FreeBSD

8 december 2004

Innehåll

1 Förord

Föreliggande arbete syftar till att delge den grundläggande kunskap en systemadministratör behöver för att upprätta ett komplett nätverk helt och hållet baserat på det unixliknande operativsystemet FreeBSD. Boken kan användas för självstudier eller utgöra undervisningsmaterial i en kurs inom datorteknik med nätverk eller dylikt. Innehållet berör installation och daglig administration av FreeBSD men syftar i huvudsak till att presentera hur ett nätverk på en arbetsplats kan upprättas bestående enbart av FreeBSD.

Det finns böcker skrivna om hur FreeBSD kan användas som serveroperativsystem i ett nätverk bestående av Windows-klienter. Däremot saknas det böcker som tar steget fullt ut och använder enbart FreeBSD – alltså både som server och klient. Det är idag fullt möjligt att framgångsrikt bestycka en hel arbetsplats, datorsal eller laboratorium med FreeBSD.

Min förhoppning är att denna bok på ett tillräckligt utförligt sätt beskriver hur FreeBSD kan användas i en sådan miljö. Att FreeBSD kan användas — och används — till en mängd uppgifter står utom tvivel. På de senaste åren har FreeBSD vuxit från att vara ett operativsystem enbart för avancerade användare till att även passa som personlig arbetsstation för en bredare allmänhet.

Syftet är inte att vara en detaljerad användarmanual utan mer en inspirationskälla till vidare egna undersökningar. Ett antal grundläggande konfigurationer visas dock.

2 Introduktion

Boken behandlar de delar som är relevanta för att upprätta en praktisk arbetsmiljö bestående av ett serverbaserat nätverk bestående av enbart FreeBSD och öppna programvaror. Den valda vägen innebär att jag ibland hoppar över material som torde ingå i en mer ”traditionell” framställning. Texten gör inget anspråk att vara heltäckande inom något ämne men är förhoppningsvis tillräckligt noggrann och beskrivande för att på ett klart och enkelt sätt presentera de grunder som behövs för att sätta upp och administrera ett FreeBSD-baserat nätverk. Betoningen ligger på generella, långsiktigt användbara kunskaper. Ett studium av FreeBSD enligt denna modell ger inte bara kunskap om FreeBSD utan även kunskaper som kan tillämpas i många andra miljöer.

Boken inleds med en kort introduktion till vad FreeBSD är och lite om dess historia. Sedan följer kapitel som ska underlätta för den första installationen och något om XFree, det förhärskande fönstersystemet hos fria operativsystem. Resten av boken ägnas åt nätverk, nätverkskonfiguration och några vanliga nätverkstjänster. Tyngdpunkten är mot praktisk tillämpbar kunskap, bland annat visas hur FreeBSD kan användas som webserver.

De senare kapitlen innehåller flera utdrag ur konfigurationsfilerna från en fungerande server att tjäna som underlag för egen installation. Det finns i allmänhet mycket stora möjligheter till individuell konfiguration och det är således inte säkert att de presenterade utdragen är optimala i varje enskilt sammanhang.

Läsaren förutsätts vara flitig, nyfiken och relativt ”självgående”. Då det behandlade ämnet är omfattande och under ständig utveckling är ett nyfiket och undersökande arbetssätt mycket viktigt.

Studera alltid de filer som medföljer använda program, mycket nyttig information döljer sig där. De referenser som texten hänvisar till är beskrivna i slutet av boken och de flesta är mycket läsvärda. Jag förutsätter att läsaren själv kan ta reda på den information jag utelämnat genom att söka efter den bland annat genom att studera referenslitteraturen, vilken förhoppningsvis ger nödvändiga grunder för att studera vidare på egen hand. De kommandon som anges med siffra inom parentes, exempelvis `ls(1)`, finns beskrivna i systemets manualsidor.

