root]# /sbin/hdparm /dev/hda

/dev/hda:
 multcount    = 16 (on)
 I/O support  =  0 (default 16-bit)
 unmaskirq    =  0 (off)
 using_dma    =  0 (off)
 keepsettings =  0 (off)
 nowerr       =  0 (off)
 readonly     =  0 (off)
 readahead    =  8 (on)
 geometry     = 39703/16/63, sectors = 40020624, start = 0
```

Nogle af de parametre du kan dreje på er følgende:

- -u[0|1] Styrer interrupt-masker for disken. Det er noget der giver en del.
- -m[tal] »Multiple sector«-tilstand (dvs. IDE Block Mode), er om man kan hente mere data pr. gang. De fleste chipsæt understøtter nu dette. 2, 4, 8, 16 og 32 er lovlige værdier. 16 eller 32 er nok interessant for de fleste.
- -d[0|1] er tilkobling/fravalg af DMA-overførsel. Dette genkendes oftest automatisk, og sættes default til -d1 hvis det er muligt.
- -X[tilstand] sættes til højest mulige hastighed pr. default. Anvend -X66 for UltraDMA-tilstand 2, -X67 for UltraDMA-tilstand 3 osv. Dog er UltraDMA-tilstand 3, 4 og 5 meget eksperimentelle.
- -c[0|1|2|3] E)IDE 32-bit I/O-support. 0 for at fravælge 32-bit I/O-support, 1 for at tilkoble 32-bit dataoverførsel og 3 for 32-bit dataoverførsel med synkronisering (mange chipsæt kræver dette.

```
[root@linus /root]# /sbin/hdparm -u1 -m16 -c3 -d1 -X66 /dev/hda

/dev/hda:
 setting 32-bit I/O support flag to 3
 setting multcount to 16
 setting unmaskirq to 1 (on)
 setting using_dma to 1 (on)
 setting xfermode to 66 (UltraDMA mode2)
 multcount    = 16 (on)
 I/O support  =  3 (32-bit w/sync)
 unmaskirq    =  1 (on)
 using_dma    =  1 (on)

[root@linus /root]# /sbin/hdparm -tT /dev/hda
/dev/hda:
 Timing buffer-cache reads:   128 MB in  3.23 seconds = 39.63 MB/sec
 Timing buffered disk reads:  64 MB in  5.76 seconds = 11.11 MB/sec
```

I eksemplet giver jeg kommandoer som passer til de nyere Ultra DMA 66-controllere, og får derved en lidt højere hastighed. Målefunktionen bør dog køres en del gange før man kan få en statistisk korrekt vurdering af ydelsen. Du skal være opmærksom på, at du kan låse din maskine, hvis du angiver mere end hvad din maskine kan yde. I praksis ser du nok et reboot hvis du går for langt. Tilføj den endelige optimale opsætning til `/etc/rc.local` (SuSE `/etc/rc.d/boot.local`) eller tilsvarende opstartsfil, når du er sikker på at maskinen kører godt. Se også **man hdparm**.

Med Red Hat kan du justere i filerne: `/etc/sysconfig/harddiskhd[a-h]` og så gør `/etc/rc.sysinit` resten ved systemstart. Udfør **cp /etc/sysconfig/harddisks /etc/sysconfig/harddiskhdX**, hvor X er a, b, c osv. for hda, hdb, hdc osv. Der er følgende parametre:

```
USE_dma=1
# det samme som -d1
# brug DMA   fra = 0, til = 1

lookahead=1
# det samme som -A1

MULTIPLE_io=16
# det samme som -m16

EIDE_32bit=3

EXTRA_params= -X69
# -X69 UDMA5 for ATA/100
# UDMAX er -X(64 + X) hvor X er et number 1-5.
```

Læs også Red Hat 7.1-filen `/etc/sysconfig/harddisks` hvor generel opsætning kan laves for alle diske.

Samme program, **hdparm**, kan bruges til at få harddisken til at spinne ned i hastighed - f.eks. efter 5 sekunder. Det betyder mere slid på harddisken, men din maskine bliver stille.

```
[root@linus /root]# /sbin/hdparm -S 1 /dev/hda
```

Hvis man har et eller flere programmer der jævnligt tjeker om nogle filer er ændret skal disse filers access-tid også opdateres jævnligt - det giver skrivninger til disken selvom filsystemets egentlige data ikke ændres (kun dets meta-data). Det kan være en god idé at montere disken via `/etc/fstab` noget i stil med

```
/dev/hda1  /  ext2  defaults,noatime  1 1
```

Kernens IDE-driver skal understøtte det IDE-chipsæt, der anvendes af bundkortet, og ikke alle IDE-chipsæt er understøttet med (U)DMA. Så hvis en given funktion ikke tillades af **hdparm**, så kan det være forklaringen.

hdparm har kun begrænset anvendelse med scsi. Til scsi skal man dels se på driveren, dvs. tilføje eventuelle modulparametre. Symbios scsi-driverne kan dog fintunes via **/proc/scsi** under drift. Et nyttigt grafisk værktøj til at tune scsi-diske er **scsi-config**, selvom der normalt ikke er så meget at "tune".

Du kan måske også være interesseret i at se mere på http://www.iozone.org/ hvor der er et program til at benchmark-teste harddisk-ydelse.

Mere info om **hdparm** kan finds på http://linux.oreillynet.com/pub/a/linux/2000/06/29/hdparm.html.

3.7. Hvilke filer har programmet åbne?

3.7.1. lsof

Ofte vil dit program have filer åbne som du måske ikke lige vidste. Med **lsof** kan du se hvad der sker. Vi kan f.eks. se hvad programmet arbejder med. Som eksempel kan vi se hvad der sker, når man har **ping** kørende. Først finder vi den *pid* - proces-ID - som **ping** kører med. Der er mindst to muligheder, enten med **ps aux** eller **pidof**.

```
[root@linus /root]# ps aux | grep ping
pto        9149  0.0  0.1  1264  228 pts/4    S    Apr29   0:00 ping eric
root       9418  0.0  0.4  1360  516 ttyp0    S    00:06   0:00 grep ping
[root@linus /root]# pidof ping
9149
[root@linus /root]# lsof -p 9149
COMMAND PID USER  FD TYPE DEVICE    SIZE   NODE NAME
ping   9149 root cwd  DIR    3,7    4096  30919 /home/pto/tmp
ping   9149 root rtd  DIR    3,5    4096      2 /
ping   9149 root txt  REG    3,5   17968 114094 /bin/ping
ping   9149 root mem  REG    3,5  340663  16023 /lib/ld-2.1.3.so
ping   9149 root mem  REG    3,5  169720  16071 /lib/libresolv-2.1.3.so
ping   9149 root mem  REG    3,5 4101324  16030 /lib/libc-2.1.3.so
ping   9149 root mem  REG    3,5  246652  16061 /lib/libnss_files-2.1.3.so
...
```

Så man kan se, at programmer har fat i mange filer. Det skal nævnes at **pidof** kræver at SysVinit-pakken er installeret og stien til **pidof** varierer desværre alt efter Linux-distribution. Det skal også nævnes, at **ps aux** bruges på en Linux-maskine, mens mange andre Unix-varianter anvender **ps -ef**.

Tilsvarende kan man have stor glæde af at kunne se, hvilke programmer der har filer åbne under et givet katalog i filtræet. Det er meget relevant ved afmontering af f.eks. cd-rom-drev. Prøv følgende:

```
[root@linus /root]# lsof +D /var/spool
COMMAND  PID USER   FD   TYPE DEVICE SIZE  NODE NAME
atd      459 root  cwd    DIR    3,5 4096 32594 /var/spool/at
crond    473 root  cwd    DIR    3,5 4096    12 /var/spool/
```

```
lpd     9299 root    4w   REG    3,5    5  4507 /var/spool/lpd/lpd.lock
```

Eksemplet viser, at der er tre programmer **at**, **crond** og **lpd** som pt. anvender filer under `/var/spool`. Meget nyttige ting for en systemadministrator.

3.7.2. fuser

Et tilsvarende nyttigt program til nogenlunde samme formål er **/sbin/fuser DEVICE**. F.eks. kan man se hvilket proces-ID, som låser lydenheden `/dev/dsp` ved at skrive **/sbin/fuser /dev/dsp**. Med **/sbin/fuser -k /dev/dsp** kan man endda direkte dræbe den proces som låser enheden.

3.7.3. Afmontering af filsystem via /proc

Et alternativ til ovenstående er at anvende `/proc/`. Antag at man vil afmontere `/usr/src/postgresql-7.2.1/`

```
[root@linus /root]# umount /usr/src/postgresql-7.2.1/
umount: /usr/src/postgresql-7.2.1: device is busy
```

Hvem pokker er synderne?

```
[root@linus /root]# ls -l /proc/*/cwd|grep /usr/src/postgresql-7.2.1/
lrwxrwxrwx 1 postgres postgres 0 Apr 30 03:49 /proc/3943/cwd -> /usr/src/postgresql-7.2.1/
lrwxrwxrwx 1 postgres postgres 0 Apr 30 03:49 /proc/3944/cwd -> /usr/src/postgresql-7.2.1/
lrwxrwxrwx 1 postgres postgres 0 Apr 30 03:49 /proc/3950/cwd -> /usr/src/postgresql-7.2.1/
```

Aha! Det er således processerne 3943 3944 3950 som låser filsystemet.

```
[root@linus /root]# kill -9 3943 3944 3950
```

Nogen kunne sikkert lave et script, der gjorde dræbningen nemmere:

```
perl -e '$dir=shift; map {m:/proc/(\d+)/cwd -> $dir[/\n]: and do {print
"$1\n"; kill 9, $1 } } `ls -l /proc/*/cwd`' /usr/src/postgresql-7.2.1/
```

3.8. Fejlfinding

Nogle gange er programmer ret dårlige til at forklare, hvorfor det gik galt. Kommandoen **strace** er god til at fejlsøge med. Det gælder specielt ved fejl som:

- Segmentation fault (core dumped)

- File not found
- Access denied

Her vil **strace** ofte kunne give dig en idé om, hvad der skete lige inden fejlen indtrådte, og med lidt held kan du ud af dette gætte, hvad der gik galt. Måske mangler du en fil eller måske bruger programmet den forkerte version af et library.

strace står for system trace og viser alle de kald, der er til library-rutiner. Hvis man vil se, hvilke kald **ls** laver, så skriver man:

```
[tyge@hven ~]$ strace ls
```

Det går hurtigt op for een, at der kommer *ret* meget output. Da problemerne ofte er fil-relateret, så kan man nøjes med at få vist de fil-relaterede systemkald:

```
[tyge@hven ~]$ strace -e trace=file ls
```

Hvis programmet starter andre processer (f.eks. ved at kalde andre programmer), så kan man strace disse med '-ff'. Prøv at se forskellen på disse to:

```
[tyge@hven ~]$ strace time ls
[tyge@hven ~]$ strace -ff time ls
```

Hvis du strace'r programmer, der kører under X, så vil du ofte få utroligt meget output. Du kan proppe det i en fil ved at redirigere stderr:

```
[tyge@hven ~]$  strace -ff time ls 2>/tmp/fil
```

En anden mulighed er at starte **strace** fra en shell i **emacs**:

```
[tyge@hven ~]$ emacs
M-x shell
[tyge@hven ~]$ strace ...
```

Herved får du mulighed for at bladre i outputtet med det samme.

Kapitel 4. Linux-kernen

I dette kapitel ser vi nærmere på Linux-kernen. Først lidt teknisk om hvordan kernen virker, dernæst mere om hvordan processer håndteres. Det vises hvordan kernen oversættes og moduler håndteres og endelig ser vi nærmere på kerne 2.4.

4.1. Hovedkarakteristika ved Linux-kernen

- *Multi-tasking* Linux understøtter ægte multi-tasking. Alle processer kører helt uafhængigt af hinanden. Ingen processer behøver at tage højde for processor-tid til andre processer.
- *Multi-bruger tilgang* Linux tillader flere brugere at benytte systemet samtidigt.
- *Demand load executables* Kun de dele af et program, som faktisk er påkrævet for udførslen, bliver hentet ind i hukommelsen. Når en ny proces er skabt ved hjælp af **fork()**, afsættes der ikke straks hukommelse, men i stedet bruges hukommelsen fra forældre-processen af begge processer. Hvis den nye proces så på et tidspunkt søger adgang til en del af hukommelsen i "write mode", kopieres denne sektion før den ændres. Dette koncept kendes som "copy-on-write"; det er med til at øge hastigheden og sænke hukommelsesforbruget.
- *Paging* På trods af forsøg på at anvende fysisk hukommelse effektivt, kan det ske, at den tilstedeværende hukommelse er brugt op. Linux ser så efter 4 kb hukommelsessider, som kan frigives. Sider med deres indhold lagret på harddisk (f.eks. programfiler) frigives. Alle andre sider kopieres ud på harddisken. Hvis der herefter søges adgang til en af disse hukommelsessider, hentes den igen. Denne procedure kaldes "paging". Den adskiller sig fra "swapping", som anvendes i ældre Unix-systemer, hvor hele hukommelsen for en proces skrives til harddisken, hvilket er betydeligt mindre effektivt.
- *Dynamisk cache for harddisken* Brugere af MS-DOS og Windows er kendt med behovet for at reservere hukommelse af en bestemt størrelse til harddisk-cache-programmer som SMARTDRIVE. Linux justerer dynamisk størrelsen af cache-hukommelse som bruges, for at tilpasse sig den nuværende hukommelsessituation. Hvis der ikke er mere hukommelse tilbage på et givet tidspunkt, reduceres størrelsen af cachen for at frigive hukommelse. Når først hukommelse er frigivet, øges cache-området igen.
- *Delte biblioteker* – biblioteker (på engelsk "libraries") er samlinger af rutiner, som kan bruges af forskellige programmer. Linux har et antal standardbiblioteker, der typisk bruges af mange programmer der kører samtidigt. Det er derfor fornuftigt at indlæse disse biblioteker i hukommelsen én gang for alle og lade alle programmerne deles om den samme kode. Dette muliggøres med *delte biblioteker* (på engelsk "shared libraries"). Da disse først indlæses og sammenkædes med et program, når det kører og har brug for dem, kendes de også som dynamisk sammenkædede biblioteker (på engelsk "dynamically linked libraries" eller "DLLs").
- Understøttelse for *POSIX 1003.1-standard* og delvis *System V* og **BSD**. POSIX 1003.1 definerer et minimum interface for et operativsystem af Unix-typen. Dette interface er beskrevet i C-funktionsdeklarationer. Denne standard understøttes nu af alle nyere og relativt sofistikerede Unix-systemer. I dag understøtter Linux tilnærmelsesvist POSIX 1003.1.

- *Forskellige formater for kørbare filer* Da det meste software i dag er skrevet til Microsoft Windows, er det ønskværdigt at kunne køre programmer til disse systemmiljøer under Linux. Derfor er emulatorer for MS-DOS og Microsoft Windows under udvikling. Linux kan også køre programmer fra andre Intel-baserede Unix-systemer, som går under iBCS2-standard. Det inkluderer f.eks. mange kommercielle programmer brugt under SCO-Unix. iBCS2-emulering er endnu ikke en del af standard-kernen, men den kan hentes fra internettet.
-

  Understøttelse af nationale tastaturer og tegnsæt Under Linux kan mange nationale tastaturer og tegnsæt anvendes. ISO-standarden UTF-8 definerer blandt andet Latin1-tegnsættet, som indeholder de europæiske specialtegn med mere.

- *Forskellige filsystemer* Linux understøtter forskellige filsystemer. Det mest almindeligt brugte filsystem er Second Extended (ext2) File System. Dette understøtter filnavne op til 255 tegn, og har et antal træk der gør det mere sikkert end almindelige Unix-filsystemer. Det er dog ikke et *Journaling filesystem*, som giver endnu større sikkerhed imod datatab. Som noget nyt i kerne-version 2.4.1 understøttes det journaliserende filsystem ReiserFS, som sikrer en hurtig reetablering af systemet efter en ikke-planlagt nedlukning (f.eks. efter strømsvigt).

  Oftest anvendes ext2-diskformat, men Linux kan læse og skrive et hav af andre diskformater, FAT, vfat, NTFS (kun læse), UDF (DVD-format), ISO 9660 (cd-rom-format) og mange andre. I netværk kan man få adgang til filsystemet på andre systemer transparent ved hjælp af NFS (Network File System) eller SMB (Windows filshare).

- *TCP/IP, SLIP og PPP-understøttelse* Linux kan integreres i lokale Unix-netværk. I princippet kan alle netværkstjenester som NFS og Remote Login bruges. SLIP og PPP understøtter brugen af TCP/IP-protokollen over serielle linjer. Det betyder at link til internettet via telefonnetværk ved hjælp af et modem er muligt.
- *BSD-sockets* Netværkskommunikation kræver naturligvis midler til interproces-kommunikation mellem forskellige computere. Interfaces hertil er i BSD-sockets.
- *System V IPC* Linux bruger dette til at lave 'message queues', semaforer og 'shared memory'. Disse er klassiske varianter til interproces-kommunikation.
- *Virtuelle konsoller* Linux understøtter virtuelle konsoller. Tastekombinationen <ALT> + <Function key> bruges til at skifte imellem dem.
- *Multiprocessing* Fra kerne 2.0 understøtter Linux symmetrisk multiprocessing – samtidig kørsel af processer på et antal processorer – op til 16 processorer. I kerne 2.2-serien var man i praksis begrænset til 4 processorer, mens man med kerne 2.4-serien kan opnå skalérbar performance på 16 og 32 processorer.

4.2. Linux er ikke

Linux er ikke et real-time system. Bl.a. virtuel hukommelse gør det umuligt for Linux at overholde kravene til et "hard" real-time system, men det kan laves mere real-time end det er i dag, bl.a. ved at arbejde med scheduling'en og optimere interruptene. Der er bl.a. et projekt, RTLinux, som har lavet en overbygning på kernen til realtidsbrug.

Linux var tidligere (før kerne 2.0) en monolitisk kerne, mens man med kerne 2.0 fik adgang til at lave kerne-moduler. Dette var en nødvendig måde at kunne understøtte meget forskellig hardware på. To brugere med forskelligt lydkort anvender hver sit kernemodul, og derfor har de to brugere ikke unødvendig meget programkode læst ind i hukommelsen.

Linux er ikke et distribueret operativsystem – alting foregår på én maskine. Oven på Linux kan man så køre systemer såsom PVM og MPI til at afvikle programmer på mange maskiner. Både PVM og MPI starter processer på flere maskiner, udveksler data mellem maskinerne og udnytter den store fælles datakapacitet til at køre programmerne parallelt. Det kræver dog at programmerne er skræddersyet til det. Linux kan også snildt bruges til at lave et cluster af maskiner. Med MOSIX http://www.mosix.org kan man binde en række maskiner sammen, så de udefra ser ud som én stor maskine. MOSIX-clusteret fordeler selv programmerne efter laveste belastning af de enkelte maskiner, og man kan transparent flytte programmer fra én maskine til den næste. Det er særdeles elegant.

Linux er ikke sikret efter de strengeste sikkerhedsprincipper, men der er indbygget en vis beskyttelse mod uforvarende at komme til at ødelægge systemet. Brugeren har ikke adgang til alle systemfiler, og kan ikke slette andet end egne filer.

4.3. Proces-management

Unix-operativsystemer er multi-programming og multi-user-systemer. Det betyder at flere programmer kan køre på én computer samtidig, og dermed også at flere personer kan bruge én computer som server samtidig. For brugeren ser det ud som om programmerne kører parallelt, og at flere processer dermed har CPU'en samtidig. Dette fungerer i praksis ved, at hver proces har CPU'en i et bestemt tidsrum, hvorefter den ryger bag i en kø og næste proces hentes ind (multi-tasking).

Linux understøtter, ligesom Unix, ægte multi-tasking. Alle processer kører helt uafhængigt af hinanden, så ingen processer behøver at tage højde for at give processor-tid til andre processer. Hver proces har et hukommelses-område, som er beskyttet mod ændringer fra andre processer.

På http://www-106.ibm.com/developerworks/linux/library/l-rt7/?Open&t=grl,l=252,p=mgth findes en interessant artikel om Linux i forhold til Windows 2000 og XP med hensyn til proces og tråd-håndtering.

4.3.1. Proces-struktur

Det vigtigste og mest specielle ved et multi-tasking-system er processerne og proces-strukturen. En proces er et program, som er under udførelse.

x86-arkitekturen understøtter 4 privilegieniveauer (eng. »privilege levels«), hvor niveau 0 er det mest privilegerede, og 3 det mindst privilegerede. Et kørende program vil altid være på et af disse niveauer.

Linux bruger kun 2 niveauer, nemlig kerneniveau (eng. »kernel mode«) og brugerniveau (eng. »user mode«). Ved at skelne mellem brugerniveau og kerneniveau kan man forhindre, at brugeren har direkte adgang til de forskellige I/O-enheder.

Der er kun adgang til disse fra kerneniveau, og alle brugerprogrammer kører på brugerniveau. Brugerprogrammer har derved kun adgang til I/O-enheder igennem et på forhånd specificeret systemkald, som resulterer i et skift fra brugerniveau til kerneniveau. Derved slipper programmøren af brugerprogrammer for at bekymre sig om detaljerne omkring f.eks. I/O. Hardwarens kompleksitet skjules derved for brugeren og udvikleren.

Et brugerprogram kan således bruge operativsystemet til at få udført forskellige instruktioner, ved at lave et systemkald. Et brugerprogram, der kører under Linux, vil derfor se Linux-kernen som en udbyder af servicefunktioner.

Når en proces kører i systemtilstand, kan den være i en af følgende tilstande:

- *Running* Running illustrerer et kørende program på brugerniveau. Det meste af tiden kører et program på brugerniveau. En gang imellem er det dog nødvendigt at skifte til kerneniveau. Det kan kun ske ved et interrupt eller et systemkald.
- *Interrupt-rutine* Interrupt-rutinen bliver aktiv, når der kommer et hardware-signal, f.eks. nye inddata fra tastaturet.
- *Systemkald* Systemkald aktiveres af software interrupts. Et systemkald kan suspendere en proces, så den skal vente på en hændelse.
- *Waiting* Processen venter på en ekstern hændelse, og processen vil ikke fortsætte, før denne hændelse indtræffer.
- *Retur fra systemkald* Denne tilstand opnås efter hvert systemkald og efter nogle interrupts. Herfra kan scheduleren skifte processen til Ready og aktivere en anden proces.
- *Ready* Processen konkurrerer om at komme til processoren, som er optaget af en anden proces.

4.3.2. Opstart af Linux

Når kernen er loadet starter den én enkeltbruger-proces, nemlig /sbin/init. Som parameter får init-programmet det "run-level" som man angiver ved lilo-prompten, f.eks. "single" der normalt fortolkes

som "run-level 1".

init kigger så i `/etc/inittab`, og udfører de kommandoer der står som aktive for run-level 1, eller alle run-levels. På min Red Hat er det f.eks.

```
# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
l1:1:wait:/etc/rc.d/rc 1
```

rc.sysinit bliver altså kørt uanset run-level, mens **/etc/rc.d/rc 1** – som starter alle de kommandoer der ligger i `/etc/rc.d/rc1.d/`-kataloget – kun kører i run-level 1.

Det er formentlig i **rc.sysinit** at du finder et kald til f.eks. **sulogin**, der spørger dig efter "root"s adgangskode førend den starter en kommandofortolker.

Det er forresten ikke nogen naturlov, at kernen starter **/sbin/init** som det første program. Man kan faktisk angive hvilket program der skal startes med en lilo-parameter, f.eks. vil `init=/bin/bash` starte en root-shell helt uden om opstarts-scripts osv.

4.3.3. Procesrelationer

Proces-hierarkiet i Linux er et forælder-barn-hierarki. En ny proces kan kun skabes med systemkaldet **fork()**. Den nye proces bliver barn-proces til den proces, der udførte **fork()**.

Den nye proces bliver oprettet ved at lave en næsten identisk kopi af den proces der kaldte **fork()**. Ofte er det første, en ny proces gør, at udføre et execve-kald, som overskriver den arvede kode, stak, registre mv., så den ikke længere er en kopi af forælderprocessen, men en ny proces.

Det kan være meget ressourcekrævende at oprette en ny proces, da der kan være mange data fra forælderprocessen, som skal kopieres. Derfor bruges der i Linux "copy-on-write"-teknikken. Tanken bag denne teknik er, at et antal processer kan have adgang til den samme hukommelse, så længe der ikke er nogen af processerne, der laver ændringer i data. Således bliver de relevante pages af hukommelse ikke kopieret ved systemkaldet **fork()**, men den fælles hukommelse bliver skrivebeskyttet. Hvis en af processerne forsøger at skrive til hukommelsen, bliver processen afbrudt, og kernen laver en kopi af den relevante page. Herefter kan der tildeles en kopi til hver proces. Den store fordel ved denne metode er, at data kun bliver kopieret hvis det er nødvendigt.

4.3.4. Proces-ID

Alle processer har deres eget ID-nummer (pid) og er i en gruppe og en session. I Linux kan en proces være tilknyttet flere grupper.

Når en ny proces bliver genereret, får den et nyt pid-nummer, og **fork()** returnerer 0 til barn-processen og barn-processens pid til forælder-processen. Derved kan de to processer se, hvilken proces der er barn-proces, og hvilken der er forælder-proces.

For at kunne bestemme hvor en proces har adgang, har hver proces et bruger-ID (userid = uid) og et gruppe-ID (gid). Disse ID-numre har børne-processerne arvet efter deres forældre-processer. Når en adgangskontrol skal foretages, er det dog den effektive bruger-ID (euid) og gruppe-ID (egid) der bliver brugt. Generelt er uid = euid og gid = egid, undtagen for set-UID programmer.

I disse programmer bliver euid og egid sat til bruger-ID og gruppe-ID for ejeren af den kørbare fil. På denne måde er det muligt for superuseren at give almindelige brugere adgang til brug af systemadministrative programmer på en kontrolleret måde. Et set-UID program er et program hvor set-UID-bitten er sat. Af sikkerhedsårsager understøtter Linux kun set-UID på binære filer, ikke på shell scripts.

4.4. proc-filsystemet

Som systemadministrator (root) har man gode muligheder for at følge, hvad Linux-kernen foretager sig. Prøv at skrive **mount**. En af de linjer, som kommer frem er

```
none on /proc type proc (rw)
```

I modsætning til de almindelige filsystemer, som er monteret (f.eks. `/dev/hda1`, `/dev/hda2` osv.), så er `/proc` meget speciel. Prøv f.eks. som det første eksperiment at se følgende fil.

```
[tyge@hven ~]$ ls -l /proc/kcore
-r--------    1 root     root     134156288 jan 22 23:18 /proc/kcore
```

Betyder dette, at der ligger en ca. 128 Mb stor fil? Nej – alt hvad der ligger under `/proc` er virtuelt. Det er Linux-kernen som på en elegant måde giver adgang til en række nyttige kerne-informationer. I eksemplet er `/proc/kcore` en adgang til hele den fysiske RAM i maskinen. Lad os se mere på hvad der sker i `/proc`

```
[tyge@hven ~]$ ls -l /proc/
1/     1150/  529/ 741/  9725/ 9794/       fs/        partitions
10023/ 15656/ 539/ 765/  9726/ 9795/       ide/       pci
10030/ 15657/ 554/ 809/  9727/ 9797/       interrupts rtc
10043/ 15658/ 570/ 825/  9765/ 9798/       ioports    scsi/
1017/  17917/ 571/ 961/  9767/ 9800/       kcore      self@
10313/ 17931/ 581/ 965/  9769/ 984/        kmsg       slabinfo
10317/ 17964/ 596/ 966/  9771/ 9950/       ksyms      sound
1051/  18243/ 6/   967/  9776/ 9958/       loadavg    stat
1052/  18246/ 61/  968/  9777/ apm         locks      swaps
1053/  18250/ 698/ 969/  9780/ bus/        mdstat     sys/
1054/  18524/ 701/ 970/  9782/ cmdline     meminfo    tty/
```

```
1055/  2/     702/ 971/  9784/ cpuinfo     misc       uptime
10571/ 3/     703/ 9717/ 9786/ devices     modules    version
10723/ 307/   704/ 972/  9788/ dma         mounts
1083/  4/     717/ 9722/ 9789/ fb          mtrr
11294/ 5/     732/ 9723/ 9791/ filesystems net/
```

Som det kan ses, er der en mængde kataloger med numre. Der er et katalog for hver proces på maskinen. Lad os se på et af katalogerne mere detaljeret. Der kører netop nu en **emacs** på maskinen med pid=18243 (PID betyder proces-ID). Dette findes ved brug af `ps`. Tilsvarende kan vi se hvad kataloget i `/proc/18243` indeholder.

```
[tyge@hven ~]$ ps aux | grep emacs
pto  18243  0.1  4.7  8940 6076 pts/1   S  22:11  0:09 emacs kerne.sgml
[tyge@hven ~]$ ls /proc/18243
cmdline  cwd   environ  exe    fd  maps
mem      root  stat     statm  status
```

Det første underlige er, at filerne har filstørrelse nul. Det skal man ikke lade sig snyde af. Alt er virtuelt her – og nogle gange kan filer i `/proc` have indhold. Prøv selv at køre **more /proc/PID/cmdline**, hvor PID er en proces-ID på din maskine, f.eks. for en editor, du har startet på kommando-linjen.

Via `/proc/PID/cwd` er der direkte adgang til det katalog, som programmet blev startet op i. Filerne `/proc/PID/stat*` (tre filer) fortæller mere om f.eks. hukommelsesforbrug og andre status-parametre.

Nu har vi set på hvad der gemmes for den enkelte proces. Lad os nu se på hvad der ellers kan findes i `/proc`.

Belastning af maskinen (load) – eller rettere – hvor mange processer, der prøver at tilgå kernen i snit, kan ses direkte ud fra `/proc/loadavg`. Det er samme tal som **uptime** har til sidst – belastning det sidste minut, sidste 5 minutter, og endelig de sidste 15 minutter.

Vil du vide mere om, hvor meget hukommelse der er brugt, så kan du få dette fra kommandoen **free**, men en endnu mere detaljeret visning kan hentes fra filen `/proc/meminfo`.

```
[tyge@hven ~]$ free
             total   used  free shared buffers cached
Mem:        127760 124336  3424 135860    4684  52104
-/+ buffers/cache:  67548 60212
Swap:        68504   7120 61384
[tyge@hven ~]$ more /proc/meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  130826240 127242240  3584000 139255808  4788224 53280768
Swap: 70148096  7290880 62857216
MemTotal:    127760 kB
MemFree:       3500 kB
MemShared:   135992 kB
Buffers:       4676 kB
```

```
Cached:        52032 kB
BigTotal:          0 kB
BigFree:           0 kB
SwapTotal:     68504 kB
SwapFree:      61384 kB
```

I `/proc/net/` er der en masse nyttig information om hvordan netværket ser ud lige nu. De fleste filer er direkte læsbare for alle, mens f.eks. firewall-filerne (for kerne 2.2) `/proc/net/ip_fwchains` (med firewall-opsætningen) og `/proc/net/ip_fwnames` kun kan læses af root (sikkerhed).

4.5. Devfs

En af de nye ting i kerne 2.4.x er devfs, som er et godt alternativ til `/dev`-katalog-systemet.

4.5.1. Hvad er /dev

`/dev`'s funktion er at strukturere alle mulige (og umulige) enheder, som *filer*. Hver enkelt *fil* definerer en enkelt enhed vha. 2 pointere (også kaldet major & minor number). Major angiver enhedens type (f.eks: disk-, terminal-, seriel-enhed), minor angiver den enkelte enhed indenfor den enkelte type (f.eks: tty1, tty2,..., hda1, hda2). Disse 2 pointere benytter kernen til at "slå op" i en drivertabel, for at finde den rigtige driver til en given enhed:

```
[root@linus /root]# ls -la /dev/hda*
brw------- 1 root root 3, 0 jan 1 1970 /dev/hda
brw------- 1 root root 3, 1 jan 1 1970 /dev/hda1
.
.
[root@linus /root]# ls -la /dev/tty?
crw------- 1 root root 4, 1 maj 24 05:38 tty0
crw------- 1 root root 4, 2 maj 24 05:38 tty1
.
.
```

Hvor 1. nummer efter gruppe er Major og 2. nummer er Minor.

For at være sikker på at en given enhed vil virke, skal der altså være en enhedsfil i `/dev`. Da der er mange forskellige (typer) enheder, skal der også være mange filer. Og for at mindske administrationen, er de oprettet på forhånd. I et typisk system, som f.eks. Red Hat 7.0, er der godt 12000 filer. Og det er nok de færreste filer, det enkelte system har brug for. Det er ikke fordi de fylder noget (omkring 300 kb i alt), men de optager hver en inode (se f.eks. http://e2fsprogs.sourceforge.net/ext2intro.html). Hvis enhedsfilen for en bestemt enhed nu ikke lige er oprettet, kan man benytte programmet **/dev/MAKEDEV** som kender til (næsten) alle enheder og deres respektive major- og minor-numre.

Hvorfor denne redundans, kan man så spørge? At både `/dev`, **MAKEDEV**-programmet og kernen skal have alle disse informationer? Det har nok været den mest indlysende og overskuelige måde at designe det på i tidernes morgen. Men som antallet af enheder stiger, bliver det mere og mere uoverskueligt og tidskrævende at administrere, og det kan Devfs gøre noget ved.

4.5.2. Hvad er Devfs

Devfs er et virtuelt filsystem, som har samme funktion som `/dev`, men som ikke ligger fysisk på harddisken. Umiddelbart vil det ligne `/dev`, da man tilgår enhederfiler igennem kataloget `/dev`. Linket går bare ikke ned på disken, men direkte ind i kernen (efter samme princip som `/proc`). Dermed er man fri for administrationen og redundans, da filerne altid svarer til de enheder som kernen har fundet.

4.5.2.1. Installation af Devfs

Devfs kræver en Linux-kerne oversat med parametrene CONFIG_DEVFS_FS og CONFIG_DEVFS_MOUNT, og user-space-dæmonen **devfsd** fra Richard Goochs FTP-server (ftp://ftp.atnf.csiro.au/pub/people/rgooch/linux/daemons/devfsd/). Se i øvrigt Afsnit 4.6.

Udpak devfs-daemonen et sted og kør følgende:

```
[root@linus devfs]# make
[root@linus devfs]# make install
```

Som default vil **devfsd** ligge i `/sbin`.

For at et system kan gøre brug af devfs, er det noget af det første **init** skal starte. Det er rimeligt system-specifikt hvor opstart af devfsd skal ske, da systemfilerne (se Afsnit 1.10 er forskellige. I System V-baserede systemer er det **/etc/rc.d/rc.sysinit**, hvor det i BSD-baserede er i **/etc/rc**. Men under alle omstændigheder skal den første linje i disse filer være:

```
/sbin/devfsd /dev
```

Bemærk: Fra og med Red Hat 7.0 er følgende linje allerede indsat i **/etc/rc.d/rc.sysinit**:

```
[ -e /dev/.devfsd -a -x /sbin/devfsd ] && /sbin/devfsd /dev
```

Lidt forklaring: Hvis der ligger en fil ved navn `.devfs` i `/dev` og en kørbar fil ved navn **devfsd** i `/sbin`, så start devfs-daemonen. Så det eneste man behøver for at aktivere devfs på et Red Hat-system er at køre følgende kommando:

```
[root@linus /root]# touch /dev/.devfsd
```

Og så er det bare at reboote. Så enkelt er det.

4.5.2.2. Problemer med Devfs

Forhåbentlig skulle der ikke opstå problemer, men det kan være at der i enkelte tilfælde er device-drivere som ikke er understøttet af Devfs endnu. Her er der 2 løsninger. Den ene er at hvis det kun drejer sig om nogle få enhedsfiler, kan de oprettes under opstart vha. **mknod**-programmet. Den anden er at pakke alle de ikke-devfs-kompatible enhedsfiler ind i en tar-fil, som pakkes ud i `/dev` under opstart.

Man kan også komme ud for at programmer benytter *letforståelige* links til enhedsfiler. Som f.eks. museprogrammet **gpm**. **gpm** benytter som standard enhedsfilen `/dev/mouse`. I dette tilfælde vil det være en god idé at sørge for at **gpm** gør brug af `/dev/psaux` direkte (eller en anden "rigtig" enhedsfil for mus). Da der ikke er en opsætningsfil til **gpm** bliver man nødt til at gå direkte ind i scriptet for startup-scriptet til **gpm** (se Afsnit 1.10) og tilføje parameteren: "*`-m /dev/psaux`*". Red Hat 7.1 og senere versioner har forbedret det lidt, ved at man kan specificere en enhed i `/etc/sysconfig/gpm`. Opstartsprogrammet `/etc/rc.d/init/gpm` indeholder dog en fejl som gør at enheden altid vil være `/dev/mouse`. Ændr derfor følgende i `/etc/rc.d/init/gpm`:

```
if [ -n "$DEVICE" ]; then
        device="/dev/mouse"
fi
```

til:

```
if [ -z "$DEVICE" ]; then
        device="/dev/mouse"
fi
```

som tester om miljøvariablen "DEVICE" er tom. Hvis den er det, er der ikke angivet nogen enhed i `/etc/sysconfig/gpm`, og "DEVICE" sættes til `/dev/mouse`.

Vil du vide mere om Devfs, så kan vi anbefale Richard Gooch's Linux Devfs (Device File System) FAQ, som findes på http://www.atnf.csiro.au/~rgooch/linux/docs/devfs.html.

4.6. Omkonfigurere Linux-kernen

Kernen er det program, der styrer computeren. Den består af en række forskellige funktioner, og vi vil i dette afsnit diskutere, hvordan du kan omkonfigurere kernen, så den kommer til at passe bedre til din computer. Sagen er nemlig den, at den kerne, som følger med f.eks. Red Hat, indeholder en række funktioner, som du måske ikke har brug for. Ved at omkonfigurere kernen slipper du for at spilde en masse hukommelse (og måske tid). Du kan måske også mangle understøttelse for en hardware-del, f.eks.

USB (Universal Serial Bus). USB er for nylig kommet med i Linux-kernen, hvorfor du kan blive nødt til at lave en ny Linux-kerne.

Et par bemærkninger omkring drivere til Linux er på sin plads. Drivere kan eksistere på to måder under Linux: Som en del af kernen eller som et modul. Hvis driveren er en del af kernen, ligger den fast i hukommelsen hele tiden, men et modul indlæses først i det øjeblik, der er brug for det – ja, faktisk kan et modul også fjernes fra hukommelsen, når der ikke er brug for det længere (se Afsnit 4.7). Det er klart en fordel at benytte moduler, hvis man ikke har megen hukommelse.

4.6.1. Bliv klar til at oversætte kernen

Første trin til at lave din egen kerne er at se, om du fik installeret hele kildeteksten til Linux-kernen. Der er to muligheder; enten fra den installations-cd-rom du har, eller fra en helt ny kildetekst. Du skal bruge ca. 150 MB diskplads.

Hvis du vil installere kildeteksten til kernen, så skriv:

```
[root@linus /root]#  rpm -ivh /cdrom/RedHat/RPMS/kernel-headers-2.2.5-15.i386.rpm
[root@linus /root]#  rpm -ivh /cdrom/RedHat/RPMS/kernel-source-2.2.5-15.i386.rpm
```

Dette gør, at kildeteksten lægges ind i `/usr/src/linux`.

I øvrigt kan det tilrådes at lade `/usr/src/linux` være et symbolsk link til et versionsafhængigt underkatalog:

```
[root@linus /root]#  ls -l /usr/src/linux
lrwxrwxrwx   1 root root   11 Jan 31 12:01 linux -> linux-2.2.5
drwxr-xr-x  17 root root 1024 Jan 25 21:22 linux-2.0.36
drwxr-xr-x  15 root root 1024 Jan 31 20:45 linux-2.2.5
```

Antag, at du vil opgradere til kerne 2.3.1, som du henter hjem fra http://www.kernel.org (eller et spejl – f.eks. ftp://ftp.sunsite.dk), dvs. nu vil du installere kildeteksten selv (uden RPM). Typisk henter man kildeteksten i filformat tar-bzip, f.eks. `linux-2.4.17.tar.bz2`.

Du behøver ikke at lægge kildeteksten ind som root – faktisk bør nu nok hellere lade være, da du så ikke ødelægger noget i `/usr/src/linux`. Pak filen ud et sted som passer dig og oversæt kernen der.

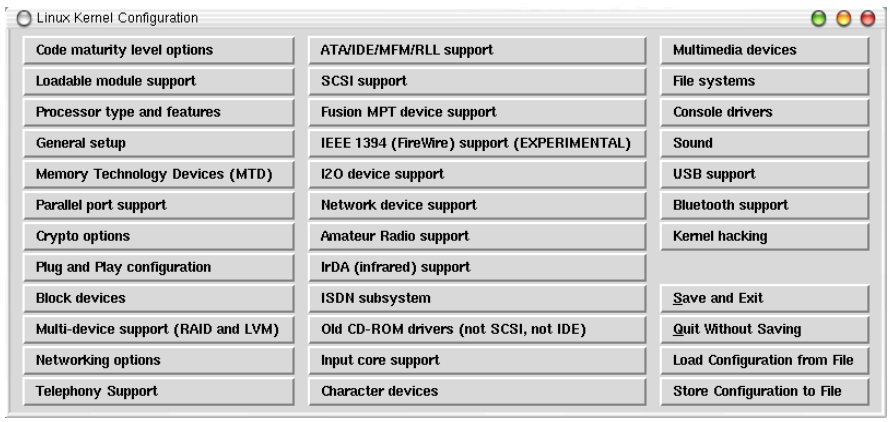
```
[tyge@hven ~]$ cd ~
[tyge@hven ~]$ tar xjvf linux-2.4.17.tar.bz2
...
```

Gå (brug **cd**) ned til `linux` og læs `README` (med **less README**). I denne fil er det forklaret, hvordan du oversætter kernen. Du bør også altid læse `linux/Documentation/Changes`, som beskriver, hvordan dit system bør være konfigureret, for at du kan oversætte kernen.

4.6.2. Oversæt Linux-kernen

For det første skal du skifte til kataloget `linux`, hvor kildeteksten til kernen er placeret. Skriv nu **`make xconfig`**. Du vil nu se et vindue som det nedenfor.

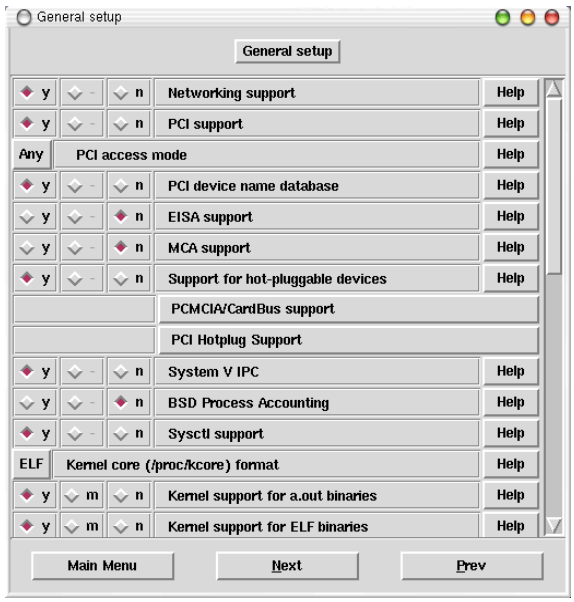
Figur 4-1. Hovedmenu for opsætning af Linux-kernen



Som du kan se, er der en række punkter, som du kan konfigurere. Du kan nu trykke på "General Setup", og du vil få et vindue som nedenfor. I langt de fleste punkter i menuen kan du vælge mellem "y", "m" og "n". Vælges "y", bliver funktionen en del af kernen, mens "n" betyder, at den ikke kommer med. Hvis du vælger "m", bliver den ikke en del af kernen, men funktionen vil eksistere som et modul. Som du også kan se, er der mulighed for at få en forklaring ved at trykke på knappen "help". Når du har konfigureret kernen, trykker du på "Save and exit".

Tip: Til bogens eksempler på www.linuxbog.dk/admin/eksempler (http://www.linuxbog.dk/admin/eksempler) kan der findes en `.config`-fil under `kerne/`. Denne fil (`linux/.config`) dannes når man kører **make xconfig** eller **make menuconfig** (tekstbaseret). Filen vil tilsvarende anvendes når man kører en af de to make-kommandoer.

Figur 4-2. "General Setup" menuen



Du er nu klar til at generere en ny kerne. Der er en række skridt, som du bare skal følge. Først gennemgås alle afhængigheder i kernen (**make dep**). Selve oversættelsen af kernen (**make bzImage**) og moduler (**make modules**) kan godt tage en rum tid afhængig af hvor meget du har taget med, og hvor hurtig din maskine er.

Bemærk at det kun er ved selve installationen af eventuelle moduler til sidst at du bliver nødt til at være root.