2.1 FreeBSD:s ursprung

Ursprunget till FreeBSD kan spåras tillbaka till december 1993. Då släpptes FreeBSD 1.0 med relativt lyckat resultat. Nästa version, FreeBSD 1.1, kom i april 1994 och fick mycket stor spridning. Systemet har utvecklats mycket sedan dess, med version 2.0 i december 1994, version 3.0 i oktober 1998 och version 4.0 i mars år 2000. Utvecklingen har inte stannat av utan fortsätter med oförminskad styrka, version 5 väntas kunna släppas under 2002.

FreeBSD använde från början mycket av den källkod som utvecklats av Berkeleyuniversitetets CSRG, Computer Systems Research Group, sedan 1970-talet. Deras källkod kallades Berkeley Software Distribution eller kortare BSD. På grund av vissa patentstrider kunde inte hela CSRG:s kod användas direkt för FreeBSD men flera väsentliga delar var dock tillgängliga. CSRG hade vid den tiden släppt 4.4BSD men var tvungna att rensa ut den kod som inte gick under BSD-licensen ur den. Den kvarvarande koden kallades 4.4BSD-Lite. En effekt av detta är att FreeBSD går under den unika BSD-licensen som ger användarna full frihet att göra vad dom vill med koden. Licensen återges i sin helhet i appendix.

Det utvecklingsarbete som CSRG genomförde fick stor spridning som framgår av stamtavlan i Figur 2.1. Flera kommersiella operativsystem baserar sig också på BSD.

Den fria spridningen av systemet och källkoden och den liberala BSD-licensen kännetecknar FreeBSD och dess användare än idag. Senare års framgångar hos Open Source-rörelsen passar också väl in i FreeBSD:s tänkesätt. På diskussionsgrupper och epostlistor på Internet sker informationsutbytet fritt. Det finns ingen central support på programvaran, men uppstår problem eller frågor kan man via Internet nå miljontals användare världen över. Sannolikheten är stor att någon kan svara på just dina frågor. I Sverige finns en svenskspråkig epostlista för användare av BSD. Den administreras av föreningen BUS - BSD Users Sweden - och är ett praktiskt sätt att träffa likasinnade inom landet.

2.2 Definitioner

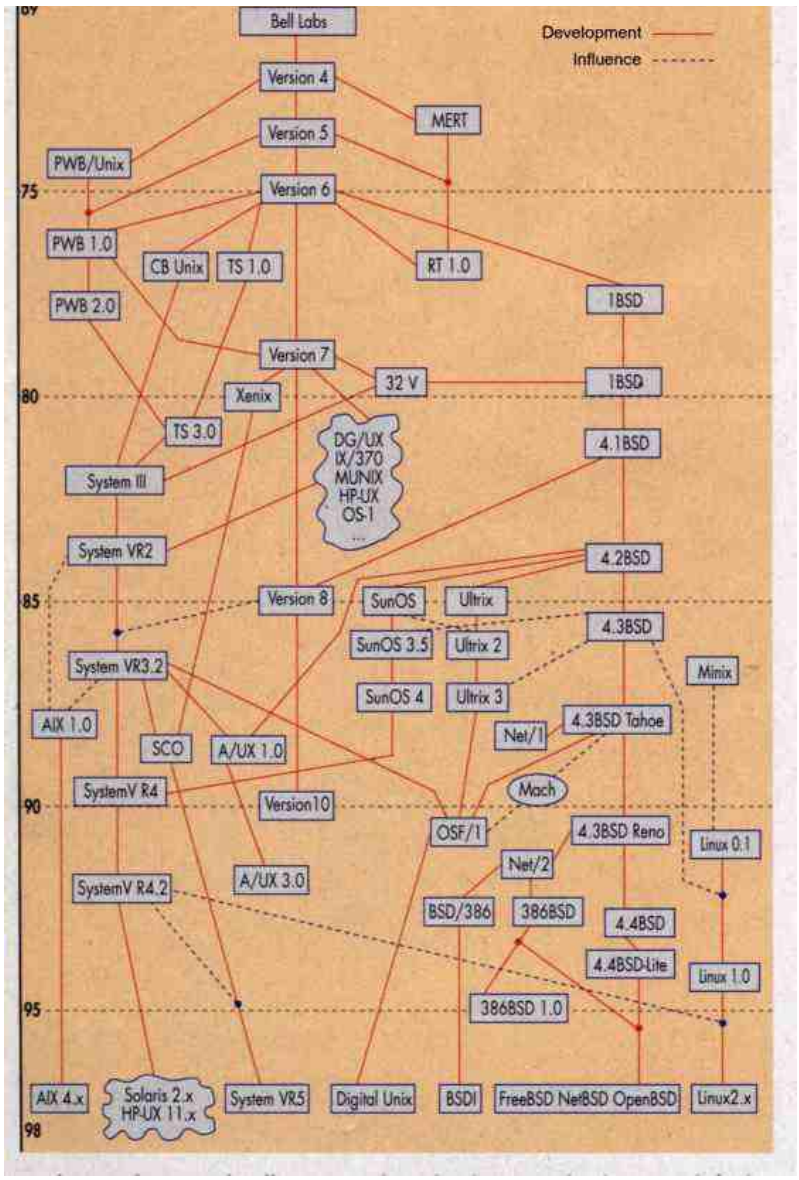
FreeBSDserver-FreeBSDklient detta.

Använda typsnitt, megabyte=MB, Gigabyte = GB

uppdelning server-client, root-andra

2.3 Vad är FreeBSD?

FreeBSD är ett generellt unixliknande operativsystem. Det baseras på Berkeley-universitetets 4.4BSD Lite-distribution från början av 1990-talet. Samma



Figur 1. Källkoden från CSRG, Computer Systems Research Group, fick stor spridning. Lägg märke till 4.4BSD nere till höger som är ursprunget till FreeBSD.

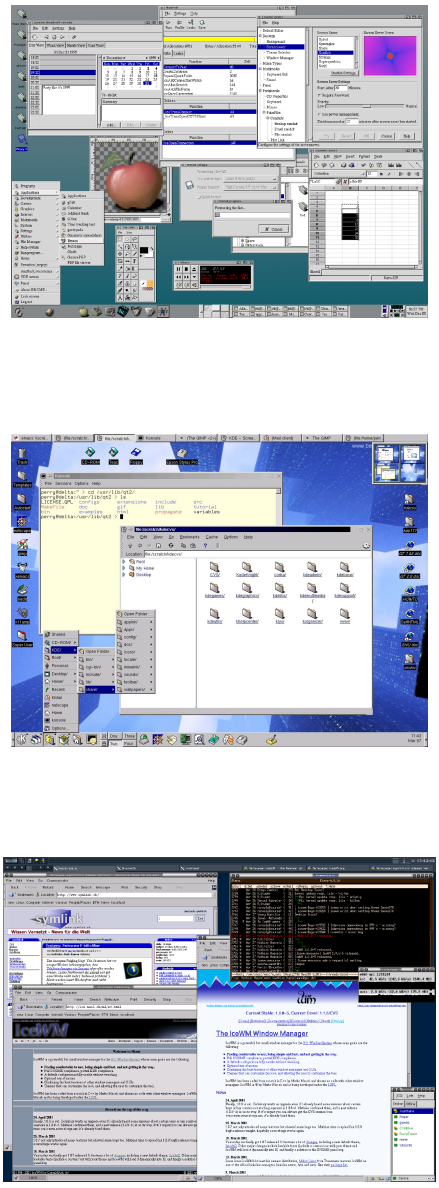
teknologi har varit grunden för flera kommersiella operativsystem, exempelvis Ultrix och SunOS.

Det finns ett antal fördelar med unix-liknande operativsystem och FreeBSD är inget undantag. Bland några väsentliga egenskaper hos FreeBSD kan nämnas

- Fleranvändarstöd. Flera - dussintals - användare kan vara inloggade mot samma dator samtidigt.
- Säkerhet, både för systemet i sig och de användardata systemet hanterar.
- Pålitlighet. FreeBSD är känt för sina långa gångtider. Det är inte ovanligt att operativsystemet bara går och går, år efter år, utan omstarter!
- ”The principle of least surprise”. Operativsystemet gör det man säger åt det att göra och försöker inte ”hjälpa till” på oförutsägbart sätt.