```
[tyge@hven ~]$ cd linux
[tyge@hven ~/linux]$ make clean
[tyge@hven ~/linux]$ make dep
[tyge@hven ~/linux]$ make bzImage
[tyge@hven ~/linux]$ make modules
[tyge@hven ~/linux]$ su root
[root@linus /home/tyge/linux]#  make modules_install
```

Bagefter kan du finde den nye Linux-kerne i `linux/arch/i386/boot/bzImage` (hvis du bruger en Intel-kompatibel maskine). Du kan teste den nye kerne ved at kopiere den til en diskette og så starte fra denne. Brug **dd if=linux/arch/i386/boot/bzImage of=/dev/fd0** for at teste på denne måde. Er du tilfreds med den nye kerne, kan du kopiere kernen til `/boot` og rette i LILO's opsætningsfil (`/etc/lilo.conf`) og køre **/sbin/lilo**, hvis du anvender LILO til at starte maskinen.

Moduler laves til kernen svarende til kernens versionsnummer under `/lib/modules`, dvs. for kerne

2.4.10 gemmes de i `/lib/modules/2.4.10`. Nu spørger du måske om, hvorfor man i det hele taget laver en kerne og så moduler. Dels kan kernen blive mindre og derfor ofte hurtigere. Men er der fejl i et modul, kan man hurtigt rette i modulets kildetekst, genoversætte kildeteksten og starte modulet på ny – uden at genstarte maskinen!

Tip: Man kan også skrive **make menuconfig** i stedet for **make xconfig**. Man får et hurtigere tekst-baseret program frem – og man har eksakt de samme muligheder for at styre opsætningen.

4.6.3. Opgradering af kernen

Hvis man har lavet en god opsætning af Linux-kernen og noget tid senere får brug for at opgradere til en ny kerne-version, så kan det være irriterende at skulle afkrydse eksakt de samme valgmuligheder igen med risiko for at noget bliver gjort forkert. En god procedure er at kopiere den gamle `.config`-fil, som ligger i dit `linux`-katalog over til et sted du vælger – og det kan være klogt at gemme den under et navn som viser hvilken dato og kerneversion, den passer med. F.eks. kan man køre **cp ~/linux/.config ~/kernecompile/kerne.config-DATO-KERNEVERSION**.

Start med at hente den nye kerne-kildetekst hjem (bemærk, at hvis du henter et patch, så passer den typisk ikke til den Linux-version du fik med din Linux-distribution). Hvis du hentede version 2.4.18, så flyt eller slet eventuelle andre kerne-kataloger.

Gå nu til `linux` og kør følgende (I dette eksempel er brugeren »root« – men det er ikke nødvendigt). Pak først den nye kerne-kildetekst ud som vist lidt tidligere (**tar xjvf KERNE.tar.bz2**).

```
[root@linus /root]# cd linux
[root@linus linux]# make mrproper
[root@linus linux]# cp /root/kerne.config-DATO-KERNEVERSION .config
```

I den sidste linje anvender vi den forrige kernes parametre som grundsten for den nye – sagen er at vi nu kører **make oldconfig**, hvor man kun skal svare på om de nye drivere skal være med i den nye kerne. Alt der er uændret spørges man ikke om.

```
[root@linus linux]# make oldconfig
[root@linus linux]# make dep
[root@linus linux]# make bzImage
[root@linus linux]# make modules
[root@linus linux]# make modules_install
[root@linus linux]# cp arch/i386/boot/bzImage /boot/vmlinuz-x.y.z
```

Endelig skal man rette `/etc/lilo.conf` til at pege på `/boot/vmlinuz-x.y.z` og køre **/sbin/lilo**.

4.6.4. Lav en redningsdiskette

Har du brug for at lave en redningsdiskette (eng. rescue disk), med Linux-kernen på, som du kan starte din Linux-maskine fra, så kan du gøre det med **mkbootdisk**. Find først ud af hvilken kerne der er på dit system. Kør f.eks. **ls /lib/modules/**. Står der 2.2.12-20, så skal du – som root - skrive

```
[root@linus /root]#  /sbin/mkbootdisk --device /dev/fd0 2.2.12-20
```

Disken har Linux-kernen, og du kan også bruge denne disk, hvis du skal flytte rundt på dine harddiske (fra primær til sekundær osv).

4.7. Moduler

Moduler (eller mere korrekt: indlæsbare kerne-moduler) er en af de mere smarte ting ved Linux. Kort fortalt er et modul en driver, som systemadministratoren (root) kan indlæse og nedlægge, mens maskinen kører, dvs. du kan indlæse en driver uden at genstarte. I dette afsnit vil vi introducere det mest basale omkring moduler.

Modul-systemet blev skrevet om til kerne-version 2.2 (som følger med Red Hat 7.0 og SuSE 6.4). Vi vil antage, at du har en version 2.2-kerne. Du kan evt. tjekke det ved at skrive **`uname -r`**. For en standard-installation af Red Hat 6.2 vil din maskine svare "2.2.14-20".

4.7.1. Hvad findes der allerede indlæst?

Inden vi går i krig med at indlæse moduler, kan det være rart at vide, hvilke der allerede er indlæst. For at finde ud af det, skal du bruge kommandoen **lsmod**, som er en forkortelse for "list modules". Kommandoen kan udføres af alle brugere, men vær opmærksom på, at kommandoen ligger i kataloget `/sbin`, og at almindelige brugere ikke har dette katalog med i deres søgesti. Derfor skal du skrive **/sbin/lsmod**, hvis du ikke er logget ind som systemadministrator (root).

```
[root@linus /root]#  lsmod
Module                  Size  Used by
ppp                    18316   0  (autoclean)
slhc                    4328   0  (autoclean) [ppp]
autofs                  9028   1  (autoclean)
nfs                    29944   1  (autoclean)
lockd                  30856   1  (autoclean) [nfs]
sunrpc                 52356   1  (autoclean) [nfs lockd]
nls_iso8859-1           2020   1  (autoclean)
nls_cp437               3548   1  (autoclean)
vfat                   11516   1  (autoclean)
fat                    25664   1  (autoclean) [vfat]
awe_wave              157804   0
sb                     33204   0
```

```
uart401                  5968   0  [sb]
sound                   57208   0  [awe_wave sb uart401]
soundlow                  300   0  [sound]
soundcore                2372   7  [sb sound]
```

Tabellen, som **lsmod** producerer, indeholder navn på modulet, hvor meget hukommelse det bruger, hvor mange programmer der bruger modulet, samt om der er andre moduler, som bruger modulet. Tag nu PPP-modulet: det fylder 18315 bytes, og ingen programmer eller andre moduler benytter det (logisk, idet forbindelsen til internet-udbyderen er lukket ned).

4.7.2. Indlæs et modul

Typisk ligger moduler i en undermappe til `/lib/modules`. På en standard-installation af Red Hat 6.2 hedder undermappen `2.2.14-5.0` efter kernen (husk svaret fra **uname**). Nede i `/lib/modules/2.2.14-5.0` finder du en række underkataloger. Alle modulerne er grupperet efter deres anvendelse, f.eks. indeholder undermappen `cdrom` drivere til cd-rom-drev (alle andre end ATAPI-drev). Til at indlæse et modul bruger du kommandoen **modprobe**. Du kan kun indlæse moduler som systemadministrator. Nedenfor viser vi, hvordan du kan indlæse en driver til HPFS (OS/2's filsystem).

```
[root@linus /root]#  modprobe hpfs
```

Det fine ved modul-systemet er, at du kun behøver at angive navnet på modulet; **modprobe** finder selv ud af, hvor det er placeret.

Det kan være, at du kommer til at angive det forkerte modul, dvs. du forsøger at indlæse et modul til et stykke hardware, som du ikke har. Nedenfor forsøger vi at indlæse driveren til et Intel EtherExpress Pro/100-kort; et kort, som *ikke* sidder i maskinen.

```
[root@linus /root]#  modprobe eepro100
/lib/modules/2.2.5-15/net/eepro100.o: init_module: Device or resource busy
```

4.7.3. Nedlæg et modul

Det kan ske, at du bliver træt af et modul eller blot i længere tid ikke har brug for det, og du derfor gerne vil nedlægge det igen. Der er to måder, du kan gøre det på. Den første kræver simpel tålmodighed: efter 60 sekunder vil ubrugte moduler automatisk blive nedlagt. Dette sørger en daemon ved navn **kerneld** for. Den anden måde er at gøre det selv. Til denne metode skal du bruge kommandoen **modprobe -r** ("remove module").

```
[root@linus /root]#  modprobe -r hpfs
```

Kigger du i mappen `/lib/modules/2.2.14-5.0`, ser du filen `modules.dep`. Filen indeholder information om, hvilke moduler et modul afhænger af. Filen er en tekstfil, så du kan bladre i den ved hjælp af **less** eller **more**. Filen skal du ikke pille ved – den bliver automatisk genereret under opstart af Linux ved at **depmod -a** køres. Næste gang din maskine starter, kan du se efter linjen "Finding module dependencies" – på det tidspunkt genereres filen. Derved sikrer man sig at der anvendes det rigtige modul, dette er specielt vigtigt hvis man har flere kerneversioner installeret.

4.7.4. Opsætning

Når vi taler om moduler, findes der en meget vigtig fil, som systemadministratoren kan rette i, nemlig `/etc/modules.conf`. Det skal lige siges, at filen bliver rettet af mange af de værktøjer, som vi omtaler i dette kapitel, f.eks. **sndconfig**, så måske kommer du aldrig selv til at rette i den. Lad os se på indholdet af en ikke helt ualmindelig `modules.conf`.

```
# /etc/modules.conf
alias parport_lowlevel parport_pc
pre-install pcmcia_core /etc/rc.d/init.d/pcmcia start
alias sound sb
pre-install sound modprobe sound dmabuf=1
options opl3 io=0x388
alias midi awe_wave
post-install awe_wave /bin/sfxload /etc/midi/GU11-ROM.SF2
options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
```

Syntaksen for filen er som følger: først et nøgleord, derefter et navn på et modul og til slut en række parametre.

Lad os begynde med det "lette" nøgleord først. Det lette nøgleord er "alias". Et alias betyder simpelthen, at når systemet spørger efter et modul, mener systemet i virkeligheden et andet. Det lyder lidt mærkeligt, men det er meget nyttigt. Et alias som **`alias eth0 eepro100`** vil betyde at når Linux forsøger at initialisere ethernettet (eth0), skal driveren `eepro100` bruges. Ofte kan Linux ikke automatisk detektere udvidelseskort i gamle maskiner, og det kan derfor være nyttigt at fortælle Linux, hvilket netkort der sidder i maskinen. Har du et gammelt kort, tilføjer du derfor en "alias"-linje som passer til dit kort. Et andet eksempel kan være hvis du skal sætte dit 3Com 90x-netkort i 10 Mbit-tilstand (og ikke 100 Mbit), så skal du indsætte følgende linjer:

```
alias eth0 3c90x
options 3c90x media_select=1
```

Et andet nøgleord er "options" (parametre). Her kan systemadministratoren angive en række parametre til en driver/modul. I det ovenstående eksempel på `modules.conf` står der f.eks. **`options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330`**. Denne linje fortæller modulet `sb` en række ting – i dette tilfælde de relevante oplysninger omkring et SoundBlaster-lydkort (f.eks. at I/O-adressen er 220, og interruptet er 5).

Nøgleordet "pre-install" tillader at udføre kommandoerne efter modulnavnet før modulet installeres. . F.eks. kald af "/etc/rc.d/init.d/pcmcia start" inden pcmcia_core kan loades. Alt hvad der kommer efter modulets navn, tolkes som en kommando med tilhørende parametre. Ligeledes findes "post-install", som angiver en kommando, der skal udføres efter modulet er indlæst.

4.8. Kerne 2.4

Den seneste generation af Linux-kernen er 2.4, som udkom den 5. januar 2001. Linus Torvalds annoncerede i vittig og underspillet tone, at nu var kerne 2.4.0 færdig:

```
In a move unanimously hailed by the trade press and industry analysts as
being a sure sign of incipient braindamage, Linus Torvalds (also known as
the "father of Linux" or, more commonly, as "mush-for-brains") decided
that enough is enough, and that things don't get better from having the
same people test it over and over again. In short, 2.4.0 is out there.

Anxiously awaited for the last too many months, 2.4.0 brings to the table
many improvements, none of which come to mind to the exhausted release
manager right now. "It's better", was the only printable quote. Pressed
for details, Linus bared his teeth and hissed at reporters, most of which
suddenly remembered that they'd rather cover "Home and Gardening" than the
IT industry anyway.

Anyway, have fun. And don't bother reporting any bugs for the next few
days. I won't care anyway.

                Linus
```

Hvad er det så kerne 2.4 har opnået i forhold til de meget anvendte kerner fra 2.2-serien? Svaret er – mange ting.

For det første er antallet af platforme, der understøttes øget. Således er der nu også understøttelse for S/390 (IBM mainframe-system), nye 64-bit MIPS-processorer samt Intels kommende Itanium. Specielt S/390 synes mange er spændende, idet det bringer Linux ind på mainframe-markedet. F.eks. har Telia i december 2000 købt en sådan maskine som erstatning for 70 Sun-computere. Desuden udnytter 2.4 MMX-instruktioner og andre udvidelser til det klassiske x86-instruktionssæt bedre end den gamle kerne.

En anden ting som er blevet klart forbedret i Linux-kernen er SMP, altså muligheden for at anvende flere processorer i samme maskine. Hvor kerne 2.2 var fin med 2 processorer og endda 4, så var ydelsen ved 8 og 16 processorer ikke imponerende. Med kerne 2.4 er ydelsen på 8 og 16 processorer forbedret markant.

Kerne 2.4 skulle umiddelbart ikke kræve mere hukommelse end kerne 2.2-serien – men den understøtter meget mere hukommelse. En af de ting, som nogle var utilfredse med i kerne 2.2-serien, var mængden af hukommelse, som Linux-kernen kunne bruge. Når man konfigurerer 2.4-kernen, er der spørgsmål om

hvor meget hukommelse man har. Tre grænser kan man vælge – under 1 Gb (off), op til 4 Gb eller 64 Gb RAM. Det er efter sigende en 5-6% mindre ydelse, hvis man vælger 64 Gb hukommelse.

En ting som store firmaer har haft problemer med, er antallet af brugere som kunne understøttes. Tidligere var grænsen på ca. 65000 brugere, mens den nu er 4,2 milliarder. Problemet var meget aktuelt for f.eks. e-post-servere, som skulle have konti for alle firmaets medarbejdere.

Med hardware kan man nu samtidig understøtte 16 Ethernet-kort, 10 IDE-controllere, og meget andet. Alt i alt har Linux taget et stort spring fremad med hensyn til high-end Linux-markedet.

Ultra-DMA 66 eller 100 er nu med i kerne 2.4. Det er noget som har været meget efterspurgt, da mange nye harddisk-controllere har været af den type.

For en Linux-laptop, så var PCMCIA med kerne 2.2 ikke ret smart. Skulle man oversætte en kerne, så skulle man have styr på sine PCMCIA-drivere ud over selve kernen. De to ting var ikke integreret, selvom de burde have været det. Med kerne 2.4 er PCMCIA en del af Linux-kernen på linje med USB.

USB-understøttelse er ny i Linux-kernen. USB-tastatur og mus skulle virke fint, men det kan godt være at din USB-skanner eller kamera ikke virker, da USB-standarden stadig er under udvikling.

Ud over PCI plug-and-play (eller er det pray :-), så er der en tilsvarende auto-detektion af enheder til ISA-bus.

En ting som netop er kommet med i kerne 2.4.0, er LVM "logical volume management", som gør det muligt at kombinere harddiske til store enheder (ligesom RAID-0), men med LVM kan man også ændre størrelsen af de enkelte partitioner løbende. Det er en klar fordel i forhold til tidligere, hvor man skulle bestemme størrelsen på partitionerne i forvejen – og ofte ud fra mere eller mindre gode estimater for hvad man skulle bruge. Læs mere om LVM på http://www.linux-mag.com/2000-11/guru_01.html.

Man kunne nemt lave en firewall med en kerne 2.2, idet kernen havde et rimelig enkelt grænsesnit til formålet, kaldet *ipchains*. Med **ipchains** kunne man sætte regler op for hvilke pakker som kunne tillades igennem firewall'en. I kerne 2.4 er hele firewall-kontrollen omskrevet til et meget stærkere koncept kaldet *netfilter*, som kan styre tilstandshåndterende firewalls. Dette er forklaret i bogen "Linux – Friheden til sikkerhed på internettet", som kan findes på www.linuxbog.dk (http://www.linuxbog.dk/).

IPv6-understøttelsen, dvs. næste generation netværksstandard, er med som eksperimentel. Man kan således ikke forvente, at dette er 100% ok.

Under `/dev` har man tidligere haft enheder, dvs. indgange til hardware, såsom harddisk `/dev/hda`. Med kerne 2.4 *kan man* anvende DevFS, som er en total restrukturering af `/dev`, så man ikke længere har ekstremt mange filer i `/dev`, men i stedet får man en hierarkisk opbygning med mange underkataloger.

Som eksempel vil man ikke længere kunne finde `/dev/hda1-/dev/hda16`, men i stedet have tilsvarende filer under `/dev/ide0/` - altså et niveau længere nede.

For dem som laver hurtige web-servere er kHTTPd interessant. Man kan accelerere web-servere kraftigt for statiske web-sider, dvs. ikke for CGI-programmer eller PHP-løsninger, ved at kernen cacher sider intelligent. Det er også markeret som eksperimentel kildetekst.

Linux avancerer i mange retninger. En ny retning er telefoni. Linux-kernen har nu fået understøttelse af telefonkort til Voice-over-IP.

For dem som arbejder med video er der godt nyt. Både IEEE-1394- (Firewire) og I2O-grænsesnit er med. Det er også ting som stadig er under udvikling. HIPPI (gigabit-netkort) er der også blevet arbejdet meget med, så der nu er fire forskellige understøttede kort.

For ADSL-brugere er kerne 2.4 (eller sene versioner af 2.3-serien) et must. Der er lavet meget inden for ATM-understøttelsen.

Og på de mere interne linjer er store dele af kernen skrevet om, så selve kildeteksten skulle være meget lettere at læse. Der er arbejdet meget med at strukturen skulle forbedres, så man lettere kunne inkorporere nye ting, som ikke tidligere var med.

Hvad der ikke kom med i kerne 2.4.0, men som er kommet med i kerne 2.4.1, er et *journaling filesystem*. Det sidste år har der været arbejdet meget hårdt på tre forskellige avancerede filsystemer, som skriver data til disk på en meget sikker måde i forhold til det gamle ext2-filsystem. I tilfælde af f.eks. strømsvigt eller lign. vil de nye filsystemer være meget mere robuste. ReiserFS vil være det første til at komme med i kernen, mens SGI's XFS-filsystem og JFS, men endnu er det uklart, hvornår de er produktionsklare. Filsystemet ext3 fra Red Hat er allerede i produktion som det primære filsystem i nyere Red Hat versioner. En glimrende oversigtsartikel, der sammenligner det gamle ext2, det nyere ext3, ReiserFS og endelige XFS kan læses på http://www.linuxgazette.com/issue68/dellomodarme.html.

En fremragende artikel (allerede en klassiker i Linux-kredse) om kerne 2.4 er Joe Pranevich: "Wonderful World of Linux 2.4", som kan findes på http://linuxtoday.com/news_story.php3?ltsn=2001-01-05-007-04-NW-LF-KN. Denne tager endnu flere aspekter op end dette afsnit har præsenteret.

Endelig kan det nævnes at en interessant kerne-patch fra http://www.tech9.net/rml/linux/ med fordel kan prøves, hvis man mener at Linux-kernen låser maskinen lidt for lang tid, når der er meget load på maskinen (bl.a. ved start af store programmer). En samling af artikler på området kan findes ud fra http://linuxtoday.com/news_story.php3?ltsn=2001-10-14-008-20-NW-KN.

Kapitel 5. Netværksprogrammer

En vigtig start-kommentar, som kunne stå mange steder i denne bog er at der med Linux-kerne 2.4 (f.eks. Red Hat 7.1 og Mandrake 8.0) er problemer med at få tilgang til alle hjemmesider – det er et problem som er forstået nu, se f.eks. http://eltoday.com/article.php3?ltsn=2001-04-17-001-14-PS. Løsningen er simpel. Tilføj følgende til f.eks. `/etc/rc.d/rc.local` (SuSE `/etc/rc.d/boot.local`):

```
echo "0" > /proc/sys/net/ipv4/tcp_ecn
```

Lad os nu se på de programmer en netværksadministrator bør kende.

5.1. Ping

ping er et meget simpelt, men utroligt nyttigt stykke værktøj, som bruges til at finde ud af, om der kan opnås forbindelse til en IP-adresse. Sagt på en anden måde er **ping** det program, du kan bruge til at finde ud af, om der er "hul igennem".

ping bør altid være det første program, du bruger til at diagnosticere et netværksproblem. Hvis det lykkes at pinge til en adresse, ved du, at den fysiske forbindelse er i orden, og de basale dele af netværket fungerer.

ping køres sådan her:

```
[tyge@hven ~]$ ping -c 4 www.linux.org
```

Ved at skrive "-c 4" beder vi **ping** om at nøjes med at sende fire pakker, ellers bliver den ved, indtil vi stopper den med Ctrl-c.

ping svarer tilbage:

```
PING www.linux.org (198.182.196.51): 56 data bytes
64 bytes from 198.182.196.51: icmp_seq=0 ttl=42 time=479.3 ms
64 bytes from 198.182.196.51: icmp_seq=1 ttl=42 time=710.0 ms
64 bytes from 198.182.196.51: icmp_seq=1 ttl=41 time=730.0 ms
64 bytes from 198.182.196.51: icmp_seq=2 ttl=42 time=600.0 ms
64 bytes from 198.182.196.51: icmp_seq=3 ttl=42 time=370.0 ms

--- www.linux.org ping statistics ---
4 packets transmitted, 4 packets received, +1 duplicates, 0% packet loss
round-trip min/avg/max = 370.0/577.8/730.0 ms
```

Vi kan først og fremmest se, at vi kan komme igennem til www.linux.org, som har adressen 198.182.196.51, og at alt, hvad vi sendte, kom tilbage igen, ja faktisk kom en af de fire pakker tilbage to

gange! Hele turen frem og tilbage tog i gennemsnit 577,8 millisekunder – altså lidt over et halvt sekund. Som Alan Cox på et tidspunkt vittigt skrev: "Afstanden mellem Bombay og New York i den nye tidsalder er kun 250 millisekunder".

Hvis du ikke skulle få nogen pakker tilbage, er dette dog ikke ensbetydende med, at der er noget galt med forbindelsen til den anden maskine, da den kan være sat op til IKKE at svare på **ping**.

5.2. telnet

Med **telnet** er det muligt at logge ind på andre computere på netværket og arbejde tekstbaseret, som om du sad ved computeren. Dette kan også foregå på tværs af platforme, f.eks. kan du køre **telnet** på en Windows 95-maskine og derfra logge ind på en Linux-maskine.

Ved normal brug startes **telnet** simpelthen med navnet på den computer, du ønsker en forbindelse til – i nedenstående eksempel `gonzo`. Når forbindelsen er oprettet, vil du blive præsenteret for en `login:`-prompt, og du kan nu arbejde videre, som om du sad ved den anden maskine.

```
[tyge@hven ~]$  telnet gonzo
Trying 192.168.100.5...
Connected to gonzo.codehell.lokal.
Escape character is '^]'.

Red Hat Linux release 6.0 (Hedwig)
Kernel 2.2.5 on an i586
login:
```

Når du ikke længere har brug for telnet-forbindelsen, afbryder du den med **exit**, og du er så tilbage på din egen maskine.

Hvis du angiver en port, som er bundet til en anden protokol, kan du også have glæde af **telnet** til fejlfinding og testbrug. I en snæver vending kan du bruge **telnet** til at hente post hos din internetudbyder!

For at beskytte din installation er standardinstallationen lavet, så "root" ikke kan logge ind fra en anden maskine med **telnet** og **ftp**. Det er gjort for, at man ikke helt så nemt kan få adgang til "root"s adgangskode. Hvis du har maskinen på et usikkert net – f.eks. internettet – bør du ikke ændre på dette. I stedet kan du bruge krypteret netkommunikation – f.eks. med "Secure Shell" eller den frie udgave "OpenSSH" (sikker shell). Har du et lukket net og mener, at du ikke gider at gå ind som almindelig bruger og derefter lave **su - root**, kan du godt få lov til at bruge **telnet** og **ftp** som root udefra. Filerne `/etc/securetty` og `/etc/ftpusers` kan rettes eller direkte slettes, så kan root logge ind udefra.

Som telnet-klient på en Microsoft Windows-maskine kan benyttes PuTTY, som også understøtter SSH. Med PuTTY ligner det langt mere det, at sidde direkte på en Linux-konsol i forhold til den telnet-klient,

som følger med Microsoft Windows. Du kan se mere om PuTTY på
http://www.chiark.greenend.org.uk/~sgtatham/putty/

5.3. ftp

Til at hente og sende filer over TCP/IP bruges programmet **ftp**.

Hvis du skal bruge en fil fra `sunsite.dk`, skriver du sådan her:

```
[tyge@hven ~]$  ftp sunsite.dk
```

Når forbindelsen er klar, spørger computeren om brugernavn og adgangskode. Du kan logge på som 'anonymous' (anonym) og give din e-post-adresse som adgangskode. Bemærk, at e-postadressen ikke er synlig, når den tastes ind.

```
Connected to sunsite.dk.
220 sunsite.dk FTP server (NcFTPd 2.2.2) ready.
Name (sunsite.dk:tyge): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-You are user #57 of 300 simultaneous users allowed.
230-
230-                Welcome to SunSITE Denmark
230-                ==========================
230-
230-SunSITE Denmark is located at Aalborg University, Institute of
230-Electronic Systems, Denmark, and is running on a SPARCserver 1000 with
230-4 CPUs, 416 MB Memory and approximately 110 GB storage.
230-

[lang velkomsthilsen klippet ud]

230-
230 Logged in anonymously.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Så står du ved en `ftp>`-prompt, hvor du kan bevæge dig rundt i katalogerne med **cd** og se indholdet med **ls** - ganske som med en almindelig kommandolinje.

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
drwxrwxr-x  60 ftpuser  ftpusers      8192 Oct 10 16:23 disk1
drwxrwxr-x   9 ftpuser  ftpusers      2048 Oct 10 15:45 disk2
drwxr-xr-x   3 ftpuser  ftpusers        96 Jun  2 17:02 disk3
drwxr-xr-x   4 ftpuser  ftpusers        96 Sep  1 09:15 disk4
... [mange kataloger klippet væk]
```

```
226 Listing completed.
```

Hvis du nu ved hjælp af **cd** navigerer ned i `/pub/os/linux/Red Hat` og kører **ls**, vil du se, at der ligger en fil ved navn `README`, som du gerne vil hente hjem på din egen maskine med kommandoen **get README**.

Nu er filen ikke særlig stor, så den er hentet ned på et øjeblik. Hvis du henter større ting, er det som regel rart at kunne se, om der er "flow" i overførslen. Med kommandoen **hash** kan du få skrevet et #-tegn ("hash mark"), hver gang en blok er blevet overført.

```
ftp> hash
Hash mark printing on (1024 bytes/hash mark).
```

Så henter du filen:

```
ftp> get README
local: README remote: README
200 PORT command successful.
150 Opening BINARY mode data connection for README (1155 bytes).
#
226 Transfer completed.
1155 bytes received in 0.227 secs (5 Kbytes/sec)
```

`README` er en ganske lille fil, så det vil kun blive til et enkelt #-tegn. Læg i øvrigt mærke til, at den er overført som binær fil (BINARY) i modsætning til ren tekst (ASCII). Hvis du bruger ASCII til overførslen, vil **ftp** konvertere linjeskift i tekstfiler til det korrekte format for den maskine, der modtager filen. Det er udmærket til tekstfiler, men kan smadre f.eks. grafik og komprimerede filer fuldstændigt. Til at skifte mellem binær og ASCII bruges ftp-kommandoerne **bin** og **ascii**.

Nu er du færdig med at hente, hvad du skal bruge, og du logger ud igen med kommandoen **quit**. Derefter er du tilbage på kommandolinjen på din egen maskine.

Hvis du har brug for at tilbyde anonym ftp på din Linux-maskine, kan du i Red Hat Linux og nogle andre distributioner installere pakken "anonftp". Pakken foretager hele opsætningen for dig. Filer, der skal kunne hentes via anonym ftp, lægges så blot i `/var/ftp/pub`.

Alt det ovenstående handler om anonymt ftp. Hvis du er oprettet som bruger på den maskine, du skal i kontakt med, kan du logge på med dit rigtige brugernavn og adgangskode. På den måde får du så rettigheder til at hente og sende filer i forhold til, hvad du er blevet tildelt af administratoren.

Man kan også bruge Midnight Commander til at hente filer fra en FTP-server. Du åbner Midnight Commander med kommandoen **mc**. Med funktionstasten F9 får du fat i menuen og trækker ned med pil-ned. Vælg så "FTP link...".

Indtast evt. brugernavn og adgangskode adskilt med et kolon (:), så et snabel-a (@), hvorefter navnet på selve FTP-serveren angives. Hvis det drejer sig om en anonym FTP-server kan man udelade alt andet end serverens navn. Det kan se sådan ud:

```
brugernavn:adgangskode@ftp.server.net
```

En hurtig og grafisk metode til at overføre filer via FTP.

Opsætning af ftp-tjeneste se afsnit Afsnit 7.2.

5.4. wget

Fra tid til anden har du brug for at kunne hente en stribe hjemmesider, som du sidenhen vil kunne læse - for eksempel programvejledninger. Med Netscape er det besværligt, idet du skal trykke "gem" for hver enkelt side. Meget smartere er programmet **wget**, der kan hente rekursivt fra en http- eller ftp-side, såsom SSLUG's adresse.

```
[tyge@hven ~]$  wget -r http://www.sslug.dk/
```

Ovenstående kommando henter hele SSLUG's websted. Er det kun de underliggende kataloger der skal hentes, tilføjes **-np** (no parents). Dette kan bruges til at hente denne bog fra www.sslug.dk. I nogle HTML-sider er der angivet direkte link. Dette kan fjernes med **-k**.

```
[tyge@hven ~]$  wget -r -k -np www.linuxbog.dk/friheden/bog/
```

Et lille trick er at bruge **wget -c ftp://ftpsted/storfil.tgz**, hvor `-c` står for continue og hvis man husker den hver gang man henter store filer ned, så kan programmet finde ud af at fortsætte afbrudte nedhentninger. Fed funktionalitet hvis man vil hente noget stort over en ustabil forbindelse.

Se også Afsnit 5.5 om **ncftp** som kan anvendes til at hente filer fra nettet.

5.5. ncftp

Programmet **ncftp** er en ftp-lignende klient, som er nem at bruge til anonym FTP. Man skal ikke angive brugernavn og adgangskode. Prøv f.eks.:

```
[tyge@hven ~]$ ncftp sunsite.dk
```

ncftp kan en masse ting der gør det væsentligt nemmere at bruge end det basale ftp-program.

5.5.1. ncftpget

ncftpget er en videreudvikling af **ncftp** som er specielt velegnet til at hente mange filer fra en anden server, ved blot at angive hvad man vil hente på kommandolinjen. Skal du hente filer i `/home/mig/public_html` fra maskinen ftp.serv.dk, hvor du har en brugerkonto *mig*, kan det gøres med følgende kommando:

```
[tyge@hven ~]$ cd temp
[tyge@hven temp]$ ncftpget -R -u mig ftp.serv.dk . public_html
```

`-R` gør at alle filer i underkataloger også bliver hentet. `-u mig` angiver brugernavn og da man ikke bør angive adgangskode på kommandolinjen, bliver man spurgt om adgangskode. Punktum angiver det katalog du står i, hvilket giver et godt overblik over hvor filerne havner. **ncftpget** har mange flere muligheder der er velegnede til automatiserede formål.

5.5.2. ncftpput

ncftpput er beregnet til at lægge mange filer op på en server. Skal du lægge filer op fra */home/linus/public_html* til *ftp.serv.dk* i kataloget */home/mig/html* hvor du har en brugerkonto *mig*, kan det gøres med følgende kommando:

```
[tyge@hven ~]$ ncftpput -R -u mig ftp.serv.dk html public_html
```

Se også Afsnit 5.5.1.

5.6. Linux som newsserver

Et naturligt spørgsmål ville være: "Hvorfor skulle man overhovedet køre en lokal newsserver?". Det er der faktisk flere gode grunde til:

- Hvis der er flere på lokalnettet, som læser de samme nyhedsgrupper, så skal de kun hentes én gang.
- Hvis man ikke lige kan overskue hvilken newsreader man ønsker at bruge, så står man med Leafnode frit for at teste 1.000 newsreadere og skal stadig kun hente serverlister én gang.
- Hvis man læser news fra mere end én newsserver f.eks. `news.sslug.dk` og `news.get2net.dk`, så kan man "slå" disse serveres nyhedsgrupper sammen til en lokal server.
- Man kan læse news Offline som hvis man var Online.

Et valg af newsserver er Leafnode.

5.6.1. Leafnode, en NNTP-server til hobby-brug

Ud over de generelle argumenter for at køre nyhedsserver selv, kan vi tilføje følgende som angår newsserveren Leafnode:

- Rigtig nem opsætning
- Ingen vedligeholdelse.
- Leafnode er en dynamisk newsserver. Den henter kun de grupper der faktisk bliver læst.

Så der er faktisk mange gode grunde til at bruge Leafnode, hvis man kun har brug for en simpel NNTP-server og ikke har brug for at "feede" til andre NNTP-servere.

5.6.1.1. Installation af Leafnode

Installationen af Leafnode er nem. På et Debian-system klares det med **apt-get install leafnode**. Hvis din distribution ikke kommer med Leafnode må du hente kildeteksten hentes fra http://www.leafnode.org/. Når man så har hentet tar-pakken, pakker man den ud:

```
[root@linus /root]# tar xzvf leafnode-*.tar.gz
...
[root@linus /root]# cd leafnode-*; less INSTALL
```

I denne fil er installationen beskrevet trin for trin, men vi løber den lige igennem alligevel. Dette er beskrevet fra en Red Hat. Det kan være at opsætningsfilerne ligger andre steder på andre distributioner.

```
[root@linus leafnode]# ./configure && make && echo O.k.
...
O.k.
```

Herefter skal der oprettes en bruger *news*:

```
[root@linus leafnode]# adduser news
```

Så må man køre:

```
[root@linus leafnode]# make install
```

Så skal der rettes i opsætningsfilen til Leafnode, som er placeret i `/etc/leafnode/config`. Det interessante er:

```
## Dette er en del af Leafnode opsætningsfilen /etc/leafnode/config
## This is the NNTP server Leafnode fetches its news from.
## You need read and post access to it. Mandatory.
server = news.sslug.dk
```

Her kan man så bare tilføje flere servere hvis man ønsker at læse fra flere. Skal der bruges brugernavn/adgangskode på en af dem, tilføjes det bare nedenunder:

```
# username = gulbrandsen
# password = secret
```

Og dette er faktisk alt der skal gøres ved opsætningsfilen. De andre parametre er rigtig godt beskrevet og omhandler ting som:

- I hvor lang tid indlæg skal gemmes.
- I hvor lang tid der skal hentes nye indlæg, hvis gruppen ikke bliver læst.

Disse parametre kan man altid stille på hvis man synes den ikke fungerer tilfredsstillende, men default er faktisk rimelig gode indstillinger.

Så skal crontab redigeres for brugeren news. Som standard anvender crontab editoren vi. Du kan evt. starte med at ændre dette forhold med kommandoen: **editor=emacs; export EDITOR** eller **editor=mcedit; export EDITOR**. Samme kommando kan tilføjes til din `~/.profile`.

```
[root@linus leafnode]# crontab -u news -e
```

Heri tilføjes linjen:

```
0 0 * * * /usr/local/sbin/texpire
```

Dette får den til at slette gamle indlæg hver midnat. Det er smart hvis man sørger for at det er et tidspunkt hvor computeren er tændt. Nå, ja, hvis den er tændt, og man sover i samme rum, så lad være med at sætte den til at køre mens du sover. Der er rimeligt meget tryk på disken, når denne køres, da der er mange filer der skal tjekes igennem, så det kan man godt vågne af :-).

Så skal inet-dæmonen sættes op til at starte Leafnode. Først kontrolleres at inet er startet (Der skal være en stjerne foran i ntsysv). Så redigeres filen `/etc/inetd.conf`. Det er følgende linje der skal ses på:

```
nntp stream tcp nowait root /usr/sbin/tcpd /usr/local/sbin/leafnode
```

Her er Leafnode sat til at gå igennem tcpwrapperen så der skal også lige åbnes for denne. I `/etc/hosts.deny` skal der helst (af sikkerhedsmæssige årsager) stå:

```
ALL: ALL
```

hvorefter man redigerer `/etc/hosts.allow` og indsætter linjen:

```
leafnode: localhost
```

Her kan så tilføjes andre maskiner som skal have lov til at læse (se **man hosts.allow**). Og så skal netværks-funktionerne genstartes:

```
[root@linus leafnode]# /etc/rc.d/init.d/inet restart
```

Eller hvis du bruger Red Hat 7.0

```
[root@linus leafnode]# /etc/rc.d/init.d/xinetd restart
```

Eller hvis du anvender SuSE 6.4

```
[root@linus leafnode]# /etc/rc.d/init.d/inetd restart
```

Så er det hele ved at være sat op. Nu køres **/usr/local/sbin/fetchnews** for første gang. Første gang køres den med `-f` som parameter. Dette får Leafnode til at læse serverlisterne på de angivne servere. Denne skal køres som "root" eller "news". Man kan igen putte den i "news"s crontab hvis ens forbindelse er oppe hele tiden, ellers kan den med fordel puttes i ens opkaldsscripts eller man kan køre den manuelt når man ønsker at poste/hente news. Det sidste har jeg valgt, og derfor har jeg redigeret `/root/.bashrc` og indsat denne linje:

```
alias news='/usr/local/sbin/fetchnews -vvv'
```

De 3 v'er giver bare meget output fra **fetchnews**, og nu kan jeg skrive news som root og køre **fetchnews**. Når man poster gennem Leafnode, bliver postningerne lagt i kø indtil fetchnews bliver kørt igen, og indlægget bliver postet mens nye indlæg fra de grupper, der abonneres på, bliver hentet. Nu mangler vi kun at sætte newsreaderen op til at læse fra localhost i stedet for den gamle newsserver.

Ønsker du automatisk hentning af nyheder hvert kvarter, kan du yderligere tilføje denne linje i `/etc/crontab` (kun interessant hvis du har en forbindelse med fast pris):

```
*/15 * * * * /usr/sbin/fetchnews
```

5.7. NTP - tidssynkronisering

(x)ntp er et værktøj til synkronisering af tiden på computere. Dette program sørger for hele tiden at holde systemuret synkroniseret med en anden ntpserver et sted på internettet, derfor er det mest oplagt at benytte det på maskiner der konstant er på internettet.

På installations-CD'erne til de fleste distributioner ligger der (x)ntp-pakker lige til at installere ellers kan programmet hentes på http://www.ntp.org.

ntp-pakken består af 2 værktøjer **ntpd** (ntp-dæmonen) og **ntpdate**, den sidste køres manuelt før man starter **ntpd** for at få tiden synkroniseret fra start, da ntpd nægter at starte hvis tidsforskellen mellem de 2 maskiner er for stor. Når ntpdate kaldes under boot, kan man med fordel bruge option "-b", der tvinger ntpdate til at sætte tiden, selv om tidsforskellen er for stor.

Denne kan også kaldes fra ens internet-opkaldsscripts, hvis man har en opkaldsforbindelse og syntaksen er bare: **ntpdate -s ntp-server-navn**. NTP-servere er der en hel del af rundt omkring på nettet og din internetudbyder har højst sandsynligt en:

```
Blandet liste: europe.pool.ntp.org
Worldonline/Tiscali: ntp.worldonline.dk
TeleDK/TDC: ntp.inet.tele.dk
```

Har du fast internetforbindelse på din server så kan det være fordelagtigt at sætte dæmonen op så uret hele tiden holdes synkroniseret. Denne dæmon kompenserer for hardware-urets drift der er afhængigt af varme og andre ting. Det gøres meget nemt ved blot at tilføje til filen `/etc/ntp.conf`.

```
server NTP-SERVER-NAVN
driftfile /etc/ntp/drift
```

og ntpserverens navn som eneste linje i filen `/etc/ntp/step-tickers`. Så kører ntpdæmonen en **ntpdate** mod serveren inden den starter synkroniseringen.

Og så skal dæmonen blot startes med: **/etc/rc.d/init.d/ntpd start** og hakkes af så den starter automatisk i "ntsysv" så den starter i det runlevel du normalt starter i.

Dette er beskrevet ud fra en Mandrake 8.0 og filernes placering kan afhænge af distribution og installationsmetode.

5.8. Opsætning af netkortparametre

Et problem som flere Linux-brugere har angår netkort, hvor man måske har lyst til at styre mellem 10 MBit og 100 MBit, samt og der anvendes fuld duplex eller ej. Der er ofte en DOS/Windows driver med til ens netkort, som kan dette - og under Linux skal man bl.a. se efter **mii-tool** (nyeste version hedder **mii-diag**. Desværre er det vist ikke alle netkort som programmet virker sammen med.

Findes **mii-diag** ikke på din Linux-maskine, så er det på adressen http://www.scyld.com/network.html muligt at hente dette sammen med andre Linux-programmer for opsætning af de mest gængse netkort. Programmerne hentes som C-kildetekst og skal derfor oversættes. Nederst i programmerne står der hvordan man gør. Programmerne giver normalt hjælp, hvis de startes uden parametre, eller man kan finde en tekst beskrivelse i sourcekoden.

5.9. Synkronisering af data mellem to steder med rsync

At synkronisere data mellem to kataloger eller to maskiner er meget nyttigt. Tanken er at man har ca. samme data to steder - det kan enten være to kataloger på samme maskine eller to forskellige maskiner (f.eks. en laptop og en stationær maskine). Man ønsker så elegant at kunne overføre *de ændrede filer* fra den ene maskine til den anden - i forhold til sidste gang de to maskiner blev synkroniseret mod hinanden.

Det første man skal gøre er at installere `rsync`-pakken på ens maskine (tjek dette med **rpm -q rsync**) samt rsync på den anden maskine sammen med OpenSSH (Secure Shell) hvis man vil synkronisere mellem to maskiner. Opsætning og installation af OpenSSH er forklaret i "Linux - Friheden til sikkerhed på internettet", der kan findes på www.linuxbog.dk (http://www.linuxbog.dk/). Man skal også sætte miljø-variablen `RSYNC_rsh=ssh` i brugerens opstartsfil (`~/.bashrc` "export RSYNC_rsh=ssh" eller `~/.tcshrc` "setenv RSYNC_RSH ssh").

Programmet **rsync** er genialt - det sammenligner filer i de to ender via tjeksummer og opdaterer filen i den ene ende kun med ændringerne mellem de to filer. Ofte har man få ændringer - og dermed kan **rsync** spare rigtig meget tid og båndbredde.

Antag at brugerens email-katalog `~/mail` evt. med en række under-kataloger findes på egen maskine (den vi kører kommandoer fra) samt maskinen `asterix`.

Antager vi først at `asterix` har de nyeste filer som skal synkroniseres over til vores egen maskine, da skriver vi:

```
[tyge@hven ~]$ rsync -azv asterix:mail ~
```

Bemærk at hvis der også er ændrede filer på egen maskine siden sidste synkronisering så overskrives disse.

Antager vi modsat at vores egen maskine har de nyeste filer som skal synkroniseres over til `asterix`, og vi vil slette eventuelle andre filer på asterix, da skriver vi

```
[tyge@hven ~]$ rsync -azv --delete ~ asterix:mail
```

Derefter vil asterix have eksakt de samme filer som vores maskine i `~/mail`. Med parameteren `--delete` sørger vi også for at slette eventuelle filer på `asterix` under `~/mail` som ikke findes på vores egen maskine.

Med de to ovennævnte eksempler vil filer blive overskrevet fra den ene maskine til den anden uanset hvilke filer der er de nyeste. Lad os nu antage at vi gerne vil synkronisere mail-katalogerne så det altid er de nyeste filer som findes *begge* steder.

```
Egen maskine:
mail/fil1   April 2
```

```
mail/fil2   April 3
mail/fil3   April 1

asterix:
mail/fil1   April 5
mail/fil2   April 1


Bagefter har begge maskiner:
mail/fil1   April 5
mail/fil2   April 3
mail/fil3   April 1
```

Dette gøres ved

```
[tyge@hven ~]$ rsync -auvz mail/ asterix:mail/
[tyge@hven ~]$ rsync -auvz asterix:mail/ mail/
```

Bemærk: At **rsync** fungerer forskeligt alt efter om der kommer `/` efter katalog-navnet. Pas på med dette.

Kapitel 6. TCP/IP og DNS

Det er i dag blevet meget almindeligt, også for privatbrugere, at koble flere computere sammen i et netværk. For at computerne kan "tale" sammen kræves en standard for hvorledes kommunikationen skal foregå. En sådan standard kaldes en protokol.

En protokol består basalt set af nogle regler som blandt andet forhindrer computerne i at "snakke i munden på hinanden". Samtidig er det defineret hvor store portioner data som skal sendes ad gangen, hvor lang tid computeren skal vente på at modparten svarer og lignende.

Gennem tiden er forskellige standarder inden for datakommunikation blevet udviklet. En dominerende teknologi, kompatibel med flertallet af computersystemer, har ikke eksisteret før internettet for alvor brød igennem i 1990'erne.

For at muliggøre kommunikation mellem vidt forskellige computersystemer er talrige standarder blevet vedtaget – og bliver løbende vedtaget. Kommunikation over internettet er bygget op om protokollen "TCP/IP", Transmission Control Protocol/Internet Protocol. Linux har i modsætning til andre styresystemer altid indeholdt denne protokol, hvilket betyder at Linux er meget overlegen når det gælder internet-kommunikation.

TCP/IP har rødder helt tilbage til slutningen af 1960'erne, hvor den amerikanske militær-organisation DARPA (Defense Advanced Research Projects Agency) startede et forskningsprojekt. Målet var at udvikle en teknologi til sikker datakommunikation, som ikke var afhængig af en central enhed, og kunne fungere selvom en stor del af netværket var destrueret af fjender. Samtidig skulle teknologien kunne fungere på vidt forskellige computersystemer og styresystemer. Teknologien kom til at hedde ARPANET.

ARPANET ekspanderede kraftigt, og delte sig i flere uafhængige men forbundne netværk. Inden længe havde ARPANET bredt sig til mange amerikanske universiteter og senere ud over USA's grænser, det blev herefter til det som vi i dag kalder "internettet".

For at få det hele til at snakke sammen, eksisterer forskellige organisationer, som hver har deres ansvarsområde. Disse består af frivillige teknikere/ingeniører, som tager stilling til forskellige forslag og diskuterer fremtiden for internettet. Debatten er meget åben; før en standard bliver vedtaget, kommer den ud til "internet-befolkningen" hvor enhver har mulighed for at kommentere. Herefter vil forslaget blive vedtaget og gjort til en endelig standard som de individuelle firmaer kan implementere i deres systemer.

Internettets historie er der skrevet mange bøger om, stort set hver bog har sin egen version af historien. Der er ikke altid enighed om hvornår hvilken begivenhed indtraf, eller hvem der stod bag. Men rent faktisk er det kun filosofien, som datidens "internet" har til fælles med nutidens internet.

Fordi du nu bruger Linux, kommer du til at have fingrene lidt længere nede i netværksmaskineriet end med andre styresystemer. Derfor er du nødt til at lære lidt mere om, hvordan det hele er skruet sammen, i

stedet for bare at skrive en stribe magiske tal i nogle dialogbokse.

Du kan i mange tilfælde teste tingene på din egen maskine uden at være forbundet til et eksternt netværk, og dermed undgår du at genere andre. Vær frem for alt ikke bange for at lege med tingene – du kan kun blive klogere af det!

6.1. IP-adresser

Kommunikation på internettet foregår ved at hver enkelt computer tilkoblet internettet, har et unikt nummer, kaldet en IP-adresse. IP-adresser ses i det daglige som decimaltal, eksempelvis: 195.249.116.158.

Men rent faktisk behandler computere dem ikke i 10-talssystemet, men derimod i det binære talsystem. Det binære talsystem består af bit. Dem skal der bruges 32 af for at have en IP-adresse. Hver bit kan enten have værdien 1 eller 0, på hardwareniveau behandles 1 som forbindelse mellem 2 kredsløb og 0 som ikke forbindelse.

I decimaltal er en IP-adresse delt ind i 4 felter adskilt af punktum. For at sammenligne binær og decimal skal den binære version derfor også opdeles i 4 felter; vi deler derfor de 32 bit med de 4 felter, og kan derved udlede at hvert felt skal bestå af 8 bit.

I binært talformat vil det første felt i ovenstående IP-adresse (195) hedde: 11000011. Følgende metode benyttes for at omregne feltet til decimaltal:

Tabel 6-1. Omregn binær til decimal

Bit-nr: (talt bagfra)	Repræsenteret værdi:	Bit-værdi:
1	1	1
2	2	1
3	4	0
4	8	0
5	16	0
6	32	0
7	64	1
8	128	1

Hvis bitværdien er 1 betyder det blot at den repræsenterede værdi skal medregnes, hvis den er 0, skal den springes over, det giver følgende regnestykke: 1+2+64+128 = 195.

Dette regnestykke bruges for hvert af de 4 felter, og man kan på den måde omregne en IP-adresse fra binær til decimal. Hvert felt kan således antage en værdi mellem 0 og 255 (begge inklusive).

Har du brug for at regne om mellem binært, decimalt og hexadecimalt, så har KDE er god lommeregner, **kcalc**, som gør dette nemt.

6.2. Netklasser

For at kunne opdele IP-adresserne i portioner som kan delegeres til firmaer/institutioner, er et system defineret til at inddele IP-adresserne i net, også kaldet netklasser. Der eksisterer 3 typer af netklasser: A, B og C. Hver klasse har et specifikt adresseområde. Derudover afgør netklassen, hvor mange af felterne som er variable; følgende skema giver en oversigt over de omtalte netklasser:

Tabel 6-2. Netklasser

Netklasse:	**A**	**B**	**C**
Adresseområde:	0.*.*.* - 126.*.*.*	128.*.*.* - 191.*.*.*	192.*.*.* - 223.*.*.*
Fast/Variabel:	Fast.*.*.*	Fast.fast.*.*	Fast.fast.fast.*
Mulige klasser:	127	16.383	2.097.151
Mulige IP-adresser:	16.777.214	65.534	254

* (stjerne) angiver at værdien kan være mellem 0 og 255 (begge inklusiv).

Som det kan ses i "adresseområde"-feltet, er det udelukkende det første felt i IP-adressen som afgør hvilken netklasse en IP-adresse tilhører. Feltet "Fast/Variabel" viser hvilken del af IP-adressen man selv kan bestemme når man får delegeret et net af den pågældende klasse. Den faste del kaldes også for netværksnummeret mens den variable del kaldes værtsnummer eller host-id. "Mulige klasser" fortæller hvor mange klasser af de forskellige typer som kan eksistere med de gældende standarder. Feltet "Mulige IP-adresser" illustrerer hvor mange IP-adresser hver klasse kan indeholde.

Hvis man sammenligner vores test-IP-adresse: 195.249.116.158 med ovenstående skema, fremgår følgende: Det første felt i IP-adressen, "195" fortæller at IP-adressen tilhører et C-klasse-net, da tallet "195" ligger i området 192-223. For C-klasse-net gælder det at de første 3 felter er faste. Det er således kun det sidste felt man selv kan råde over. Samtidig kan det ses at der i alt kan eksistere 2.097.151 C-klasse-net. Hvert C-klasse-net kan indeholde 254 IP-adresser.

6.2.1. Subnet-maske

Når et IP-net bliver delegeret til et firma/en institution, får firmaet tildelt et netværksnummer og skal derfra selv tildele værtsnumre til de enkelte computere. Eftersom computerne sidder i et lokalnetværk og derfor er fysisk koblet sammen, skal de formentlig primært udveksle data indbyrdes. For at muliggøre dette kræves det at en specifik subnet-maske sættes samtidig med den individuelle IP-adresse.

Subnet-masker består ligesom IP-adresser af 32 bit, til et C-klasse-net hedder subnet-masken i decimal

255.255.255.0. De første 3 felter af en C-klasse IP-adresse er som bekendt faste. I subnet-masken vises det ved at feltet har den maksimale værdi 255. Mens det sidste felt, som er variabelt, har værdien 0, hvilket betyder at dette felt i IP-adressen kan antage en hvilken som helst værdi mellem 0 og 255 og stadig være inden for det givne IP-net. Med en C-klasse-net subnet-maske skal de 3 første felter i en IP-adresse altså være ens før 2 IP-adresser stammer fra samme IP-net.

Tabel 6-3. Test af IP-net

IP-adresse:	Samme IP-net som 195.249.116.158?
212.112.128.10	Nej
195.125.13.2	Nej
195.249.116.15	Ja!

Ovenstående skema viser med nogle eksempler hvordan man ser om C-klasse IP-adressen 195.249.116.158 er i samme IP-net som en anden adresse. Kriteriet er at de 3 første felter skal være ens, da vi har at gøre med en C-klasse subnet-maske (255.255.255.0).

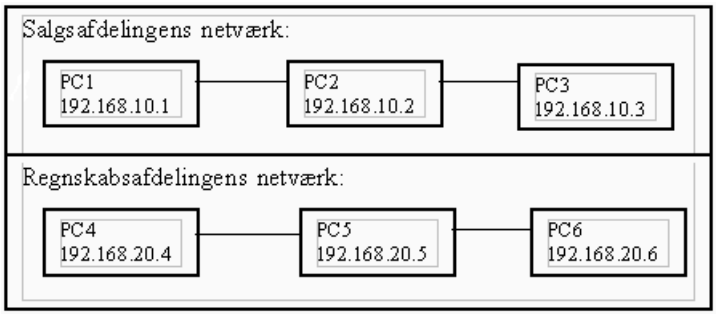
Det første felt i den første adresse (212) er ikke lig 195, og derfor er de 2 adresser ikke i samme net. Ved den næste adresse matcher det første felt, men det andet felt (125), er ikke lig det andet felt i test-adressen (249). Den sidste adresse er på samme IP-net fordi de 3 første felter matcher.

6.3. Sammenkobling af flere netværk

En computer i et TCP/IP-netværk benytter altså subnet-masken til at verificere om en anden computer som den skal kommunikere med, er på det samme IP-net. Hvis det ikke skulle være tilfældet, skal computeren bruge en "gateway". En gateway er et slags mellemled som fysisk er tilkoblet flere netværk og samtidig har en IP-adresse for hvert netværk. På denne måde kan data komme fra et netværk, passere igennem gateway'en, og ud på et andet netværk til sin destination. Gateway'en er nærmere en funktion end en enhed. Denne "funktion" kaldes routning og kan udføres af en computer, en router eller en lignende hardware-enhed.

Et elementært eksempel kunne være en virksomhed bestående af en salgsafdeling og en regnskabsafdeling, med hver deres lokalnetværk:

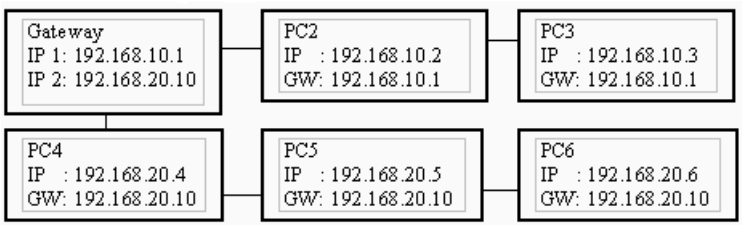
Figur 6-1. Netværks-eksempel



Disse pc'er har hidtil kørt i 2 separate netværk, men skal nu forenes således at pc'erne kan kommunikere på tværs af hinandens netværk.

Vi starter med at udnævne salgsafdelingens PC1 til at være gateway. Denne skal have et ekstra netkort installeret som fysisk skal tilkobles regnskabsafdelingens netværk. Dette ekstra netkort skal sættes til at kommunikere med de øvrige pc'er på regnskabsafdelingens netværk. Derfor tildeles dette netkort IP-adressen 192.168.20.10. Netværket ser herefter således ud:

Figur 6-2. Netværk med gateway



Gateway er nu mellemledet som direkte kan kommunikere med samtlige pc'er i virksomheden. De øvrige pc'er har fået en gateway sat op (angivet som GW). For PC2 og PC3's vedkommende er denne adresse gateways IP 1, mens de øvrige har gateway's IP 2.

Ud fra ovenstående kan det konkluderes at en gateway er nødvendig når en computer skal kommunikere med en anden computer som ikke er inden for det samme IP-net. Det er derfor logisk at en gateway vil være påkrævet når man eksempelvis skal benytte internettet, da størstedelen af adresserne ligger uden for ens eget IP-net.

6.4. Klasseløst internet

Igennem 1990'erne er antallet af brugere på internettet mildest talt eksploderet, hvilket har udløst en akut mangel på IP-adresser. Som følge deraf er internettet i dag rent faktisk blevet klasseløst, idet man ikke længere har nok adresser tilbage til at delegere A-, B- eller C-net til firmaer/institutioner. Man bliver derfor nødt til at uddele IP-adresserne i meget mindre portioner. Følgende er en oversigt over de nye og langt mindre net:

Tabel 6-4. IP-net oversigt

Antal adresser:	Netklasse:	Subnet-maske:	Bit:
1		255.255.255.255	32
2		255.255.255.254	31
4		255.255.255.252	30
8		255.255.255.248	29
16		255.255.255.240	28
32		255.255.255.224	27
64		255.255.255.192	26
128		255.255.255.128	25
256	C-Net	255.255.255.0	24
65536	B-Net	255.255.0.0	16
16777216	A-Net	255.0.0.0	8

(Nettene mellem A-, B- og C-klasserne er ikke medtaget!)

Nettene benævnes på deres subnet-maske, eller subnet-maskens bit. Tallet i kolonnen "Bit" fortæller hvor mange af de 32 bit som er sat. Det første net med subnet-masken 255.255.255.255 har 32 bit sat og kaldes derfor et "32-net".

Hvert IP-net indeholder en netværksadresse og en broadcast-adresse. Netværksadressen er den første adresse i IP-nettet og bruges til at definere nettet. Broadcast-adressen er den sidste adresse, og benyttes til at kontakte samtlige andre adresser i IP-nettet på én gang. Broadcast-adressen kan være nyttig hvis man eksempelvis vil pinge (se om en computer er aktiv) samtlige computere på et IP-net, i stedet for at pinge hver enkelt kan man blot pinge broadcast-adressen.

Den første og sidste adresse i et IP-net er derfor altid reserveret og kan ikke tildeles andre enheder. Det mindste IP-net som kan bruges til at tildele IP-adresser fra er derfor et 30 bit net som indeholder 4 IP-adresser. Efter netværksadressen og broadcast-adressen er talt fra, er 2 adresser i overskud som kan tildeles computere.

IP-nettet som vores test-IP-adresse 195.249.116.158 tilhører, vil man umiddelbart kalde 195.249.116.0/24, da IP-adressen tilhører et C-klasse-net som jo netop består af 24 bit.

Netværksadressen vil i dette IP-net hedde 195.249.116.0 og Broadcast-adressen 195.249.116.255.

6.5. Eksempel på netværk tilkoblet internettet

Hvis man forestiller sig et firma som netop er blevet forbundet til internettet via en fast forbindelse, og som af deres internetudbyder er blevet tildelt IP-nettet 195.249.116.144/28, vil følgende informationer gøre sig gældende:

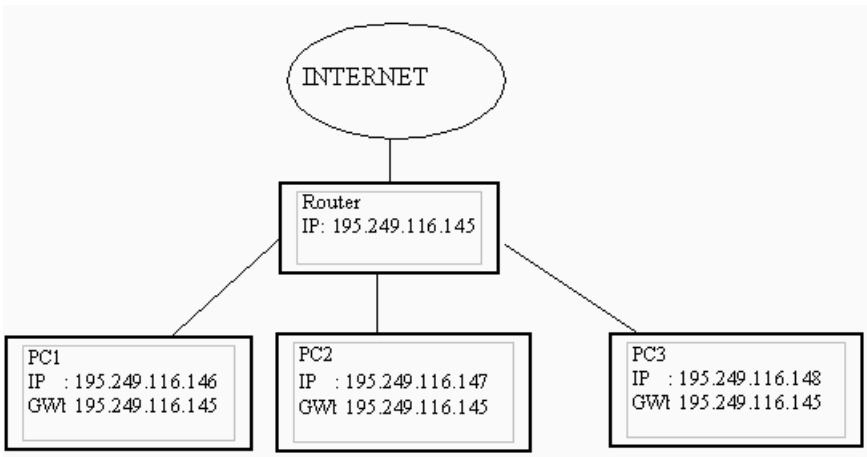
Tabel 6-5. IP-net eksempel

Subnet-maske	255.255.255.240
Antal IP-adresser	16
Netværksadresse	195.249.116.144
Broadcast-adresse	195.249.116.159
Første tilgængelige adresse	195.249.116.145
Sidste tilgængelige adresse	195.249.116.158

Internetudbyderen har samtidig leveret en router, som er blevet fysisk tilkoblet netværket. Hensigten med denne er at den skal fungere som gateway. Den har derfor fået tildelt den første af de tilgængelige adresser.

Firmaet har 10 computere som er tilkoblet netværket. TCP/IP skal på hver af disse sættes op til at bruge internettet ud fra nedenstående oplysninger:

Figur 6-3. Netværk med internet



Ovenstående eksempel viser hvorledes de første 3 pc'er skal sættes op, de resterende sættes op på samme måde.

Computerne vil nu være fuldt ud i stand til at kommunikere med brugere på internettet, ligeså gælder den anden vej. Hver enkelt bruger på internettet kan nu også kommunikere direkte med hver af computerne.

6.6. Private IP-adresser

Der er fordele ved at gøre sine computere direkte tilgængelige over internettet, men der bestemt også ulemper! For hackere er det et rent slaraffenland; eftersom de kan kommunikere med samtlige adresser direkte, er der ingen mellemled til at bremse deres indbrudsforsøg.

For at hindre direkte kommunikation mellem internettet og lokalnettet benytter man såkaldte private IP-adresser som ikke kan nås fra internettet.

Private IP-adresser er specielle adresseområder som man har valgt at "frede", netop med det formål at firmaer/institutioner kan vælge et af disse og benytte det på deres lokalnetværk. Private IP-adresser er altså derfor modsætningen til offentlige IP-adresser, da private adresser ikke er brugbare på internettet. De bliver simpelthen sorteret fra når de forsøger at komme igennem en gateway. Følgende oversigt viser de private IP-adresseområder:

Tabel 6-6. Private IP-adresser

Netklasse:	Antal klasser:	IP-adresser:
A	1	10.*.*.*
B	16	172.16.*.* op til 172.31.*.*
C	256	192.168.*.*

I praksis fungerer et netværk med private adresser ikke meget anderledes end et netværk med officielle adresser. Når de enkelte computere skal kommunikere med en IP-adresse som er uden for deres eget IP-net, og derfor kommunikerer via deres gateway, er denne sat op til at "oversætte" deres adresse til en officiel adresse.

Basalt set fungerer det på den måde at de beder gateway'en om at kommunikere med en given internet-computer på deres vegne. internet-computeren "tror" at den udelukkende kommunikerer med gateway'en - og kender derfor ikke til computerne bag ved gateway'en. Man kan derfor også kalde det en slags "envejs-kommunikation", da det kun er computeren med den private adresse der kan kontakte en computer på internettet og ikke omvendt.

Denne form for adresse-oversættelse kaldes også NAT, Network Address Translation, hvilket er en slags udvidet routnings-funktion. Mange routere kan udføre denne funktion. For at sætte Linux op med kerne 2.2 til det, skal programmet **ipchains** benyttes, og med kerne 2.4 anvendes **netfilter**. Dette er forklaret

nøjere i bogen "Linux – Friheden til sikkerhed på internettet".

Det er meget vigtigt, at man sørger for at vælge private IP-adresser når man kører med NAT som er inde for det tilladte område. Hvis man i stedet vælger at benytte et IP-net som i forvejen er i brug, vil konsekvensen være, at man ikke kan kommunikere med de enheder som retmæssigt har fået tildelt IP-adresserne.

Det skyldes, at ens computere er sat op til at benytte en gateway, når de skal kommunikere med enheder uden for det lokale IP-net. Hvis IP-adressen derimod er inden for det lokale IP-net vil de forsøge at kommunikere med den pc som har den aktuelle IP-adresse, eftersom det er den forkerte pc, vil det naturligvis mislykkes.

Hvis man alligevel forestillede sig at computerne kommunikerede via gateway'en og kunne kontakte de rigtige enheder, ville der opstå en konflikt idet der nu eksisterede flere enheder med samme IP-adresser. Dette er ikke tilladt, og styresystemet vil straks advare om problemet. Hvis IP-adresser ikke er unikke, kan computere ikke finde ud af at bruge dem.

6.7. Eksempel på netværk med NAT

Vi bygger videre på det foregående eksempel og benytter derfor informationerne fra Tabel 6-5 igen, men denne gang vil vi benytte NAT. Primært for at øge sikkerheden, men også for at øge fleksibiliteten, i og med at vi ikke behøver at kontakte vores internetudbyder og ansøge om flere IP-adresser, hvis de 16 viser sig at være for få.

Der skal tages stilling til 2 ting: Hvilken enhed skal udføre NAT-oversættelsen, og hvilket privat IP-net skal benyttes. Enten vælger man at købe en internet-løsning med NAT, hvilket vil sige at den router internetudbyderen leverer allerede er sat op med NAT. Eller også sætter man en computer op til at køre NAT.

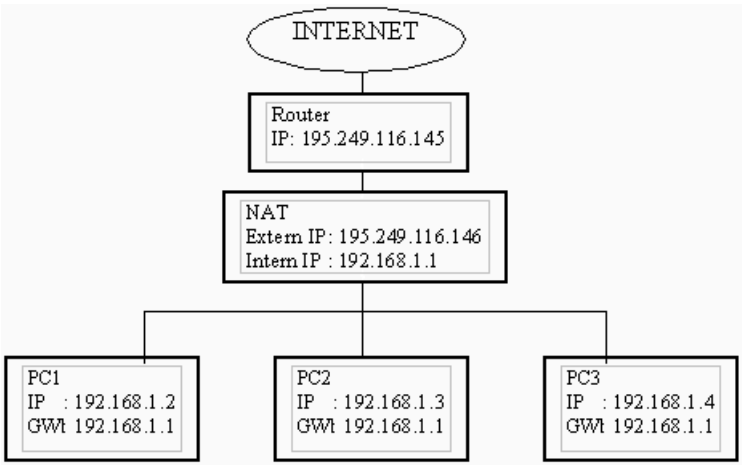
Valget af det private IP-net afhænger meget af ens behov, og hvor stort antal computere man regner med at have på sit netværk inden for en overskuelig årrække. I vores eksempel vælger vi 192.168.1.0/24:

Tabel 6-7. Privat IP-net eksempel

Subnet-maske	255.255.255.0
Antal IP-adresser	255
Netværksadresse	192.168.1.0
Broadcast-adresse	192.168.1.255
Første tilgængelige adresse	192.168.1.1
Sidste tilgængelige adresse	192.168.1.254

Routeren skal bibeholde IP-adressen 195.249.116.145, mens der i næste led skal placeres en enhed til at udføre NAT-oversættelsen. Denne skal på "ydersiden" (mod routeren) have tildelt en officiel IP-adresse. Her vælger vi den næste ledige, 195.249.116.146. Desuden skal den have tildelt en privat IP-adresse på indersiden, her vælger vi den først tilgængelige adresse, 192.168.1.1.

Figur 6-4. Netværk med NAT



Ovenstående eksempel illustrerer hvorledes netværket skal sammenkobles hvis man vælger selv at sætte en enhed op til at udføre NAT-oversættelsen. Routeren bliver fysisk tilkoblet NAT-enheden, som samtidig via et sekundært netkort er tilkoblet det eksisterende netværk. Denne vil herefter fungere som gateway for PC1-PC3.

Alternativt kan man, som omtalt, købe en løsning hvor routeren indeholder NAT-opsætningen. I dette tilfælde vil internetudbyderen foretage den nødvendige opsætning. I forhold til ovenstående figur vil man kunne skippe NAT-enheden, idet routeren selv klarer opgaven. Routeren vil i så fald få tildelt både en ekstern og en intern IP-adresse.

Hvis man senere ønsker at tilføje yderligere pc'er til denne opsætning, kan det gøres problemfrit, indtil man har opbrugt de resterende adresser i IP-nettet. Herefter vil det være nødvendigt at skifte IP-net. En simpel løsning vil være blot at skifte subnet-maske, eksempelvis til /16, hvilket vil betyde 255 gange flere IP-adresser.

6.8. Domain Name Service (DNS)

På internettet ser man ofte adresser, som hedder noget i retning af www.smartnavn.dk. De 3 w'er står for World Wide Web. Internettet er verdensomspændende og nogle betragter det som værende et stort spindelvæv.

Men faktisk gemmer der sig en IP-adresse bag hver enkelt af disse web-adresser. Dette er dog ikke noget den normale bruger lægger mærke til. Men faktisk er det således, at der slet ikke eksisterer egentlige web-adresser. Det er udelukkende IP-adresser.

Når en bruger skriver en web-adresse i sin browser, anmoder hans computer en DNS-server om IP-adressen på den pågældende web-adresse. Det er kun IP-adresser, som er brugbare på et TCP/IP-netværk.

En DNS-server er en service som en server kan køre ved siden af de almindelige services. Hvis der er meget belastning, kan man vælge at have en DNS-server, som udelukkende kører DNS-service.

Hver gang DNS-serveren får en anmodning om en web-adresse, slår den web-adressen op i sin database og finder den tilhørende IP-adresse.

For at dette kan virke over hele verden, er der et vist system i det. Det går ud på, at hvert land har et domæne. Ligesom institutioner og firmaer i det pågældende land har et "underdomæne/subdomæne". For eksempel hedder SSLUG's domæne sslug.dk, hvilket er et subdomæne til .dk-domænet. 'dk' angiver Danmark.

Hos SSLUG har man så tilføjet et yderligere subdomæne, www. Den komplette adresse hedder således www.sslug.dk. Hvis man skriver denne adresse i sin browser, vil computeren automatisk spørge sin DNS-server om IP-adressen på domænet og herefter lave en webforespørgsel til IP-adressen.

Det vil sige at domænenavne er hierarkisk opbygget. Landet er øverst, derefter navnet på firmaet/institutionen (eventuelt i en forkortet udgave), og til sidst subdomænet som "peger" på serveren for domænet.

De øverste domæner i hierarkiet kaldes også TLD (Top Level Domains = Topniveau-domæner), og disse administreres af de såkaldte root-servere. En root-server besvarer kun forespørgsler på TLD-niveau. I dag eksisterer der på verdensplan 13 root-servere, som hver besvarer cirka 200 millioner forespørgsler om dagen.

Almindelige computere kommunikerer ikke direkte med root-serverne. De sender i stedet en web-adresse-forespørgsel til enten en lokal DNS-server, eller DNS-serveren hos deres internetudbyder.

Hvis vi forestiller os at DNS-serveren ns.sslug.dk som er ansvarlig for domænet sslug.dk, modtager en forespørgsel på subdomænet www.sslug.dk, ser serveren med det samme at det er det domæne som den selv er ansvarlig for, og kan derfor besvare denne forespørgsel uden at skulle kontakte andre DNS-servere.

Hvis serveren i stedet modtog en forespørgsel på subdomænet `www.linux.dk` ville den straks kunne se at det ikke var et domæne den selv var ansvarlig for. Den ville derfor være nødt til at starte med det

øverste rent hierarkisk set. Serveren vil derfor indlede med at spørge en root-server om topniveau-domænet `.dk`, den vil af root-serveren blive bedt om at kontakte DK-Hostmasters server, som er ansvarlig for det domæne. DK-Hostmasters server vil henvise til serveren ns.linux.dk som er ansvarlig for domænet `linux.dk`. Serveren vil derfor ende med at få besvaret sit spørgsmål når den spørger `ns.linux.dk`.

Man kan betragte en DNS-server som en telefonbog, hvor personerne er web-adresser. Begge steder får man et nummer (telefonnummer/IP-adresse), som systemet kan bruge til at lokalisere informationen/personen.

For at begrænse DNS-trafikken benytter DNS-servere sig af DNS-cache. Her bliver de svar serveren har fået fra øvrige DNS-servere gemt. Hvis man gentog ovenstående eksempel, ville serveren derfor ikke starte forfra med at spørge alle serverne, men i stedet blot kigge på det foregående svar.

Eftersom der dagligt bliver lavet ændringer på mange servere, er det vigtigt at en DNS-server ikke tror den bare kan gemme sin cache evigt. Hvis eksempelvis `www.sslug.dk` bliver ændret til at pege på en anden webserver, vil det ikke være meget værd hvis vores DNS-server stadig fortæller alle og enhver hvad den hed førhen! Derfor har DNS-servere det man kalder en TTL (Time To Live), hvilket simpelthen definerer hvor lang tid et svar må leve. Hvis vores server igen modtager forespørgslen på `www.sslug.dk`, og TTL'en er udløbet, vil serveren være nødt til at starte forfra med at spørge de forskellige DNS-servere.

Som et konkret eksempel på et DNS-opslag kan vi f.eks. bede om SSLUG's IP-adresse. Til dette bruger vi kommandoen **nslookup**.

```
[tyge@hven ~]$  nslookup www.sslug.dk
Server:  danpost.uni-c.dk
Address:  129.142.6.64

Non-authoritative answer:
Name:    sslug.sslug.dk
Address:  130.228.2.150
Aliases:  www.sslug.dk
```

Vi kan i eksemplet se, at vi beder `danpost.uni-c.dk` med IP-adresse 129.142.6.64 om adressen på SSLUG's webserver. SSLUG har IP-adressen 130.228.2.150, og maskinen er åbenbart også kendt som `sslug.sslug.dk`. Svaret er "Non-authoritative", idet danpost DNS-serveren ikke er herre over sslug-domænet, men har fået informationen fra en anden navneserver.

DNS indeholder også andre informationer end IP-adresser. I DNS-sprog sætter man et punktum bag navnene for at angive, at disse er absolutte. Normalt kan man dog ignorere det sidste punktum.

6.8.1. DNS-zoner

DNS er baseret på filer. Disse filer indeholder zone-data. I mange tilfælde indeholder hver fil/zone ét domæne, i det følgende kan man derfor blot betragte "zone" som et synonym for "domæne".

Navneserverne for en zone listes ved:

```
[tyge@hven ~]$ host -t ns sslug.dk
sslug.dk name server ns-soa.darenet.dk
sslug.dk name server ns.sslug.dk
sslug.dk name server ptah.dkuug.dk
```

For at finde post-serverne for et domæne skal man kende en af navneserverne for domænet. Derefter spørger man denne navneserver:

```
[tyge@hven ~]$ host -t mx domæne.dk navneserver.for.domæne.dk
```

Den ansvarlige administrator for en zone findes i zonens SOA-record (eng. Start Of Authority). SOA-recorden findes med:

```
[tyge@hven ~]$ host -t soa domæne.dk navneserver.for.domæne.dk
```

Hvis SOA starter med: `sslug.dk. IN SOA ns.sslug.dk. root.sslug.dk. (` så er zonen `sslug.dk`, den primære navneserver `ns.sslug.dk` og den ansvarlige administrators e-post-adresse `root@sslug.dk` (udskift første . med @).

Nogle navne er blot et alias for et andet navn: de har et kanonisk navn (eng: "canonical name"). Dette kan ses ved at spørge navneserveren for domænet om alt, hvad den ved om et givet navn:

```
[tyge@hven ~]$ host -t any dette.navn.domæne.dk nameserver.for.domæne.dk
```

Til nogle domæner er knyttet tekstinformation. Denne kan kaldes frem med:

```
[tyge@hven ~]$ host -t txt navn.med.info.domæne.dk
```

Hvis informationen er til stede, indeholder den ofte information om hvem der bestyrer navn.med.info.domæne.dk

Da DNS ikke er helt simpelt at sætte op korrekt, kan de ovenstående eksempler bruges, hvis du skal fejlfinde din egen DNS. De kan også bruges til at finde ud af, hvorfor netværksfejl opstår eller til at finde en ansvarlig for et domæne, som man har modtaget reklamer fra.

I forbindelse med installationen af Red Hat Linux kan du vælge pakken "caching-nameserver". Pakken foretager en simpel opsætning af navneserveren **bind**. Fra starten kender den ikke selv svaret på

navneopslag, men sender blot spørgsmålet videre og husker svaret (som beskrevet ovenfor). Den glemmer desværre alt, hvad den har lært, når maskinen lukkes ned, så du får mest glæde af en caching navneserver, hvis den kører på en maskine, der er i gang altid - f.eks. en server på et lille lokalnetværk. Den kan selvfølgelig sagtens bruges som navneserver for Windows-pc'er på netværket.

Hvis du vil bruge Linux på en server i et mindre netværk, kan du have glæde af at lade den fungere som navneserver for et lokalt domæne. Opsætningen af dette er ikke det første Linux-eksperiment, du skal starte med, så eventuelle interesserede henvises til den detaljerede beskrivelse i DNS-HOWTO (under Red Hat, se `/usr/doc/HOWTO/DNS-HOWTO`) og http://www.sslug.dk/artikler/dnsbind.shtml er sikkert også af interesse.

6.9. DNS-opsætning

DNS er interessant, når man har flere end f.eks. 5 maskiner på netværket og hvis man ikke længere ønsker at kopiere `/etc/hosts`-filer (Se Afsnit 1.6) rundt til alle maskiner. Har du mange maskiner på netværket, kan du i stedet have en fælles navneserver (eng. nameserver) til at kunne håndtere navneopslag.

Som gennemgående eksempel antager vi her at vi har domænet "intranet" med følgende maskiner. IP-adresserne her er private, dvs. at maskinerne ikke er på internettet. Vi vælger derfor adresser fra Tabel 6-6 i Afsnit 6.6.

```
192.168.0.1      linus.intranet    linus
192.168.0.2      alan.intranet     alan
192.168.0.3      richard.intranet  richard
192.168.0.4      eric.intranet     eric
```

Vi vedtager også at maskinen "linus" skal være DNS-server, og at samme maskine også skal være kendt som "ns.intranet" eller i kort form blot "ns".

6.10. Bind

Dette afsnit vil gennemgå opsætningen af et registreret domæne på en DNS-server. Strukturen fra BIND 8.x er benyttet, da dette er den version de fleste DNS-servere anvender, modsætningen er den noget ældre BIND 4.x. De følgende eksempler kører på BIND 8.1.2, men nye versioner udvikles hele tiden. Opgraderinger kan hentes fra http://www.isc.org.

Den overordnede opsætningsfil er `/etc/named.conf`. I denne defineres hvilke zoner (domæner) DNS-serveren administrerer og i hvilke filer de enkelte zoner er defineret.

Som standard vil man skulle definere 4 zoner for domænet "intranet" med IP-adresser i intervallet 192.168.0.1-192.168.0.254 (ikke 255, da dette er broadcast-adressen).

- `.` (punktum) : Først og fremmest root, denne zone henviser til root-serverne således at DNS-serveren kan spørge disse om domæner den ikke selv er ansvarlig for, hvilket vil sige at det er denne zone som DNS-serveren benytter når den modtager forespørgsler om IP-adresser på et domæne, som den ikke selv er ansvarlig for. Nedenfor er dette angivet i filen `/var/named/root.cache`.
- `0.0.127.in-addr.arpa`: Reverse lookup betyder, at man har en IP-adresse og gerne vil vide hvilket domæne denne tilhører. For at DNS-serveren kan besvare denne slags forespørgsler, kræves det at reverse-zonerne bliver defineret. Reverse-zonen 127.0.0 refererer blot til DNS-serveren selv (filen `/var/named/127.0.0.rev`), TCP/IP-software kræver i nogle tilfælde at denne IP-adresse henviser til DNS-serveren selv. Derfor skal denne zone altid medtages.
- `0.168.192.in-addr.arpa`: Den anden (eller de andre) reverse-zoner er de "rigtige" IP-adresser. Hvis man for eksempel råder over adresserne 192.168.0.1-192.168.0.254, laver man en reverse-zone-fil til disse IP-numre. (De nævnte adresser er kun et eksempel, de eksisterer ikke på internettet da de er nogle af de "private" adresser). Denne zone gemmes i `/var/named/192.168.0.rev`.
- `intranet`: Du skal også lave en fil, der kan afbilde fra navn til IP-nummer for det domæne, du sætter op. I filen `/var/named/intranet` kan du erklære navne og aliaser for alle maskiner.

6.10.1. /etc/named.conf

Følgende er et eksempel på en `/etc/named.conf` fil som indeholder ovenstående grundlæggende opsætning:

```
options {
  /* Hvor er filerne med DNS-opsætning */
  directory "/var/named";

  /* Indsaet 1-3 forwarder nameservere */
  /* Brug dem fra din ISP og indsaet IP-adresserne */
  /* i stedet for NNN.NNN.NNN.NNN og MMM.MMM.MMM.MMM */
  forwarders {
          NNN.NNN.NNN.NNN;
          MMM.MMM.MMM.MMM;
  };

  /* Lyt på DNS-forespørgsler til to IP-adresser */
  listen-on { 192.168.0.1; 127.0.0.1 ; };

  /* Udkommenter følgende linje hvis dit domæne
     er kendt på internettet */
  notify no;

};

logging {
  category lame-servers { null; };
  category cname { null; };
```

```
};

zone "." in {
  type hint;
  file "root.cache";
};

zone "0.0.127.in-addr.arpa" in {
  type master;
  file "127.0.0.rev";
};

zone "0.168.192.in-addr.arpa" in {
  type master;
  file "192.168.0.rev";
};

zone "intranet" in {
  type master;
  file "intranet";
};
```

De første linjer definer i hvilket bibliotek de senere omtalte filer skal placeres. Standard er: `/var/named`, men du kan også anvende f.eks. `/etc/named` eller andre steder.

Logging-linjen forhindrer at man får en masse unødigt i sin syslog. (Log-funktionen er meget avanceret, en masse forskellige kriterier kan defineres).

Næste linje er en zone-definition. Her gælder det root-zonen, selve filen hentes fra ftp://ftp.internic.net. De resterende linjer fortæller først hvilket domæne det handler om, og derefter i hvilken fil selve domænet er defineret. Samtidig får serveren af vide om den er "master" eller "slave" for det pågældende domæne. I ovenstående tilfælde er DNS-serveren master for alle zoner. For at kunne registrere et domæne kræves det at man har 2 DNS-servere til domænet, master og slave, eller primær og sekundær. Følgende er et eksempel på en slave-zone:

```
zone "intranet" in {
  type slave;
  file "intranet";
  masters { 192.168.0.1; };
};
```

Udover at der nu står slave i stedet for master, er en linje tilføjet som fortæller hvilken server er master. Grunden til at slave-serveren skal kende master-serveren er, at slave-serveren henter zone-filerne fra master-serveren. Dette betyder at ændringer i et domæne på master-serveren automatisk vil blive overført til slave-serveren.

6.10.2. /var/named/root.cache

Vi kan f.eks. bruge følgende udgangspunkt for `/var/named/root.cache` – dvs. root-zonen.

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Feb 28, 1997
;       related version of root zone:   1997022800
;
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                        3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;
```

```
; formerly NS.NIC.DDN.MIL
;
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
;
; temporarily housed at NSI (InterNIC)
;
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File
```

6.10.3. /var/named/intranet

Den næste fil er for maskinerne i 192.168.0.*-området:

```
$TTL   1D
@   IN SOA ns.intranet. hostmaster.intranet. (
                                        1999102400 ; serial number
                                        60000      ; refresh (1 day)
                                        3600       ; retry (1 hour)
                                        2600000    ; expire (1 month)
                                        60000      ; minimum (1 day)
)

;
```

```
; NB: BRUG TABULATOR SOM MELLEMRUM
;

;
; Dette domænes DNS-servere:
;
      IN   NS   ns.intranet.

;
; Mailserveren for dette domæne:
;
      IN   MX   0   mail.intranet.

;
;  Standardhostnavne afbildet til IP-adresser:
;
localhost   IN   A   127.0.0.1
linus       IN   A   192.168.0.1

;
; Øvrige hostnavne afbildet til IP-adresser:
;
alan        IN   A   192.168.0.2
richard     IN   A   192.168.0.3
eric        IN   A   192.168.0.4

;
; Alias adresser:
;
ns          IN   CNAME   linus.intranet.
mail        IN   CNAME   linus.intranet.
```

Husk: Den første linje $TTL - "Time to live" skal med, og der anvendes tabulator til at adskille hvert felt, og der må ikke anvendes mellemrum. Bemærk specielt, at der er punktum efter hvert maskin-navn i CNAME-linjerne.

Når en computer ude på nettet har spurgt vores server om noget på vores zone vil den have modtaget noget data. Efter et par dage vil det være muligt at disse data er forældet. Derfor kan man med "TTL"-parameteren angive hvor lang tid en klient må beholde data som den har fået fra en DNS-server.

Linje to til syv angiver diverse parametre for selve domænet. Det er ikke så vigtigt at forstå ordenen i disse linjer, derimod er det vigtigt at forstå betydningen. Det første tegn "@" er en variabel for selve domænet. Her kunne altså i stedet stå "intranet.". Grunden til at man i stedet sætter en variabel, er at man på den måde ikke behøver ændre dette felt hvis man bruger denne zone-fil til at lave en anden zone-fil.

"IN" står for "INTERNET" og angiver den klasse data man benytter. I dag bliver der meget sjældent benyttet andre klasser, så derfor er IN default, og man behøver derfor ikke angive dette. SOA står som tidligere skrevet for "Start Of Authority" og fortæller efterfølgende hvilken navneserver der er ansvarlig for denne zone.

"ns.intranet." fortæller hvilket domæne navneserveren er ansvarlig for, "hostmaster.intranet" er e-post-adressen til vedkommende som er ansvarlig for domænet. Bemærk, at det første . (punktum) skal læses som et @.

"Serial" er serienummeret for zonen. Hver gang man laver en ændring, skal serienummeret opdateres. Eneste regel er at tallet skal blive mindst én højere (inkrementeres). De fleste vælger at bruge dags dato sammen med et 2-cifret tal. Det gør det muligt at vide hvornår sidste opdatering blev foretaget. Grunden til at man har serienummeret, er at man kun opdaterer sine zone-filer på sin master-server. Slave-serveren vil derefter selv hente ændringerne, men den ved kun at der er foretaget ændringer *hvis* serienummeret er blevet inkrementeret.

"Refresh" fortæller hvor tit slave-serveren skal spørge master-serveren om serienummeret på det pågældende domæne og herved beslutte om der skal foretages en overførsel.

Hvis slave-serveren ikke er i stand til at skabe kontakt til master-serveren i første forsøg, definerer man med "Retry" hvor lang tid den skal vente før den prøver igen.

I tilfælde af at slave-serveren slet ikke kan få kontakt til master-serveren, vil det efter nogle dage kunne være sandsynligt at der er foretaget ændringer på master-serveren som slave-serveren ikke har. Hvilket vil sige at den primære og den sekundære server giver modstridende oplysninger til deres klienter! For at undgå dette kan man definere hvor lang tid slave-serveren skal tro på at dens zone-data er gode nok. Dette gøres med "expire"-parameteren. Det skal forstås således, at når slave-serveren ikke kan kontakte master-serveren og det angivne tidsrum er udløbet, vil slave-serveren smide sine data om den pågældende zone ud, og altså ikke besvare forespørgsler omkring denne zone.

Ovenstående parametre som angives i tidsrum, kan skrives i S, M, H, D (Sekunder, Minutter, Timer, Dage). De fleste vælger at angive dem alle i sekunder, det gør dog ingen forskel, og det er rimelig besværligt at skulle omregne eksempelvis 86400 sekunder (1 dag).

Der findes mange meninger om hvorledes ovenstående parametre skal sættes. De 4 første handler udelukkende om DNS-serverne, og derfor er det yderst begrænset hvor store fejl, der kan opstå ved at sætte nogle af disse 4 parametre "forkert", hvis man selv administrerer både master og slave. Derimod er det mindre hensigtsmæssigt at sætte TTL helt forkert. Resultatet kan eksempelvis blive at man ikke kan modtage post i en uges tid, hvis man er kommet til at angive en forkert IP-adresse til mail-serveren, og TTL er sat til en uge.

Resten af zone-filen er selve "records". Den første kolonne fortæller hvilket navn man vil omsætte. Eksempelvis kunne der stå "mail.intranet" for at fortælle hvad "mail.intranet" skulle referere til. Hvis man ikke angiver noget i den første kolonne, vil selve zonen være default, altså "intranet."

En vigtig ting som altid gælder når man angiver navne: Hvis man ikke afslutter med et "." (punktum) vil serveren automatisk tilføje det pågældende domæne til navnet. "mail.intranet" vil derfor blive til "mail.intranet.intranet".

Herefter skal der angives hvilken klasse man arbejder med – her er "IN" igen default. Derpå skal man fortælle hvilken type record man vil angive (de mest brugte vil blive gennemgået i det efterfølgende). Det sidste der skal angives, er hvilket navn eller hvilken IP-adresse DNS-serveren skal svare tilbage med, når den bliver spurgt om denne record.

Den først record er af typen "NS" (Name Server), her angiver man navnet på DNS-serveren, først den primære, herefter den sekundære.

Den næste record er af typen "MX" (Mail eXchanger), her angives navnet på post-serveren.

Herefter kommer en stribe standard "A"-records (A = Address). Denne record angiver hvilket IP-nummer den omtalte host har.

"CNAME" står for Canonical name og betyder alias. Denne record-type bruges til at have flere navne til at referere til en IP-adresse.

6.10.4. /var/named/127.0.0.rev

Den første "reverse"-fil er for "localhost", som vi i `/etc/named.conf` valgte at gemme i `/var/named/127.0.0.rev`.

```
$TTL    1D
@ IN  SOA   ns.intranet. hostmaster.intranet. (
                                        1999102400 ; serial number
                                        60000     ; refresh (1 day)
                                        3600       ; retry (1 hour)
                                        2600000    ; expire (1 month)
                                        60000     ; minimum (1 day)
)

;
; NB: BRUG TABULATOR SOM MELLEMRUM
;

;
; DNS-serveren for denne zone:
;
IN   NS    ns.intranet.

;
; Reverse mappings:
;
1   IN   PTR   localhost.
```

Husk den første linje $TTL - "Time to live" - skal med, og der anvendes tabulator til at adskille hvert felt. Bemærk specielt, at der er punktum efter hvert maskin-navn.

Igen har vi SOA-recorden (fra linje 2 til 7). Det er en fordel at have en generel SOA-record for alle sine zoner, dog kan man naturligvis have behov for forskelle nogle steder, men rent administrativt er det langt mere overskueligt med samme parametre.

Grunden til at der kun er angivet en DNS-server, er at denne zone udelukkende er for 127.0.0 som altså kun kan være den lokale server, hvilket ikke har noget at gøre med den/de andre DNS-servere.

Reverse records er af typen PTR. Da forespørgsler til reverse-zoner skal besvares med et navn i stedet for en IP-adresse er strukturen i disse zoner også anderledes. Der er simpelthen byttet om på navnet og IP-adressen. Derfor står IP-adressen først, i det her tilfælde 1 (1 + zone-navnet (127.0.0) = 127.0.0.1), herefter klassen (IN), record-typen (PTR) og til sidst navnet (localhost).

6.10.5. /var/named/192.168.0.rev

Den næste "reverse"-fil er for maskinerne i 192.168.0.*-området:

```
$TTL   1D
@ IN SOA   ns.intranet. hostmaster.intranet. (
                                        1999102400 ; serial number
                                        60000      ; refresh (1 day)
                                        3600       ; retry (1 hour)
                                        2600000    ; expire (1 month)
                                        60000      ; minimum (1 day)
)

;
; NB: BRUG TABULATOR SOM MELLEMRUM
;

;
; Name Servers for this reverse zone:
;
IN   NS    ns.intranet.

;
; Reverse mappings:
;
1   IN   PTR   linus.intranet.
2   IN   PTR   alan.intranet.
3   IN   PTR   richard.intranet.
4   IN   PTR   eric.intranet.
```

Ovenstående er reverse zone-filen for zonen 192.168.0.*. Nok en gang - den første linje $TTL - "Time to live" skal med, og der anvendes tabulator til at adskille hvert felt. Bemærk igen, at der er punktum efter hvert maskin-navn.

Her gælder de samme regler som for 127.0.0 zonen. 4 reverse records er defineret fra 1 til 4 svarende til 192.168.0.1-192.168.0.4, dvs. vores fire maskiner. Har du brug for et andet eksempel på opsætning af DNS, så kan du se i http://www.sslug.dk/artikler/dnsbind.shtml.