En vanlig missuppfattning är att det inte finns några program att köra på en unixdator. Detta är fel. I FreeBSD:s paketsystem finns det fler program att installera än man hinner prova, inom en mängd olika kategorier. Några av kategorierna är backup och komprimering av data, astronomi, biologi och matematik, databaser, CAD, programspråk, spel, videokonferenser, datasäkerhet och ordbehandling. Praktiskt taget alla Linuxprogram kan köras under FreeBSD tack vare den inbyggda ”linuxulatorn”. Dessutom finns ett flertal fönsterhanterare som erbjuder i det närmaste oändliga möjligheter till konfigurering av skrivbordet. För närvarande (2004) återfinns mer än 10 000 program färdiga att installera.

Internetprotokollet TCP/IP utvecklades på datorer som körde operativsystemet BSD och FreeBSD har ärvt mycket av den källkoden vilket gör FreeBSD till ett välbeprövat nätverksoperativsystem. Sin största spridning har FreeBSD hittills haft inom olika nätverksfunktioner på företag där det uppskattas bland annat på grund av sin enastående pålitlighet och funktion. Tills tämligen nyligen återfanns FreeBSD på skrivbordet bara hos datorentusiaster och programmerare. Emellertid har en stor förändring skett de senaste åren. Dels har fönstersystemen (det som utgör ”skrivbordet” på skärmen) utvecklats så att de mycket väl kan konkurrera med kommersiella varianter, dels finns numera lättanvända program för de vanligaste kontorsuppgifterna ordbehandling, kalkyl, databas med mera. Tiden är alltså mogen att låta FreeBSD komma in i värmen på våra kontor och skrivbord.



Figur 2. Det finns flera moderna fönsterhanterare för FreeBSD. Här visas KDE, GNOME och icewm, men det finns många fler att välja mellan. Samtliga kan anpassas efter eget tycke och smak.

FreeBSD går under den så kallde BSD-licensen (Berkeley Software Distribution) som gör att all källkod kan levereras med systemet, och det dessutom gratis. BSD-licensen betyder i stort ”här är koden, gör vad du vill med den”. Det är svårt att hitta någon licens som är friare än så!

2.3.1 FreeBSD i företag

Beroende på företagets verksamhet kan man ha mindre eller större nytta av FreeBSD. Till de klassiska uppgifterna, web- och mailserver samt router och brandvägg ut mot Internet, är FreeBSD en lysande kandidat. Om företaget vill ha inringande linjer kan FreeBSD även konfigureras som PPP-server. Andra ofta förekommande uppgifter är backup- och skrivarserver. Med upptider som kan räknas i år, inte veckor eller dagar, borgar FreeBSD för en pålitlig server oavsett tillämpning.

Det råder ingen tvekan om att ordbehandling och epost är bland de mest använda funktionerna för datorer, oavsett företag. FreeBSD kan användas till båda. Det är numera vanligt att man förser sitt intranät med en egen webserver för att sprida gemensam, lokal, information. Till exempel kan olika avdelningar presentera information för de anställda inom avdelningen och välja att visa annan information utåt. Projektgrupper inom organisationen kan centralisera nödvändiga data för egen koordination på webservern och använda den som anslagstavla för gemensamma dokument.

Man behöver dock inte begränsa sig till de klassiska uppgifterna med FreeBSD i ”bakgrunden”. Det fria StarOffice-paketet med sin WYSIWYG ordbehandlare och kalkylprogram kan ersätta andra Office-paket. StarOffice har även s.k. *filter* för att konvertera MS-Office-dokument. Ett antal rit- och bildmanipuleringsprogram och flera webb-läsare finns tillgänliga och det finns även en mängd epost-klienter anpassade för FreeBSD.

FreeBSDs traditionella kännemärken funktion, säkerhet och stabilitet kombinerat med moderna användargränssnitt och program resulterar i ett system som väl kan matcha andra operativsystem för skrivborden. Att dessutom FreeBSD går under BSD:s licens gör att en FreeBSD-lösning ofta kan erbjudas till en oslagbar kostnad.