6.11. Start named

Efter at have redigeret navneserver-filerne færdig er du klar til at starte **named**. Kører du Red Hat, er det nemt. Du skal installere **bind**-pakken og køre

```
[root@linus /root]#  /sbin/chkconfig --level 345 named on
```

Dette er en smart måde at få startet `/etc/rc.d/init.d/named` i run-levels 3, 4 og 5. Du kan nu eksperimentere og se om din opsætning er i orden. Start en xterm og skriv **tail -f /var/log/messages**. I en anden xterm skriver du **/etc/rc.d/init.d/named start**. Kommer der ikke fejl ud, er du igennem første syntakstjek. Dernæst kan du starte med at spørge din lokale DNS, f.eks.:

```
[root@linus /root]# nslookup linus linus
[root@linus /root]# nslookup alan linus
[root@linus /root]# nslookup 192.168.0.3 linus
[root@linus /root]# nslookup localhost localhost
```

Tjek både på IP-adresser og maskinnavne. Går alt glat, skal dine Unix-maskiner nu have `/etc/resolv.conf` med indhold for at bruge din nye DNS-server.

```
search intranet
nameserver 192.168.0.1
```

Endelig skal det nævnes, at der i tidens løb har været en del sikkerhedsfejl i DNS-serveren BIND, og det kan kraftigt anbefales at følge sikkerhedslisterne på internettet.

- Bugtraq http://www.securityfocus.com bør du følge med på for at læse om de nyeste afsløringer af sikkerhedsbrister (exploits).
- Root Shell http://www.rootshell.com har alt indenfor diskussion af sikkerhed.
- Linux Today http://linuxtoday.com bringer også sikkerhedsbrister frem sammen med andre nyheder. Det er muligt at tilpasse hvilke nyheder man får, f.eks. nyheder om sikkerhed.
- http://www.securityportal.com/ er et andet interessant sted for folk, som interesserer sig for sikkerhed.
- http://lwn.net/ er Linux Weekly News. Der er hver torsdag en ny udgave, hvor der er et afsnit om sikkerhed.

Du kan finde et mindre eksempel på DNS-filerne under ’Eksempler’ til denne bog på hjemmesiden www.linuxbog.dk/ (http://www.linuxbog.dk/).

6.12. Sikret-DNS

Hvis en angriber har held med et angreb på en DNS server og kan udføre root kommandoer på systemet, kan konsekvenserne blive uoverskuelige. Der findes en sikkerhedsteknik til begrænsning af de skader en angriber kan forvolde samt til at gøre det umuligt at bruge den til at bryde ind i det øvrige system den kører på. Teknikken går ud på, at afspærre angrebet i en isoleret del af filsystemet, det såkaldte "chroot-jail". For at opnå uautoriseret adgang kan angriberen bruge forskellige svagheder i programmer som har suid/sgid bitten sat. Den største fordel ved denne teknik er begrænsningen af adgang til sådanne programmer. I denne vejledning har DNS-processen ingen adgang til programmer og den sættes op til at køre som en upriviligeret bruger. Denne kombination gør livet *meget* besværligt for den eventuelle angriber. Når en proces køres på denne måde kan den ikke se det øvrige filsystem der er udenfor afgrænsningen og den vil ikke kunne tilgå det øvrige filsystem. Dem der har brugt et ftp program for at koble op til en ftp-server, har sandsynligvis set et "chroot-jail" som det ser ud indenfra.

Man skal dog huske at denne teknik ikke løser alle sikkerhedsproblemer men blot en del af dem. Der findes sågar rapporter og vejledninger om hvordan man som angriber bryder ud af isolationen for at tilgå servers virkelige filsystem. Forudsætningen for at dette lykkes er at angriberen først skal opnå root-privilegier inden han/hun kan bryde ud fra den isolerede del af filsystemet. For en uddybning af dette se afsnit 3.8 i "Running Services in a chroot Environment":
http://www.nic.com/~dave/SecurityAdminGuide/SecurityAdminGuide-3.html
(http://www.nic.com/~dave/SecurityAdminGuide/SecurityAdminGuide-3.html) Samt:
http://www.linuxgazette.com/issue30/tag_chroot.html
(http://www.linuxgazette.com/issue30/tag_chroot.html) Denne teknik kan feks. ikke forhindre en cracker i at forespørge dns-serveren om oplysninger til brug i forbindelse med andre angreb på computeren.

Den følgende vejledning gælder kun for BIND version 9 og opefter men noget kan dog bruges til tidligere versioner. Denne vejledning tager udgangspunkt i opsætningsfilerne for en fungerende DNS-server og derfor er det oftest unødvendigt at ændre dem. Hvis DNS-serveren køres som upriviligeret bruger og beskyttes samtidig af en meget restriktiv dørvøgter kan man dog alligevel være nødt til at lave en lille ændring i opsætningsfilen. Dette forklares i Afsnit 6.12.7.

I disse afsnit bruges variablen `$JAIL` hyppigt. F.eks. skal kommandoen **echo $JAIL** bogstaveligt skrives sådan.

6.12.1. Indsamling af oplysninger om serverens opsætning før "chroot".

Der findes flere distributioner og de placerer filer på hver sin måde. Det ville være for snævert hvis denne vejledning, var lavet således at den kun kan bruges til én af disse distributioner. For at sætte serveren op skal man kende navnet og placeringen på de relevante filer i hver distribution. Vi nøjes med 4 forskellige distributioner i håb om at det rammer tilstrækkelig bredt.

Vi skal bruge stinavnet på opstartsfilen der anvendes til at starte og stoppe processen.

- Red Hat: `/etc/rc.d/init.d/named`
- Debian: `/etc/init.d/bind9`
- Mandrake: `/etc/rc.d/init.d/named`
- SuSE: `/etc/init.d/named`

Placeringen af den overordnede opsætningsfil. Variablen `KONFIL` sættes lig med denne mappe, dog uden det første skråstreg.

- `konfil=`
- Red Hat, Mandrake og SuSE: `etc/named.conf`
- Debian: `etc/bind/named.conf`

Serverens forvalgte placering af zone filer. Denne mappe nævnes som regel i den overordnede opsætningsfil (se eksemplet i Afsnit 6.10.1) og den er angivet som `var/named` i det følgende eksempel:

```
 options { directory "/var/named";}
```

Variablen `OPTDIR` sættes lig med denne mappe, dog uden det første skråstreg.

- `optdir=`
- Red Hat, Mandrake og SuSE: `var/named`
- Debian: `var/cache/bind`

En mappe hvor masterzone filer normalt placeres. Denne mappe nævnes som regel i den overordnede opsætningsfil og den er angivet som `etc/bind` i det følgende eksempel:

```
zone "127.in-addr.arpa" {
 type master;
 file "/etc/bind/db.127";
};
```

Variablen `MASTERZD` sættes lig med denne mappe, dog uden det første skråstreg.

- `masterzd=`
- Red Hat, Mandrake og SuSE: `var/named`
- Debian: `etc/bind`

Det er et potentielt sikkerhedshul at lægge sine "master zone"-filer i mapper som selve processen har skriverettigheder til idet en angriber derved kunne ændre dem.

En mappe hvor processen har skriveadgang og hvor der normalt placeres slave-zone-filer. Denne mappe nævnes undtagelsesvis i den overordnede opsætningsfil og den er angivet som `etc/named/slave` i det følgende eksempel:

```
  zone "intranet" in {
  type slave;
  file "/etc/named/slave/intranet";
  masters { 192.168.0.1; };
```

Den forvalgte mappe kan sagtens bruges her hvis master filerne lægges i en anden mappe. Variablen `SLAVEZD` sættes lig med denne mappe, dog uden det første skråstreg.

- `slavezd=`
- Red Hat, Mandrake og SuSE: `var/named`
- Debian: `var/cache/bind`

For at få variabler med fornuftige værdier skal man køre det følgende som passer for den enkelte distribution.

I Red Hat, Mandrake og SuSE:

```
export konfil=etc/named.conf
export optdir=var/named
export masterzd=var/named
export slavezd=var/named/slave
```

I Debian:

```
export konfil=etc/bind/named.conf
export optdir=var/cache/bind
export masterzd=etc/bind
export slavezd=var/cache/bind
```

Der kan opstå forvirring vedrørende placeringer af filer og mapper hvilket nok ikke er særlig underligt idet der grundlæggende set er tale om 3 forskellige filsystemrødder, nemlig systemets egentlige rod `/` , den nye rod der etableres ved "chroot" og som er angivet ved variablen `$JAIL` samt DNS-serverens forvalgte mappe angivet bl.a. ved variablen `$OPTDIR`.

Det kan forøge forvirringen at DNS-serveren fortolker *relative* filnavne i forhold til serverens forvalgte mappe mens *absolutte* filnavne fortolkes i forhold til den nye rod der etableres ved "chroot".

Variablerne `OPTDIR` , `MASTERZD` samt `SLAVEZD` skal allesammen fortolkes i forhold til filsystemets rod, nemlig `/` eller `$JAIL`.

Et fiktivt eksempel hvor variablen `SLAVEZD` er forskellig fra værdien i den overordnede opsætningsfil.

```
export slavezd=var/named/slave
```

Mens der i den overordnede opsætningsfil bl.a. står:

```
  options { directory "/var/named";
  };
  zone "intranet" in {
  type slave;
  file "slave/intranet";
  masters { 192.168.0.1; };
};
```

Variablerne `$OPTDIR` , `MASTERZD` samt `SLAVEZD` skal allesammen fortolkes i forhold til filsystemets rod `/` eller `$JAIL`.

6.12.2. DNS-serverens brugernavn

Nogle distributioner opretter automatisk en bruger der svarer til DNS-serveren. Et hurtigt blik på `/etc/passwd` kan afsløre en bruger der hedder f.eks. named, bind eller dns og så kan man lave de nødvendige rettelser i kommandoerne. Her antages at brugeren hedder **named** samt at gruppen hedder ligeledes **named**.

Hvis denne bruger ikke findes i forvejen, kan den i Debian oprettes med den følgende kommando:

```
[root@linus /root]# adduser --system --group --home /opt/chroot/named named
Adding system user named...
Adding new group named (116).
Adding new user named (116) with group named.
Creating home directory /opt/chroot/named.
```

6.12.3. Tilføjelse til filsystemtabellen

DNS-serveren bruger 3 /dev filer. Disse er `log`, `null`, og `random`. For hver af disse /dev filer skal der oprettes en forbindelse til de tilsvarende filer i den isolerede del af filsystemet. Når det er gjort kan serveren skrive til syslog.

Tidligere er der lavet forskellige krumspring for at DNS-serveren kan skrive til syslog. Med version 2.4 af kernen kom muligheden for at løse dette let og elegant med kommandoen **mount**.

I det følgende er der givet 2 forskellige metoder for at opnå forbindelse til filerne i den isolerede del af filsystemet. I den første metode bruges en **mount** kommando for hver af disse `/dev` filer. Den første metode giver mulighed for at afprøve opsætningen men den vil være tabt ved genstart af computeren.

```
mount --bind /dev/log $JAIL/dev/log
mount --bind /dev/random $JAIL/dev/random
```

```
mount --bind /dev/null $JAIL/dev/null
```

Erfarne brugere vil vide at mount kommandoen også kan bruges i forbindelse med filsystemtabellen `/etc/fstab` og derved kan man opnå en varig opsætning af disse filer. Der tilføjes nogle linjer til filsystemtabellen (derfor er det tilrådeligt at man lige tager en kopi af denne fil idet den er meget vigtig for systemet). Dette gøres lettest med de følgende kommandoer:

```
export jail=/opt/chroot/named &&
cd /etc &&
cp --backup=t fstab fstab.orig &&
echo -e '/dev/log\t'$JAIL'/dev/log\tnone \t rw,bind \t 0 0' >>fstab &&
echo -e '/dev/null\t'$JAIL'/dev/null\tnone \t rw,bind \t 0 0' >>fstab &&
echo -e '/dev/random\t'$JAIL'/dev/random\tnone \t rw,bind \t 0 0' >>fstab
```

Ovenstående kommandoer er nok til at sætte computeren op så ændringerne er varige også ved genstart.

Hvis mappestrukturen er blevet opsat (Se Afsnit 6.12.4) , kan man med følgende kommando afprøve om dette fungerer som det skal:

```
[root@linus /root]#
mount --all
```

Derpå vil kommandoen **mount** afsløre de interne forbindelser i kernen.

```
[root@linus /root]#  mount

/dev/log on /opt/chroot/named/dev/log type none (rw,bind)
/dev/null on /opt/chroot/named/dev/null type none (rw,bind)
/dev/random on /opt/chroot/named/dev/random type none (rw,bind)
```

6.12.4. Oprettelse af mappestruktur

Der oprettes en mappestruktur, til DNS-serveren hvori den skal køre isoleret. Set fra DNS-serveren vil denne mappestruktur udgøre hele serverens filsystem, og den vil således ikke kunne se den resterende del af filsystemet. Dette vil således også gælde for en eventuel angriber der har haft held med at tage kontrollen over DNS-processen.

I nedenstående eksempel oprettes mappestrukturen med roden `/opt/chroot/named` hvilket er i overensstemmelse med standarden for filhierarki i Linux (for flere oplysninger om FHS se:

http://www.pathname.com/fhs/ (http://www.pathname.com/fhs/)). Hvis man ønsker en anden rod (feks. `/chroot/named` kan man blot ændre variablen `$JAIL` i starten af kommandoerne. Ligeledes kan man rette brugernavnet i kommandoerne så det svarer til det brugernavn DNS-processen skal køre som. De følgende kommandoer kan klippes og klistres med et par museklik ind i et terminalvindue hvor root allerede har logget ind:

```
export jail=/opt/chroot/named &&
export piddir=var/run &&
export lokaltid=etc/localtime &&
export unamed=named &&
export gnamed=named &&
mkdir -p $JAIL &&
cd $JAIL &&
mkdir -p dev $OPTDIR $PIDDIR $MASTERZD $SLAVEZD &&
cd /
cp $LOKALTID $JAIL/$LOKALTID  &&
cd $JAIL/dev &&
touch log null random &&
mount --all &&
cd / &&
cp $KONFIL $JAIL/$KONFIL &&
cd $MASTERZD &&
find . | cpio -pm  $JAIL/$MASTERZD &&
cd $JAIL &&
chown $UNAMED.$GNAMED .  $PIDDIR $SLAVEZD &&
chmod 700 . .. &&
chattr +i  dev etc var $LOKALTID
```

Kommandoen i femte nederste linje, kopierer hele DNS-serverens opsætning til mappen $JAIL/$MASTERZD . Den tredje øverste linje sætter DNS-serveren som ejer af bl.a. mappen `$JAIL` , hvilket jo også er meget passende idé det jo er dens hjemmekatalog (det burde det i hvertfald være hvis alting gik godt i Afsnit 6.12.2). Den næstnederste linje siger adgang forbudt for uvedkommende. Den allersidste er kommandoen **chattr** der på et Linux filsystem (ext2, ext3 og evt. andre) kan sætte "immutable"-bitten på bestemte filer. Filer med denne bit sat, kan ikke ændres, omdøbes eller slettes. Dette er nødvendigt idet `named` brugeren ejer mappen `$JAIL` og har dermed ret til at omdøbe og slette enhver fil i denne mappe, uanset hvem ejer dem. Denne bit vil altså forhindre `named` brugeren i at omdøbe feks. `etc` mappen til noget andet og oprette en ny `etc` mappe med falske eller ændrede domæner. Dette kan named brugeren gøre uden videre hvis denne bit ikke er sat. Hvis en angriber opdager at denne bit ikke er sat, kan han/hun juble ligesom Benny i "The Julekalender" da han havde eksperimenteret med nissernes magiske bog og udbrød i ekstase: "Sikke muligheder man! Sikke muligheder!" Hvis man er i gang med at eksperimentere med denne opsætning, gør man dog klogest i at undgå denne bit idet den forbyder selv root-brugeren at slette eller ændre hele `$JAIL` mappen. Denne bit ("immutable"-bitten) skal resettes med kommandoen **chattr -i** før root kan ændre eller slette filer hvor den er sat. Så er folk blevet advaret.

Hvis der har været fejl i opsætningen eller kommandoerne kan man fjerne mappestrukturen med nogle eller alle af de følgende kommandoer og derpå kan man prøve endnu en gang at installere mappestrukturen. Som den Linuxkyndige læser vil se, er der sat to små sikkerhedsventiler på kommandoen **rm -rf** for det tilfælde at variablen `&JAIL` ikke findes.

```
umount $JAIL/dev/* &&
cd $JAIL &&
chattr -i  dev etc var $LOKALTID &&
cd /tmp &&
rm -rf $JAIL findesikke
```

6.12.5. Parametre til bind-processen.

Hvis det er lykkedes at oprette mappestrukturen samt montere `/dev` filerne i foregående afsnit, er filsystemet klart til at isolere DNS-serveren og dermed også isolere en eventuel angriber.

Det eneste der mangler nu er at fortælle DNS-serveren at den skal isoleres i denne del af filsystemet og køre som en bestemt bruger. Tidligere versioner af DNS-serveren måtte genoversættes til formålet, men BIND version 9 kan bruges uden videre ved at tilføje nogle parametre til kommandolinjen. Disse parametre er:

- `-u named` processen skal køre som brugeren named i stedet for root.
- `-t /opt/chroot/named` processen skal isoleres i "chroot-jail" der er blevet klargjort.

Der tilføjes nogle linjer til opstartsfilerne:

```
opts="-u named -t /opt/chroot/named" &&
/usr/sbin/named $OPTS
```

For at gøre opsætningen varig, kan ovenstående sættes ind i opstartsfilen.

I Red Hat ser opstartsfilen således ud med ovenstående ændringer:

```
#!/bin/sh
#
# named           This shell script takes care of starting and stopping
#                 named (BIND DNS server).
#
# chkconfig: - 55 45
# description: named (BIND) is a Domain Name Server (DNS) \
# that is used to resolve host names to IP addresses.
# probe: true

# Source function library.
```

```
   . /etc/rc.d/init.d/functions

   # Source networking configuration.
   . /etc/sysconfig/network

   # Check that networking is up.
   [ ${NETWORKING} = "no" ] && exit 0

   [ -f /usr/sbin/named ] || exit 0

   [ -f /etc/named.conf ] || exit 0

   retval=0

   # See how we were called.
   case "$1" in
   start)
   # Start daemons.
   echo -n "Starting named: "
# *** De foelgende linjer er udkommenteret for chroot
#   daemon named
# *** De foelgende linjer er indsat for chroot
opts="-u named -t /opt/chroot/named"
daemon named $OPTS
# *** De ovenstaaende linjer er indsat for chroot
   retval=$?
   [ $RETVAL -eq 0 ] && touch /var/lock/subsys/named
   echo
   ;;
   stop)
   # Stop daemons.
   echo -n "Shutting down named: "
   killproc named
   retval=$?
   [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/named
   echo
   ;;
   status)
   /usr/sbin/ndc status
   exit $?
   ;;
   restart)
   $0 stop
   $0 start
   ;;
   reload)
   /usr/sbin/ndc reload
   exit $?
   ;;
   probe)
   # named knows how to reload intelligently; we don't want linuxconf
   # to offer to restart every time
   /usr/sbin/ndc reload >/dev/null 2>&1 || echo start
```

```
   exit 0
   ;;

   *)
   echo "Usage: named {start|stop|status|restart}"
   exit 1
   esac

   exit $RETVAL
```

I Debian ser opstartsfilen således ud for bind version 9:

```
#!/bin/sh

path=/sbin:/bin:/usr/sbin:/usr/bin
# for a chrooted server: "-u nobody -t /var/lib/named"
opts="-u named -t /chroot/named"

test -x /usr/sbin/named || exit 0

case "$1" in
    start)
        echo -n "Starting domain name service: named"
        start-stop-daemon --start --quiet --exec \
        /usr/sbin/named -- $OPTS
        echo "."
    ;;

    stop)
        echo -n "Stopping domain name service: named"
        start-stop-daemon --stop --quiet  \
            --pidfile /var/run/named.pid --exec /usr/sbin/named
        echo "."
    ;;

    reload)
        /usr/sbin/rndc reload
    ;;

    restart|force-reload)
        $0 stop
        sleep 2
        $0 start
    ;;

    *)
        echo "Usage: /etc/init.d/bind {start|stop|reload|restart|force-reload}" >&2
        exit 1
    ;;
esac
exit 0
```

Med ovenstående opstartsfil kan serveren køre efter genstart, uden indgreb fra administrator.

6.12.6. Afprøvning af DNS-serveren

For at starte DNS-processen uden at genstart maskinen, giver man følgende kommando i henholdsvis Red Hat og Debian (den er desuden pyntet med lidt && krusiduller og Co. for at man kan se hvordan opstarten går):

```
[root@linus /root]# /etc/rc.d/init.d/named start && tail -f /var/log/messages
```

```
[root@linus /root]#  /etc/init.d/bind9 start && tail -f /var/log/daemon.log
```

```
Jun  8 14:29:28 nse named[308]: starting BIND 9.2.0 -u named -t /opt/chroot/named \
-c /etc/bind/named.conf
Jun  8 14:29:28 ns named[308]: using 1 CPU
Jun  8 14:29:29 ns named[311]: loading configuration from '/etc/bind/named.conf'
Jun  8 14:29:29 ns named[311]: listening on IPv4 interface eth0, 192.168.8.1#53
Jun  8 14:29:29 ns named[311]: command channel listening on 127.0.0.1#953
Jun  8 14:29:29 ns named[311]: zone 0.in-addr.arpa/IN: loaded serial 1
Jun  8 14:29:29 ns named[311]: zone 127.in-addr.arpa/IN: loaded serial 1
Jun  8 14:29:29 ns named[311]: zone 168.192.in-addr.arpa/IN: loaded serial 2000111605
Jun  8 14:29:29 ns named[311]: zone 255.in-addr.arpa/IN: loaded serial 1
Jun  8 14:29:29 ns named[311]: zone localhost/IN: loaded serial 1
```

Hvis alting er som det skal være, udskrives noget der ligner det ovenstående.

Opsætningen kan desuden afprøves ved at se om bind virkelig kører samt kigge i logfiler. Hvis bind kører, vil kommandoen **ps aux | grep named** vise en eller flere linjer som den følgende:

```
named 2043 0.0  2.8 10396 1432 ?  S  03:11 0:00 /usr/sbin/named \
  -u named -t /opt/chroot/named -c /etc/bind/named.conf
```

Ovenstående linje bekræfter at BIND kører som den upriviligerede bruger `named` og at den er isoleret i `/opt/chroot/named` som hensigten var.

Den afgørende bekræftelse af opsætningen får man hvis serveren svarer når den forepørges om forskellige domæner. Hvid den er authoritativ/primær DNS-server for et domæne bør man selvfølgelig forespørge DNS-serveren om dette domæne. Man kan konkludere at "chroot" opsætningen er korrekt hvis serveren svarer.

Hvis der er noget i vejen, kan man få yderligere oplysninger kan findes i logfilen med kommandoen **tail /var/log/daemon.log** eller **tail /var/log/messages**.

Hvis serveren ikke kører, bør man kigge nærmere efter om alle filer og mapper er placeret som de skal og om der er manglende sammenhæng mellem de filnavne der er angivet i opsætningsfilerne og de filnavne og mappenavne der blev oprettet.

Alle øvrige problmemer med DNS-serveren har at gøre med den generelle opsætning i opsætningsfilerne. Yderligere oplysninger om opsætning og test af DNS, findes i Afsnit 6.8.

6.12.7. DNS-serveren som upriviligeret bruger

Dette afsnit handler hovedsagelig om hvordan DNS-serveren BIND 9 opfører sig når den kører som upriviligeret bruger. Selve opsætningen omhandles i foregående afsnit.

Traditionelt set, tilhører alle porte lavere end 1024 til root eller daemoner der kører som root. Når BIND køres som en upriviligeret bruger, har den ikke rettigheder til at sende førespørgsler til andre servere fra port 53. Som udgangspunkt vil den vælge et tilfældigt upriviligeret port at sende fra. Hvis netværket beskyttes af en dørvogter der udtrykkeligt specificerer hvilket port DNS-serveren kan sende fra, vil forespørgslerne ikke kunne komme forbi dørvogteren. For at afhjælpe dette kan man vælge en bestemt upriviligeret port, f.eks. 1053 og sætte BIND op til kun at bruge denne port som afsenderport. I den overordnede opsætningsfil til BIND bør den nederste af de følgende linjer tilføjes i sektionen options {}:

```
listen-on port 53 { 192.168.8.1; };
query-source address * port 1053;
```

I eksemplet ovenfor, har serveren IP-nummeret 192.168.8.1 på trods af at det er en offentlig DNS server. Forklaringen er, at serveren ikke har fået tildelt et offentligt IP-nummer, men trafik til den bliver dirigeret til den gennem en dørvogter. Dørvogteren skal blot ændres så den tillader udgående trafik med dette source port. Dermed kan BIND fortsætte med at fungere bag selv den mest restriktive dørvogter.

6.12.8. Testkørsel af DNS serveren.

Hvis serveren kører, bør den kunne svare på forespørgsler. Der findes flere værktøj til dette men her skal nævnes det nu snart forældede "**nslookup**" samt "**dig**". Man kan bede "**dig**" om at forespørge en vilkårlig DNS server ved blot at angive dens IP-nummer med parameteren `@`. I det følgende eksempel, skal den nyopsatte DNS-server give os oplysninger om domænet "sslug.dk".

```
[tyge@hven tyge]$ dig @192.168.8.1 sslug.dk
; <<>> DiG 9.2.0  <<>> @192.168.8.1 sslug.dk
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 45711
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
```

```
;; QUESTION SECTION:
;sslug.dk.                      IN      A

;; ANSWER SECTION:
sslug.dk.               86400   IN      A       130.228.2.150

;; AUTHORITY SECTION:
sslug.dk.               86400   IN      NS      ns.sslug.dk.
sslug.dk.               86400   IN      NS      ptah.dkuug.dk.
sslug.dk.               86400   IN      NS      ns-soa.darenet.dk.

;; ADDITIONAL SECTION:
ns.sslug.dk.            86400   IN      A       130.228.2.150
ptah.dkuug.dk.          71539   IN      A       195.215.30.66
ns-soa.darenet.dk.      71539   IN      A       130.226.1.4

;; Query time: 237 msec
;; SERVER: 192.168.8.1#53(192.168.8.1)
;; WHEN: Sat Apr 13 18:11:17 2002
;; MSG SIZE  rcvd: 161
```

Sådan kan man afprøve serveren ved at pege den på forskellige domæner. Hvid den er authoritativ/primær DNS-server for et domæne bør man selvfølgelig forespørge DNS-serveren om dette domæne. Man kan konkludere at "chroot" opsætningen er korrekt hvis serveren svarer.

En af de ting der gør **dig** interessant sammenlignet med **nslookup** er at man let kan få fat i alle de forskellige oplysninger der ligger i DNS. Til gengæld giver **dig** også en masse oplysninger man normalt ikke er interesseret i, så det kan være nødvendigt at filtrere programmets uddata lidt.

Hvis man vil vide hvor post til en bestemt maskine afleveres giver man, udover maskinens navn, **dig** argumentet `MX`:

```
[tyge@hven tyge]$ dig kaoslx07.nbi.dk MX

;; <<>> DiG 9.2.1 <<>> kaoslx07.nbi.dk MX
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28989
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;kaoslx07.nbi.dk.               IN      MX

;; ANSWER SECTION:
kaoslx07.nbi.dk.        86400   IN      MX      10 alf.nbi.dk.

;; AUTHORITY SECTION:
nbi.dk.                 86400   IN      NS      alf.nbi.dk.
```

```
nbi.dk.                 86400   IN      NS      garm.adm.ku.dk.
nbi.dk.                 86400   IN      NS      sarastro.nbi.dk.

;; ADDITIONAL SECTION:
alf.nbi.dk.             86400   IN      A       130.225.212.55
garm.adm.ku.dk.         8272    IN      A       130.225.127.34
sarastro.nbi.dk.        86400   IN      A       130.225.212.46

;; Query time: 24 msec
;; SERVER: 130.225.212.46#53(130.225.212.46)
;; WHEN: Thu Sep  5 11:36:33 2002
;; MSG SIZE  rcvd: 164
```

Der kommer altså en hel masse data, hvoraf det eneste vi egentlig er interesserede i er "ANSWER SECTION". Med tilvalget `+short` kan vi få **dig** til kun at give os det egentlige svar:

```
[tyge@hven tyge]$ dig kaoslx07.nbi.dk MX +short
10 alf.nbi.dk.
```

Man kan også bruge **dig** til at få oplysninger som maskinens IP-adresse:

```
[tyge@hven tyge]$ dig kaoslx07.nbi.dk A +short
130.225.212.98
```

Hvilke maskiner der står for et domænes navnetjeneste:

```
[tyge@hven tyge]$ dig linuxlab.dk NS +short
www.itu.dk.
ns-soa.darenet.dk.
```

Eventuelt også lidt oplysninger om selve isenkrammet:

```
[tyge@hven tyge]$ dig kaoslx07.nbi.dk HINFO +short
"Pentium 4/1.5GHz" "Linux 7.3/2.4.18"
```

Endelig kan man også få lidt løse uformaterede oplysninger om domænet:

```
[tyge@hven tyge]$ dig nbi.dk TXT +short
"Niels Bohr Institute, Copenhagen University"
```

Kapitel 7. Linux som server

Vi er tidligere kommet ind på en mere grundlæggende forklaring af netværk, TCP/IP og netværksadresser, idet dette er helt centralt for alt med netværk.

I dette kapitel ser vi nærmere på en række server-funktionaliteter DHCP, NFS (Unix-fildeling), NIS (adgangskode-deling) og AppleTalk (Mac-server). Desuden er der afsnit om hvordan man får en X-server og klient til at virke. Endelig behandles også hvordan du kan få adgang til din Linux-maskine via modem (eng. dial-in modem).

7.1. Opsætning af DHCP-server

Hvis man har mere end et par computere på sit netværk, eller hvis man ofte flytter computerne mellem forskellige netværk, så bliver det hurtigt besværligt at skulle ændre deres IP-adresser, standard-gatewayadresse og navneserver hver gang man flytter dem rundt.

Løsningen på det problem er at bruge DHCP – "Dynamic Host Configuration Protocol". Som navnet antyder, så er det en speciel protokol som er beregnet til at en computer kan blive sat op automatisk, uden at man skal ind og pille i opsætningsfiler på hver enkelt maskine.

Bemærk, at hvis du har mere end én DHCP-server på samme netværks segment bør disse være koordineret. Hvis de ikke er koordineret vil du kunne opleve at det kun virker en gang imellem – nemlig når klienten får svaret fra den server, der er sat korrekt op.

7.1.1. Hvordan virker DHCP?

Når man tænder en computer der bruger DHCP, så vil den sende en forespørgsel ud på sit lokale netværk for at spørge efter en DHCP-server. Hvis der er en DHCP-server på nettet, så tager serveren en ubrugt IP-adresse fra den pulje af adresser, den er sat op til at bruge, og sender et svar tilbage om at den nye computer kan bruge den pågældende IP-adresse. Sammen med IP-adressen sendes også andre oplysninger som er nødvendige for at netværket kan fungere – som regel er det netmasken, IP-adressen for default gateway og navneservernes IP-adresser som fås fra DHCP-serveren.

Den IP-adresse som den nye computer får tildelt, må den bruge i et begrænset tidsrum – den får det der kaldes et "lease" på adressen, som skal fornyes med jævne mellemrum. Hvis ikke computeren beder om at forny sit lease, så inddrager DHCP-serveren det igen, og kan dele det ud til andre maskiner der har brug for en IP-adresse.

Når man bruger DHCP, kan man derfor ikke være sikker på, at ens computere altid har en fast IP-adresse – det afhænger af, hvor længe ad gangen de er tændt. Så længe de er på nettet, bliver de ved med at have

den IP-adresse de har fået fra serveren. Men hvis de bliver lukket ned eller taget af nettet så længe at deres lease udløber, så vil de sandsynligvis få en ny IP-adresse næste gang de kontakter serveren.

Derfor er DHCP mest anvendelig på de systemer, som ikke kører nogen form for server-funktion – dvs. almindelige arbejdsstationer.

Det er dog muligt at sætte sin DHCP-server op på en måde, så visse IP-adresser bliver reserveret til bestemte maskiner, og derved kan man sikre at den pågældende maskine altid har en bestemt IP-adresse. Derfor kan man godt bruge DHCP til servere, f.eks. hvis man vil styre netværks-opsætningen centralt.

7.1.2. Valg af DHCP-server/servere

Der følger naturligvis en DHCP-server med din Linux-distribution – det er en server som er udviklet af ISC (Internet Software Consortium). De har også udviklet navneserveren BIND. I næste afsnit kan du se, hvordan man sætter den op.

Det anbefales derfor at du ikke har mere end én dhcp-server på et netværkssegment, og at du koordinerer hele opsætningen af DHCP i denne server. Du *kan* dog have mere end en DHCP server på samme netværkssegment. Dette forudsætter at de alle giver korrekte oplysninger til klienterne, og at deres puljer af adresser ikke overlapper hinanden.

Hvis du har en eller anden form for router på dit netværk, så kan det være at du allerede har en DHCP-server på nettet. Små ISDN- og ADSL-routere har ofte en DHCP-server indbygget, og den kan man sagtens bruge. Man kan endda opleve at printere og andet udstyr med netværkstilslutning har en DHCP-server indbygget.

Hvis ikke alle DHCP-serverne er sat korrekt og koordineret op kan du opleve at det kun virker en gang imellem. Du skal være opmærksom på at du kan komme ud for udstyr som kun *tænder* for DHCP-serveren når det ikke kan finde en *anden* DHCP-server. I praksis betyder det at du kan opleve at det kun virker, når du tænder / tilslutter netværksenhederne i en bestemt rækkefølge.

7.1.3. ISC DHCP-server

ISC's DHCP-server bruger én opsætningsfil. Normalt ligger den i `/etc/dhcpd.conf`. Derudover bruger den en midlertidig fil til at holde styr på hvilke IP-adresser den har uddelt til de forskellige maskiner på nettet; den ligger normalt i `/etc/dhcpd.leases`, og den administrerer selv serveren. Du skal blot sørge for at filen eksisterer inden du starter serveren – f.eks. ved at give kommandoen **touch /etc/dhcpd.leases** (for Red Hat 7.0 **touch /var/lib/dhcp/dhcpd.leases**) inden du starter serveren første gang. Installér dernæst følgende RPM-pakke for at få DHCP-serveren ind på Red Hat:

```
[root@linus /root]# rpm -ivh dhcp-*.rpm
```

For at sætte DHCP-serveren op skal du bestemme dig for, hvilke oplysninger den skal uddele til de computere, der spørger den om en IP-adresse. Dvs. du skal vælge:

- Hvor længe skal de have lov til at bruge den tildelte adresse?
- Hvilket IP-adresseområde skal der tages adresser fra?
- Hvilken netmaske har dit netværk?
- Hvad er default gateway for nettet?
- Hvilken IP-adresse bruges til navneserver?

Derudover er der en frygtelig masse oplysninger, som man kan levere via DHCP. Det er f.eks. domæne-navnet for maskinerne på dit netværk, navn på post-server, time-server, WINS-server etc. Det er dog langtfra alle disse oplysninger, som rent faktisk bruges til noget af de maskiner som kontakter DHCP-serveren, så derfor er det ikke altid at man får noget ud af at konfigurere dem på DHCP-serveren. Du kan se hvilke muligheder der er i "dhcp-options" man-siden.

7.1.3.1. dhcpd.conf filen

I netværket der bruges som eksempel, har man valgt at bruge følgende værdier til DHCP:

- Et lease skal være fornyet efter 10 timer, og er max. gyldigt i 24 timer.
- IP-adresser tildeles fra puljen 192.168.0.100 - 192.168.0.250
- Netmasken er 255.255.255.0
- Default gateway er 192.168.0.1
- Navneserveren er 192.168.0.1
- Domæne-navnet er "intranet"

Opsætningsfilen `/etc/dhcpd.conf` skal så se således ud:

```
ddns-update-style ad-hoc;

subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.100 192.168.0.250;
        option subnet-mask 255.255.255.0;

        # default gateway
        option routers 192.168.0.1;

        option domain-name "intranet";
        option domain-name-servers 192.168.0.1;

        default-lease-time 36000;
        max-lease-time 86400;
}
```

"range" fortæller så, hvilke IP-adresser serveren kan tildele klienter på dette netværk – det er her, vi skriver at adresserne skal være i intervallet 192.168.0.100 til 192.168.0.250.

"option subnet-mask" definerer netmasken.

"option routers" definerer standard-gateway.

"option domain-name" og "option domain-name-servers" definerer DNS-domæne-navnet og navneserveren. DHCP-serveren laver så et opslag inden den tildeler IP-adressen til klienten. En smart ting hvis navneserveren skulle skifte IP-adresse. Hvis du ikke selv har en navneserver på dit netværk, så kan du sagtens skrive IP-adressen på din internetudbyders navneserver her.

```
option domain-name-servers 192.168.0.1, 192.168.0.2;
```

"default-lease-time" er det antal sekunder, et lease er gyldigt. Herefter skal klienten forny sit lease hos DHCP-serveren.

"max-lease-time" er det antal sekunder en IP-adresse må bruges. Når den tid er gået *skal* klienten holde op med at bruge IP-adressen.

Forskellen på "default-lease-time" og "max-lease-time". Funktionen af de to tider er defineret i RFC 2131, afsnit 4.4.5 "Reacquisition and expiration". Kort fortalt går det ud på, at når default-lease-time ("T1" i RFC'en) udløber, skal klienten begynde at forsøge at forny sit lease på IP-adressen ved at kontakte den DHCP-server, den oprindelig fik adressen fra. Det kan klienten så blive ved med at forsøge indtil den får svar, eller indtil "max-lease-time" ("T2" i RFC'en) udløber. Hvis T2 udløber *skal* klienten stoppe med at benytte IP-adressen, og i stedet begynde at broadcaste efter en ny DHCP-server.

Tiden mellem default-lease-time og max-lease-time er altså den frist, som en klient har til at forny sit lease på en IP-adresse. Da servere kan være nede, er det nødvendigt at der er lidt tid til at klare sådan en ekspeditionssag – helst så længe, at klienterne ikke begynder at falde af nettet pga. et enkelt nedbrud af DHCP-serveren.

Hvordan vælger man en fornuftig lease-time? Det der giver mindst belastning af DHCP-serveren er, hvis arbejdsstationerne ikke behøver at forny deres lease hele tiden – en gang om dagen er rimeligt, og derfor er default-lease-time sat til lidt længere end en normal arbejdsdag (10 timer) og max-tiden til 24 timer (86400 sekunder). Omvendt vil man måske godt have, at klienterne hurtigt opdager hvis man skifter DNS-server – for at det skal ske hurtigt, skal default-lease-time sættes ned, så klienterne får friske oplysninger fra serveren noget oftere. Det er en afvejning, man må gøre.

DHCP-serveren kan håndtere flere "subnet" – f.eks. hvis serveren også fungerer som router, så der er flere netkort installeret i maskinen. Så kan man sætte én DHCP-server op, som deler adresser ud til de forskellige netværk. Almindeligvis har man dog kun ét netværk, der får adresser fra sin DHCP-server – men derfor skal der være en "subnet"-erklæring, der fortæller hvad det er for et fysisk netværk denne DHCP-pulje handler om. Har man f.eks. to netkort hvor man kun ønsker at DHCP-serveren skal tildele IP-adresser i det ene net – skal man alligevel definere DHCP-serveren for begge net, men vise at DHCP-serveren ikke skal tildele IP-adresser i det andet net, f.eks. for et klasse C-netværk 192.168.36.X. I det tilfælde kan følgende tilføjes til `/etc/dhcpd.conf`.

```
subnet 192.168.36.0 netmask 255.255.255.0 {
          deny booting;
```

```
}
```

Hvis man gerne vil styre hvilke netkort DHCP-serveren lytter på, kan man tilføje det/dem som afsluttende argument(er) i kaldet af **dhcpd** (sikkert fra et skript i `/etc/init.d/`).

7.1.3.2. Start DHCP-serveren

Efter at have redigeret `/etc/dhcpd.conf` færdig er du klar til at starte DHCP-serveren. Kører du Red Hat, er det nemt. Du skal installere dhcp-pakken og køre

```
[root@linus /root]# /sbin/chkconfig --level 2345 dhcpd on
```

Så starter DHCP-serveren automatisk når maskinen starter op. Det virker dog først efter en genstart, så hvis du vil have serveren i gang med det samme, så kører du

```
[root@linus /root]# /etc/rc.d/init.d/dhcpd start
```

Du kan nu prøve at konfigurere enten en Linux-maskine eller en Windows-maskine til at bruge DHCP, og så se om den får oplysningerne fra serveren. Hvis det er en Windows-maskine du bruger til at teste med, så kan du bruge **winipcfg**-programmet (Win95 og Win98) eller **ipconfig /all** (Windows NT) til at se, hvordan maskinen er blevet konfigureret af en DHCP-server.

Når serveren kører, logger den forespørgsler til en log-fil, almindeligvis `/var/log/messages`. Det kan se således ud:

```
dhcpd: DHCPDISCOVER from 00:00:b4:c7:1e:e6 via eth0
dhcpd: DHCPOFFER on 192.168.0.101 to 00:00:b4:c7:1e:e6 via eth0
dhcpd: DHCPREQUEST for 192.168.0.101 from 00:00:b4:c7:1e:e6 via eth0
dhcpd: DHCPACK on 192.168.0.101 to 00:00:b4:c7:1e:e6 via eth0
```

Først er det klienten, som leder efter en DHCP-server. Så tilbyder DHCP-serveren at klienten kan bruge adressen "192.168.0.101". Den beder klienten så om, og serveren svarer tilbage at det er OK.

De underlige adresser der står i hver linje, er Ethernet-adressen (MAC adressen) på det netkort, som spørger DHCP-serveren. Dem kan man bruge til at give servere en fast IP-adresse, selv om de får oplysningerne via DHCP.

7.1.3.3. Konfigurér en server med DHCP

Hvis man ønsker at en bestemt maskine altid får den samme IP-adresse fra DHCP-serveren, så kan man give den en statisk tildeling i DHCP-server-opsætningen. Her bruger man den maskinens ethernet-kortets

adresse, hvis maskinen tidligere har fået en IP-adresse fra DHCP-serveren, så kan man se ethernet-kortets -adresse i server-maskinens log-fil. På en Linux-maskinen kan du se dens egen ethernet-adressen med kommandoen **/sbin/ifconfig:**

```
[root@eric]$ /sbin/ifconfig eth0
          Link encap:Ethernet  HWaddr 00:00:B4:C7:1E:E6
          inet addr:192.168.0.104  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:55 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0x6100
```

"HWaddr" er Ethernet-kortets adresse. Hvis nu vi vil have at denne maskine altid får IP-adressen "192.168.0.4" selv om den får sin IP-adresse fra DHCP-serveren, så tilføjer vi nogle linjer til `/etc/dhcpd.conf`, inden i den "subnet"-blok som vi lavede før – så kommer den til at se således ud:

```
ddns-update-style ad-hoc;

subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.100 192.168.0.250;
        option subnet-mask 255.255.255.0;

        # default gateway
        option routers 192.168.0.1;

        option domain-name "intranet";
        option domain-name-servers 192.168.0.1;

        default-lease-time 36000;
        max-lease-time 86400;

        host eric {
                hardware ethernet 00:00:B4:C7:1E:E6;
                fixed-address 192.168.0.4;
        }
}
```

"host eric" indleder en særlig specifikation af hvordan en enkelt vært skal sættes op. For at DHCP-serveren kan kende værten bruger vi Ethernet-adressen – den tager vi fra **ifconfig**-beskeden. For at tildele denne host en fast adresse bruges "fixed-address 192.168.0.4", og den adresse vil den pågældende vært så altid få.

Bemærk, at man godt kan tildele faste IP-adresser, der ligger uden for det område af IP-adresser, der står i "range"-specifikationen.

Fordelen ved også at sætte servere op med IP-adresser fra en DHCP-server er, at hvis nogle af de andre oplysninger ændrer sig - f.eks. navneserveren eller standard-gateway – så skal man kun ændre det ét sted, nemlig på DHCP-serveren. Alle andre servere og arbejdsstationer får oplysningerne fra den.

Der er dog en enkelt undtagelse – DHCP-serveren kan ikke spørge sig selv om, hvilken IP-adresse den skal have. Den er man nødt til at konfigurere manuelt med en statisk IP-adresse, standard-gateway osv.

7.1.4. DHCP og DNS

Dette afsnit beskriver DHCP 3.0 fra ISC (http://www.isc.org). Tidligere versioner i 2.X-serien har ikke haft samme funktionalitet.

Installationen er ret nem, enten henter du en rpm (ftp://angus.ind.wpi.edu/pub/packages/isc/dhcp)- eller en tarpakke (ftp://ftp.isc.org/isc/dhcp/dhcp-latest.tar.gz) med kildeteksten. Til Debian findes pakken dhcp3-server.

Når du har installeret DHCP 3.0, forestår der kun minimale ændringer. Du skal ændre i to filer: named.conf og dhcpd.conf. Eksempler kan hentes her www.linuxbog.dk/admin/eksempler (http://www.linuxbog.dk/admin/eksempler).

I named.conf skal der tilføjes en option til de zoner hvor vores DHCP-server uddeler ip-numre til. Tilføjelsen giver named ret til at foretage opdateringer i vores zonefiler, og tilladelsen gives med følgende kommando: allow-update {localhost;}. Se eksempel nedenfor.

```
options {
 directory "/var/named";
 notify no;
 forwarders{
  XXX.XXX.XXX.XXX;
  XXX.XXX.XXX.XXX;
 };
};

logging {
 category lame-servers { null; };
};

zone "." in {
 type hint;
 file "named.ca";
};

zone "0.0.127.IN-ADDR.ARPA" in {
 type master;
 file "127.0.0";
};

zone "domænenavn" in {
 type master;
 file "filnavn";
 allow-update {localhost;};
```

```
};

zone "XXX.YYY.ZZZ.IN-ADDR.ARPA" in {
 type master;
 file "filnavn";
 allow-update {localhost;};
};
```

I dhcpd.conf skal der tilføjes en række options, der vedrører dynamisk dns. Disse options starter alle med ddns, og de væsentligste er følgende:

- ddns-domainname "domænenavn"
- ddns-rev-domainname "IN-ADDR.ARPA"
- ddns-update-style interim
- zone domænenavn. {
 primary DNS-serverens ip-adresse;
}
- zone reverse ip-adresse.IN-ADDR.ARPA. {
 primary DNS-serverens ip-adresse;
}

Et eksempel følger nedenfor.

```
 option domain-name "domænenavn";
 ddns-update-style interim;

subnet WWW.XXX.YYY.ZZZ. netmask 255.255.255.0 {
 range WWW.XXX.YYY.100 WWW.XXX.YYY.150;
 ddns-domainname "domænenavn";
 ddns-rev-domainname "IN-ADDR.ARPA";
# log-facility local7;

 # default gateway
 option routers WWW.XXX.ZZZ.YYY;
 option domain-name-servers WWW.XXX.ZZZ.YYY;
 option interface-mtu 1500;
 option broadcast-address WWW.XXX.ZZZ.255;

 option netbios-name-servers WWW.XXX.ZZZ.YYY;
 option netbios-dd-server WWW.XXX.ZZZ.YYY;
 option netbios-node-type 8;

 default-lease-time 36000;
 max-lease-time 86400;
}

host prtsrv {
  hardware ethernet 00:40:8C:30:6F:CE;
  fixed-address prtsrv.domænenavn;
}

zone domænenavn. {
```

```
  primary WWW.XXX.ZZZ.YYY;
}

zone ZZZ.XXX.WWWW.in-addr.arpa. {
  primary WWW.XXX.ZZZ.YYY;
}
```

Hvis du bruger Redhat 7.2, mangler der en sidste rettelse. Rettelsen skal foretages i filen `/etc/sysconfig/network`, hvor du skal tilføje følgende:

- DHCP_hostname="hostnavnet på klienten"

Det skal bemærkes, at ovenstående hostnavn er uden angivelse af domænet. Hvis vi f.eks. har domænet intranet, og ønsker at tildele klienten eric en adresse, skal hostnavnet angives som eric og ikke som eric.intranet.

Undlad at lave manuelle opdateringer i dynamiske zone filer. Hvis man vil lave manuelle opdateringer skal man først stoppe DNS serveren og dernæst slette .jnl filerne, så lave opdateringerne og starte serveren igen. Man kan desuden bruge **nsupdate** til at ændre i dynamiske zoner.

7.2. FTP-server

De fleste Linux-distributioner installerer automatisk en ftp-server, hvis man har valgt en server-installation. Har du valgt en skrabet installation til en gammel maskine, er det nemt senere at få ftp-serveren installeret efterfølgende.

`wu-ftpd` er en ftp-server der følger med næsten alle distributioner. Et bedre alternativ til wu-ftpd er `proftpd` http://www.proftpd.org/, som også følger med mange distributioner. Med wu-ftpd installeret er det muligt for alle brugere af systemet at få adgang til deres filer med ftp. Installationen er nem.

```
[root@linus RPMS]# rpm -ivh wu-ftpd*.i386.rpm
```

Ftp-serveren er nu startet og klar til brug. Af sikkerhedsmæssige årsager kan **root** ikke logge ind på ftp-serveren. Dette skyldes at **root** er nævnt i filen `/etc/ftpusers`. Du kan have dine grunde til at give root lov til at logge ind, men tænk dig om.

Som før nævnt er det kun brugere af systemet, der kan logge ind. Skal ftp-serveren være åben for alle, f.eks. til brug for at installere Linux på en anden maskine over nettet, så skal der lige en pakke mere ind.

```
[root@linus RPMS]# rpm -ivh anonftp*.i386.rpm
```

For at du kan starte din ftp-server op, skal du redigere filen `/etc/xinetd.d/wu-ftpd` (Red Hat 7). Ret `disable = yes` til `disable = no` og kør **/etc/init.d/xinetd restart**. Grunden til dette er sikkerhedsgrunde, så ftp-serveren vil *ikke* automatisk blive startet op.

Nu er maskinen åben for at alle kan hente filer fra din maskine, hvor brugerne logger ind som **anonymous** (eller **ftp**) og giver sin egen e-post-adresse som adgangskode. Skal du lægge filer, så brugerne kan hente dem ned, skal de lægges i `/var/ftp/pub/` eller et af dets underkataloger. Du kan evt. linke til andre dele af filsystemet, hvis du f.eks. vil give almindelige brugere lov til at lægge filer ud til nedhentning.

Det skal nævnes, at der har været en hel del sikkerhedsproblemer i wu-ftpd, derfor bør du tjeke hvad der er nyeste udgave før du installerer på et usikkert netværk, såsom internettet.

7.3. NFS – Deling af filer mellem Unix-maskiner

I Unix-verdenen og dermed også Linux har man i mange år kunnet dele filer mellem forskellige maskiner i et netværk. Den mest udbredte protokol til dette er NFS (netværksfilsystem – eng. Networked File System). Det er selvfølgelig et område, man skal være forsigtig med, hvis man har et alt for åbent netværk. Her skal vi kun give et simpelt eksempel på, hvordan NFS virker.

Du skal først installere RPM-pakken `nfs-utils` (eller tilsvarende Debian-pakke) som indeholder nye udgaver af nfs-programmer og dæmonerne.

nfs (dvs. dæmonerne `rpc.rquotad`, `rpc.mountd` og `rpc.nfsd`) og nfslock (dvs. dæmonerne `rpc.lockd` og `rpc.statd`) skal startes når maskinen genstarter:

```
[root@linus /root]# /sbin/chkconfig nfs on
[root@linus /root]# /sbin/chkconfig nfslock on
[root@linus /root]# /sbin/chkconfig --list | grep nfs
nfs 0:off 1:off 2:off 3:on 4:on 5:on 6:off
nfslock 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

Herefter er maskinen klar som fornuftig NFS-server.

Lad os antage at maskinen alfa skal dele `/home` med maskinen beta, så skal alfa-maskinen have installeret RPM-pakken `knfsd`. I `/etc/exports` skriver du først hvilken del af filtræet der skal deles, derefter hvilke klienter der skal bruge det (du kan anvende * for alle), og sidst om der deles med læse/skrive- (rw) eller kun læse-rettigheder (ro). Brug **man exports** for flere detaljer.

```
# /etc/exports - NFS export of /home til beta
# Tilsvarende gives kun læseadgang til /mnt/cdrom til alle maskiner
/home beta.domænenavn.dk(rw)
/mnt/cdrom *(ro)
```

Derefter skal du genstarte nfs-serveren ved at skrive **/etc/rc.d/init.d/nfs restart**, og som root på beta-maskinen kan du nu nemt få data fra alfa-maskinen.

```
[root@beta /root]# mount -t nfs alfa:/home /home
```

Tip: Vil du vide hvad der kan monteres fra en NFS-server, så kan du anvende **/usr/sbin/showmount -e NFSSERVER**, hvor "NFSSERVER" er navn eller IP-adresse på NFS-serveren.

7.4. NIS – deling af adgangskodefiler mellem Unix-maskiner

Er der flere maskiner i lokalnettet, og har brugerne brug for adgang til dem alle (eller nogle af dem), er NIS en god løsning.

NIS, The Network Information System, sørger for at flere maskiner kan dele oplysninger om brugere, grupper og endda også maskiner og en del andre oplysninger.

NIS blev tidligere kaldt "Yellow Pages", forkortet "yp", efter telefonbøgernes gule sider, men det måtte ændres, da Yellow Pages er et registreret varemærke ejet af British Telecom. Men de fleste programmer i NIS starter stadig med yp.

NIS arbejder, ligesom DNS, med et domænenavn. Men disse to domænenavne har intet med hinanden at gøre. I det følgende har vi to Red Hat-maskiner i (DNS-)domænet strangeparty.dk. De skal dele informationer om kodeord mv. i (NIS-)domænet spo.

Serveren hedder "supernaut". Klienten hedder "omnitec".

På serveren skal pakken "ypserv" installeres:

```
[root@supernaut /home/kwv]#  rpm -Uvh /cdrom/RPMS/ypserv-1.3.9-1.alpha.rpm
ypserv                  ##################################################
```

NIS-domænet skal være "spo". Det sættes med kommandoen **domainname**.

```
[root@supernaut /home/kwv]# domainname spo
```

Yp-serveren skal nu startes:

```
[root@supernaut /home/kwv]#  /etc/rc.d/init.d/ypserv start
Starting YP server services:                               [  OK  ]
```

Det kan betale sig at gennemse, og måske tilrette, filen `/var/yp/Makefile`. Du bør nok ændre MINUID og MINGID, så disse starter UID (bruger-ID) og GID (grupper-ID) ved 300, og ikke ved 500 som er standard i Red Hat. Du kan også udkommentere linjen "GSHADOW = $(YPPWDDIR)/gshadow" og indsætte "MERGE_group=false".

Herefter initieres NIS-serveren:

```
[root@supernaut /home/kwv]# /usr/lib/yp/ypinit -m

At this point, we have to construct a list of the hosts which will run NIS
servers.  supernaut.strangeparty.dk is in the list of NIS server
hosts.  Please continue to add
the names for the other hosts, one per line.  When you are done with the
list, type a <control D>.
 next host to add:  supernaut.strangeparty.dk
 next host to add:
The current list of NIS servers looks like this:

supernaut.strangeparty.dk

Is this correct?  [y/n: y]
We need some  minutes to build the databases...
Building /var/yp/spo/ypservers...
Running /var/yp/Makefile...
gmake[1]: Entering directory `/var/yp/spo'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating mail.aliases...
gmake[1]: Leaving directory `/var/yp/spo'
[root@supernaut /home/kwv]#
```

Med ypbind installeret på serveren, kan man nu kontrollere de services der stilles til rådighed. Ypbind skal startes først:

```
[root@supernaut /home/kwv]# /etc/rc.d/init.d/ypbind start

Binding to the NIS domain...                                [  OK  ]
Listening for an NIS domain server: supernaut.strangeparty.dk
```

Kommandoen **ypcat -x** viser de tjenester serveren stiller til rådighed:

```
[root@supernaut /home/kwv]# ypcat -x

Use "ethers" for map "ethers.byname"
Use "aliases" for map "mail.aliases"
Use "services" for map "services.byname"
Use "protocols" for map "protocols.bynumber"
Use "hosts" for map "hosts.byname"
Use "networks" for map "networks.byaddr"
Use "group" for map "group.byname"
Use "passwd" for map "passwd.byname"
```

Det interessante i vores lille netværk er `passwd` og `group`. Yderligere kan `aliases` og `mail.aliases` være smart til at styre lokal e-post.

Men hvad indeholder de så?

```
[root@supernaut /home/kwv]# ypcat passwd
hanne:BmXEt39P1By2I:301:300:Hanne:/supernaut/hanne:/bin/bash
kwv:/g10xcDPwwaA2:405:400:Kristian Vilmann:/home/kwv:/bin/bash
[root@supernaut /home/kwv]# ypcat group
hjemme:x:300:
teknik:x:400:
```

På klienten skal **ypbind** selvfølgelig være installeret. Og den skal startes:

```
[root@omnitech /root]# rpm -q ypbind
ypbind-3.3-24
[root@omnitech /root]# domainname spo
[root@omnitech /root]# /etc/rc.d/init.d/ypbind start
Binding to the NIS domain...                                [  OK  ]
Listening for an NIS domain server: .supernaut.strangeparty.dk
```

Hvis klienten har problemer med at finde NIS-serveren, eller der er flere NIS-domæner i netværket, kan man sætte sine præferencer i filen `/etc/yp.conf`.

Lad os se om det ser rigtigt ud. **ypwhich** fortæller hvilken NIS-server klienten er bundet til:

```
[root@omnitech /root]# ypwhich
supernaut.strangeparty.dk
[root@omnitech /root]# ypcat passwd
hanne:$1$d5dYr98N$/k.0aNyvq/BMX0UlMoyGZ1:301:300:Hanne:/home/hanne:/bin/bash
kwv:$1$I1/h6uyj$Spk11r0bYTyo9r.sc7nQ..:405:400:Kristian Vilmann:/home/kwv:/bin/bash
```

Det ser fint ud. Vi lader nu en bruger prøve telnet fra serveren til klienten. For en god ordens skyld kontrolleres at brugeren hanne ikke optræder i filen `/etc/passwd`:

```
[root@omnitech /root]# grep hanne /etc/passwd
```

Her var der intet svar (nul linjer).

```
[hanne@supernaut ~]# telnet omnitec
Trying 192.168.10.126...
Connected to omnitec.strangeparty.dk.
Escape character is '^]'.

Red Hat Linux release 6.1 (Cartman)
Kernel 2.2.12-20 on an i586
login: hanne
Password:
Last login: Wed Mar 29 13:42:12 from supernaut.strangeparty.dk
[hanne@omnitec ~]$ id
uid=301(hanne) gid=300(hjemme) groups=300(hjemme)
[hanne@omnitec ~]$ pwd
/supernaut/hanne
[hanne@omnitec ~]$ whoami
hanne
[hanne@omnitec ~]$ ls -al
total 142
drwx------   2 hanne    hjemme       1024 Mar 29 13:13 WebDB/
-rw-r--r--   1 hanne    hjemme      34862 Mar 12 23:24 headbanging.jpg
-rw-r--r--   1 hanne    hjemme      59995 Oct 18 22:16 insoundout.gif
drwxr-xr-x   4 hanne    hjemme       1024 Mar 12 00:10 objectsystem1.0/
-rw-r--r--   1 hanne    hjemme      43971 Mar 21 19:31 sslug.httpd.conf
```

Det ser rigtigt ud. Læg mærke til at Hannes hjemmekatalog er NFS-monteres fra serveren, og at hun derfor har samme hjemmekatalog på begge maskiner.

NIS tilbyder en række værktøjer til manipulering af brugerinformationerne på NIS-serveren: **yppasswd**, **ypchfn** og **ypchsh**.

De fungerer på samme måde som de tilsvarende kommandoer uden yp-præfikset. Hanne vil gerne skifte sit kodeord:

```
[hanne@omnitec ~]$ yppasswd
Changing NIS account information for hanne on supernaut.strangeparty.dk.
Please enter old password: GAMMELKODE
Changing NIS password for hanne on supernaut.strangeparty.dk.
Please enter new password: NYKODE
Please retype new password: NYKODE

The NIS password has been changed on supernaut.strangeparty.dk.
```

Lad os se om det virker:

```
[hanne@omnitec ~]$ telnet supernaut
Trying 192.168.10.1...
Connected to supernaut.strangeparty.dk.
Escape character is '^]'.
```

```
Red Hat Linux release 6.0 (Hedwig)
Kernel 2.2.14 on an alpha
login: hanne
Password: NYKODE
Last login: Wed Mar 29 13:02:19 from omnitec.strangeparty.dk
```

Fint. Kodeordet er ændret på serveren. Men hvad med klienten?

```
[hanne@supernaut ~]$ telnet omnitec
Trying 192.168.10.126...
Connected to omnitec.strangeparty.dk.
Escape character is '^]'.

Red Hat Linux release 6.1 (Cartman)
Kernel 2.2.12-20 on an i586
login: hanne
Password: NYKODE
Last login: Wed Mar 29 13:46:21 from supernaut.strangeparty.dk
```

Hvis man som administrator ændrer i adgangskodefilen, opretter eller sletter en bruger, eller ændrer andet der påvirker NIS, skal man sørge for at NIS-databaserne er konsistente efter ændringerne:

```
[root@omnitech /root]# cd /var/yp
[root@omnitech /root]# make
```

Og så skal vi lige sørge for at det starter næste gang maskinerne genstartes: På serveren skal **ypserv**, **yppasswdd** og **ypbind** startes i runlevel 3, 4 og 5.

```
[root@supernaut /home/kwv]# /sbin/chkconfig --list | grep yp
ypbind 0:off 1:off 2:off 3:on 4:on 5:on 6:off
yppasswdd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
ypserv 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

ypserv starter ikke automatisk. Det får vi den til:

```
[root@supernaut /home/kwv]# /sbin/chkconfig ypserv on
[root@supernaut /home/kwv]# /sbin/chkconfig --list | grep yp
ypbind 0:off 1:off 2:off 3:on 4:on 5:on 6:off
yppasswdd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
ypserv 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

NIS-domænet skal sættes, både på server og klient. Det gøres i `/etc/sysconfig/network`:

nisdomain=spo

På klienten skal ypbind også startes ved boot:

```
[root@supernaut /home/kwv]# /sbin/chkconfig --list ypbind
ypbind 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

NIS kan en del mere end beskrevet her. Men dette er normalt det, der er størst behov for.

For mere information: Linux Network Administrators Guide, NIS-HOWTO, Managing NFS & NIS, O'Reilly & Associates, Manualsiderne fra ypserv- og ypbind-pakkerne. Find dem således:

```
[hanne@supernaut ~]$ rpm -ql `rpm -qa | grep ^yp` |grep man
```

7.5. AppleTalk

Linux kan ikke bare forbindes til Apple via ftp, men også via almindelig fildeling. For brugerne af lokalnettet kommer Linux-serveren så til at optræde som en Mac-server.

Det er ikke svært at anvende en Linux-maskine som filserver hvis de andre klienter i netværket kører Microsoft Windows®, da Samba-protokollen (som beskrevet i Kapitel 10) følger med stort set alle distributioner af Linux. Dette er ikke så lige til med Mac-protokollen, da du selv må hente denne. Til at forbinde et Mac-netværk med en Linux-maskine bruges et program/protokol der hedder Netatalk. Dette program får på listig vis din Linux-maskine til at se ud som en Macintosh-filserver. Programmet understøtter to typer af AppleTalk-protokollen; nemlig Classic AppleTalk og AppleShareIP (sidstnævnte kaldes også TCP/IP til Mac).

Programmet blev oprindeligt udviklet af University of Michigan's Research Systems Unix Group og understøttede oprindeligt kun Classic AppleTalk. Sidenhen er det blevet udviklet og vedligeholdt af Adrian Sun, der ud over at debugge programmet også har implementeret understøttelse af AppleShareIP og andre udvidelser.

Netatalk kan hentes fra: ftp://ftp.cobaltnet.com/pub/users/asun/release/netatalk-1.4b2+asun2.1.3.tar.gz (ftp://ftp.cobaltnet.com/pub/users/asun/release/netatalk-1.4b2+asun2.1.3.tar.gz)

og færdigoversat fra:
http://thehamptons.com/anders/netaltalk/mirror/netatalk-1.4b2+asun2.1.1.x86-shadow-linux.tar.gz (http://thehamptons.com/anders/netaltalk/mirror/netatalk-1.4b2+asun2.1.1.x86-shadow-linux.tar.gz)

Til Debian: http://packages.debian.org/stable/non-US/netatalk (http://packages.debian.org/stable/non-US/netatalk)

Til RPM-baserede systemer (SuSE, RedHat, Mandrake):
ftp://contrib.redhat.com/pub/contrib/libc6/SRPMS/netatalk-1.4b2+asun2.1.3-6.src.rpm (husk at køre **rpm --rebuild -sti-til-source** for at oversætte og installere den)

Hvis du har hentet en af `.tar.gz`-filerne, skal du gøre sådan her:

```
[root@linus /root]# tar xzvf <filnavn>.tar.gz
...
[root@linus /root]# cd netatalk<tab><enter>
[root@linus netatalk]# make install
```

Netatalk "bor" i kataloget `/usr/local/atalk/`. Server-binaries og opsætningsfiler ligger i `/usr/local/atalk/etc` og userland-binaries i `/usr/local/atalk/bin`. Hvis du ikke kan finde **netatalk** efter du har installeret den, så brug kommandoen **find / -name atalkd** for at finde den.

For at kunne bruge Classic AppleTalk-protokollen, skal understøttelsen af denne være slået til i kernen. For at finde ud af, om den er det, gør du følgende: dmesg | grep -i apple. Hvis den ikke er der, skal du oversætte din kerne om, med dette tilvalg slået til.

Det program der gør AppleTalk-shares tilgængelige i både Classic AppleTalk og AppleShareIP hedder **afpd** og får sin opsætning fra en fil der hedder `afpd.conf`.

Den simpleste `afpd.conf`-fil indeholder kun ét - (minus), hvilket står for "Denne computer" med alle standard options koblet til. En lidt mere kompliceret opsætning kunne være følgende:

```
-
MinLinuxMac -port 12000
"Denne Linux Leger Mac" -port 12001
"Der Ude" -port 12000 -address 206.114.89.46
"Guest Volume" -nocleartxt  -loginmesg "Vær venlig ved min gæsteserver !"
"Boksen" -noguest -port 12000
"Det Helligste" -notcp -defaultvol <path> -systemvol <path>
```

Forklaring af opsætningsfilen:

Den første linje kun med en bindestreg gør at Linux-maskinen laver en server med sit hostnavn. Den anden linje laver en server ved navn MinLinuxMac på port 12000. Den tredje linje laver en server kaldet "Denne Linux Leger Mac" på port 12001. Den fjerde linje laver en server kaldet "Der Ude", bundet på port 12000 og med den virtuelle IP-adresse 206.114.89.46.

I næste sektion laver linjen "Guest Volume" -nocleartxt -loginmesg "Vær venlig ved min gæsteserver !" en dedikeret gæsteserver, der ikke spørger om adgangskode og som giver login-beskeden "Vær venlig...". Linjen "Boksen" -noguest -port 12000 laver en volume på port 12000 der ikke tillader gæsteadgang. Linjen "Det Helligste" -notcp -defaultvol <path> -systemvol <path> laver en server der ikke bruger TCP/IP (kun DDP) og som bruger to alternative opsætningsfiler i stedet for de to standardfiler `AppleVolumes.default` og `AppleVolumes.system`. Husk at rette i <path> hvis du vil gøre dette.

Lav et Classic AppleTalk-share:

For at lave et Classic AppleTalk-share skal du ændre i filen `/usr/local/atalk/etc/AppleVolumes.system` Eksempelvis hvis du indsætter linjen `/data/www` "Webserver" laves der en volume kaldet Webserver, hvis indhold så vil kunne ses.

Den anden ting denne fil styrer, er "type-creator mappings". På Macintosh består alle filer (ulig på Unix) af to dele. En "resource fork" og en "data fork". Resource fork'en gemmer på ting som ikoner, filtyper, og mapping til den applikation der i sin tid blev brugt til at lave filen. Siden Linux-filer generelt er en lang liste af bogstaver i stedet for delt i to forke/forks, skriver netatalk resource forkene i filer i et katalog kaldet `.AppleDouble`. Derfor skabes én Macintosh-fil af to filer på Linux-maskinen; eksempelvis `enfil.txt` og `.AppleDouble/enfil.txt`. Denne fil ville så, hvis det var en tekstfil og Word havde lavet den, skulle have haft følgende type-creator mapping i `afpd.conf`: .txt TEXT MSWD for at fortælle **afpd** at .txt-filer er TEXT-filer og at det var Microsoft Word der blev brugt til at fremstille filen. Afpd vil så fortælle dette til Macintosh-maskiner der støder på filer der ender på .txt og som ikke er blevet oprettet på en Macintosh-maskine.

ATALKD-daemonen:

Atalkd er det kerne-interface der bruges til Classic Appletalk. Det vil agere forbindelse mellem AppleTalk-kernemodulet og resten af Classic AppleTalks funktioner i netatalk. Den vil sågar også tage sig af AppleTalk-routing mellem flere netkort.

Atalkd er styret af opsætningsfilen atalkd.conf. Den enkleste form for opsætningsfil (og også den enkleste måde at konfigurere atalkd på) er en tom fil. Denne fil vil atalkd så selv overskrive med nogle ganske fornuftige opsætninger. Det er tilrådeligt at specificere alle de netkort man har i atalkd.conf-filen. Hvis man specificerer mere end ét netkort, vil atalkd selv route AppleTalk-pakker mellem netkortene.

Følgende opsætning kunne overvejes:

```
eth0 -net 153-174 -addr 154.212 -zone "Langt Ude"
eth1 -net 175-200 \adr 182.318 -zone "På Landet" -seed
```

Hvad betyder så dette? Forklaringen er ret simpel; hver linje i filen `atalkd.conf` specificerer ét netkort. I dette eksempel betyder det at eth0 (første netkort i Linux-maskinen) bruges på et netværk med AppleTalk-adresser mellem 153 og 174 (se eventuelt afsnittet om TCP/IP for nærmere forklaring om netværksadressering). Dette netkort (eth0) er her konfigureret til at bruge AppleTalk-adressen 154.212 og eksisterer i zonen "Langt Ude".

Næste linje betyder at det sekundære netkort eth1 eksisterer på et netværk mellem 175 og 200 med adressen 182.318 og pakker til zonen "På Landet". Derfor vil atalkd agere router mellem de to netværk. Bemærk, at netværksnumrene ikke må overlappe hinanden; ej heller må der være andre routere på eth1-netværket. Hvis atalkd finder en anden router end den der er specificeret i filen, vil programmet afslutte sig selv. Bemærk også, at hvis du har flere end 5 maskiner på netværket, skal du angive dette

over for afpd (se nedenfor) - ellers virker det bare ikke! Derfor anbefales det at læse afsnittet om TCP/IP - det er ikke så svært som det umiddelbart ser ud til.

Så kommer den sjove del af det hele - vi skal prøve at starte Netatalk-serveren op!

Først skal du finde ud af hvor Netatalks starup-script ligger, hvis du ikke allerede ved det.

Dette kan du gøre sådan her:

```
find /* -name "*atalk*" -print
```

eller med

```
locate "*atalk*"
```

(Husk at køre **updatedb** som root efter installationen af Netatalk, ellers vil den ikke kunne findes af databasen af kommandoen **locate**).

Når du så er i det katalog hvor scriptet er, skriver du: **./rc.atalk start** og venter et par sekunder på at Netatalk starter op. Dette vil så starte en filserver der kører Classic AppleTalk. Kun hvis du har specificeret nogle TCP/IP-adresser i `afpd.conf`-filen vil en tilsvarende Appletalk IP-server blive startet også. Atalkd er den der tager længst tid at starte op, da den først kontrollerer netværket ud før den registrerer sig selv. Alternativt kan du nøjes med kun at starte en TCP-server, hvis du på forhånd har konfigureret en i `atalkd.conf`-filen. Det gøres sådan her:

```
/usr/local/atalk/etc/afpd -F
/usr/local/atalk/etc/afpd.conf
```

Bemærk! Hvis du skal bruge mere end 5 forbindelser til Netatalk-serveren skal du angive et antal maksimumforbindelser når du starter afpd. Det gøres med argumentet -c: **afpd -c 25** Dette vil så give mulighed for 25 forbindelser til Netatalk-serveren.

Hvis denne kommando lykkes uden at brokke sig, så gå hen til en Mac og åbn "Chooser" (under Apple-menuen i venstre side). Klik på AppleShare og se om din Netatalk-server er der. Hvis du bruger AppleShareIP skal du klikke på AppleShareIP-knappen og manuelt indtaste maskinens navn eller IP-nummer. Hvis alt så er lykkedes vel, vil du blive bedt om at indtaste din adgangskode.

Husk, at du af sikkerhedsmæssige årsager ikke kan logge ind på filserveren som root, og kun hvis dit brugernavn består af 8 eller færre tegn. Guest access (gæsteadgang) til filserveren vil være tilladt, hvis det ikke er blevet forbudt i `afpd.conf`.

Hvis det lykkes dig at logge ind, vil du blive præsenteret for en liste af "volumes" du så kan vælge - og montere. Det er tilrådeligt at bruge et af de startup-scripts der følger med Netatalk, til at starte denne med

når maskinen booter. Alt afhængigt af hvilken distribution du bruger, skal du følge fremgangsmåden for hvordan din maskine gør. På en Red Hat skal du nok kigge på `/etc/rc.d/rc.local`. Ellers er et andet godt bud `/etc/init.d`, for at finde ud af hvor dine startup-scripts befinder sig.

Andre ting man kan gøre med Netatalk:

Tovejs-krypteret "password authentication" er understøttet af Netatalk. Men for at dette skal kunne bruges, må serveren nødvendigvis vide hvor klartekst-adgangskoderne kan findes henne. Da systemadgangskoderne er krypterede ved serveren ikke hvad klartekst-adgangskoderne er, så det skal den have at vide. Ved at oprette en `.passwd`-fil i brugerens hjemmekatalog med klartekstadgangskode i kan tovejskryptering opnås (punktummet foran i filen betyder at den er skjult og vil kun blive vist, hvis man bruger **ls -a**).

Det ganske usikkert at gemme sine adgangskoder på et så forudsigeligt sted. Derfor kræver Netatalk at hver `.passwd`-fil ejes af den pågældende bruger. Filen må kun være læsbar og skrivbar for denne bruger. I enkle ord skal du en gang for alle bare gøre sådan med `.passwd`-filen:

```
[root@linus /root]# chown BRUGER.GRUPPE /home/<brugernavn>/.passwd
[root@linus /root]# chmod 600 /home/<brugernavn>/.passwd
```

Hvor BRUGER er brugerens gruppe, og GRUPPE er brugerens standardgruppe.

7.5.1. Udskrivning via Netatalk

Netatalk kan ikke kun agere filserver for Mac-maskiner. Den kan også lave print-spooling og endda printe til eksisterende AppleTalk-spools. AppleTalk-spooling opnås med et program der kaldes papd.

Vi antager at du allerede har en lokal printer der er sat op under Linux og som har Mac-drivere installeret. papd vil så annoncere sig selv på netværket som et print spool og acceptere jobs fra Mac-maskiner på netværket. Dette kan lade sig gøre, da printerjobs er Postscript-filer, som papd så vil føde Linux-printsystem (lpd) med. Eksempler på hvordan papd sættes op kan findes på:

http://www.giub.unibe.ch/~eugster/appleprint.html (http://www.giub.unibe.ch/~eugster/appleprint.html)

Det modsatte af dette - altså at printe fra en Linux-maskine til en Macintosh-printer - kan opnås ved at bruge pap. Se hvordan med kommandoen **man pap**.

7.5.2. Anden nyttig viden om AppleTalk-server

Lige som på Linux og Microsoft Windows® er der værktøjer til at pinge maskiner med. Over TCP/IP findes der **aecho** til AppleTalk, der virker ligesom ping. Kommandoen **getzones** vil returnere en liste

over de Macintosh-zoner der findes på netværket. Ydermere vil kommandoen **nbplkup** uden options returnere en uddybende liste over alle de AppleTalk services der findes på det lokale netværk.

Der findes mange andre programmer til at bruge med Netatalk - så vær ikke bange for at gå i gang med at udforske området - held og lykke !

Andre steder på nettet man bør besøge hvis man bruger Netatalk:

The Netatalk HOWTO: http://thehamptons.com/anders/netatalk/ (http://thehamptons.com/anders/netatalk/)

The Netatalk Faq-O-Matic: http://www.zettabyte.net/fomserve/netatalk/cache/1.html (http://www.zettabyte.net/fomserve/netatalk/cache/1.html)

The Original Netatalk pages: http://www.umich.edu/~rsug/netatalk/ (http://www.umich.edu/~rsug/netatalk/)

The Netatalk Admins list: netatalk-admins-request@umich.edu (mailto:netatalk-admins-request@umich.edu)

7.6. Linux som ring-ind-server

Til fjernadministration kan man sætte sin maskine op, så det er muligt at ringe ind til serveren og dermed administrere den via modem.

Programmerne **ppp** og **mgetty** (eller lignende) skal være installeret på den maskine, som skal kunne ringes op.

Find ud af hvilken serielport dit modem sidder på. Eksterne modemer er typisk tilknyttet enheden `/dev/ttyS0` (COM1 under Windows) eller `/dev/ttyS1` (COM2 under Windows). Interne modemer er typisk tilknyttet enheden `/dev/ttyS2` (COM3 under Windows) eller `/dev/ttyS3` (COM4 under Windows).

Den fundne enhed skal så linkes til enheden `/dev/modem` således:

```
[root@linus /root]# ln -s /dev/ttyS0 /dev/modem
```

Tilføj nedenstående linje i opsætningsfilen `/etc/inittab`

```
S0:2345:respawn:/usr/sbin/mgetty ttyS0 -D /dev/ttyS0
```

Efter den er blevet gemt, opdateres oplysningerne med kommandoen **init q**.

For at brugeren der kommer ind skal kunne køre **pppd** skal rettighederne lige ændres, så de ændres til SUID:

```
[root@linus /root]# chmod u+s /usr/sbin/pppd
```

Så er turen kommet til tilretning af **mgetty** via dets opsætningsfiler.

Opret en ny opsætningsfil `/etc/ppp/options.server` for **ppp**-serveren med følgende indhold:

```
-detach
asyncmap 0
modem
crtscts
lock
proxyarp
require-pap
refuse-chap
```

Tilføj de loginnavne (client), som skal have mulighed for at få adgang til maskinen, samt tilhørende adgangskode (secret) i klar tekst. Angiv stjerner for tjeneste (server) og IP-adresse i filen `/etc/ppp/pap-secrets`

```
# Secrets for authentication using PAP
# client        server  secret                  IP addresses
  martin        *       nitram                  *
```

Disse loginnavne og adgangskoder skal helst ikke være de samme som for selve systemet, da de jo står i klar tekst. De skal udelukkende bruges til at få forbindelse med maskinen, så man kan benytte IP-baserede tjenester på maskinen. Du bør nok ikke lade `/etc/ppp/pap-secrets` være læsbar for almindelige brugere. Kør derfor **chmod og-rw /etc/ppp/pap-secrets**.

Til sidst der i filen `/etc/ppp/options.ttyS0` angives henholdsvis IP-nummeret for værten og IP-nummeret for den maskine der ringer ind:

```
192.168.1.1:192.168.1.222
```

Vær opmærksom på, at IP-nummeret for den maskine som ringer ind, skal være en del af samme subnet som værten.

Så er det muligt at ringe ind til din maskine.

Denne vejledning er baseret på Linux Gazette #38's vejledning "Linux Dialin Server Setup", som er langt mere udtømmende: http://www.linuxgazette.com/issue38/gentry.html (http://www.linuxgazette.com/issue38/gentry.html)

7.7. Linux som X-server og terminal

En Linux-maskine kan være en glimrende X-server for andre mindre grafiske terminaler. Tilsvarende kan en Linux-maskine snildt virke som en X-terminal.

7.7.1. Linux som X-server

For at en Linux-maskine kan virke som grafisk login-server, skal der køre en X-server på maskinen. Normalt kræver dette, at man kører i runlevel 5, dvs. man ændrer linjen i `/etc/inittab` (eksemplet er fra Red Hat)

```
id:3:initdefault:
```

til

```
id:5:initdefault:
```

og tilsvarende længere nede i samme fil

```
# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

ændres til

```
# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm
```

I `/etc/X11/xdm/xdm-config` skal man derefter indsætte et udråbstegn foran linjen der starter med `DisplayManager.requestPort`.

Anvender du Gnomes login-system, så skal du endvidere rette i `/etc/X11/gdm/gdm.conf`.

```
[xdmcp]
enable=0
```

rettes til

```
[xdmcp]
enable=1
```

og root skal så genstarte i runlevel 5 ved at køre **init 5**. Den option `-nodaemon` der blev fjernet gør at andre maskiner kan koble sig ind på serveren. Der er således noget med sikkerhed, man bør tænke på. På et mindre lukket netværk spiller det ingen rolle. Nu har du en X-server, og lad os nu se på en Linux-klient.

7.7.2. Linux som X-klient

Du skal logge ind på din Linux-maskine i runlevel 3 og som root skrive **/usr/X11R6/bin/X -query SERVER**, hvor `SERVER` er navnet på din X-server. Denne kommando kan du evt. så senere indsætte som en del af `/etc/inittab`.

```
x:3:respawn:/usr/X11R6/bin/X 2>/var/log/Xlog -dpi 100 -query SERVER
```

hvor SERVER byttes ud med navnet på den maskine der er X-server.

7.8. Printserver med CUPS

En af de virkelig gode ting ved CUPS er at det er rigtig nemt at bruge i et netværk.

Du skal sætte alle printerne op på en maskine (serveren), som beskrevet i CUPS-afsnittet i "Friheden til at vælge installation". På den maskine skal du desuden i `/etc/cups/cupsd.conf` sørge for at der er adgang til printerne, ved at tilføje linjer a la:

```
Allow From 192.168.42.0/24  # Erstat med de IP-numre du bruger
```

til **<Location />**-afsnittet.

På de andre maskiner skal du så installere klientdelen af CUPS (hvis det er muligt at nøjes med den) på Debian hedder den nødvendige pakke `cupsys-client`. Herudover skal du så i `/etc/cups/client.conf` tilføje linjen

```
ServerName server.domæne
```

og skulle det spille. For at tjekke om det virker kan du prøve at køre **lpstat -p**, den skulle gerne give dig en liste over de tilgængelige printere, der skulle ligne hvad samme kommando giver på serveren.

Bemærk at CUPS-dæmonen (cupsd) ikke behøver at køre på klienterne!

7.9. High Availability

Der har været meget fokus det sidste år på "High Availailty", dvs. at få stor sikkerhed i drift. Typisk ved at have ekstra servere klar, som automatisk tager over, hvis der er en defekt med en server.

Her er nogle gode steder at søge efter information:

- http://www.LinuxVirtualServer.org/
- http://linux-ha.org/
- http://www-1.ibm.com/servers/esdd/articles/linux_clust/index.html
- Red Hats produkter http://www.redhat.com/software/rha/cluster/ og http://www.redhat.com/software/rha/cluster/manager/.
- http://www.linux-high-availability.com/

7.10. LDAP

LDAP er en forkortelse for *Lightweight Directory Access Protocol* og er en database service, hvorved man kan håndtere bruger-information effektivt over et stort netværk. Flere informationer kan findes fra http://linux.oreillynet.com/pub/a/linux/2001/11/08/ldap.html eller http://www.linuxjournal.com/article.php?sid=5689.

Kapitel 8. E-post servere

Hvis du installerer rpm-pakken `imap`, kan din Linux-maskine bruges som server for indgående post i et netværk. Ved installationen af pakken skulle hele opsætningen faktisk være på plads, så brugerne på de enkelte maskiner blot skal angive Linux-maskinen som postserver i deres e-post-programmer – og så naturligvis være oprettet som brugere på Linux-maskinen!

Hvis du kun har en modemforbindelse til internettet, kan du lade Linux-serveren sende og hente post for alle brugere på et lokalt netværk. Programmet **fetchmail** (se bogen "Linux - friheden til at lære Unix") er ret nemt at sætte op til at hente post til flere forskellige brugere på samme tid hos internetudbydere.

Udgående post håndteres af programmet **sendmail**. Hvis **sendmail** skal acceptere at videresende udgående e-post fra andre maskiner på et lokalnet, er det nødvendigt at angive dette i filen `/etc/mail/ip_allow`, ellers vil forsøg på at bruge Linux-maskinen som SMTP-server blive afvist med beskeden "We do not relay".

I `/etc/mail/ip_allow` skriver du adresser på enkelte maskiner eller netværk, som må bruge Linux-maskinen som udgående postserver. Hvis for eksempel alle maskiner på netværket `192.168.100.0` skal accepteres, skal indholdet af `/etc/mail/ip_allow` være dette:

```
192.168.100
```

Hvis post kun skal accepteres fra nogle af maskinerne på netværket, skrives adresserne på de enkelte godkendte maskiner. Alternativt kan man tillade alle maskiner i eget domæne at være "relay"-maskine ved at tilføje `FEATURE(`relay_entire_domain')` i `/etc/sendmail.mc` og derefter køre **m4 /etc/sendmail.mc /etc/sendmail.cf**. Bemærk, at man skal have sendmail-cf-pakken installeret.

At tilpasse **sendmail**'s opsætning er i øvrigt et emne, som kan fremkalde nervøse trækninger og koldsved hos store voksne systemadministratorer. Red Hat Linux sætter **sendmail** ganske fornuftigt op fra starten, så medmindre du er meget videbegærlig eller masochistisk anlagt, vil vi foreslå dig at glemme alt om opsætning af **sendmail**, indtil det er absolut nødvendigt. Du kan med fordel erstatte Sendmail med Postfix. Se Afsnit 8.1.

8.1. Postfix

Postfix er en moderne MTA (mail transfer agent), dvs. et program til at flytte e-post fra én postserver til en anden. I mange år var Sendmail den MTA, alle brugte, men i de sidste par år har det vist sig, at Sendmail ikke kan leve op til de sikkerhedskrav, man typisk stiller.

Postfix har umiddelbart tre fordele:

- Skrevet af Wietse Venema, som er kendt for sin store viden og kunnen inden for edb-sikkerhed.

- Systemet er brudt ned i mange små delprogrammer. Hver komponent er lettere at overskue, og det mindsker risikoen for fejl.
- Langt de fleste komponenter behøver ikke at blive kørt som root, hvilket betyder at evt. programmerings- eller konfigureringsfejl ikke bliver nær så alvorlige.

Mandrake Linux kommer med Postfix og Sendmail, og du kan under installationen vælge hvilken MTA, du vil bruge. Dette valg har du endnu ikke under Red Hat eller SuSE, og i det følgende vil vi antage at du har Red Hat kørende.

Først skal du stoppe og afinstallere sendmail inden du går i gang med Postfix.

```
[root@linus /root]# cd /etc/rc.d/init.d
[root@linus init.d]# ./sendmail stop
Shutting down sendmail:                 [ OK ]
[root@linus init.d]# rpm -e --nodeps sendmail
```

Du er nu klar til at hente og installere Postfix. Find den på http://www.postfix.org/ eller http://rpmfind.net. Se efter en fil kaldet `postfix-*.i386.rpm`.

Efter du har hentet filen, skal du installere den.

```
[root@linus /root]# rpm -i postfix-1.1.7-1.i386.rpm
postfix-script: Warning: Creating missing Postfix pid directory
postfix-script: Warning: Creating missing incoming directory
postfix-script: Warning: Creating missing bounce directory
postfix-script: Warning: Creating missing defer directory
postfix-script: Warning: Creating missing deferred directory
postfix-script: Warning: Creating missing saved directory
postfix-script: Warning: Creating missing corrupt directory
postfix-script: Warning: Creating missing public directory
postfix-script: Warning: Creating missing private pid directory
[root@linus /root]# mkdir /var/log/mail
[root@linus /root]# cd /etc/rc.d/init.d
[root@linus init.d]# ./postfix start
Starting postfix: postfix-script: starting the Postfix mail system
[root@linus init.d]# chkconfig --level 3 postfix on
```

Nu har du fået udskiftet Sendmail med Postfix, og den nye MTA vil altid blive startet op, når du kører i runlevel 3 (den sidste kommandolinje).

Postfix er god til at tilpasse sig til dit system og behøver sjældent nogen opsætning. Du skal dog sikre dig, at kommandoen hostname giver det korrekte navn på din maskine:

```
[root@linus /root]# hostname
linus.kongeh.dk
```

Navnet skal eksistere i DNS.

Hvis du ikke kan sende email til root@linus.kongeh.dk, så se i `/var/log/maillog` eller `/var/log/mail/*`. Her skriver postfix hvis der er noget galt.

Tip: Typisk skal man sætte at man afleverer epost til den SMTP-gateway ens internet-udbyder har (TDC har `smtp.mail.dk`). Dette gøres ved at udkommentere linjen med `relayhost` i filen `/etc/postfix/main.cf` og skrive navnet på den server man anvender:

```
...
relayhost = smtp.mail.dk
...
...
```

Postfix har hjemmesiden www.postfix.org (http://www.postfix.org/), hvor der også er en god FAQ.

8.1.1. Modtage e-mail for flere domæner samtidigt

Postfix kan nemt sættes op til at modtage e-mail fra flere domæner samtidigt, og sende disse e-mails videre til flere forskellige steder. Filen der styre dette ligger i filen `/etc/postfix/virtual`, der er grundlaget for databasen `/etc/postfix/virtual.db`.

Det første der skal sættes op, er at få Postfix til at bruge filen `/etc/postfix/virtual` til at slå domæner og brugere op. I filen `/etc/postfix/main.cf` indsættes følgende linje:

```
# Filnavn: /etc/postfix/main.cf
virtual_maps = hash:/etc/postfix/virtual
```

Og så lige den obligatoriske reload:

```
[root@linus /root]# /etc/init.d/postfix reload
```

Postfix kan ved hjælp af koden *hash:* se at det er filen `/etc/postfix/virtual.db` der skal slås op i. Dette er en binær fil der skal genereres hver gang der sker ændringer i `/etc/postfix/virtual`.

Herefter kan domæner oprettes i filen `/etc/postfix/virtual`. For overskuelighedens skyld, er her først vist opsætning for eet domæne. Der er kun e-mailadressen *postmaster@eksempel.dk*, og e-mail sendes til brugeren *root*, der har en konto lokalt på maskinen.

```
# Filnavn: /etc/postfix/virtual
eksempel.dk anything
postmaster@eksempel.dk root
```

`virtual` kan nu afprøves ved at kompile den til `virtual.db`, hvilket *skal* gøres hver gang der er ændret i `virtual`.

```
[root@linus /root]# postmap hash:/etc/postfix/virtual
```

Efter at have kompilet `virtual` kan man udføre en *reload* til postfix. Gør man ikke det, vil der gå ca. 1 minut inden postfix selv opdager at `virtual.db` er ændret, og så reloader denne.

Hvis man gerne vil have at al anden mail end ovenstående, skal sendes til brugeren *linus*, tilføjes en ekstra linje hvor der ikke er noget user-name.

```
# Filnavn: /etc/postfix/virtual
eksempel.dk anything
@eksempel.dk           linus
postmaster@eksempel.dk root
```

Mail kan også sendes videre til en helt anden bruger på en helt anden server. Herunder er brugeren *abuse@eksempel.dk* tilføjet, og sendes videre til et domæne udenfor serveren.

```
# Filnavn: /etc/postfix/virtual
eksempel.dk anything
@eksempel.dk           linus
abuse@eksempel.dk      nogen@tagersigafdet.com
postmaster@eksempel.dk root
```

Når der skal tilføjes et domæne mere, skal der være en blank linje til næste blok. Herunder er der tilføjet domænet *linux.dk*. E-mailreglerne for disse to domæner er de samme, så al mail der måtte komme sendes blot videre *eksempel.dk*. Mail til *postmaster@linux.dk* vil således blive sendt videre til *root*.