2.3.2 FreeBSD i skolor

Skolor, från grundskola till universitet, kan likaväl som företag dra nytta av FreeBSD som web- och mailserver, och i och med StarOffice, har FreeBSD blivit ett attraktivt och billigt system även för de klientmaskiner elever och lärarkollegiet använder. En nyligen genomförd studie av Skolverket(ref) visar att, åtminstone vad gäller svensk skolverksamhet, datorn först och främst används för ordbehandling, följt av en viss del kalkyl. StarOffice har både ordbehandling och kalkyl men innehåller dessutom XXX.

Även inom andra områden kan FreeBSD hjälpa till. För matematikundervisningen finns program för symbolisk algebra, och programmeringskurser kan dra full nytta av FreeBSD oavsett om programspråket skulle vara C, C++, Fortran, Perl, Awk, Html, Python eller något annat. Vissa av dessa programspråk stöds med kompletta IDE-miljöer. För utbildningar i nätverkskommunikation kan FreeBSD med sin fullt utbyggda och beprövade TCP/IP-stack utgöra basen. Dessutom finns hela operativsystemets

erkänt välskrivna källkod (oftast skriven i C) tillgängligt för studium — och ändringar...

BILD på labsal?

Det är lätt att tänka sig fler användningsområden. FreeBSD kan användas för att skriva rapporter och specialarbeten. Klassen kan presentera sig med en egen hemsida. Varför inte lägga upp genomförda specialarbeten och rapporter på hemsidan?

Tack vare filsystemets stöd för behörigheter kan administratören ge behörighet för skrivning och läsning med mera, ner till filnivå. Systemet kan därmed konfigureras så att inte ens den mest klåfingrige elev kan förstöra något för någon annan, varken för systemet eller för någon annan elev.

2.3.3 FreeBSD hemma

Av tradition har FreeBSD alltid haft ett starkt fäste hos hemanvändarna. FreeBSD:s programvarusortiment utvecklas faktiskt fortfarande till stor del av entusiatiska hemanvändare som ständigt utvecklar och förfinar operativsystemet. Är man intresserad av programmering finns det ingen anledning till att inte ha FreeBSD hemma. Oavsett om din hobby är astronomi eller att spela nätverksspel har FreeBSD något för dig, och med ett vanligt modem erbjuder systemet dessutom en stabil och säker surfplattform för Internet. Har man turen att ha kabelmodem eller ISDN kan FreeBSD mycket väl tjänstgöra som brandvägg också.

Överhuvudtaget bör operativsystemets höga säkerhet ha en lugnande inverkan på hemanvändaren som oftast inte har tid eller kunskap att detektera och åtgärda intrångsförsök från *crackers*. Det senare blir mer och mer aktuellt då hemdatorerna allt oftare är ständigt uppkopplade mot Internet. FreeBSD är en mycket säker plattform och minimerar risken för datorintrång.

2.3.4 FreeBSD överallt?

Efter att ha varit en pålitlig arbetshäst i närmare tio år inom nätverk och industriella tillämpningar och med en stamtavla som sträcker sig över trettio år är FreeBSD ett vuxet operativsystem. Traditionellt har BSD använts som server för olika funktioner i nätverk och andra ”tekniska” tillämpningar. Fortfarande används FreeBSD huvudsakligen i den ”osynliga” delen av nätverk, som filserver, webserver med mera. De som använt FreeBSD har ofta varit specialister i sitt fack och ofta programmerare eller nätverkstekniker. På grund av BSD:s rötter i tidigt 1970-tal har varken användarvänlighet (för ”normala” användare) eller utseende varit prioriterat. Funktion, säkerhet och stabilitet har alltid kommit i främsta rummet då driftstörningar och instabila program varit absolut portförbjudna i dessa miljöer. Det är dessutom allmänt erkänt som ett operativsystem som inte

knäcks av tunga arbetsbördor - olikt flera av den senare tidens andra operativsystem.