```
# Filnavn: /etc/postfix/virtual
eksempel.dk anything
@eksempel.dk           linus
abuse@eksempel.dk      nogen@tagersigafdet.com
postmaster@eksempel.dk root

linux.dk anything
@linux.dk              @eksempel.dk
```

8.2. Mailman

En af de rigtig gode programmer til håndtering af postlister er Mailman. Det kommer med grafisk administration via en webbrowser, kan lave mail<-%gt;news gateway automatisk, email-arkiv og meget andet.

8.2.1. Installation af Mailman

Installation af mailman fra RPM er ikke nødvendigvis det nemmeste. Man kan hente kildeteksten og oversætte denne med standard udpak (tar xvzf), **./configure**, **make** og **make install**.

Herved kommer man igennem et par skridt med at lave en "mailman" bruger og gruppe, lave kataloget `/usr/local/mailman` ejet af mailman:mailman, og det går ret nemt.

På Mandrake skal mailman-pakken installeres og så er man ret tæt på at være klar. Red Hat er ikke helt så nem – man de hemmelige tricks er forklaret i det følgende :)

8.2.2. Opsætning af Mailman

Følgende ting skal køres en gang for alle.

Find først programmet **mmsitepass**.

```
[root@linus /root]# rpm -ql mailman | grep mmsitepass
/var/mailman/bin/mmsitepass
```

Installerede du fra kildetekst, så brug **find / -name mmsitepass**.

I dette tilfælde er det `/var/mailman/bin/mmsitepass` som skal køres som root for at sættes postlisteserverens adgangskode.

```
[root@linus /root]# /var/mailman/bin/mmsitepass
New site password:
```

Angiv en god adgangskode som skal anvendes når der skal laves nye lister i det følgende!

Hernæst skal man finde den standard opsætningsfil som mailman anvender til bl.a. at sætte internet-information.

```
[root@linus /root]# rpm -ql mailman | grep mm_cfg.py
/var/mailman/Mailman/mm_cfg.py
```

Indsæt i denne fil URL-adressen på maskinens webserver som `DEFAULT_URL_HOST` og domænenavnet for postserveren står i `DEFAULT_EMAIL_HOST`. I eksemplet sættes lister op for LISTENAVN@hven.dk, hvor LISTENAVN kommer senere og www.hven.dk skal passe med at der er en web-server på http://www.hven.dk.

```
DEFAULT_EMAIL_HOST = 'hven.dk'
DEFAULT_URL_HOST = 'www.hven.dk'
```

Det kan også være man bliver nødt til (nogle gange hos Red Hat), at eksplicit anføre brugerid og gruppe på mailman-brugeren. Er det bruger-ID nr 41 og gruppe 42 (se `/etc/passwd`) så tilføj til `mm_cfg.py`.

```
MAILMAN_UID = 41
MAILMAN_GID = 42
```

Anvender man postfix som MTA, så skriv også i `mm_cfg.py`

```
MTA = 'Postfix'
unknown_local_recipient_reject_code = 550
POSTFIX_ALIAS_CMD = '/usr/sbin/postalias'
POSTFIX_MAP_CMD = '/usr/sbin/postmap'
```

Sikr dig at **/usr/sbin/postalias** og **/usr/sbin/postmap** eksisterer.

Der er masser af gode forklaringer på hvordan mailman fungerer internt i filen `Defaults.py` som ligger samme sted som `mm_cfg_py`, dvs. det kan være interessant at læse den igennem på et tidspunkt. Generelle opsætninger for alle lister på maskinen bør ske igennem denne fil.

På Red Hat skal man nu tjeke at `mailman`-scriptet kan startes. Kør **/etc/init.d/mailman start** og se om det går godt. Mandrake har ikke dette script.

På Red Hat skal man køre følgende en gang for alle for at mailman startes sammen med maskinen

```
[root@linus /root]# /sbin/chkconfig --level 35 mailman on
[root@linus /root]# /sbin/chkconfig --list mailman
mailman         0:off   1:off   2:off   3:on    4:off   5:on   6:off
```

Dette starter mailman (kun nødvendigt på Red Hat) i runlevel 3 og 5 (se Afsnit 2.5).

Det smarte med Mailman er at man kan administere systemet via en web-browser. Derfor skal man nu tilføje `/mailman/` (webadministration) og `/pipermail/` (postarkiv) til Apache webserveren. Der er en fil `/etc/httpd/conf/mailman.conf`, som følger ned Mailman i RPM-form, som nu skal tilføjes i Apaches `Vhosts.conf` eller `httpd.conf` under den webserver man anvender i forvejen. Erstat `/var/mailman/` med den korrekte sti hvis du ikke anvender Red Hat RPM-pakker.

```
ScriptAlias /mailman/ /var/mailman/cgi-bin/
<Directory /var/lib/mailman/cgi-bin>
  Options -Indexes -FollowSymLinks -Includes ExecCgi
  AllowOverride None
  order allow,deny
  allow from all
</Directory>

#
# Configure the public archives
#
```

```
Alias /pipermail/ /var/mailman/archives/public/
<Directory /var/lib/mailman/archives/public>
  Options -Indexes FollowSymlinks -Includes
  AllowOverride None
  order allow,deny
  allow from all
</Directory>
```

8.2.3. Oprette postlister i Mailman

For at oprette en enkelt post-liste så kan man enten lade mailman automatisk generere nye post-aliaser eller man kan køre det mere manuelt. I det følgende er det vist manuelt.

Find først kommandoen til at lave en ny liste **newlist**.

```
[root@linus /root]# rpm -ql mailman | grep newlist
/var/mailman/bin/newlist
```

(samme sted ligger kommandoen til at slette en liste **rmlist**).

Find herefter kommandoerne til at oprette en liste **newlist**

```
[root@linus /root]# /var/mailman/bin/newlist LISTENAVN admin@hven.dk MIT_HEMMELIGE_PASSWORD_FOR_DENNE
```

Vil man senere slette en liste helt (inkl. mailarkiv)

```
[root@linus /root]# /var/mailman/bin/rmlist -a LISTENAVN
```

Den første liste man skal oprette er listen "mailman", der muligvis allerede optræder i `/etc/aliases` – slet dette og lav en "mailman"-liste i stedet. Erstat admin@hven.dk med systemadministratorernes post-konto.

```
[root@linus /root]# /var/mailman/bin/newlist mailman admin@hven.dk MIT_HEMMELIGE_PASSWORD_FOR_DENNE_L
```

8.3. Exim

Exim er en mailserver som er nem at sætte op til almindeligt brug. Den har også en del avancerede funktioner, som for eksempel indbygget filter og vacation-program. Og så skulle den iøvrigt være sendmail-kompatibel.

Man finder den her: http://www.exim.org. Her er der dog ingen rpm-filer, men dem kan man finde på http://rpmfind.net.

Der følger desværre ikke nogen fil med til automatisk at lave en rpm-fil, så hvis man selv vil lave den, så er det vigtigt at følge installationsvejledningen.

Exim styres fra 1 stor opsætningsfil, som ligger i `/etc` eller for SuSE 9.0 i `/etc/exim/`. Opsætningsfilen er indelt i 7 dele, og det er normalt ikke nødvendigt at ændre i den for at få et virksomt system, da det enten er noget som er defineret i systemet, eller fordi standard-værdierne er fornuftigt valgt.

Dog skal man sikre sig at kommandoen hostname giver det rigtige navn. Ellers skal man rette det i opsætningsfilen.

Her er de 7 dele af opsætningsfilen kort beskrevet:

1. *Main configuration*: Her indstiller man værtnavn, videresendelse mm. Dette er det eneste sted det kan være nødvendigt at rette noget for at få det til at virke.
2. *Transports configuration*: Dette er hvordan en mail bliver leveret til destinationen. Dette er kun selve leveringen som er defineret her.
3. *Directors configuration*: Når en mail skal afleveres lokalt, er det her man definerer hvad der skal ske med den. Man kan for eksempel indsætte et kald af en virusskanner, og det er også her man kan bestemme om .forward filer skal bruges. Herfra kalder man en transport som blev defineret ovenfor.
4. *Routers configuration*: Dette er opsætning af hvordan post skal sendes videre til andre maskiner. Man kan også her skanne for virus, men i modsætning til ovenstående, så forhindrer man her virus i at komme ud, og ikke ind. Også her kaldes en transport.
5. *Retry configuration*: Her indstiller man timeoutværdier, både for levering af post til andre maskiner, men også lokalt, som hvis en virus-skanner er gået død, og det er et krav at al post skal skannes. Man kan for eksempel også lade nogen domæner have en meget lang timeout, mens man kan sætte den til meget lidt for andre.
6. *Rewrite configuration*: Man kan her lave regler til at omskrive emailadresser, men dette skal man normalt holde sig fra at gøre, medmindre man ved hvad man gør, og har en god grund til dette. At gøre dette bliver af nogen betragtet som noget forbudt, og det er kun nødvendigt i sjældne og specielle tilfælde.
7. *Authentication configuration*: Under authentication har man mulighed for at åbne for relay af post fra specifikke maskiner. Det sker ved at man skal identificere sig over for mailserveren. Dette bliver defineret i denne del.

8.3.1. Brug af procmail under exim

Hvis procmail altid skal bruges til at afleve posten til indbakken, så skal dette rettes til i opsætningsfilen.

Følgende skal sættes ind i `exim.conf` under `Transports configuration`:

```
# This transport is used for local mail deliveries with procmail

procmail_pipe:
  driver = pipe
  command = /usr/bin/procmail -d $local_part
  return_path_add
  delivery_date_add
  envelope_to_add
  check_string = "From "
  escape_string =">From "
  user = $local_part
  group = mail
```

Og under directors skal følgende indsættes lige før linjen med `localuser`:

```
procmail:
  driver = localuser
  transport = procmail_pipe
```

8.3.2. Brug af Mailman listmanager sammen med Exim

Når man bruger Exim og Mailman sammen, så behøver man ikke at tænke på at rette forskellige alias-adresser til når man opretter og nedlægger lister. Dette sker automatisk, men kræver at man retter lidt i `exim.conf`.

Hvis man ikke vil dette, så skal man i stedet for huske at rette i filen `/etc/alias` hver gang man opretter eller nedlægger en liste.

I afsnittet Main configuration indsætter man følgende lige før linjen hvor der står "end":

```
## Mailman definitions
MAILMAN_HOME=/var/mailman
MAILMAN_DATA=/var/mailman
MAILMAN_WRAP=MAILMAN_HOME/mail/wrapper
MAILMAN_UID=mail
MAILMAN_GID=mail
```

Disse rettes selvfølgelig til, så de passer med installationen.

Under Transports configuration skal man indsætte:

```
# Mailman definitions

list_transport:
```

```
  driver = pipe
  command = MAILMAN_WRAP post ${lc:$local_part}
  current_directory = MAILMAN_HOME
  home_directory = MAILMAN_HOME
  user = MAILMAN_UID
  group = MAILMAN_GID

list_request_transport:
  driver = pipe
  command = MAILMAN_WRAP mailcmd ${lc:$local_part}
  current_directory = MAILMAN_HOME
  home_directory = MAILMAN_HOME
  user = MAILMAN_UID
  group = MAILMAN_GID

list_admin_transport:
  driver = pipe
  command = MAILMAN_WRAP mailowner ${lc:$local_part}
  current_directory = MAILMAN_HOME
  home_directory = MAILMAN_HOME
  user = MAILMAN_UID
  group = MAILMAN_GID
```

Dette skal igen være før "end"

Og endelig under Directors configuration skal følgende indsættes (det _skal_ stå lige før "end", da det er de sidste regler som skal kontrolleres).

```
# Mailman definitions

# Accept mail to listname-owner and send it to listname-admin
list_owner_director:
  driver = smartuser
  require_files = MAILMAN_DATA/lists/${lc:$local_part}/config.db
  suffix = "-owner"
  new_address = "${lc:$local_part}-admin@${domain}"

# Accept mail to owner-listname and send it to listname-admin
owner_list_director:
  driver = smartuser
  require_files = MAILMAN_DATA/lists/${lc:$local_part}/config.db
  suffix = "owner-"
  new_address = "${lc:$local_part}-admin@${domain}"

list_admin_director:
  driver = smartuser
  suffix = "-admin"
  require_files = MAILMAN_DATA/lists/${lc:$local_part}/config.db
  transport = list_admin_transport

list_request_director:
```

```
  driver = smartuser
  suffix = "-request"
  require_files = MAILMAN_DATA/lists/${lc:$local_part}/config.db
  transport = list_request_transport

list_director:
  driver = smartuser
  require_files = MAILMAN_DATA/lists/${lc:$local_part}/config.db
  transport = list_transport
```

Herefter kan man blot oprette og nedlægge lister uden at bekynre sig om at sørge for at posten bliver sendt til Mailman programmet.

Kapitel 9. Tynde klienter med Linux

Tynde klienter er maskiner der blot virker som en terminal og lader al væsentlig databehandling ske på en server.

Terminalen kan derfor være ganske begrænset udstyret. Fx. kan man benytte ældre pc'er af pentiumtypen med blot en 100 MHz CPU, 16 Mb RAM og uden disk, og få ganske rimelig ydelse ud af den. Man kan også købe nye maskiner med begrænset udstyr, såsom kun lidt RAM og ingen disk, og derved spare penge på hver arbejdsstation.

En anden fordel er at i større miljøer med mange arbejdsstationer, kan en maskinpark med tynde klienter betyde en væsentlig arbejdsbesparelse, nogen undersøgelser viser op mod 50 % besparelse i de totale driftsomkostninger (TCO) i.fh.t. almindelige fritstående Linux-maskiner med netværk, og op mod 75 % besparelse i.fh.t. almindelige Microsoft Windows fritstående maskiner i netværk.

En tredje fordel kan være et bedre miljø, idet den manglende disk og mindre behov for CPU-kraft kan betyde mindre støj og mindre varmeudvikling.

En løsning med tynde klienter kan således være aktuel i både større og mindre virksomheder, som arbejdsstationer på skoler, og hjemme, hvor en gammel pc måske kan gives et helt nyt liv.

Tynde klienter, som det omtales her, kræver et lokalnet, (mindst 10 Mbit, gerne 100 Mbit) for at fungere ordentligt. Tung grafik såsom film eller grafikintensive spil vil køre meget langsomt, mens de fleste andre anvendelser, såsom netlæser, kontorpakker vil køre ganske fortrinligt. Microsoft Windows programmer kræver særlig opsætning og formentlig køb af særligt programmel.

Tynde klienter kræver en server, der skal have den nødvendige kapacitet. Dette behøver ikke at være alverden: folk har været imponeret over ydeevnen på en server med en 650 MHz CPU og 200 Mb RAM, hvilket er en del mindre end de mindste maskiner der sælges i dag på markedet. Der skal også være nødvendig diskkapacitet, selve det system, der omtales nedenfor fylder under 300 Mb udover det basale Linux-driftssystem. Det vigtigste er at have meget RAM, 20-80 Mb RAM per klient, afhængigt af om der køres mange store programmer såsom kontorpakker, og aftagende med antallet af klienter. CPU-hastigheden er ikke så afgørende, 1 GHz er fint. Hertil skal man beregne diskplads til hver enkelt bruger ud fra deres forventede behov.

9.1. LTSP

Tynde klienter har været en mulighed under Linux og UNIX i mange år, via X Windows. En pakke, som gør det nemt at installere, er *LTSP – Linux Terminal Server Project* som findes på http://www.ltsp.org. Hovedmanden bag projektet hedder Jim McQuillan.

Det, LTSP giver dig er først og fremmest et system til at køre X Windows på en pc-klient. På serveren kan man bruge de fleste distributioner, for eksempel Red Hat, Mandrake, Debian, SuSE, og det er altså serverens system, de enkelte klienter vil komme til at køre.

De 4 pakker som skal installeres for at lave en X skærm kan hentes fra www.ltsp.org, hvor de findes i de forskellige pakkeformater der passer til den valgte distribution (.rpm, .deb, .tgz etc.) Pakkerne er:

- ltsp-core – skal installeres først.
- ltsp-kernel – linux-kerne mv.
- ltsp-x-core – kan undværes hvis man kun vil køre kommando-skal.
- ltsp-x-fonts – kan undværes hvis man vil benytte serverens skriftsnit.

Desuden findes der en masse anden information på http://www.ltsp.org om generel installation, postlister for diskussion, udvikling og annoncering, opsætning af lyd, floppy, kørsel af programmer på klientens CPU, støtte for gamle skærme mm. Der findes en dansk side om LTSP på http://www.klid.dk/sw/ltsp/.

9.1.1. LTSP virkemåde

LTSP har en række trin som gennemgås før pc'en kører som X-terminal.

1. Klienten bootes via diskette, eller en bootprom på netkortet
2. Klienten får en IP-adresse fra serverens DHCP
3. Linux-kernen overføres via TFTP
4. initrd RAM-filsystem overføres via TFTP
5. klienten bootes fra RAM-disk og kerne
6. Klienten får parametre til X-terminal fra LTSP
7. Klient monterer diske på server via NFS
8. Klienten kører X
9. Klienten får login fra server

Når man logger ind, kører programmerne så på serveren. Man skal derfor have en konto for hver bruger der skal bruge serveren. Samme bruger kan godt være logget ind flere gange samtidigt.

9.1.2. Opsætning af server

1. LTSP installeres fra de ovennævnte 4 pakker.
2. Opsætningsfiler laves for DHCP, TFTP, NFS, se LTSP-vejledning.
3. LTSP opsætningsfil opsættes i `/opt/ltsp/i386/etc/lst.conf` med info om skærm, tastatur, mus, lyd etc.

4. Der skal opsættes DHCP per klient med IP-nummer ud fra Ethernet MAC-adresse og evt info om X-driver for ældre skærmkort der ikke kan køre XFree86 version 4.
5. `/etc/hosts` skal have en indgang per klient
6. NFS `/etc/exports` skal opsættes

9.1.3. Opsætning af klient

Der skal genereres en diskette, eller en bootprom. Dette kan gøres via netstedet http://www.rom-o-matic.net, der indeholder drivere fra Etherboot-projektet for en masse netkort, til at generere både en opstartsdiskette og kode til en bootprom. Brænding af bootprommer er et kapitel for sig, der findes info om dette på http://www.ltsp.org.

Desuden skal der laves opsætning på serveren for hver klient, som nævnt ovenfor.

9.1.4. Problemløsning i LTSP

Grafikkort virker som regel med udgave 4 af XFree86, men ved ældre grafikkort kan man blive nødt til at finde ud af hvilken X-driver der skal bruges, samt andre parametre for grafikkortet og skærmen. Det kan her være en hjælp at installere grafikkort og skærm på en fritstående Linux-maskine med disk og X installeret. Her kan programmet Xconfigurator være en stor hjælp.

Standardværdierne for netværk, gateway mm er måske ikke rigtige, de skal evt. rettes til i en række filer.

Der kan være problemer hvis man har mere end én DHCP-server på nettet, ved at klienten accepterer data fra den forkerte server. Man kan have flere DHCP-servere på samme net, men det må anbefales at integrere dem til én. Ofte er det dog ikke nemt at gøre, fx hvis man har en fritstående ruter med DHCP-server i forbindelse med et ADSL-abonnement. Fra Etherboot version 5.0.7 vil klienten ikke bruge opstartsdata fra en DHCP-server, hvis denne ikke giver nogen opstartsfil, og dermed ignoreres normalt simple DHCP-servere. Således kan man på driftsikker måde have en ekstra DHCP-server installeret, for eksempel på en Linux-maskine, der kun tager sig af de tynde klienter, via deres explicitte MAC-adresser.

Kapitel 10. Linux som server i Windows-netværk (Samba)

Selv om denne bog handler om Linux, skal man ikke glemme, at der findes andre styresystemer. Et af de mere udbredte er Windows – herunder wfw/95/98/NT/2000, som alle har indbygget muligheden for at fungere i netværk. Hvis man allerede har et lokalnet med både Linux- og Windows-computere, kan man med fordel kigge nærmere på Samba.

Samba er frit programmel, og kan dermed erhverves og bruges helt gratis. I Danmark kan Samba lettest hentes fra http://mirrors.sunsite.dk/sambawww (http://mirrors.sunsite.dk/sambawww/) (d.v.s. dansk mirror; pas på, for sunsite ændrer ofte struktur og dermed url'er!) ellers fra home-site: http://www.samba.org/.

Det som Samba først og fremmest kan tilbyde er at fungere som fil- og printerserver. Derudover kan Samba også fungere som Domain Controller for Windows-klienter.

Med Samba installeret (og ikke mindst sat op) vil serveren kunne ses, hvis man klikker på "andre computere" (på engelsk: "network neighbourhood") eller på samme netværk i Windows 95/98/NT.

Den letteste måde at installere eller opgradere Samba på er at bruge en færdigoversat pakke. Sådanne pakker kan hentes på http://mirrors.sunsite.dk/samba/ (http://mirrors.sunsite.dk/samba/), og de findes til flere forskellige Linux-distributioner, bl.a. Red Hat, Caldera, Debian, Slackware og SuSE. Hvis man holder af at oversætte sin kildetekst selv, kan den også hentes fra sunsite.

Vi vil dog meget anbefale at bruge en færdigoversat pakke, og vi vil i eksemplet gå ud fra, at du har Red Hat installeret. Når du har hentet rpm-pakken, skriver du bare:

```
[root@linus /root]# rpm -ivh samba-*.i386.rpm
```

10.1. Opsætning af Samba-serveren i Linux

Når du har fået installeret Samba-serveren, skal du have den sat op. Dette kan gøres på flere måder. I sædvanlig Unix-stil kan man rette direkte i filen `/etc/smb.conf`. Denne fil kan virke meget skræmmende, da der er op mod 300 parametre, man kan sætte. Det skal du dog af to grunde ikke lade dig gå på af: For det første har de fleste parametre udmærkede forudvalgte værdier, så dem behøver man slet ikke at bekymre sig om. For det andet bliver der installeret en udmærket standard `/etc/smb.conf`, som virker uden de store ændringer.

Denne standard `smb.conf` er desuden meget velkommenteret, hvilket yderligere letter tilpasning. Her er et eksempel på en lille `/etc/smb.conf`-fil, som ud over at sætte Samba op som fil- og printerserver også sætter Samba op som domænekontroller.

```
[global]
   workgroup = hjemme

   printing = bsd
   printcap name = /etc/printcap
   load printers = yes

   log file = /var/log/samba-log.%m

   short preserve case = yes
   preserve case = yes

   lock directory = /var/lock/samba
   locking = yes
   strict locking = yes

   security = user

   socket options = TCP_NODELAY

   domain master = yes
   domain logons = yes
; Hvis du vil have Windows klienter med æ,ø og å
   client code page = 850
   character set = ISO8859-1
   valid chars = æ:Æ ø:Ø å:Å

[homes]
   comment = Home Directories
   read only = no
   create mode = 0750

[deskjet870cxi]
   comment = All Printers
   path = /var/spool/samba
   browseable = yes
   printable = yes
; Set public = yes to allow user 'guest account' to print
   public = no
   writable = no
   create mode = 0700
```

Tip: I Windows 2000 skal man udføre kommandoen **chcp 850** i et terminalvindue. Hvis man ikke gør dette på Windows-maskinerne vil Samba oversætte ø:Ø forkert til o:O i stedet, da Windows 2000 *ikke* selv sætter codepage til 850 men derimod til 457 selvom man sætter sin locale til dansk under installationen.

10.1.1. Tillidsforhold mellem maskinerne

For at dine Windows NT/2000/XP-klienter skal få lov til at logge sig på Samba-serveren, skal Samba kende til dem – ikke brugerne, men de enkelte maskiner. Første gang en maskine tilsluttes genereres et ID-nummer for maskinen som serveren skal huske, til dette formål bruger Samba såkaldte "Machine Trust Accounts" (MTA). Med samba består sådanne af to dele:

- En sambakonto, sædvanligvis gemt i din `smbpasswd`-fil.
- En tilhørende Unix-konto, sædvanligvis gemt i `/etc/passwd`.

Der er flere måder at klare oprettelsen af disse på:

- Man kan oprette begge dele direkte på serveren, før man slutter arbejdsstationerne til domænet. Det åbner en mulighed for at andre kan nå at tilslutte en maskine og "stjæle" kontoen. Hvis man kan overskue netværket eller stoler på de folk der har adgang til bygningen er dette næppe et problem.
- Man kan få samba til at oprette det hele undervejs. Det kræver at root er oprettet som sambabruger (bemærk at root skal være den første bruger i `smbpasswd`-filen), men til daglig kan man jo deaktivere kontoen. Brug for en sikkerheds skyld også forskellige kodeord for root som unix- og sambabruger. Derudover kræver det lidt arbejde at få lavet et script der kan klare oprettelsen af Unixkontoen.
- Man kan oprette Unixkontoen på forhånd, og lade samba oprette sambakontoen undervejs. Det kræver at root er oprettet som sambabruger (bemærk at root skal være den første bruger i `smbpasswd`-filen), men til daglig kan man jo deaktivere kontoen. Brug for en sikkerheds skyld også forskellige kodeord for root som unix- og sambabruger. Til gengæld slipper man for at skriver et script til oprettelse af Unixkontoen og for at oprette sambakontoerne ved håndkraft.

Hvis windowsmaskinen hedder WS1 (hvor det sættes afhænger lidt af windowsversionen, men det er noget i retning af "kontrolpanel/netværk"), skal Unix- og sambakontoen hedde WS1$. Dette dollartegn fortæller serveren at der er tale om en maskinkonto.

Da forfatteren ikke kan se nogen god grund til ikke at oprette unixkontoerne på serveren, vil vi ikke gennemgå metode 2. Lige meget hvilken af de to andre metoder du vil benytte skal du læse næste afsnit, og hvis du vil benytte metode 1 skal du også læse det næste igen.

10.1.1.1. Oprettelse af unixkontoen

Unixkontoen skal oprettes i filen `/etc/passwd`, for eksempel med en kommando i stil med **useradd -u 801 -g 800 -c "Arbejdsstation 1" -d /dev/null -s /bin/false ws1$** ,

Det er selvfølgelig en forudsætning af der findes en gruppe med gid 800 til maskinerne. Du kan oprette gruppen med en kommando i stil med:

```
groupadd –g 800 maskiner
```

10.1.1.2. Oprettelse af sambakontoen

Når vi har tilføjet vores Windows-klient i `/etc/passwd`, skal vi få Samba til at generere vores MTA. Det gøres med kommandoen

```
[root@linus /root]# smbpasswd -a -m WS1
```

Når man har afviklet **smbpasswd**, vil man kunne se følgende linje i filen `/etc/samba/smbpasswd` (hvor noget af linjen er forkortet):

```
WS1$:801:996FE367692....05AA1BE70:[W    ]:LCT-397A3E6E
```

N.B. Det kan være at `smbpasswd` ligger i `/etc/`.

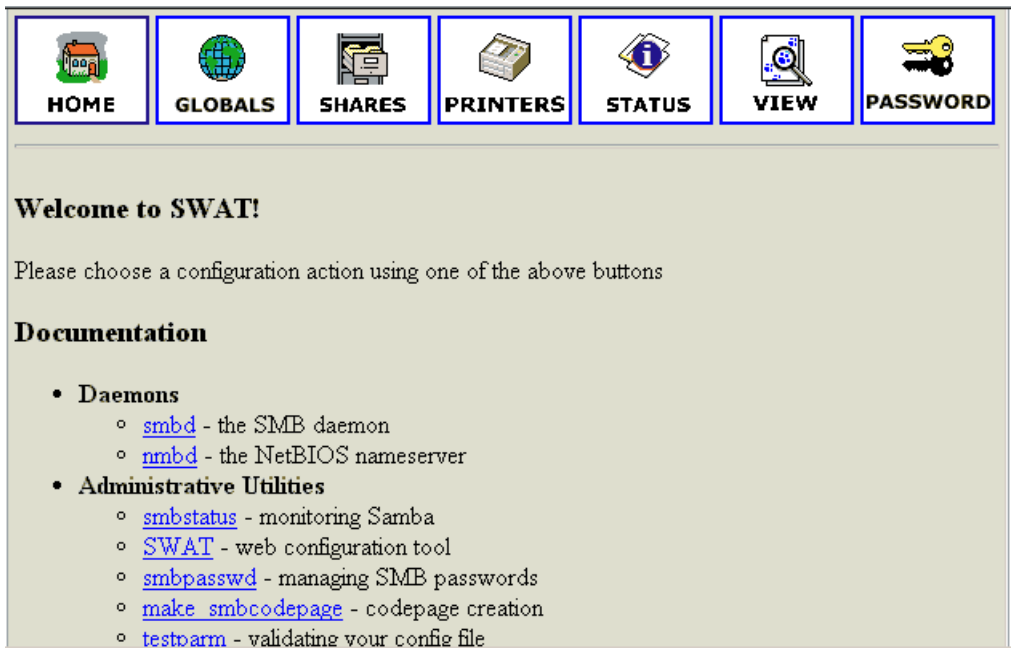
10.1.2. Samba som primær domænekontroller

Som noget nyt i version 2.x.x har man nu mulighed for at sætte Samba op som primær domænekontroller for et Windows-netværk med 95-, 98-, ME- og NT-klienter. Det vil dog føre for vidt at komme ind på det her. Hvis man har mod på at prøve, kan man finde mere information her http://us1.samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html (http://us1.samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html)

10.2. Swat (Samba Web Administration Tool)

Sammen med Samba kommer der også et grafisk opsætningsværktøj, der hedder **swat** (Samba Web Administration Tool). Med dette værktøj er det muligt at klikke sig gennem opsætningen af Samba.

Figur 10-1. Swat

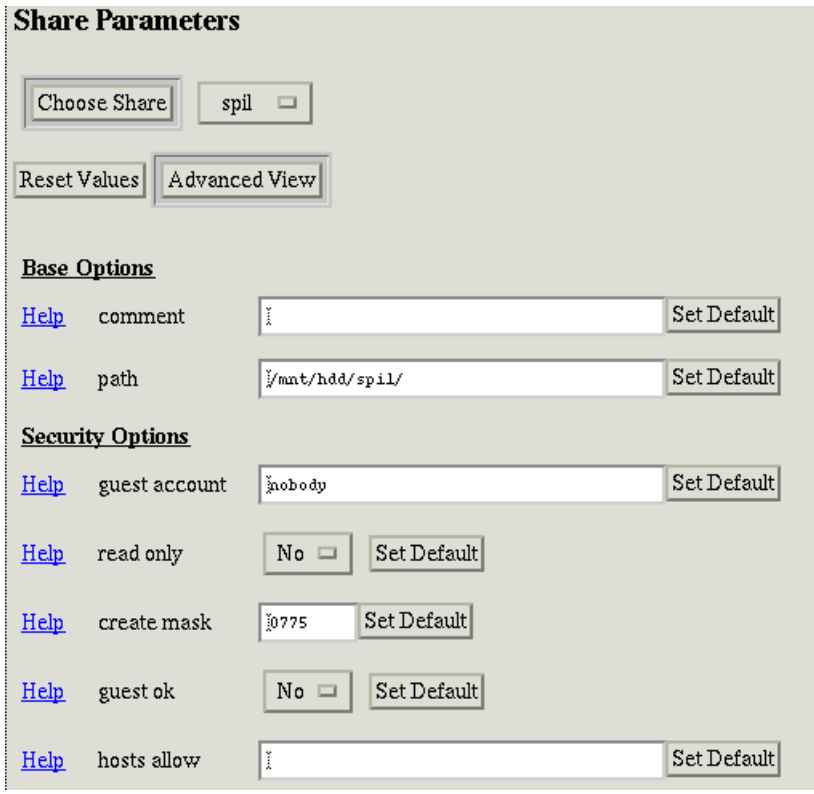


Her ses et eksempel på, hvordan **swat** ser ud, jeg har defineret et "share" (en delt mappe), der hedder spil. I `smb.conf` ser det således ud:

```
[spil]
  create mask =  0775
  read only = no
  path = /mnt/hdd/spil/
```

Hvis man bruger swat, ser det således ud:

Figur 10-2. Swat spil



Her er det kun den øverste del af siden, der er taget med. Det skal bemærkes, at der findes en del andre indstillinger, der også kan sættes med swat. Den største fordel ved at benytte swat er, at man har adgang til hjælp.

Der findes efterhånden en del andre grafiske opsætningsværktøjer. http://mirrors.sunsite.dk/sambawww/ (http://mirrors.sunsite.dk/sambawww/) indeholder en liste over disse.

Inden man kan gå videre til at sætte klienterne i ens netværk op, er der yderligere to andre ting, der skal tages stilling til. Den første beslutning, der skal tages er, hvorvidt man ønsker at benytte krypterede adgangskoder eller ej. Opsætning af dette og en diskussion af fordele og ulemper følger i næste afsnit. Den anden beslutning angår, om man vil benytte TCP/IP eller LAN-manager (Netbios) til navneopslag. Hvis man vælger LAN-manager-løsningen er opsætningen af Samba overstået på Linux-serveren, mens TCP/IP-løsningen kræver, at der installeres DNS på Linux-serveren.

Hvordan man installerer DNS på Linux, kan du læse mere om her. Se Afsnit 6.9.

10.3. Krypterede adgangskoder

SMB bruger en krypteringsteknik, der ligner standard Unix-kryptering. Det er dog kun tilsyneladende, at de to teknikker er ens, Unix-variationen sender nemlig typisk kodeordet i klar tekst over netværket, når man skal logge ind. SMB sender i modsætning hertil en kodet version af adgangskoden.

Hvorvidt man vælger at bruge krypterede adgangskoder eller ej, er op til den enkelte at afgøre, da der er både fordele og ulemper, som listet nedenfor. Bruger man nyere versioner af Microsoft Windows og NT vil vi dog anbefale at bruge kryptering, idet man slipper for at skulle sætte alle sine klienter op. Som det vil fremgå lidt senere, er det uhyre enkelt at få Samba til at benytte krypterede adgangskoder.

Fordele ved krypterede adgangskoder:

- Sikkerhed. Adgangskoder bliver ikke sendt over netværket i klar tekst.
- Windows 95 med OSR2, Windows 98 og NT 4.0 SP4 vil som standard kun tale med en server, der anvender krypterede adgangskoder.

Ulemper ved krypterede adgangskoder:

- Der skal vedligeholdes endnu en passwd-fil.
- Hvis du allerede bruger services, der sender adgangskoder i klar tekst (f.eks telnet og ftp), gør det ikke den store forskel, om SMB også gør det.

Windows 98 og Windows NT 4 (SP3 og senere) kræver som standard brug af krypterede adgangskoder. Det betyder, at du har to valg:

- Du sætter Samba til at bruge krypterede adgangskoder.
- Du ændrer Windows 98/95/NT registry, så de ikke kræver krypterede adgangskoder.

10.3.1. Opsætning af Samba

I modsætning til tidligere er Samba i dag født til at kunne håndtere krypterede adgangskoder. Det eneste man skal gøre, er at tilføje følgende linje til [global]-sektionen i `/etc/smb.conf`

```
encrypt password = yes
```

Hvis du gerne vil have flyttet dine Unix-adgangskoder over til Samba, skal du lave en ny passwd-fil til brug for Samba alene. Der findes et program, som kan generere en Samba-adgangskodefil (`/etc/samba/smbpasswd`) ud fra den eksisterende Unix-adgangskodefil (`/etc/passwd`):

```
[root@linus /root]#  mksmbpasswd.sh < /etc/passwd > /etc/samba/smbpasswd
```

I SuSE 6.3 er `smbpasswd.sh` placeret i kataloget `/usr/sbin`, og katalogplaceringen skal med i kommandoen. Desuden har filen i den distribution ikke rettigheder til at kunne køres, hvilke den således skal tildeles, før kommandoen udføres.

Kommandoen opretter filen til de krypterede adgangskoder, men den opretter ikke selve de krypterede adgangskoder. Uanset om du vælger, at brugerne skal have samme Samba-adgangskode, som de er tildelt som brugere i dit Linux-system, skal du alligevel bagefter tildele dem krypterede Samba-adgangskoder ved hjælp af kommandoen **smbpasswd**.

Selv hvis du med linjen `null passwords = yes` i `/etc/smb.conf` under global-sektionen har tilladt, at Samba-adgangskoden er blank, skal du alligevel oprette en "blank" adgangskode med kommandoen **smbpasswd**.

Du bør bruge samba-1.9.18p10 eller senere (2.0.7 er pt. den seneste), da kryptering er slået til som standard – dette gælder i hvert fald for Red Hat. Bruger du andre distributioner, bør du tjekke, hvordan Samba er sat op. Hvis du gerne vil oversætte dit eget programmel, skal du anvende linkerflaget `-lcrypt` til LIBSM.

10.3.2. Opsætning af Windows 98/NT

Som udgangspunkt er Windows sat op til at sende krypterede adgangskoder, så hvis man ikke ønsker kryptering, skal dette fortælles til Windows, da man ellers ikke vil kunne logge sig på.

På Windows 95/98-maskiner skal følgende skrives i registreringsdatabasen for at få Windows til at undlade at sende krypterede adgangskoder:

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"EnablePlainTextPassword"=dword:00000001
```

Tilsvarende for Windows NT:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\Parameters]
"EnablePlainTextPassword"=dword:00000001
```

Hvis man ikke har mod på at ændre i registreringsdatabasen selv, kan man kopiere en af følgende filer til sin Windows-maskine og afvikle dem fra **stifinder**:

- `Win95_PlainPassword.reg`
- `Win98_PlainPassword.reg`
- `NT4_PlainPassword.reg`
- `Win2000_PlainPassword.reg`

Hvis man afvikler dem, vil ovenstående ændringer automatisk blive tilføjet i registreringsdatabasen.

En meget udførlig beskrivelse af brugen findes i filen `ENCRYPTION.txt`, som følger med Samba-distributionen.

10.4. Opsætning af klienterne

Nu har du sat Samba op på Linux-serveren samt besluttet dig for, om du vil benytte krypterede adgangskoder eller ej, så det eneste du mangler, er at få Windows-klienterne til at tilslutte sig domænet. I det følgende afsnit vil vi anvise en løsning på dette. Der knytter sig følgende forudsætninger til det efterfølgende eksempel:

Maskiner i vores netværk:

- test.domain.net; Linux-serveren
- router.domain.net; Router til internettet
- WS1.domain.net; Windows-klient

Indhold af filen `/etc/hosts`:

```
127.0.0.1 localhost.localdomain localhost
192.168.1.1 test.domain.net  test
172.10.10.1 router.domain.net router
```

Indhold af filen `/etc/smb.conf`:

```
[global]
   workgroup = testdomain

   printing = bsd
   printcap name = /etc/printcap
   load printers = yes

   log file = /var/log/samba-log.%m

   short preserve case = yes
   preserve case = yes

   lock directory = /var/lock/samba
   locking = yes
   strict locking = yes

   security = user
```

```
   socket options = TCP_NODELAY

   domain master = yes
   domain logons = yes

[homes]
   comment = Home Directories
   read only = no
   create mode = 0750

[deskjet870cxi]
   comment = All Printers
   path = /var/spool/samba
   browseable = yes
   printable = yes
; Set public = yes to allow user 'guest account' to print
   public = no
   writable = no
   create mode = 0700
```

For at kunne få forbindelse til Samba-serveren, kræves der en fungerende løsning på navneopslag. Det kan løses på 2 måder:

- Hosts og Lmhosts – netbios-way (har i øvrigt suffixet .sam): I disse filer skal samba-serveren være nævnt. Når du har foretaget ændringerne i filerne, skal deres navne ændres til Hosts og Lmhosts – efternavnene `.sam` skal altså udelades. Se eksempel på disse filer senere.
- Sæt Linux-serveren til at køre named – TCP/IP-way, og fortæl Windows at den skal bruge Linux-serveren som navneserver (DNS). Åbn indstillinger->kontrolpanel->netværk. Åbn egenskaber for TCP/IP. Under fanebladet "Gateway" tilføjes 192.168.1.1. Under fanebladet "DNS-opsætning" aktiveres DNS, i "Vært" skrives maskinens navn WS1 (det skal være det samme navn som blev angivet som "Computernavn" i fanebladet "identifikation": Indstillinger->kontrolpanel->netværk). "Domæne" sættes til testdomain. Feltet "Rækkefølge" til søgning efter navneserver tilføjes 192.168.1.1 (hvis den står som den første, går det hurtigere).

DOMÆNENAVN: Workgroup/domain-navn må ikke være sammenfaldende med host-navnet. Hvis samba, som i vores eksempel, kører på hosten test.domain.net, og har følgende stående i filen: `/etc/hosts`:

```
127.0.0.1 localhost.localdomain localhost
192.168.1.1 test.domain.net test
172.10.10.1 router.domain.net router
```

er følgende navne ikke tilladte for workgroup/domain:

- localhost.localdomain

- test.domain.net
- router.domain.net

Disse vil formentlig være ulovlige i sig selv, da de indeholder et forbudt tegn.

- localhost
- test
- router

I dette afsnit kommer så vores eksempel på, hvordan filerne `Hosts` og `Lmhosts` skal se ud.

10.4.1. Hosts

For Windows 95/98 gemmes filen som `c:\windows\Hosts`.

For Windows NT gemmes filen som `c:\winnt\system32\drivers\etc\Hosts`.

```
# Copyright (c) 1998 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP stack for Windows98
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
# 102.54.94.97 rhino.acme.com # source server
# 38.25.63.10 x.acme.com # x client host

127.0.0.1 localhost
192.168.1.1 test
172.10.10.1 router
```

10.4.2. Lmhosts

For Windows 95/98 gemmes filen som `c:\windows\Lmhosts`.

For Windows NT gemmes filen som `c:\winnt\system32\drivers\etc\Lmhosts`.

```
# Copyright (c) 1998 Microsoft Corp.
#
# This is a sample LMHOSTS file used by the Microsoft Wins Client (NetBios
# over TCP/IP) stack for Windows98
#
# This file contains the mappings of IP addresses to NT computernames
# (NetBIOS) names. Each entry should be kept on an individual line.
# The IP address should be placed in the first column followed by the
# corresponding computername. The address and the comptername
# should be separated by at least one space or tab. The "#" character
# is generally used to denote the start of a comment (see the exceptions
# below).
#
# This file is compatible with Microsoft LAN Manager 2.x TCP/IP lmhosts
# files and offers the following extensions:
#
# #PRE
# #DOM:<domain>
# #INCLUDE <filename>
# #BEGIN_ALTERNATE
# #END_ALTERNATE
# \0xnn (non-printing character support)
#
# Following any entry in the file with the characters "#PRE" will cause
# the entry to be preloaded into the name cache. By default, entries are
# not preloaded, but are parsed only after dynamic name resolution fails.
#
# Following an entry with the "#DOM:<domain>" tag will associate the
# entry with the domain specified by <domain>. This affects how the
# browser and logon services behave in TCP/IP environments. To preload
# the host name associated with #DOM entry, it is necessary to also add a
# #PRE to the line. The <domain> is always preloaded although it will not
# be shown when the name cache is viewed.
#
# Specifying "#INCLUDE <filename>" will force the RFC NetBIOS (NBT)
# software to seek the specified <filename> and parse it as if it were
# local. <filename> is generally a UNC-based name, allowing a
# centralized lmhosts file to be maintained on a server.
# It is ALWAYS necessary to provide a mapping for the IP address of the
# server prior to the #INCLUDE. This mapping must use the #PRE directive.
# In addtion the share "public" in the example below must be in the
# LanManServer list of "NullSessionShares" in order for client machines to
# be able to read the lmhosts file successfully. This key is under
#\machine\system\currentcontrolset\services\lanmanserver\
#  parameters\nullsessionshares
```

```
# in the registry. Simply add "public" to the list found there.
#
# The #BEGIN_ and #END_ALTERNATE keywords allow multiple #INCLUDE
# statements to be grouped together. Any single successful include
# will cause the group to succeed.
#
# Finally, non-printing characters can be embedded in mappings by
# first surrounding the NetBIOS name in quotations, then using the
# \0xnn notation to specify a hex value for a non-printing character.
#
# The following example illustrates all of these extensions:
#
# 102.54.94.97 rhino #PRE #DOM:networking #net group's DC
# 102.54.94.102 "appname \0x14" #special app server
# 102.54.94.123 popular #PRE #source server
# 102.54.94.117 localsrv #PRE #needed for theinclude
#
# #BEGIN_ALTERNATE
# #INCLUDE \\localsrv\public\lmhosts <file://\\localsrv\public\lmhosts>
# #INCLUDE \\rhino\public\lmhosts <file://\\rhino\public\lmhosts>
# #END_ALTERNATE
#
# In the above example, the "appname" server contains a special
# character in its name, the "popular" and "localsrv" server names are
# preloaded, and the "rhino" server name is specified so it can be used
# to later #INCLUDE a centrally maintained lmhosts file if the "localsrv"
# system is unavailable.
#
# Note that the whole file is parsed including comments on each lookup,
# so keeping the number of comments to a minimum will improve performance.
# Therefore it is not advisable to simply add lmhosts file entries onto the
# end of this file.