Huvudsakligen har FreeBSD alltså återfunnits i bakgrunden och inte på skrivborden. Detta är inte förvånande, eftersom det inte funnits bra fönstersystem att tillgå. På senare år har dock situationen ändrats radikalt. Det finns numera flera dussin olika fönstersystem att välja mellan. De mest namnkunniga idag är nog KDE och Gnome. Men flera andra finns tillgängliga och det är lätt att byta från ett till ett annat.

Det pågår en kraftig utveckling av FreeBSD som inbyggt system, dvs system som är avskalat på flera av sina normala funktioner och som anpassats för att kunna köras på minimal hårdvara. Typiska tillämpningar för inbyggda system är till exempel webkameror, styrsystem till robotar och andra apparater i industrin.

Men inte bara små system är lämpade för FreeBSD. I andra ändan av tillämpningsområdet kan nämnas flerdatorsystem, *kluster*, där dussintals, kanske hundratals datorer med FreeBSD samarbetar för att lösa ett problem. Typiska problem kan vara vädersimulationer, beräkningar på biologiska och kemiska problem, och bildbehandling. Till filmen *The Matrix* upprättades ett FreeBSD-kluster med över 200 datorer för att kunna erbjuda den enorma datorkraft som behövs för att rendera bildsekvenser med biokvalitet.

Den allmänna acceptansen för Open Source-lösningar och den mångfald program som finns att tillgå för FreeBSD och de som ligger i startgroparna (Loki-games meddelar att de numera testar sina spel även mot FreeBSD) antyder en fortsatt livlig utveckling och spridning av operativsystemet. Den generösa BSD-licensen uppskattas av företag som, dels inte behöver betala licensavgifter, och dels inte behöver göra sin egenutvecklade källkod tillgänglig för andra.

2.3.5 När ska man inte använda FreeBSD?

Det kan tyckas att alla problem är lösta bara man inför FreeBSD. Tyvärr är det inte så. Det finns tillfällen då FreeBSD inte är en särskilt lämplig lösning.

- Redan framtagen eller inköpt programvara kanske inte fungerar på FreeBSD. Speciellt kan man inte köra Windows-program (MS-Office med Word/Excel/Access osv).
- Om man redan nu kör ett nätverk som inte är ethernetbaserat kan man också överväga om FreeBSD verkligen passar in. I FreeBSD finns visserligen stöd för andra nätverksprotokoll (NetBEUI, IPX eller Appletalk till exempel) men detta stöd är avsett för FreeBSD som server, inte som klient. Om du däremot vill ha en fil- eller skrivarserver för ett nätverk med dessa nätverksprotokoll så är FreeBSD ett utmärkt val. Denna bok behandlar dock inte sådana tillämpningar.

- Epost kan också utgöra ett potentiellt problem. Många windowsanvändare låter skicka sina ebrev med html-koder[1]. Om mottagarens epostklient har stöd för detta är allt lugnt och väl, men eftersom htmlbrev bryter mot den standard som gäller för epostformatet har nästan inga av FreeBSD:s epostklienter stöd för det. Undantaget är Netscapes webbläsare och epostklient, den kan både ta emot och sända htmlbrev. Använder man någon annan epostklient kanske man måste be sändaren skicka om brevet utan html för att det ska bli läsligt.

Det finns visserligen körmiljöer för både DOS och Windows (*bochs* och *dosemu* för DOS respektive *wine* för Windows) men i varje fall vad gäller den senare är stödet ibland inte tillräckligt. Om det ändå är nödvändigt att köra windowsprogram har *vmware*[2] fått goda vitsord. *Vmware* erbjuder möjlighet att samtidigt köra flera olika operativsystem på en och samma dator: Man kan till exempel köra Windows i ett fönster på sitt FreeBSD-skrivbord och ha båda systemen åtkomliga samtidigt.