192.168.1.1 test #PRE #DOM:testdomain #net group's DC
```

10.5. Videre med Samba

Ud over man-siderne og dokumentationen, som installeres i `/usr/doc`, findes der en række bøger om Samba. Den første bog er ikke blændende godt redigeret hele vejen igennem, men er ualmindelig rar at have stående som opslagsværk, hvis du administrerer en server med Samba. Sidst i bogen er der et rigtig godt kapitel om fejlfinding.

John D. Blair: *Samba: Integrating UNIX and Windows*, Specialized Systems Consultants, Inc., ISBN: 1-57831-006-7, 290 sider + cd-rom.

Der findes også en glimrende online-bog fra O'Reilly, som hedder "Using Samba" skrevet af Robert Eckstein, David Collier-Brown og Peter Kelly.

En anden virkelig god bog om Samba er efter sigende Gerald Carter & Richard Sharpe, *Sams Teach Yourself Samba in 24hrs*, Sams Publishing 1999. ISBN 0-672-31609-9.

En god og kort artikel om Samba som Domain server er at finde på http://www.freeos.com/articles/3842/. Se også http://www.linux-mag.com/2002-02/samba_01.html.

Kapitel 11. Pakke-formater

De fleste har nok lidt svært ved, at forstå og bruge Linux' tekstopsætningsfiler når de lige er startet med at bruge Linux. Dette kan - som oftest - volde problemer, da mange mennesker jo netop installerer Linux for at bruge de medfølgende funktioner til fil- og printer-deling; eksempelvis på et kollegie, eller i en virksomhed.

Det kan imidlertid godt være svært, da funktionerne til brugerstyring, netværksopsætning, indstilling af tiden osv. ofte ikke er lige nemme at gå til. Men fortvivl ikke - der er flere grafiske programmer, som kan hjælpe hvis du ikke vil gøre det tekstbaseret.

I dette kapitel ser vi først nærmere på de pakkeformater, som mange programmer distribueres i.

11.1. Pakker af data

Linux har glimrende muligheder for at pakke programmer og datafiler sammen svarende til zip-formatet kendt fra Windows-verdenen. Faktisk kan du med **unzip** udpakke zip-filer. Du skal blot installere unzip-pakken først - se Afsnit 11.1.3. Tilsvarende kan du have brug for at kunne udpakke en selvudpakkende zip-fil (som kommer som en exe-fil fra Windows). Til det kan du bruge **unzipsfx**.

Du kan teoretisk set stadig komme ud for programpakker i Z-format. De skal udpakkes med **uncompress** (og de laves med **compress**). På grund af bedre pakke-effektivitet anvendes kombinationen af **tar** og **gzip** eller **bzip2**. Dette er nærmere omtalt i Afsnit 11.1.1. Et andet alternativ som efterhånden er en stærk standard for udbredelse af Linux-programmer i binær form er RPM, som er omtalt i Afsnit 11.1.3.

Det kan nævnes, at **tar** historisk set er et program, man bruger til at lime datafiler sammen, så de kan lægges ud på bånd - deraf navnet tar som står for Tape ARchive.

11.1.1. Pakning af data med tar og gzip/bzip2

I Linux/Unix-verdenen anvendes oftest tar-formatet til at samle mange filer i én pakkefil. Denne er ikke komprimeret, men alene en samling af filerne i én pakkefil. Har du filerne `oversigt.txt`, `tekstA.txt` op til `tekstD.txt`, som du vil pakke sammen i filen `tekster.tar`, så gøres dette med:

```
[tyge@hven ~]$ tar cvf tekster.tar oversigt.txt tekst[A-D].txt
oversigt.txt
tekstA.txt
tekstB.txt
tekstC.txt
tekstD.txt
[tyge@hven ~]$ ls -l oversigt.txt tekst[A-Z].txt tekster.tar
-rw-r--r-- 1 tyge tyge  400   mar  3 17:41 oversigt.txt
```

```
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstA.txt
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstB.txt
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstC.txt
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstD.txt
-rw-r--r-- 1 tyge tyge  10240 mar  3 17:41 tekster.tar
```

Du kan således give alle de filnavne, der skal pakkes som det sidste argument, og giver du et katalognavn i listen vil dette katalog - med alle underkataloger - også blive pakket med. Som det ses af eksemplet vil programmet **tar** vise filnavne på de filer, der pakkes. Ønsker du ikke at se dette, skal du blot udelade option `v`, dvs. brug **tar cf TARFILNAVN.tar LISTE_AF_FILER** i stedet.

Hvis de filer du ville gemme er almindelige tekstfiler vil du i `tekster.tar` direkte kunne læse dig til de enkelte filer, som skulle lægges i pakken. Derfor vil tar-filen fylde lidt mere end summen af de filer, der skulle gemmes. For små filer vil der være en del ekstra fyld.

Ofte ønsker man nu at komprimere tar-filen, så den fylder mindre. Oftest vinder man en faktor 2, men både langt mere eller noget mindre er set i praksis. Der er flere muligheder for at komprimere, hvor **gzip** er langt det mest udbredte.

```
[tyge@hven ~]$ gzip tekster.tar
tekster.tar:                93.3% -- replaced with tekster.tar.gz
[tyge@hven ~]$ ls -l tekster.tar.gz
-rw-r--r-- 1 tyge tyge  706 mar  3 17:41 tekster.tar.gz
```

Programmet **gzip** fortæller at filen `tekster.tar.gz` er komprimeret med 93.3% og vi ser at filen nu kun fylder 706 bytes. Ofte vil man se at gzippede tar-filer ikke hedder `.tar.gz` men den kortere form `.tgz`, og de to fil-endelser betyder det samme.

Bemærk, at originalfilen `tekster.tar` nu er væk. Kun den pakkede fil er tilbage efter **gzip**.

Ofte laver man ikke tar-filen, men springer direkte til `.tgz`-formatet ved at danne pakkefilen med **z** (for zip) tilføjet optionerne.

```
[tyge@hven ~]$ tar cvzf tekster.tgz oversigt.txt tekst[A-D].txt
oversigt.txt
tekstA.txt
tekstB.txt
tekstC.txt
tekstD.txt
93.3%
[tyge@hven ~]$ ls -l tekster.tgz
-rw-r--r-- 1 tyge tyge  706 mar  3 17:41 tekster.tgz
```

Vi skal nu se på hvordan du udpakker dine filer igen fra pakkefilen. Først ser vi hvad der er inde i pakken (t-option), og når vi er sikre på hvad der vil ske, bliver filerne udpakket (x-option). Antag at vi har placeret pakkefilen i et katalog, hvor der ikke er andre filer end pakkefilen.

```
[tyge@hven ~]$ tar tzvf tekster.tgz
-rw-r--r-- 1 tyge tyge  400   mar  3 17:41 oversigt.txt
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstA.txt
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstB.txt
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstC.txt
-rw-r--r-- 1 tyge tyge  1024  mar  3 17:41 tekstD.txt
[tyge@hven ~]$ tar xzvf tekster.tgz
oversigt.txt
tekstA.txt
tekstB.txt
tekstC.txt
tekstD.txt
```

Igen kan v-option udelades, hvis du ikke vil have helt så meget information præsenteret på skærmen. Det skal nævnes, at **tar** har en stor mængde options, som kan læres lidt efter lidt. Brug **man tar** til at komme videre. En option du måske allerede nu kan lære er **-C DIR**, som bruges til at pakke pakkefilen ud, svarende til at du står i kataloget `DIR` og pakker ud.

Du kan også komme ud for at folk bruger **bzip2** til at komprimere data med i stedet for **gzip**. En fil der ender på `.bz2` kan du udpakke med **bunzip FILNAVN.bz2**. Er det en tar-fil som er pakket med **bzip2**, kan du med en nyere version af **tar** udpakke direkte med **tar xjvf FILNAVN.tar.bz2**, mens man med en ældre version af **tar** må tage den i to omgange: **bunzip2 FILNAVN.tar.bz2** og derefter **tar xvf FILNAVN.tar**.

11.1.2. Flytning af data til større disk

Tip: Lad os lige tage et avanceret eksempel. Det kan jo ske, at du løber tør for plads på din harddisk og køber en ekstra disk. Du beslutter nu, at du vil flytte `/home` over på den nye disk (`/dev/hdb1`) og vil køre videre med Linux-systemet på den gamle disk. Lad os antage, at du har partitioneret `/dev/hdb` med **fdisk /dev/hdb** og kørt **/sbin/mke2fs /dev/hdb1** for at formatere disken.

```
[root@linus /root]# mkdir /mnt/nydisk
[root@linus /root]# mount /dev/hdb1 /mnt/nydisk
[root@linus /root]# cd /home
[root@linus /root]# tar cvf - . | (cd /mnt/nydisk; tar xpf -)
```

De to første linjer er blot forberedelse, så vi har den nye disk til rådighed på `/mnt/nydisk`. Dernæst stiller vi os der hvor sikkerhedskopieringen skal startes fra (gør at dette er "."). I den sidste kommando laver vi en sikkerhedskopi til "-" dvs. til stdout, som kanaliseres videre til en modtage-tar-kommando, der startes under `/mnt/nydisk`. Denne gang tilføjes p-option for at de udpakkede filer får samme rettigheder og ejerforhold som de oprindelige.

Det kan bemærkes at den sidste linje af ovenstående kommandolinje kunne også være skrevet om følgende, hvor stdin/stdout er implicit.

```
[root@linus /root]# tar cv . | (cd /mnt/nydisk; tar xp)
```

Først når du *har* tjekket at alle dine filer er genskabt under `/mnt/nydisk` kan du slette dine filer og skabe kontakt med den nye disk med `/home`-filerne. Redigér `/etc/fstab` så den nye disk er med:

```
/dev/hdb1   /home    ext2     defaults         1 1
```

Bruger du en anden partition end `/dev/hdb1` til den nye disk, så skal du naturligvis erstatte dette i `/etc/fstab`

```
[root@linus /root]# mv /home /oldhome
[root@linus /root]# umount /mnt/nydisk
[root@linus /root]# mount /home
```

Hvis du kan se at alt fungerer, kan du slette den gamle `/home` som nu endte på `/oldhome`. Brug **rm -rf /oldhome**, når du er helt sikker.

Det der kan gå galt her er, at man har links, som ikke peger på de rigtige steder efter flytningen. Med `/home` er dette normalt ikke tilfældet, men er det `/usr` du flytter, så skal du være meget forsigtig og altid lave disse operationer i enkeltbrugermodus (dvs. brug **init 1**).

En guide til at opgrade harddiske kan findes på http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/Hard-Disk-Upgrade.html.

11.1.3. Installation af RPM-programpakker

Gennem mange år har Unix-systemadministratoren skullet hente nye programmer hjem med kildetekst og derefter oversætte og installere. Det gik oftest nemt, men kunne fra tid til anden være meget svært. Hvad værre var, man kunne typisk ikke afinstallere programmer uden at have 100 pct. styr på installationsfasen og styr på at andre programmer ikke anvendte samme biblioteker. Alt i alt ganske problematisk og uden reel mulighed for at kunne opgradere systemet løbende.

Firmaet Red Hat og andre er i de senere år gået over til at oversætte programmer én gang for alle og så distribuere programpakker, som indeholder binær kode, biblioteker, manualsider og andet. Fordelen ved disse programpakker er, at systemet kan holde styr på præcis hvad der er installeret. Alle pakker undersøges for afhængigheder og du kan afinstallere pakker igen, hvis andre pakker ikke påvirkes af det. Red Hats format for programpakker kaldes RPM, som er en forkortelse for "Red Hat Package Management", hvilket vil sige Red Hats pakkehåndtering.

Tabel 11-1. Miniguide i at anvende rpm-programmet.

Kommando	Forklaring
`rpm -i pakke_version.rpm`	Installér pakken

Kommando	Forklaring
rpm -ivh pakke_version.rpm	Installér pakken med status vist. Tilføj option `--nodeps` hvis pakken skal installere uden at tjekke for manglende støttepakker.
rpm -Uvh pakke_version.rpm	Opgrader pakken med status vist. Hvis pakken ikke er installeret i forvejen vil den blive installeret. Tilføj option `--nodeps` hvis pakken skal installere uden at tjekke for manglende støttepakker.
rpm -Fvh pakke_version.rpm	Opgrader pakken med status vist. Hvis pakken ikke er installeret i forvejen vil den *ikke* blive installeret. Tilføj option `--nodeps` hvis pakken skal installere uden at tjekke for manglende støttepakker.
rpm -qip pakke_version.rpm	Viser information om den pakke, som kan installeres.
rpm -e pakke	Afinstallér den installerede pakke.
rpm -q PAKKE	Viser version af den installerede pakke.
rpm -qi PAKKE	Viser information om den installerede pakke.
rpm -qf filnavn	Viser hvilken RPM-pakke filnavnet kom fra.
rpm -ql PAKKE	Viser hvilke filer der blev installeret med RPM-pakken.
rpm -qlp PAKKE.rpm	Viser hvilke filer der er med i RPM-pakken.
rpm -qa	Viser alle installerede RPM-pakker.
rpm -ql PAKKE	Viser alle filnavne indeholdt i pakken.
rpm -Va	Tjek alle RPM-installerede filer for ændringer siden installation af pakkerne.

Lad os illustrere styrken i RPM-programmet: Du sidder på en maskine som systemadministrator og opdaterer alle Linux-maskinerne i dit netværk med én kommando - og endda med krypteret transmission, så ingen kan lytte med. Det er simpelthen administratorens drøm af et system. Nemt, sikkert og stabilt. Hvis alle maskiner, der skal opgraderes, er nævnt i filen `/etc/serverlist`, vil nedenstående magiske linje (ja, det er avanceret - men sejt, ikke sandt?) opgradere vim-pakken, som hentes fra ftp-serveren `SERVER`. Alle maskiner bliver lige opgraderet i ét hug!

```
[root@linux /root]# cat /etc/serverlist | \
    xargs -l1 -i= -r ssh = rpm -U ftp://SERVER/vim-4.6-4.i386.rpm
```

Tip: Under installationen eller opgraderingen af en RPM-pakke sker det at den klager over at man mangler nogle filer. Det skyldes at RPM-pakken, som man er ved at installere, afhænger af andre RPM-pakker for at fungere, dvs. RPM-pakker som man enten mangler eller som man har, men af for gammel version. Et godt trick er at gå ind på http://www.rpmfind.net/linux/RPM/. Her kan du søge efter

RPM-pakker, men også efter filnavne og derved oftest finde hvilke RPM-pakker du mangler.

11.1.4. RPM for programmøren

RPM-filer opstår selvfølgelig ikke "af sig selv". Der skal bruges et værktøj til at lave dem - nemlig det samme **rpm**-program, som man bruger til alle de andre RPM-funktioner.

rpm-programmet har en *build*-funktion, som bruges når man skal lave sine egne rpm-filer. En 'build' består af flere faser

- *prep* - udpakning af kildetekstenfiler
- *setup* - installation af patches og opsætning af softwaren
- *build* - oversættelse (bygning) af softwaren
- *install* - installation af program-, dokumentations- og opsætningsfiler i de korrekte kataloger.
- *packaging* - sammenpakning af kildetekst- og binære RPM-filer
- *clean* - oprydning

Hver af disse faser kan kontrolleres fuldt ud gennem en opsætningsfil, som rpm-programmet bruger til hele build-processen. Denne opsætningsfil kaldes en *spec*-fil, og er den vigtigste fil at forstå, når man skal lave sine rpm-pakker.

11.1.4.1. Katalogstruktur

Normalt bruger rpm-programmet en katalogstruktur under **/usr/src/redhat** til at generere rpm-filer. Man kan bruge et andet toplevel-katalog, men /usr/src/redhat er default.

Herunder er der et antal underkataloger:

- **SOURCES** - indeholder alle de kildefiler og patches, der bruges til build af en rpm-pakke
- **SPECS** - her ligger spec-filen
- **BUILD** - er et 'arbejdskatalog' hvor kildefilerne pakkes ud, og selve oversættelsen finder sted
- **RPMS** - er der hvor de færdige binære rpm-filer placeres. Dette katalog har mindst to underkataloger: **i386** til rpm-filer, der er rettet mod i386-platformen og **noarch** til rpm-filer, der ikke er platformsafhængige.
- **SRPMS** - er der hvor de færdige source rpm-filer placeres.

Inden man går i gang med at lave en rpm-fil, skal man sørge for at have source-filerne lagt ned i kataloget SOURCES og en spec-fil i kataloget SPECS. Det kan man f.eks. gøre ved at installere en source rpm-fil.

11.1.4.2. En spec-fil til ssh (Secure Shell)

Der har været en del skriveri om **ssh** på sslug mailing-listen, og for nylig er der skrevet en god webside om hvordan man bruger dette program. ssh findes ikke i rpm-format, så det kunne være nyttigt at bruge det som eksempel på, hvordan man laver en spec-fil og bruger den til at lave rpm-pakker.

Hele spec-filen `ssh.spec` ligger i underkataloget `rpm` til bogens eksempler på www.linuxbog.dk (http://www.linuxbog.dk/), men lad os kigge på den i nogle mindre bidder.

```
Summary: Secure Shell - secure network communications
Name: ssh
Version: 1.2.25
Release: 2
Copyright: GPL
Group: Utilities/Networking
Source: ftp://sunsite.auc.dk/pub/security/ssh/ssh-1.2.25.tar.gz
Patch: ssh-1.2.25-Makefile.patch
URL: http://www.cs.hut.fi/ssh/
BuildRoot: /tmp/ssh-build

%description
Secure Shell enables you to communicate securely across on unsafe
network such as the Internet. Communication is transparently
encrypted, and thus secured against eavesdropping. The package
includes drop-in replacements for telnet, rlogin, rsh, rcp and
other standard networking tools.
```

Starten på sådan en spec-fil er ret standardiseret. Der er en stribe *tags* som man skal angive - det er bl.a. al den information om pakken, som **rpm -i** kommandoen skal levere.

Rækkefølgen af de enkelte tags er ligegyldig, bare de er der.

- *Name:* Pakkens navn
- *Summary:* En kort beskrivelse (under en linje) af, hvad pakken indeholder
- *Version:* Programmets versionsnummer.
- *Release:* RPM-pakkens 'build-nummer' - det er uafhængigt af selve programmets versionsnummer og bruges blot til at holde rede på forskellige versioner af denne rpm-pakke med samme udgave af programmet i.
- *Copyright:* Hvilken copyright programmet er underlagt. Det er naturligvis vigtigt at skrive, hvis programmet ikke kan distribueres frit.
- *URL:* Hvis der er en webside om pakken, kan man skrive dens URL her.
- *Group:* Red Hat bruger dette felt til at gruppere pakkerne - f.eks. "Games", "Development tools" o.lign.
- *Description:* Det er den lange beskrivelse af softwaren, og strækker sig derfor over flere linjer - faktisk så mange man har lyst til, indtil der kommer en linje som begynder med et procenttegn.

Source-tag'en er vigtig - her fortæller man, hvor man har fundet den oprindelige source-fil til programmet, gerne med en URL. Det er en dårlig idé kun at skrive filnavnet - så ved man ikke, hvor man skal gå hen for at finde eventuelle nyere versioner af programmet. Bemærk: Filnavnet der står her, *skal* matche det filnavn, der ligger i `SOURCES`-kataloget. Så med eksemplet, skal `SOURCES`-kataloget indeholde filen `ssh-1.2.25.tar.gz`.

Patch-tag'en er også vigtig - det er ændringer til de oprindelige sources (i form af patches) som skal installeres inden programmet kan oversættes. Disse skal også ligge i `SOURCES`-kataloget. I eksemplet er der kun én patch; hvis man har flere, skal de nummereres, og så lister man dem en ad gangen (en patch uden nummer er automatisk nummer nul):

```
Patch: foo-Makefile.patch
Patch1: foo-glibc.patch
Patch2: foo-wtmpfix.patch
```

BuildRoot-tag'en er ikke krævet, men anbefales. Det er *ikke* det katalog, hvori programmerne bliver oversat, men derimod det toplevel-katalog, hvori de oversatte programmer og øvrige filer placeres i installationsfasen, inden de pakkes sammen i rpm-filerne. Hvis ikke man bruger en BuildRoot-tag er default at filerne placeres i de "rigtige" kataloger - `/usr/bin`, `/etc`, `/usr/lib` osv. Det gør måske ikke noget, men hvis der er opsætningsfiler involveret er det en rigtig god idé at bruge BuildRoot-tag'en; ellers kan man nemt risikere at komme til at overskrive sine egne omhyggeligt tilpassede opsætningsfiler undervejs i build-processen, eller (måske endnu værre) komme til at inkludere sine egne opsætningsfiler i rpm-pakken, som hentes af tusindvis af brugere. (F.eks. havde jeg en overgang password-filerne til min ISP liggende på ftp.sslug.dk, i en ppp-RPM-pakke). BuildRoot-tag'en er dog ofte lidt besværlig, da man næsten altid skal rette i programmets 'Makefile' eller installationsscript, for at de kan finde ud af at installere pakkerne et andet sted end det normale. Men det er ulejligheden værd.

11.1.4.3. prep-fasen

prep (prepare) fasen består almindeligvis blot i udpakning af source-filerne til et underkatalog under BUILD. **rpm**'s `build`-funktion kan automatisk håndtere tar-arkiver (også komprimerede), så normalt kan man blot nøjes med at skrive:

```
%prep
```

i spec-filen, og så går det af sig selv.

11.1.4.4. setup-fasen

setup-fasen er der hvor der begynder at ske noget.

```
%setup
%patch -p1
./configure --prefix=/usr
```

Denne fase skal gøre programmet klart til at blive oversat. Typisk skal der først installeres de forskellige *patches* som man har listet, og dernæst skal programmet måske konfigureres.

De linjer der står efter `%setup` er i virkeligheden almindelige **/bin/sh**-kommandoer. Dog er der defineret nogle makroer, som **rpm**'s `build`-funktion håndterer, inden de overgives til kommandofortolkeren - en af makroerne er `%patch`. Denne makro kører **patch**-programmet, med patch nummer nul som input. For at apply'e patch nummer 1 skriver man `%patch1`, patch nummer 2 er `%patch2` osv. Rækkefølgen af de forskellige patches er ligegyldig, så længe patch-programmet kan finde ud af det. Man kan også sagtens springe patches over, f.eks. hvis de ikke er relevante for den platform man builder til.

Inden kommandoerne i setup-fasen afvikles, sættes default-kataloget til `/usr/src/redhat/BUILD/`*pakkenavn-version*, dvs. `/usr/src/redhat/BUILD/ssh-1.2.25` i eksemplet. Hvis source-arkivet ikke er blevet pakket ud til det katalognavn (f.eks. bruger nogle source-arkiver en underscore i stedet for en bindestreg), så angiver man det rigtige katalognavn på `%setup`-linjen, f.eks.:

```
%setup -n foobar_1.17
```

hvis source-filerne er havnet i `/usr/src/redhat/BUILD/foobar_1.17`

Default-kataloget er også vigtigt at kende, når man vælger hvilke options man giver til patch-programmet. I eksemplet kaldes patch med **-p1**-option, fordi patch-filen indeholder `ssh_1.2.25` som første element i filnavnet. Men da default-kataloget allerede er nede i ssh-1.2.25, bruges -p1-option for at fjerne det første katalog-element fra filnavnet, inden patch leder efter de filer den skal ændre.

Opsætning af programmet kan ske på mange måder - her er det et configure-script, som kaldes med en option om, at filerne skal bruge **/usr** som præfiks (default er ofte `/usr/local`, som jeg foretrækker at reservere til ikke-rpm-styrede programmer). Andre programpakker konfigureres via en headerfil eller ved at ændre i Makefile - så må man lave en patch mellem den oprindelige fil og den rettede, og installere den som en patch i patch-listen.

Kommandoen **rpm -bp specfile** vil afvikle prep- og setup-fasen, og kan bruges til at "teste" om alle ens patches nu også kan installeres uden problemer.

11.1.4.5. build-fasen

build-fasen er der, hvor programmerne oversættes, og ofte er det blot at køre make. Ligesom ved setup-fasen, er default-kataloget sat til der hvor source-filerne blev pakket ud - hvis man brugte **-n**-optionen til setup-fasen, så bliver det "husket" så man ikke skal skrive den igen.

```
%build
make cflags="$RPM_OPT_FLAGS" ldflags="-s"
```

Her er der dog givet nogle parametre til make-kommandoen. CFLAGS bliver sat til $RPM_OPT_FLAGS - det er en miljøvariabel som **rpm**'s `build`-funktion giver med til alle kommandoerne i build-fasen og som indeholder standard "optimizer"-indstillingerne for programmer, som oversættes med rpm. Default er det `-O2 -m486 -fno-strength-reduce` på i386-platformen.

Kommandoen **rpm -bc specfile** vil først afvikle prep- og setup-faserne og dernæst build-fasen. Hvis man har travlt og allerede har brugt **rpm -bp** til at afvikle prep og setup, kan man bruge **rpm -bc --short-circuit specfile** for bare at køre build-fasen.

Det er typisk under build-fasen, at man ramler ind i de fleste problemer. Her kan det være nyttigt at glemme rpm et øjeblik og i stedet gå ned i build-kataloget og køre build-kommandoerne manuelt, indtil man har fået rettet det hele og programmet kan oversættes. Så laver man en patch-fil med sine ændringer, tilføjer den til spec-filen (husk at få den med i `%setup`-fasen!), og går så tilbage til at builde med rpm.

11.1.4.6. install fasen

install-fasen sørger for at installere de nu oversatte programmer på de rigtige steder i katalogstrukturen. Oftest sker det med kommandoen 'make install', da de fleste pakker heldigvis selv kan finde ud af at installere sig selv. Men ellers må man selv skrive de kommandoer der skal til - enten **cp**, **install** eller hvad man nu foretrækker.

Det er også vigtigt, at man sørger for at de installerede filer får korrekt ejer/gruppe-attributter og permissions.

ssh spec-filen er lidt mere kompliceret end normalt:

```
%install
rm -rf $RPM_BUILD_ROOT
mkdir -p $RPM_BUILD_ROOT
make install

%post
if [ ! -f /etc/ssh_host_key ]; then
   echo "Generating ssh host key"
   umask 022
   /usr/bin/ssh-keygen -b 1024 -f /etc/ssh_host_key -N ''
fi
```

En del af kompleksiteten skyldes brugen af `BuildRoot`-tag'en - inden filerne kan installeres, skal det sikres at det katalog vi installerer i er tomt. Derfor slettes det først, og oprettes så bagefter igen, hvorefter **make install** klarer resten. Det er dog ikke sket uden at jeg måtte ændre i Makefile - men det blev gjort med den patch, der blev installeret helt tilbage i setup-fasen.

Normalt vil **make install** til ssh sørge for at generere filen */etc/ssh_host_key*, som er en del af ssh's krypteringsnøglesæt. Men det vil jo ikke være smart at inkludere den fil i rpm-pakken; så ville alle som brugte rpm-pakken jo have denne nøgle til fælles. Derfor er denne del af installationen fjernet fra Makefile'n, og i stedet lagt ud som et *post-install script* - det står under `%post`.

Disse kommandoer vil blive afviklet *når brugeren installerer den færdige rpm-pakke*, dvs. hver eneste gang ssh-pakken bliver installeret på en pc. Derved får hver enkelt pc sin egen nøgle-fil.

`%post` bruges også til andre ting - hvis man laver en rpm-pakke med et shared library i, er det en god idé at køre **/sbin/ldconfig** i et post-install script; derved opdaterer man den dynamiske linkers tabel over hvilke biblioteker der er installeret, og det kan så bruges med det samme (uden at man skal reboote systemet).

Der er andre af sådanne scripts, som man kan definere: `%post-un` afvikles når man afinstallerer en rpm-pakke, og der er også pre-install (`%pre`) og pre-uninstall (`%pre-un`) scripts. De kan selvfølgelig kombineres.

Install-fasen køres med kommandoen **rpm -bi specfile** - man kan også her bruge `--short-circuit` for kun at afvikle denne fase.

11.1.4.7. sektionen files

Når filerne er blevet installeret, så er det eneste der mangler at pakke dem sammen i en binær rpm-fil. For at kunne gøre det, er det nødvendigt at opremse alle de filer, der er en del af pakken - det sker i sektionen `%files`.

```
%files
/usr/bin/make-ssh-known-hosts
/usr/bin/ssh
/usr/bin/ssh-add

%config /etc/ssh_config
%config /etc/sshd_config

%doc COPYING INSTALL OVERVIEW TODO
%doc README README.CIPHERS README.SECURERPC README.SECURID README.TIS RFC TODO
```

Filerne listes blot med deres fulde path-navn (man ser bort fra evt. *BuildRoot* her). Hvis en fil er en opsætningsfil, *skal* man skrive `%config` før filnavnet; det fortæller rpm-programmet, at denne fil ikke må slettes eller overskrives når man opdaterer pakken, men skal gemmes som en `.rpmsave`-fil.

Dokumentationsfiler listes med `%doc` - og dem behøver man ikke at bekymre sig om at skulle installere. De kopieres automatisk fra det katalog, hvor source-arkivet er blevet udpakket. Dokumentationsfiler placeres i `/usr/doc/`*pakkenavn-version*-kataloget. Hvilke filer man tager med som dokumentationsfiler

er selvfølgelig et skøn, men hellere lidt for mange end for få. (Når man installerer en rpm-pakke kan man bede om ikke at få dokumentationsfilerne med - ved at bruge `--excludedocs`-optionen).

Bemærk, at man-pages *ikke* regnes under dokumentation - de betragtes som en essentiel del af pakken, på lige fod med selve programmet.

Det er selvfølgelig vigtigt, at man får alle filerne med, som hører til pakken. Det kan være nyttigt at køre **make -n install** for bare at se hvad pakken foretager sig under installationen, og holde øje med hvilke filer der kommer hvorhen.

Hvis en pakke indeholder så mange filer, at de har deres eget underkatalog, kan man nøjes med at specificere kataloget - så følger alle filerne deri automatisk med. Det *skal* dog være et katalog, der kun bruges af den pågældende pakke - ellers kan man få sig nogle overraskelser.

Det er også muligt at angive, at et tomt katalog er en del af pakken - det vil typisk være et katalog til f.eks. log-filer, som der jo ikke er nogen af når man laver rpm-filen, men som skal findes for at programmet kan køre. Det skal opremses i sektionen `%files` som

```
%dir /var/log/foobar
```

11.1.4.8. Den ultimative build-kommando

Når man har en spec-fil, som er helt på plads, kan man bruge en enkelt rpm-kommando til at køre gennem hele build-fasen og generere rpm-filerne:

```
[tyge@hven ssh]$ rpm -ba --clean specfile
```

Den kommando klarer det hele: Udpakning af sources, installation af patches, opsætning af programmet, oversættelse, installation, sammenpakning i rpm-filer og oprydning bagefter så BUILD-kataloget er pænt og ryddeligt.

11.1.4.9. Sikke et besvær!

Det er selvfølgelig lidt mere omstændeligt at lave en rpm-fil, end bare at hælde et program igennem en oversætter og ned i `/usr/local`. Nogle gange kan det være frustrerende små detaljer i spec-filen, som gør at en build fejler - og det er som regel først efter, at din pc har tygget et par timer på at oversætte programmet...

Men min erfaring er, at den tid man bruger på at nørkle en ordentlig spec-fil sammen, er godt givet ud. Det er *meget* nemmere at holde styr på sin software, når man med en enkelt kommando kan få en oversigt over, hvad der er installeret og hvilke filer hver pakke indeholder.

For nogle år siden opgraderede jeg mit system fra Red Hat 4.2 til 5.1. Det var en større opgradering, bl.a. fordi der skiftes fra de gamle libc (version 5) til det nye glibc (version 6). Red Hats installationsprogram klarede det meste af opgraderingen uden problemer, men jeg havde en del software-pakker, som jeg selv havde lavet i rpm-format. Nu var det nemt at finde dem frem - jeg kunne bruge

```
[tyge@hven ~]$ rpm -qia | grep -v Manhattan
```

til at liste alle pakker på mit system og filtrere Red Hats nye pakker fra. Det blev til en liste på 10-15 pakker, som jeg selv måtte stå for at opdatere til glibc - og fordi jeg havde installeret dem liggende som RPM-pakker, kunne jeg nemt gentage build-processen, og rette de småting der skulle til for at de kunne oversættes med glibc. Selv programmer, jeg sidst havde haft fat i for 2 år siden kunne nemt genoversættes - fordi spec-filen indeholdt alle de oplysninger, der skulle til for at lave programmet. Jeg behøvede ikke at granske hukommelsen for at komme i tanke om, hvordan det nu var jeg havde konfigureret programmet sidst.

Så bare kløe på - det lønner sig.

11.1.5. Bygge RPM ud fra SRPM

I dette afsnit ser vi på hvordan man bygger en RPM-pakke ud fra kilde-RPM-filen, dvs. en `.src.rpm`-fil. Man behøver faktisk ikke at være root for at bygge en RPM-fil. Du skal blot gøre følgende:

```
[tyge@hven ~]$ cd
[tyge@hven ~]$ mkdir rpm
[tyge@hven ~]$ cd rpm
[tyge@hven rpm]$ mkdir BUILD  RPMS  SOURCES  SPECS  SRPMS
[tyge@hven rpm]$ mkdir RPMS/i386
[tyge@hven rpm]$ cd
[tyge@hven ~]$ echo "%_topdir    $HOME/rpm" > .rpmmacros
```

Når du så har hentet en `.src.rpm`-fil, f.eks. `dos2unix-3.1-3.src.rpm`, så kan du skrive

```
[tyge@hven ~]$ rpmbuild --rebuild  dos2unix-3.1-3.src.rpm
dos2unix-3.1-3.src.rpm
Installerer dos2unix-3.1-3.src.rpm
Udfører(%prep): /bin/sh -e /var/tmp/rpm-tmp.61717
+ umask 022
+ cd /home/pto/rpm/BUILD
+ cd /home/pto/rpm/BUILD
+ rm -rf dos2unix-3.1
+ /bin/gzip -dc /home/pto/rpm/SOURCES/dos2unix-3.1.tar.gz
/home/pto/rpm/SOURCES/dos2unix-3.1.tar.gz:      + tar -xvvf -
...
```

og en masse flere linjer.

Hvis alt går godt ender din `din-nyligt-hentede.i386.rpm` i `$HOME/rpm/RPMS/i386/` og du kan nu installere (som root) med **rpm -Uvh $HOME/rpmRPMS/i386/din-nyligt-hentede.i386.rpm**. Har du en Alpha og ikke en pc, da kommer pakken ikke til at hedde `i386` men `alpha`

11.1.6. chkconfig

chkconfig er et nyttigt lille værktøj til ændringer i services i et runlevel. Værktøjet følger med Red Hat og Mandrake.

Antag, at du har en service **foo**, som du ønsker at tilføje til runlevel 3. Du har installeret (måske gennem **rpm**) et script i kataloget `/etc/rc.d/init.d`, som følger de gængse retningslinjer for service-scripts (dvs. at en service startes op vha. **foo start** og lukkes ned vha. **foo stop**). Nedenfor ser du hvordan den nye service tilføjes. Bemærk, at der er to minus-tegn foran "level"

```
[root@linus root]# chkconfig --level 3 foo on
```

Bliver du træt af servicen **foo** kan du naturligvis fjerne den. Du udskifter blot *on* med *off*. Du kan også bruge **chkconfig** til at tjekke hvilke runlevels en service bliver startet op under. Nedenfor tjekker vi **named**. Bemærk igen, at der er to minus-tegn foran "list".

```
[root@linus root]# chkconfig --list named
named 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

11.1.7. Installation af DEB-programpakker

Skal du installere programmer på Debian eller Corel Linux, skal du være root. Med Corel Linux er det meningen at man skal anvende "Applications"->"System"->"update". Der anvendes et program der hedder **get_it**. Her kan man sætte cd-rom, ftp- eller http-adresser hvor updates hentes fra. Som standard er Corels ftp-update adresse medtaget. Man kan jo prøve at tilføje ftp://ftp.dk.debian.org/debian for at få adgang til alle Debian-pakkerne.

Du kan dog også være interesseret i at kunne styre installationen selv fra kommandolinjen. Nyttige kommandoer er:

- **dpkg -i PAKKE.deb** - installerer pakken.
- **dpkg -r PAKKE** - fjerner pakken igen.
- **dpkg -s PAKKE** - viser status for pakken.
- **dpkg -L PAKKE** - viser hvilke filer, der er i pakken.
- **dpkg -S FILNAVN** - viser hvilke pakker, som FILNAVN stammer fra.
- **dpkg --help** - viser hjælp.

For at konfigurere pakker til dit system skal du køre kommandoen **dpkg --pending --configure**. Det gør, at man får sat de enkelte programmer op til maskinen efter, at pakkerne er blevet installeret.

Kapitel 12. Netværksovervågning

Netværksovervågning kan groft sagt opdeles i to typer: dokumenterende og kontrollerende.

Den dokumenterende overvågning handler om at dokumentere ting som udnyttelse af båndbredde, forbrug af diskplads, osv. Det gøres ofte ved at producere grafer der viser udviklingen over tid. Typiske værktøjer er MRTG og RRDtool.

Den kontrollerende overvågning handler om at sikre at tingene kører og kunne reagere hvis de holder op med at køre. Typiske værktøjer er Big Brother og Nagios Afsnit 12.2.

12.1. Big Sister

Big Sister er en GPL'et klon af Big Brother, der (desværre) berettiget har opnået en meget ringe udbredelse på grund af ganske elendig dokumentation. Det kan derfor anbefales at følge anvisningerne her for at få installeret en virkende Big Sister.

12.1.1. Installation

Start med at tjekke "Prerequisits"-afsnittet på installationsvejledningen (http://bigsister.graeff.com/pdoc/INSTALL.html), for at se hvad du kan installere for at øge Big Sisters funktionalitet/hastighed. Jeg havde installeret de fleste ting i forvejen, og valgte at installere de sidste, for at undgå problemer.

Hent Big Sister version 0.97p2 (http://prdownloads.sourceforge.net/bigsister/big-sister-0.97p2.tar.gz) fra Big Sisters hjemmeside (http://bigsister.graeff.com/). Udpak `big-sister-0.97p2.tar.gz` et sted hvor der er en smule plads, og skift til kataloget `bs-0.97`. I følge dokumentationen skulle du nu kunne sætte Big Sister op, og bl.a. angive hvor du gerne vil have det installeret, men det kan man ikke (`configure`-scriptet findes simpelthen ikke[1]). Heldigvis er det ikke nødvendigt at ændre noget i opsætningen, man kan gå direkte til **make install**. [2] Næste skridt er at oprette en bruger bs, som skal have en brugbar kommandofortolker (ikke noget med **/bin/false**, hjemmekataloget kan passende sættes til `/usr/local/lib/bs`, som er hvor **make install** placerede tingene, men sørg for at man ikke kan logge ind som bs (sæt en * eller et ! i andet felt i `/etc/shadow`).

12.1.2. Indledende opsætning

Det første du skal gøre er at finde linjen

```
localhost bsdisplay
```

i `usr/local/lib/bs/adm/uxmon-net` og rette "localhost" til maskinens navn. Mens du er i gang med at rette i den fil, så vil du sandsynligvis også fjerne "eventlog" fire linjer højere oppe (fra listen over hvad der tjekkes – den vender vi tilbage til), det virker kun på win32-platformen.

Vær opmærksom på at Big Sister læser alle filer i `/usr/local/lib/bs/` der starter med `uxmon-net`, også en fil som `uxmon-net~`. Så hvis din editor laver en sikkerhedskopi af den gamle udgave så slet den!

Du skal også sørge for at du kan få fat i de sider Big Sister genererer. Det letteste er nok at sætte sin navneserver op så bs.domæne peger på din maskine og så tilføje:

```
<VirtualHost bs.domæne>
ServerAdmin admin@domæne
ServerName bs.domæne
DocumentRoot /usr/local/lib/bs/www/
</VirtualHost>
```

til `httpd.conf`.

12.1.3. Start Big Sister

Nu er tiden inde til at starte Big Sister, og se at det virker. Big startes med kommandoen **/usr/local/lib/bs/bin/bb_start start**. Nu skulle du kunne pege din netsøger mod bs.domæne og se hvordan Big Sister tror din maskine har det.

12.1.4. Videre opsætning

Først vil vi af med den gule advarselslampe der er under "msgs", den skyldes at brugeren bs ikke kan læse `/var/log/syslog` og `/var/log/messages`, eftersom vi har sikret os at man ikke kan logge ind som bs, er det ikke særlig farligt at tilføje bs til gruppen adm, så Big Sister kan læse disse filer.

Med mindre du bruger sendmail havde du også en rød alarmlampe under "procs", der fortalte dig at der ikke kørte mindst en sendmail-proces, den vil du sikkert gerne af med. For at gøre det skal du igen have fat i `/usr/local/lib/bs/adm/uxmon-net`, denne gang skal du finde linjen der indeholder

```
procs=init(1-1),sendmail(1-20)
```

og enten fjerne sendmail helt, eller erstatte det med noget du er interesseret i. Hvis du kører Postfix kan du f.eks. bruge

```
procs=init(1-1),master(1-20)
```

til at sikre dig at Postfix kører.

Hvis du har en webserver kørende på maskinen (og det har du, for ellers kan du ikke komme i kontakt med Big Sister), er du måske også interesseret i at få holdt øje med om den kører. Det gør du ved i `/usr/local/lib/bs/uxmon-net` at tilføje http til listen over ting der skal tjekkes.

Der er mange andre ting man kan få Big Sister til at holde øje med, det vil føre for vidt at forsøge at komme ind på dem alle her. Tjek dokumentationen (http://bigsister.graeff.com/pdoc/CONFIG.html), som omend den ikke er god så dog giver lidt på dette punkt.

Du har måske også bemærket at "History", "Alarms" og "Admin" henvisningerne til højre ikke virker. For at ordne det skal du kopiere filerne `bshistory`, `bswebalarm`, `bswebadmin` og `bsgraph` fra `/usr/local/lib/bs/bin` til `/usr/local/lib/bs/www/cgi` (det katalog skal oprettes først), og indsætte linjen

```
ScriptAlias /cgi /usr/local/lib/bs/www/cgi
```

i `httpd.conf`.

Når du har ændret i Big Sisters opsætning skal du genstarte Big Sister med kommandoen **/usr/local/lib/bs/bin/bb_start restart**. Der kan der godt gå et par minutter før Big Sister får udskiftet en lampe på oversigtssiden, selv om man ved at klikke på den kan se at status er ændret.

12.1.5. Andre maskiner

Hvis du vil tjekke om netværksforbindelsen til en anden maskine er i orden, kan du tilføje linjen

```
anden.maskine.domæne ping
```

til `/usr/local/lib/bs/adm/uxmon-net`. Så vil Big Sister jævnligt sende et UDP-ping til anden.maskine.domæne, for at se om den svarer. For at den skal have noget held med det, skal du sørge for at den anden maskine svarer på UDP-ping, det gør du ved sørge for at linjen

```
echo            dgram   udp     wait    root    internal
```

findes i `/etc/inetd.conf`.

12.2. Nagios

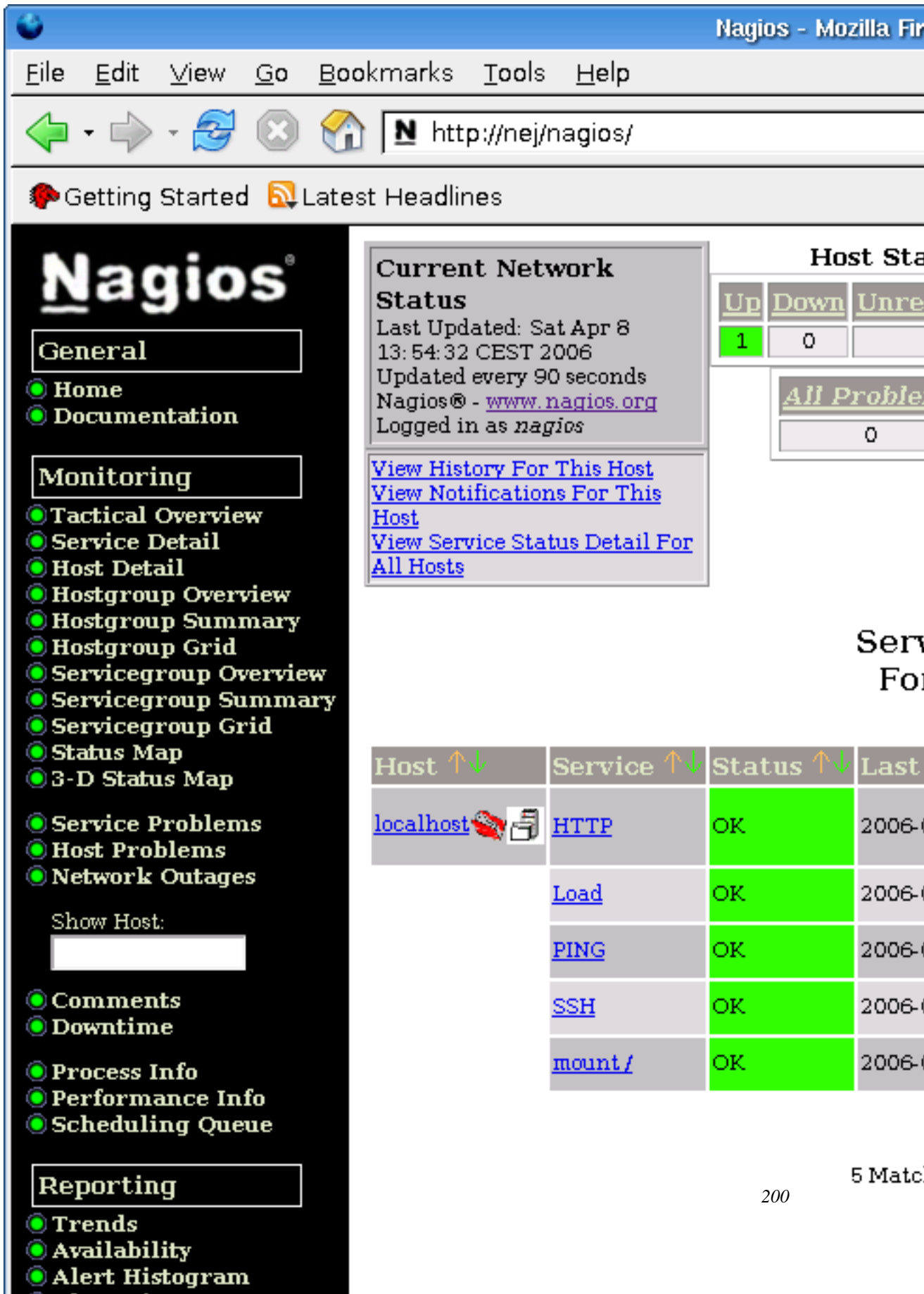
Nagios er et overvågningsværktøj der kontrollere om alle services på alle maskiner køre. En typisk brug er at sætte Nagios op til at overvåge et antal maskiner der er forbundet i netværk, og administratorene af disse maskiner bliver så alarmeret når der er noget galt med de services de har ansvaret for. Det er ganske få ting der skal sættes op for at få det hele til at virke, men følger man den vedlagte dokumentation i Nagios, er det ikke sikkert man kommer helt i mål. Som det fremgår af Nagios-manualen: "Relax - its going to take some time." - så syntes forfatteren at det skal være meget besværligt at få begyndt med et

Nagios-system. Men med de eksempler der følger med dette kapitel du læser på nu, samt beskrivelsen her, så skulle du gerne have en simpel kørende installation igang på ca. time hvis du er sådan en almindelig habil systemadministrator.

Nagios er opbygget som en central enhed der kontakter de andre maskiner og henter status fra dem. Nogle services såsom HTTP, SMTP og SSH kan checkes direkte fra Nagios maskinen selv fordi de anvender en åben port. Status vedrørende diskplads og andre interne services skal kontroleres ved at køre et program på den pågældende maskine. Programmet der skal køres på klient-maskinen hedder NRPE.

Som supplement til denne beskrivelse er der nogle konfigurationsscript du kan bruge som udgangspunkt. Sådanne start-script burde som udgangspunkt have været med i standardpakken, men det er de ikke. Disse ekstra-scripts kan downloades fra http://www.linuxbog.dk/admin/eksempler/etc/nagios (/admin/eksempler/etc/nagios/) Når de er blevet installeret vil du få et skærmbillede der ligner nedenstående.

Figur 12-1. Et skærmdump af Nagios ved almindelig drift hvor alt er OK



Følgende programpakker skal downloades for at kunne komme igang.

- http://www.nagios.org/download/ Nagios: nagios-2*.tar.gz er selve programmet der skal køre på serveren og det program der indhenter informationer fra de andre maskiner.
- nagiosplug.sourceforge.net (http://sourceforge.net/project/showfiles.php?group_id=29880) Nagios plugins: nagios-plugins-1*.tar.gz er en samling af små programstumper der checker de enkelte services. Eksempelvis findes programmet **check_http** i den pakke. og det anvendes til at checke web-serveren på en fjern maskine. Du skal have denne pakke installeret for at kunne bruge Nagios til noget. Det kan så undre at den ikke er en del af standardpakken.
- http://www.nagios.org/download/ NRPE: nrpe-*.tar.gz anvendes for at kontrollere ting der foregår lokalt på en fjern-maskine. Vil man holde øje med fx diskplads, skal man bruge NRPE-pakken på klient-maskinen.

12.2.1. Installation

I beskrivelsen her er valgt at installere fra grundpakken og ikke bruge et pakkeformat der passer til en specifik distribution. Grundinstallationen er baseret på BSD's måde at placere filerne på, så det er en værre rodebunke at hitte rundt i, når man er vant til Linux. Herunder er vist en måde som filerne kan lægges ud på. Vigtigst er det at du kan finde konfigurationsfilerne i `/etc/nagios`, men de andre subdir er nu også rare at vide hvor ligger. Kør først en **configure**, efterfulgt af **make** og **make fullinstall**.

```
[tyge@hven ~]$ [ -z "`grep ^nagios: /etc/group`" ] && groupadd nagios
[tyge@hven ~]$ [ -z "`grep ^nagios: /etc/passwd`" ] && useradd -c "Nagios" -g nagios -d /var/www/nagi
[tyge@hven ~]$ cd nagios-2.0
[tyge@hven nagios-2.0]$ ./configure \
        --sysconfdir=/etc/nagios \
        --bindir=/usr/bin \
 --libexecdir=/usr/libexec/nagios \
        --sbindir=/var/www/nagios/cgi-bin \
        --datadir=/var/www/nagios/html \
        --localstatedir=/var/nagios \
        --with-htmurl=/nagios \
        --with-cgiurl=/nagios/cgi-bin
[tyge@hven nagios-2.0]$ make all
[tyge@hven nagios-2.0]$ mkdir -p /var/www/nagios/html
[tyge@hven nagios-2.0]$ su -c "make fullinstall"
```

nagios-plugins er en særskilt programpakke der skal downloades. Den indeholder alle de check-programmer Nagios-dæmonen skal bruge, og uden nagios-plugins virker Nagios slet ikke. Hent programpakken (Afsnit 12.2.5) og installer pakken som angivet herunder:

```
[tyge@hven ~]$ cd nagios-plugins-1.4.3
[tyge@hven nagios-plugins-1.4.3]$ ./configure \
        --with-cgiurl=/var/www/nagios/cgi-bin
[tyge@hven nagios-plugins-1.4.3]$ make
```

```
[tyge@hven nagios-plugins-1.4.3]$ su -c "make install"
```

Med de mest basale programmer installeret, mangler der nu et færdigt opsat eksempel du kan bruge som udgangspunkt. Det er beskrevet i næste afsnit.

12.2.2. Konfiguration

Nu mangler der et færdigt opsat eksempel. Der følger en smule med nagios-pakken, men det er ikke helt nok til at komme i luften med. For at kunne bruge eksemplet der følger med Nagios, skal der rettes så mange ting at det mindst vil tage dig 6 timer at komme frem til mål. Hent defor den færdige pakke fra http://cvs.linuxbog.dk/admin/eksempler/etc-nagios.tar.gz (/admin/eksempler/etc-nagios.tar.gz) og stil dig i roden af din disk og pak den ud. Filerne vil lægge sig i **/etc/nagios**. Filerne kan også hentes enkeltvis og browses på http://www.linuxbog.dk/admin/eksempler/etc/nagios/ (/admin/eksempler/etc/nagios/)

Når filerne er pakket ud er det vigtigt at få sat password til brugeren **nagios**. Det er den bruger som skal kunne logge ind via web-siden. Du kan tilføje alle de brugere du har behov for. Det gøres med programmet **htpasswd2** når det er Apache 2.0 du har installeret, ellers er det **htpasswd**. Gør som nedenstående eksempel:

```
[nagios@hven ~]$ cd /etc/nagios
[nagios@hven nagios]$ htpasswd2 htpasswd.users nagios
New password:
Re-type new password:
Updating password for user nagios
```

Når password til nagios-brugeren er sat kan Nagios-dæmonen startes, og den nye konfiguration til Apache webserveren kan loades.

```
[root@hven ~]# /etc/init.d/nagios start
```

Nu er Nagios dæmonen startet og indsamling af data er begyndt. Næste trin er så at kunne se de indsamlede data. Det forventes at du har Apache-webserveren installeret, og har nogenlunde styr på at administrerer den. Det nemmeste er hvis du vil tilgå Nagios via http://localhost/nagios (http://localhost/nagios/) for så kan du nøjes med at inkludere `sample-config/httpd.conf` der ligger i filstrukturen under source nagios-pakken. Bruger du Gentoo er det nemmest blot at kopiere filen ned i filnavnet `/etc/apache2/modules.d/99_nagios.conf` (eller brug et andet nummer der er lavere end 99). Ved andre Linux-distributioner er det nok nemmest at kopiere `sample-config/httpd.conf` til `/etc/nagios/` og så gå ind i din `httpd.conf` der hører til Apache og i bunden skrive `Include /etc/nagios/httpd.conf` . Når den config-fil der hører til Nagios er blevet inkluderet, er det blot at få den loadet ind i Apache. Det gøre evt. med følgende kommando:

```
[root@hven ~]# /etc/init.d/apache reload
```

Nu kan du klikke dig ind på http://localhost/nagios/ og se status for den maskine som Nagios er installeret på. Klik på service detail. Måske er der nogle service-check der lige står til "Pending", men så

vent lige et par minutter, så kommer status også for disse. Som udgangspunkt viser eksemplerne der hører til "Friheden til at vælge" status for `localhost` og `cvs.sslug.dk` . For at rette det, skal der rettes i konfigurationsfilerne der ligger i `/etc/nagios/*.cfg` . Da der skal rettes i flere filer, kan man jo være fræk at gøre det fra kommandolinjen. Hvis du fx vil rette `localhost` til `hven.sslug.dk`, og du vil rette ip-nummer fra 127.0.0.1 til 80.80.80.80, så kan det gøres med:

```
[nagios@hven ~]$ cd /etc/nagios
[nagios@hven nagios]$ grep localhost *.cfg   # her ser du hvilke filer det er
[nagios@hven nagios]$ perl -i -pe 's/localhost/hven.sslug.dk/g' *.cfg
[nagios@hven nagios]$ perl -i -pe 's/127.0.0.1/80.80.80.80/g' *.cfg
```

12.2.3. Ændre konfiguration

Inden du kaster dig over at tilrette konfigurationsfilerne manuelt med en editor, skal du vide at der er indtil flere forskellige tredjeparts systemer der er beregnet for konfiguration via en GUI. En del af dem er baseret på PHP og MySQL, og nedarver selvklart de relationer der i forvejen er imellem Nagios' konfigurationsfiler. Download tredjepartsprogrammerne fra Afsnit 12.2.5.

Som udgangspunkt er konfigurationen fordelt på flere filer ved at der er refereret til dem fra `nagios.cfg` . Nogle gange vil man finde det irreterende at skulle åbne flere filer for at lave en ændring, men så er muligheden at man kan samle det hele i en fil. Det der handler om host og service har man ofte fat i, så det kan være en fordel at have dem i samme fil. Den måde som filerne kommer som standard er følgende:

1. `services.cfg` indeholder alle de services der skal overvåges på alle maskiner. De services der overvåges i eksemplet her, er dem som de fleste vil forvente er startet.
2. `hosts.cfg` indeholder en liste over alle de maskiner der skal overvåges. Her i eksemplet er det kun `localhost` og `cvs.sslug.dk` der er sat op til at blive overvåget. Det er så meningen at du skal gå ind og lave en søg-og-erstat af de to hostnavne, og sætte dine egne navne ind.
3. `hostgroups.cfg` er listen på alle de grupper af maskiner du har. De fleste har nok så få maskiner at de kun har én gruppe, men det er nemt at lave en gruppe mere.
4. `contacts.cfg` er listen over de personer der skal have tilsendt en mail, når der er noget galt.
5. `contactgroups.cfg` indeholder listen af de grupper der skal have sendt en mail, når der er noget galt. Når en service fejler, så er det gruppen der sendes en mail til. Det gør det nemmere at administrerer mange brugere samtidigt.
6. `servicegroups.cfg` er ikke så vigtig at have med til at starte med. Den giver fx et godt overblik af alle webservere, men på alle maskiner.
7. `checkcommands.cfg` indeholder listen over alle de kommandoer man kan bruge til at checke med i Nagios.
8. `nagios.cfg` er hoved konfigurationsfilen. Det er typisk en man kun vil rette i under installation, eller hvis skal tilføje meget avancerede funktioner. Som udgangspunkt skal du nok ikke rette i denne.

De resterende konfigurationsfiler er til mere avanceret opsætning og er ikke beskrevet her.

En ting der er fælles for den måde konfigurationsfilerne er opbygget på i Nagios, er at man definere først en standardindstilling, som man så efterfølgende bruger som basis for de andre definitioner. Har man kun et ganske lille system, der kun overvåges af ganske få systemadministratore, så vil man nok kun definere en gruppe der får tilsendt mail når noget går galt. Her vil man så i filen `/etc/nagios/services.cfg` øverst definere en template/objektklasse der indeholder navnet på den gruppe der skal have en mail tilsendt. Det er "contact_groups" der får tilsendt mail. Det kunne se ud som nedenstående:

```
define service{
 name  generic-service # navn på definitionen
 register 0  # angiver at det kun er en default
 contact_groups admins  # eksempel på noget der nedarves
 ... flere definitioner følger
 }
```

Der vil typisk være flere definitioner end vist herover. Det er blot beskrevet minimalt her, for at forklare princippet. Det man nu kan gøre med "generic-service" definitionen er at bruge den i alle de services man skal have defineret.

```
define service{
 use   generic-service # nedarv fra definition
 host_name  foo.eksempel.dk # navn på host
 service_description HTTP  # navn på service
 check_command  check_http # kommando der tjekker service
 }
```

Ved et bruge ovenstående princip, kan man så i mange tilfælde definere en service med kun fire konfigurationslinjer. Får man brug for enkelte undtagelser, kan man så blot skrive det ind ved den service. Som eksempel kunne man forestille sig at webadministratorerne også skulle havee at vide hvis der er noget galt med HTTP. Ved så at indskrive contact_groups igen, overskrives den forgående definition. I Nagios skrives lister adskilt med komma.

```
define service{
 use   generic-service # nedarv fra definition
 host_name  foo.eksempel.dk # navn på host
 service_description HTTP  # navn på service
 check_command  check_http # kommando der tjekker service
 contact_groups  admins,webmasters
 }
```

En ting der er god at huske efter man har lavet om på konfigurationen, er lige at køre et check om det man har lavet nu er rigtigt. Det gøres med kommandoen:

```
[tyge@hven ~]$ /usr/bin/nagios -v /etc/nagios/nagios.cfg
```

Herfter får man listet alle de fejl der måtte være og man kan gå igang med at rette dem.

Er der en check-kommando der driller, kan man altid prøve dem af fra kommandolinjen. En af de check-kommandoer der kan give underlige svar, er fx **check_http_string**. Den bruges til at undersøge

om en hjemmeside indeholder en bestemt tekst. Det kunne være en tekst far en database, eller andet der indikere at mange af de services der skal fungere på en hjemmeside, er til stede. For at vide hvad programmet rigtigt hedder, må det findes i `/etc/nagios/checkcommands.cfg` og finde **check_http_string**. Her kan man se at programmet blot er **check_http** med en ekstra option. Definitionen ser således ud:

```
# 'check_http_string' command definition
define command{
    command_name    check_http_string
    command_line    $USER1$/check_http -H $ARG1$ -w $ARG2$ -c $ARG3$ -s "$ARG4$"
    }
```

Hjælp til brug af check-kommandoerne fås alle ved option '-h'. '-s' bruges til at undersøge om en bestemt tekst forekommer. Nogle hjemmesider checker for hvilket sprog brugeren foretrækker, og her vil check-kommandoen spørge som default sprog, hvilket typisk er engelsk. Hvis man checker hjemmesiden med en browser, står der måske "Velkommen", men med **check_http** vil der stå "Welcome". På skærmen kunne det se således ud:

```
[tyge@hven ~]$ cd /usr/libexec/nagios
[tyge@hven /usr/libexec/nagios]$ ./check_http -H www.eksempel.dk -w 5 -c 10 -s "Velkommen"
HTTP CRITICAL: string not found|time=  0.020
[tyge@hven /usr/libexec/nagios]$ ./check_http -H www.eksempel.dk -w 5 -c 10 -s "Welcome"
HTTP ok: HTTP/1.1 200 OK -   0.020 second response time |time=  0.020
```

12.2.3.1. services.cfg

Alle services der skal holdes øje med findes i `services.cfg` . Med services forstås fx en HTTP web-service. Her er det at der bliver testet om det kan lade sig at hente en hjemmeside, og samtidigt bliv er det noteret hvor lang tid det tog. I nedenstående anvendes direktivet **use generic-service**, hvilket så gør at alle de sædvanlige direktiver er nedarvet fra denne. Derved kan man nøjes med kun at angive fire linjer parametre for hver service der skal holdes øje med.

```
define service{
 use   generic-service
 host_name  localhost
 service_description HTTP
 check_command  check_http
 }
```

Har man en bestemt service man altid gerne vil bruger på alle de hosts man har defineret i hosts.cfg, så kan man fordel bruge en '*' som host_name. Eksempelvis kunne det være at alle skulle have en ping:

```
define service{
 use   generic-service
 host_name  * ; Alle hosts får en ping
 service_description PING
 check_command  check_ping!100.0,20%!500.0,60%
 }
```

12.2.3.2. hosts.cfg

De maskiner der skal overvåges er angivet i `hosts.cfg`. Som med services anvendes også her en standardkonfiguration benævnt **use generic-host** så alle fælles direktiver kan angives et sted. Det kan virke overflødigt at IP-adressen skal med, men som angivet i Nagios-manualen kan man komme ud for at DNS-serveren er nede, og så mister man overvågning af alle sine maskiner på én gang.

```
define host{
 use  generic-host
 host_name localhost
 alias  Local Host
 address  127.0.0.1
 }
```

Til hver host kan der tilføjes ekstra information som ikke er påkrævet, men praktisk at have. Har fx en hel hjemmeside der beskriver hver enkelt maskine, kan sætte et link op til denne så man hurtigt kan klikke sig frem til informationen direkte fra Nagios overvågningssiden. Informationen kunne være som følger:

```
define hostextinfo{
 host_name       localhost
 notes           Den lokale vaert som Nagios er installeret paa
 notes_url       http://localhost/info.html
 icon_image      localhost.png # placeres i "/var/www/nagios/htdocs/share/images/logos"
 icon_image_alt  Lokal vaert
 }
```

Som tidligere nævnt er der noget rod omkring hvor de forskellige filer skal ligge, og disse "icon_image" skal ligge i et subdir der hedder noget med "share/images/logos", og det må du så lede efter.

Her i eksemplet er brugt hostnavnet 'localhost', men det bør man ikke bruge i praksis, da det er forvirende hvis rent faktisk sidder ved en anden maskine.

12.2.3.3. hostgroups.cfg

På Nagios web-status-siden er der i venstre side en menu, hvor man kan vælge at se status på forskellig måde. En måde er at se de forskellige maskiner sammen som en gruppe, og så have flere grupper med et antal maskiner i. Er man flere personer om at overvåge maskiner, og hver har et ansvarsområde for et antal maskiner, kan man med fordel dele dem op i flere hostgrupper. Det fx være en gruppe med Linux-maskiner, og en med UNIX-maskiner. Gruppen kan også være den fysiske lokation såsom København og Ruds Vedby.

Man kan godt have den samme maskine nævnt i flere grupper samtidigt, men når der så er en fejl på den maskine vil der være to røde felter på skærmen, og på afstand vil det se ud som om der er to fejl. Så nævn kun hver maskine én gang i hostgroup-filen.

```
define hostgroup{
```

```
 hostgroup_name gruppenavn ; Vælg et kort unikt navn
 alias  Langt navn for gruppen
 members  localhost, cvs.sslug.dk ; Kommasepareret liste
 }
```

12.2.3.4. contacts.cfg

De brugere der skal have en meddelelse når der er noget galt med en maskine, skrives i listen `contacts.cfg`. I eksemplet der følger med beskrivelsen her, bruges en contact-template og så er det kun ganske få linjer man typisk skal skrive om hver person.

```
define contact{
 use  generic-contact ; Den definerede contact-template
 contact_name nagios
 alias  Nagios Bruger
 email  nagios@localhost
 }
```

12.2.3.5. contactgroups.cfg

Der vil ofte være flere personer der skal sendt en e-mail når noget går galt. Derfor definere man et antal personer i en gruppe, og ved de enkelte services angives hvilken kontaktgruppe der skal have besked når der er noget galt. Man kunne fx foresille sig at man i `services.cfg` havde en overvågning af web-servere (check_http) og her ville sende en mail til gruppen "webmasters" hvis der skete noget der. I det vedlagte eksempel er dog blot defineret en enkelt gruppe der modtager alle fejl, hvilket typisk vil være det de fleste har brug for.

```
define contactgroup{
 contactgroup_name admins
 alias   Administrators
 members   root, nagios
 }
```

12.2.3.6. servicegroups.cfg

Servicegrupper bruges til at overvåge en bestemt service, men på tværs af hvilke maskiner de køre på. I det viste eksempel er det webservices der bliver holdt øje med, og det gør så at man kan få en pæn liste op på skærmen der ikke viser andet end webservere. Bemærk her at listen af "members" skrives både med hostnavn og servicebeskrivelse: members <host_name>,<service_description> . Alle <host_name> og <service_description> kan findes i filen `services.cfg` , og man kan fx lave en hurtig liste til skærmen med kommandoen:

```
[tyge@hven nagios]$ egrep "(host_name|service_description)" services.cfg
```

Herunder et eksempel:

```
define servicegroup{
 servicegroup_name HTTP
 alias   HyperText Transport Protocol
 members   localhost,HTTP, cvs.sslug.dk,HTTP
 }
```

12.2.3.7. checkcommands.cfg

`checkcommands.cfg` indeholder definitionen på alle de check-kommandoer man kan kalde i `services.cfg` . Nogle af kommandoerne skal have parametre med såsom hostname, og andre skal ikke have nogen parametre. En nem måde at lære hvordan de enkelte check-programmer virker og hvordan de giver output, er at kalde dem fra kommandolinjen. Hvis du fx 48% diskplads tilbage på din rod-partition, så prøv at lave en warning på 50%. Nagios-plugins (hjælpe-programmerne) ligger i `/usr/libexec/nagios`, så prøv denne:

```
[tyge@hven ~]$ cd /usr/libexec/nagios
[tyge@hven libexec]$ ./check_disk -w 50% -c 25% -p /
DISK WARNING - free space: / 134354 MB (48%);| /=147041MB;140697;211046;0;281395
```

I stedet for at bruge mount-pointet '/' kan man også bruge selve devicen såsom '/dev/hda1', hvis det giver mere mening. De anvendte mount-point kan læses i `/etc/mtab` . Som det ses herunder hedder kommandoen **check_local_disk**.

```
define command{
 command_name    check_local_disk
 command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
 }
```

Når man har fundet ud af hvilke parametre man til henholdsvis warning og critical, kan man oprette en definition i `services.cfg`:

```
define  service{
 use   generic-service
 host_name  localhost
 service_description Disk / mount
 check_command  check_local_disk!50%!25%!/
 }
```

For alle check-programmerne gælder at man kalder dem med kommandoen **-h** for at få den fulde hjælp. Her får man så al den hjælp der findes, da reglerne for skrivning af plugins foreskriver at det skal være sådan.

12.2.3.8. nagios.cfg

Der er mange ting der kan ændres i hovedkonfiguratiosfilen `nagios.cfg`, men det vil oftest ikke være nødvendigt. Der kan være en enkelt path der skal ændres, men ellers er standardparametrene sat fornuftigt, og det er kun de andre filer man skal koncentrere sig om.

12.2.4. Manualen

Det er ikke mange steder i denne bog "Friheden til at vælge" hvor der er en beskrivelse af hvordan læser den engelsksprogede manual. Hvad angår Nagios, så er det påkrævet da den ikke er helt nem at finde rundt i. Der er desværre heller ikke noget stiksordsregister, hvilket kunne have hjulpet en del.

Links ændre sig hele tiden for den sidst gældende manual, men her er så alligevel et link til Nagios version 2.0 manual (http://nagios.sourceforge.net/docs/2_0/) . Et emne, hvor det kunne være rart hvis det var nemt at finde, er der hvor der står noget om hvordan man definere services. Når der ikke lige er et stikordsregister, så er det lige sin sag at finde ud af at det er Table of Contents - > Configuring Nagios -> Object configuration file options -> Click here -> Service definitions (http://nagios.sourceforge.net/docs/2_0/xodtemplate.html#service) man skal klikke på. Måske du bare skulle overvej at hente hele manualen hjem, så kan du altid bruge **grep** hvis der er noget du vil finde. Og nu du har hentet manualen hjem på egen harddisk, så kan du lige sætte fontene til noget lidt større så det er til at læse. 7 og 8 pt er lige lovlig småt når man bruger "serif". De to følgende kommandoer skulle klare de værste problemer.

```
[tyge@hven ~]$ wget -mk http://nagios.sourceforge.net/docs/2_0/
[tyge@hven ~]$ ls *.html | \
 xargs perl -i -pe 's/font-size: [1-9]pt/font-size: 10pt/g'
```

12.2.5. Links

- nagios.org (http://www.nagios.org/) er hjemmesiden for Nagios. Her downloades hovedprogrammet til indsamling af data og programmet til præsentation på web.
- nagiosplug.sourceforge.net (http://nagiosplug.sourceforge.net/) er hjemmesiden for check-programmerne for Nagios. Uden check-programmerne kan Nagios intet, så man skal downloade en pakke nagios-plugins.*.tra.gz her fra.
- nagiosexchange.org (http://www.nagiosexchange.org/) indeholder alt muligt andet omkring Nagios. Der er forskellige programmer der gør det nemmere at sætte Nagios op, pakker med ikoner, og andre tillægsting til Nagios. Der findes flere pakker der handler om at gøre opsætning af Nagios nemmere, så det kan være interessant at kigge på den på et tidligt tidspunkt.

Slutbemærkning:

1. Det gør det i version 0.98, som dokumentationen flere steder ser ud til at være rettet mod, den havde jeg andre problemer med.
2. Big Sister er en samling Perl-programmer der ikke skal oversættes derfor er der ikke noget **make**-skridt.

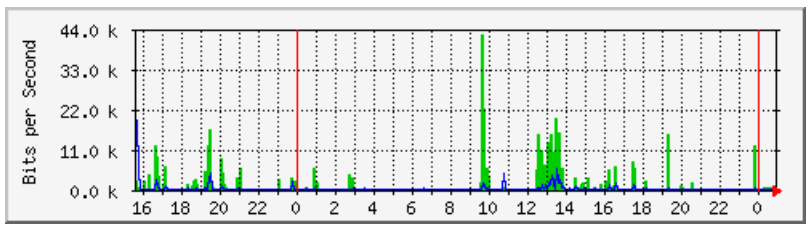
Kapitel 13. Måling og webgrafik

I dette kapitel ser vi på programmer til måling af systemparametre og efterfølgende grafisk fremstilling af hjemmesider af målingerne. Eksempler på dette kunne være nettrafik, blæserhastighed eller temperatur over tid.

13.1. MRTG – grafisk overvågning af systemet

MRTG er en god måde at overvåge dit system på. MRTG viser i pæne grafer den løbende udvikling omkring belastningen på forskellige områder, for det seneste døgn, den seneste uge, den seneste måned og det seneste år.

Figur 13-1. Traffic Analysis for eth0 -- 192.168.1.2



MRTG er først og fremmest lavet til netværksovervågning, så du kan bruge det til at for eksempel overvåge din modem-linje, din ADSL-linje eller trafikken på dine servere. MRTG kan også overvåge andre systemressourcer såsom systembelastning, antal processer, cpu-temperatur, antal brugere mv, men dette er ikke standard.

Graferne vises i et HTML-format, så man kan se det i en browser – men man behøver ikke have en webserver installeret for at se dem. Til netværksovervågningen bruges SNMP, og MRTG laver løbende overvågning typisk hver 5. minut, hvor data opsamles så de fylder ca. en konstant størrelse (der er ikke noget med at data bliver ustyrligt store).

MRTG står for Multi Router Traffic Grapher og har denne hjemmeside (http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/). Det er udgivet under GPL og findes med i de fleste Linux-distributioner, så man kan blot installere pakken derfra.

Der er mange konfigureringsmuligheder, så se installationsguiden på hjemmesiden, og også brugersiderne er interessante med andre anvendelser end netværksovervågning. SSLUG bruger MRTG til overvågning af sslug.dk, man kan se hvordan maskinen har det på denne URL

(http://www.sslug.dk/bb/html/www.sslug.dk.larrd.html) og en beskrivelse af denne opsætning kan findes på denne URL (http://www.sslug.dk/mrtg/sslug_mrtg.html).

13.1.1. Opsætning af MRTG

Hvis man skal for eksempel overvåge lokalnettet på sin maskine, skal man gøre følgende, som root:

1. Installér en SNMP-dæmon på din maskine, eller aktivér SNMP på de bokse du vil overvåge. SNMP er Simple Network Management Protocol. Fx findes en beskrivelse af aktivering af SNMP på en Cisco 677 ADSL ruter på denne URL (http://www.napoleon.dk/snmp/).

   Til Linux findes SNMP-dæmonen for eksempel i pakken net-snmp eller ucd-snmp. En simpel opsætning i `/etc/snmp/snmpd.conf` kunne se således ud (både for net-snmp og ucd-snmp):

   ```
   com2sec public     localhost        public
   com2sec public     192.168.0.0/16   public
   group     public    v1   public
   group     public    v2c  public
   group     public    usm  public
   view      all  included  .1
   access    public    ""   any  noauth    exact     all  none none
   ```

   Dette tillader maskiner på lokalnetværket 192.168.0.0 at se hvad maskinen laver, men ikke at ændre noget, og det udelukker alle andre maskiner ude fra det offentlige internet fra at kigge på SNMP-oplysninger.

2. For at starte snmp-dæmonen på linux-maskinen kan man gøre følgende:

   ```
   [root@linus /root]# service snmpd start
   ```

   Dette skal også gøres permanent ved opstart, så du skal aktivere snmpd for eksempel via Gnomes kontrolcenters behandling af tjenester.

   Du kan evt se hvilke netværksgrænseflader der er tilgængelige med programmet snmpwalk, for eksempel på maskinen med ip-adressen 'localhost' (enten net-snmp eller ucd-snmp udgaven):

   ```
   [root@linus /root]# snmpwalk -Os -c public -v 1 localhost
   [root@linus /root]# snmpwalk localhost public 2
   ```

3. Installér mrtg, enten fra din distribution eller fra kildetekst fundet på MRTG-hjemmesiden nævnt ovenfor. Du kan for eksempel lægge din opsætning i `/home/mrtg/cfg/mrtg.cfg` – som kunne se sådan her ud:

   ```
   WorkDir: /var/www/html/mrtg
   Language: Danish
   Options[_]: bits,growright
   Weekformat[_]: V

   Target[localhost_2]: 2:public@localhost
   ```

```
MaxBytes[localhost_2]: 12500000
Title[localhost_2]: Traffic Analysis for eth0
PageTop[localhost_2]: <H1>Traffic Analysis for eth0</H1>

Target[Cisco]: 2:public@192.168.1.1
MaxBytes[Cisco]: 32000
Title[Cisco]: Cisco 677 Gateway
PageTop[Cisco]: <H1>Traffic Analysis for outside ADSL</H1>
```

Hermed lægges data fra MRTG i kataloget `/var/www/html/mrtg/`

Du kan bruge programmet cfgmaker til at automatisk at lave en opsætningsfil til MRTG. Den kigger på en eller flere maskiner eller bokse og finder ud af via SNMP hvilke netværksgrænseflader der er aktive og laver så opsætningsfilen. For eksempel for din lokale maskine og en ADSL-ruter på 192.168.1.1:

```
[root@linus /root]# cfgmaker localhost 192.168.1.1 > /home/mrtg/cfg/mrtg.cfg
```

4. Initiér filen `/var/www/html/mrtg/index.html` med programmet indexmaker:

   ```
   [root@linus /root]# indexmaker --columns=1 /home/mrtg/cfg/mrtg.cfg >/var/www/html/mrtg/index.html
   ```

5. Kør MRTG for første gang:

   ```
   [root@linus /root]# mrtg /home/mrtg/cfg/mrtg.cfg
   ```

6. Kør MRTG hvert 5. minut via cron med

   ```
   [root@linus /root]# crontab -e -u root
      0/5 * * * * /usr/bin/mrtg /home/mrtg/cfg/mrtg.cfg
   ```

7. Du kan nu se resultatet af din MRTG-overvågning for eksempel med netscape:

   ```
   [root@linus /root]# netscape /var/www/html/mrtg/index.html
   ```

Appendiks A. Revisionshistorie for bogen

Igennem tiden har bogen "Linux – friheden til systemadministration" udviklet sig meget. Vi frigiver ofte nye versioner, når der er kommer en del rettelser ind, eller nye afsnit er blevet skrevet. Kommentarer, ris og ros, og specielt fejl og mangler bedes sendt til linuxbog@sslug.dk (mailto:linuxbog@sslug.dk), men er du medlem af SSLUG så skriv til sslug-bog@sslug.dk (mailto:sslug-bog@sslug.dk). Her er en liste over, hvad der er ændret i bogen.

Denne bog er skabt ud fra version 3.9 af "Linux – friheden til at vælge", som blev splittet i mindre dele.

- Version 2.8.20060113 - 2006-januar-13: Donald Axel: Retter afsnit med dødt link til Linuxcare CD og sørger for at det bliver mere generelt rescue teknik, som vises.

  Jacob Sparre Andersen: Retter sprog. Præciseret teksten om skyggeadgangskoder. Donald Axel retter døde links.

  Hans Schou sætter Big Sister indholdsfortegnelse og skriver afsnittet om Nagios.

- Version 2.8 - 25. januar 2004: Henrik Christian Grove tilføjer et lille afsnit om at styre hvilke netkort en DHCP-server lytter på. Jacob Sparre Andersen: Retter sprog. Peter Toft starter nyt kapitel om test af server-funktioner. Tiføjer flere links om "High Availability". Retter URL-fejl fundet af Esben Rugbjerg. Fjerner gamle døde links i afsnit om Samba. Henning Schou retter en sproglig bøf. Niels Jalling fanger problem med Mailman opsætningen.
- Version 2.7 - 7. oktober 2003: Henrik Grove forbedrer en reference. Peter Toft tilføjer nyt afsnit om hvordan man redder en forsvundet partitionstabel samt nyt afsnit om mailman som anvendes til at lave postlister. Rasmus Terkelsen fanger en tastefejl. Jacob Sparre Andersen: Indsætter afsnit om at finde versionsoplysninger på Debian- og Red Hat-systemer. Retter nogle sproglige fejl. Udvider stikordsregistret en smule. Gjort opmærksom på at Leafnode er med i Debian. Henrik Grove retter en fejl om quotaon. Peter Toft retter et par småting og opdaterer lidt i DHCP-afsnittet grundet nye parametre. Troels Arvin retter en "make" bøf.
- Version 2.6 - 6. april 2003: Peter Toft om test af mail-server. Niels Elgaard Larsen har lavet nyt afsnit om at få en Linux-maskine til at være stille. Jacob Sparre Andersen - skrevet om **mknod**. Erik Søe Sørensen retter sprog. Keld Simonsen starter nyt kapitel om måling med MRTG.
- Version 2.6 - 1. december 2002: Svante Vinther har rettet Samba-opsætning så ææå virker fra Windows-klient. Jacob Sparre Andersen: Skrevet mere om **dig**. Peter Toft: Lidt opdateret mht. SuSE. Opdatering om SMTP-forward under Postfix. Mere om migrering af ext2 til ext3. Poul-Erik Hansen redigerer afsnit om DHCP-server og tilføjer information om at drive mere end en DHCP-server i nettet. Keld Simonsen retter et par sproglige fejl. Claus Boje tilføjer mere information om mii-diag. Keld Simonsen tilføjer mere information om LTSP.
- Version 2.5 - 1. september 2002: Henrik Skov Midtiby retter en masse fejl. Jesper Norskov Kristensen retter et par fejl. Hans Schou har tilføjet noget om syslog-remote og postfix-virtual. Jacob Sparre Andersen - Harmoniseret brugen af **tar**. Udbedret logiske fejl i SGML-koden. Udbygget omtalen af ‘badblocks‘ med lidt om Linuxcares "bootable business card" og lidt ekstra om hvordan man finder ud

af hvad man har af filsystemer på sin maskine. Keld Simonsen tilføjer kapitel om tynde klienter - se Kapitel 9. Peter Toft: Retter "rpm --rebuild" til "rpmbuild --rebuild" efter input fra Red Hat. Link til www.LinuxVirtualServer.org ind. Flyttet afsnit om procmail til unix-bogen. Oprettet nyt kapitel om e-post-servere. Har skrevet nyt afsnit om **at** til at køre kommandoer på givne tidspunkter - inspireret af input fra Henrik Christian Grove. Morten Liljeberg tilføjer afsnit om exim. Roy Sigurd Karlsbakk retter kommando til at lave RAID-opsætning. Henrik Christian Grove har påbegyndt et kapitel om netværksovervågning med et afsnit om opsætning af Big Sister.

- Version 2.4 - 14. juni 2002: Anna Jonna Armannsdottir laver nyt afsnit om sikret DNS - se Afsnit 6.12. Mere om lm_sensors fra Anna Jonna Armannsdottir. Peter Toft: Nyt afsnit om rsync - se Afsnit 5.9. Link til SAMBA PDC-artikel. Har skrevet nyt om lsof, og andre måder til at klare problemer med lås af filsystemer - dette er direkte inspireret af Ole Tange og Mogens Kjærs indlæg på sslug-misc@sslug.dk. Ole Tange opdaterer afsnit om Postfix og tilføjer afsnit om af fejlfinde programmer - se Afsnit 3.8. Peter Toft: Ud med afsnit om ISC DHCP version 2. Morten Christensen har skrevet en del mere om styring af cron-jobs med crontab. Henrik Christian Grove har omstrukturet Sambakapitlet. Svend Erik Venstrup og Hans Christian Andersen fanger fejl. Jacob Sparre Andersen: Rettet sproglige småfejl.
- Version 2.3 - 10. marts 2002: Ny licens for bogen - Åben dokumentlicens. Hanne Munkholm har skrevet en meget nyttig vejledning til håndtering af defekte blokke på harddisken. Jesper Krogh skriver mere om procmail. Michael Rasmussen: Tilføjet afsnit om Dynamisk DNS fra Kristian Vilmann. Claus Boje og Peter Toft: Mere om rpm-kommandoer. Rasmus Laursen retter fra insmod til modprobe. Peter Toft: Skrevet en del nyt om kerne-oversættelse. Tilføjet link til artikel om tråde og processer. Har skrevet nyt afsnit om lm_sensors til måling af CPU-temperatur. Christian Treldal tilføjelse med tar-kommandoen En stribe nye stikord fra Svend Erik Venstrup. Kim Futtrup Petersen bidrager med opdaterede billeder af kerneoversættelse.
- Version 2.2 - 29. december 2001: Lars K. Schunk har lavet mange hundrede sproglige rettelser i hele bogen - en stor indsats! Jacob Sparre Andersen: Sproglige smårettelser. Peter Rasmussen retter omkring xdm. Svend Erik Venstrup finder en dum slåfejl og masser af nye stikord. Peter Toft har tilføjet lidt om "High Availablity", mii-tool, LDAP, date og hvordan man tilføjer brugere og har sammen med Torkil Zachariassen skrevet nyt afsnit om styring af diskforbrug med quota. Peter Toft inspireret af Hanne Munkholm; links til mere info om kickstart.
- Version 2.1 - 21. oktober 2001: Peter Toft: Nyt om tidszone-valg og links til gode referencer om opsætningsfilerne i `/etc` Nyt afsnit om DPMS. Lidt mere om ftp-server. Rettet op på NFS-afsnit om tilføjet nyt om "showmount". Afsnit om GPG flyttet til unix-bogen. Nyt afsnit om opgradering af kernen, inspireret af Mogens Kjær og Henrik Størners diskussion på sslug-teknik. Lidt om søg-og-erstat i filer (inspireret af Kristian Vilmann). Kim Futtrup Petersen har tilføjet rettelse i Samba afsnit om dansk tegnsæt for Windows 2000. Henrik Størner har skrevet lidt om opstart af Linux. Jacob Sparre Andersen: Rettet enheder til (MHz = megahertz, Mbit = megabit, Mb = megabyte). Rettet lidt sprog. Martin Egholm Nielsen fandt en dum trykfejl. Anna Jonna Armannsdottir har bidraget med kerne-config-fil til eksempel på `/usr/src/linux/.config`. Lars K. Schunk har lavet mange sproglige rettelser. Michael Lerskov Munk Nielsen har lavet en tilføjelse angående DHCP-server for maskine med to netkort, hvor man kun ønsker at tildele IP-adresser til det ene netværk. Leo Laursen har givet et smart procmail-filter til at afsløre potentielle post-vira.
- Version 2.0 - 11. august 2001: Christian Jørgensen fanger en trykfejl. Peter Toft: trykfejl. Har flyttet afsnit om grafiske system-værktøjer til "Linux - friheden til at vælge". Har opdateret afsnit om hdparm, så man kan få mere tryk på UDMA66 IDE-diske. Gunner Poulsen, Lars Scheele Jensen og Michael Rasmussen har hjulpet til med dette. Henrik Størner har givet lov til at inddrage hans

vejledning http://www.sslug.dk/rpm/build.html i at bygge RPM-pakker. Peter Toft har tilføjet afsnit skrevet af Kristian Vilmann (fra sslug-teknik) om at bygge RPM ud fra kilde-RPM-filer.

- Version 1.9 - 9. juli 2001: Peter Toft: conf.modules -> modules.conf i kerne-afsnit. Lidt om dynamisk DNS/DHCP. Atte André Jensen: ombytning af parametre til ncftpput. Jesper Krogh: Nyt afsnit om NTP. Owen Brotherwood: Advarsel mod flere DHCP-servere i samme netværk tilføjet. E. Sjørlund tilføjer mere om fetchmail. Erik Søe Sørensen og Claus Hindsgaul: Sproglige ændringer. Kim Hermansen: Placering af Samba password-fil. Ole Tange fangede en stribe format-fejl.
- Version 1.8 - 24. maj 2001: Peter Toft: Mindre rettelser i DHCP-afsnit. Rod i hvad Red Hat 7.0 skal have. Erling Sjørlund retter til så leafnode under SuSE 6.4 er beskrevet bedre. Mere om kerne 2.4 problemer med at se alle hjemmesider. Jacob Sparre Andersen: Rettet en henvisning i wget-afsnittet på opfordring af Svend Erik Venstrup. Mindre rettelser fra Svend Erik Venstrup. Magnus Østergaard forbedrer teksten omkring DHCP-serveren. Erik Søe Sørensen retter et par fejl om kerne 2.4. Rasmus Laursen laver nyt afsnit om devfs (kerne 2.4-devices). Peter Seidler og Thorbjørn Ravn Andersen retter indhold om kerne 2.4.
- Version 1.7 - 15. april 2001: Peter Toft: Sproglige rettelser. Har flyttet afsnittet om slrn til "Linux - Friheden til at vælge". Smårettelser af Erik Søe Sørensen og Frank Damgaard. Erling Sjørlund har tilføjet lidt mere om Samba. Mindre eksempel på DNS er tilføjet til eksempel-filerne.
- Version 1.6 - 12. marts 2001: Forlaget IDG har bidraget med professionel korrekturlæsning af hele bogen, som Peter Toft har rettet op. Dette har betydet et markant sprogligt løft af bogen. Afsnit om pakke-formater er flyttet fra "Linux - Friheden til at vælge" til denne bog. Kim Futtrup Petersen har skrevet om nye Ximian sysadm-værktøjer - Peter Toft har tilsvarende skrevet om tilsvarende KDE 2.1 værktøjer. Nyt kapitel :-) Jørgen Ramskov og Jacob Sparre Andersen har rettet sprog. Carsten Svaneborg har rettet afsnit om system-opstart til. Claus Sørensen har lavet mange gode rettelser i netværks- og server-kapitlerne, samt tilføjet nyt om Midnight Commander til at hente fra ftp-servere. Hans Schou har tilføjet flere stikord. Henrik Christian Grove fandt en dum trykfejl i oversigt over bøger. Kasper Hauge fandt en Red Hat 7.0 fejl. Erik Søe Sørensen kom med en god stribe rettelser. Kapitlet om cd-rom-brænding flyttet til "Linux - friheden til at vælge programmer".
- Version 1.5 - 4. februar 2001: Peter Toft: Ny bog om Docbook nævnt i serien. Større omrokeringer af afsnit, så det blev mere logisk. Har skrevet om proc-filsystemet og kerne 2.4. Rettelser i flere omgange fra Donald Axel, Jørgen Ramskov og Asbjørn Laurberg. Nyt kapitel om Linux-kernen. Anvender dele fra "the Linux way", som er skrevet af Hanne Munkholm et al. Rykket TCP/IP+DNS-afsnit til hver sit nye kapitel, og firewall-afsnit vil senere blive integreret i bogen "Linux - Friheden til sikkerhed på internettet". Tommy Mogensen fandt en sprogbøf. Troels Liebe Bentsen forbedrer billeder. Poul Petersen har lavet rigtig mange sproglige opdateringer til afsnittet om dørvogtere over flere omgange. Flyttet kapitel om emulatorer til applikations-bogen. Jacob Sparre Andersen: Rettet diverse sproglige fejl over flere omgange. Christian Rasmussen har rettet et par fejl op i DNS-kapitlet. Claus Sørensen har skrevet afsnit om Linux som ring-ind-server og om sikkerhedskopiering. Claus har også lavet korrekturlæsning på mange af de nye afsnit.
- Version 1.4 - 29. december 2000: Flyttet afsnit om MP3 til "Linux - friheden til at vælge applikationer". Magnus Østergaard har lavet tilføjelse til DHCP-server om "option domain-name" og "option domain-name-servers". Link til den nye bog "Linux - friheden til at programmere i C".
- Version 1.3 - 11. november 2000: Morten Christensen og Ole Tange har opdateret info om ReiserFS.
- Version 1.2 - 26. oktober 2000: Christian Rasmussen har revideret hele afsnittet om TCP/IP og introduktionen til DNS. Sproglige rettelser fra Frank Damgaard. Ekstra om kommandoen pidof - tak til Morten Christensen, Erik Sørensen, Jesper Krogh, Gitte Wange, Emil S Hansen og Jakob Hilmer.

Navne og servernavne er lavet om efter utallige opfordringer. Jesper Krogh har tilføjet mere om at brænde en cd-rom. Peter Frederiksen har tilføjet lidt mere om RAID. En bunke sproglige rettelser fra Jacob Sparre Andersen (bl.a. password->adgangskode).

- Version 1.1 - 30. august 2000: Mindre rettelse i Samba-afsnittet fra Michael Rasmussen. Fejl rettet mht. hvor man kan hente de firewall-scripts som nævnes i teksten. Tak til Alex K også for fejl i RPM-fil til NFS. Poul Petersen har lavet en lang række sproglige rettelser. Nyt afsnit om at brænde cd-rom oversat af Gert Holtoft. Peter Lund har filføjet lidt mere om harddisk tuning (-k1 og brug af noatime).
- Version 1.0 - 30. juli 2000: Jesper Krogh har skrevet om nyhedsserveren Leafnode og om nyhedslæseren slrn. Afsnittet om AppleTalk er lavet Anders Brownworth, og Kim Futtrup Petersen har oversat og bearbejdet teksten. Poul Petersen har lavet sproglige rettelser. Peter Frederiksen har skrevet lidt mere om Raid. Michael Rasmussen har lavet en solid revision og udvidelse af Samba-afsnittet.

Stikordsregister

Symboler

A

B

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z