Så långt som möjligt försöker programkonstruktörerna för FreeBSD, och unixvärlden i övrigt också, att följa öppna standarder och format. Den information som inte följer någon vedertagen, öppen standard kan därmed vara svår att hantera. Detta är inte för att trilskas utan är ett medvetet sätt att försäkra sig om största möjliga informationsflöde mellan olika användare och system. Somliga format hemlighålls dock av tillverkaren av konkurrensskäl och är inte allmänt tillgängliga, andra format kan kräva stora summor licensavgifter för att få användas. Oavsett vilket är detta helt i strid med den filosofi som Open Source-rörelsen står för.

MS-Words .doc-format är till exempel inte öppet tillgänligt och sådana filer kan vara problematiska att läsa. Om Microsoft släppte specifikationen för .doc-formatet, och flera andra format, skulle situationen förbättras över en natt, men så länge formaten hemlighålls kvarstår problemet.

Flera av de tilläggsprogram som finns gratis eller kan köpas till FreeBSD innehåller filter för konvertering mellan odokumenterade format och något öppet format, men då formaten just är odokumenterade kan filtren misslyckas med vissa dokument.

En standard som, i varje fall för textdokument, brukar vara acceptabel både för FreeBSD-användare och andra är RTF (Rich Text Format). RTF utgör en helt öppen standard framtagen av Apple, IBM och Microsoft med avsikt att vara så portabel mellan olika system som möjligt.

[1] Gör inte det!

[2] `http://www.vmware.com XXX`

3 Mot första starten

3.1 Hårdvara

FreeBSD ställer relativt små krav på den hårdvara som används. Med äldre och långsam hårdvara kan man med FreeBSD komma mycket långt. Det finns exempel på en 486:a[1] som dagligen sorterar och distribuerar 80 000 epostbrev! Det är också fullt rimligt att låta en gammal 486- eller pentiumdator utgöra filserver för ett nätverk med dussintalet datorer.

Medan FreeBSD i sig har relativt låga krav på hårdvaran kan det mycket väl vara så att de program man kör på systemet är väldigt processorkrävande och/eller minneshungriga: Alla fönstersystem tillhör en kategori av program som kan få en i övrigt snabb och kompetent dator att verka långsam. Fönstersystemet XFree är inget undantag, ska du köra XFree med full fart bör du åtminstone ha mycket, mer än 64 MB, minne. Det är i praktiken tveksamt om du vill köra XFree på någon dator långsammare än 100 MHz. För hemmabruk och experiment är det tillräckligt, men vänta dig inga applåder från professionella användare. Ett praktiskt minimal dator för XFree kan vara en 200 MHz processor med åtminstone 32 MB primärminne. I valet mellan snabbare dator eller mera minne – satsa på mer minne. En av de tidiga flaskhalsarna (swap:ning) beror just på för litet primärminne. Åtminstone 64 MB primärminne behövs innan det kan vara idé att överväga snabbare processor, för XFree:s skull.

Utan fönstersystem kan FreeBSD köras på datorer med så lite som 4-8 MB primärminne beroende på vilken version som används[2].

Köper man en ny dator idag, bärbar eller bordsmodell, räcker den med råge vad gäller prestanda för XFree. Om du tvingas använda äldre hårdvara kan FreeBSD trimmas för bästa prestanda. En bok som mycket utförligt diskuterar detta är *System Performance Tuning* av Mike Loukides[3]. Emedan boken behandlar unixsystem i allmänhet är den en ypperlig introduktion till hur program använder hårdvaruresurser och ger också konkreta tips för prestandahöjning även i BSD-miljöer.

[1] Intels processor 80486 var högsta mode i mitten av 1990-talet. I sanning en antikvitet i datorsammanhang!

[2] FreeBSD version 2.2.x behöver 5MB primärminne för att kunna installeras, sägs det. Jag har installerat det med bara 4MB, jag kanske hade tur. Senare versioner, 3.x och 4.x, behöver dock mer minne: 8MB för att köras och 12MB för att installeras.

[3] O'Reilly, 198X. ISBN XXX.

3.1.1 Hårddisk

Av det föregående kan man tro att det inte är någon skillnad på hårdvara och hårdvara. I allmänhet är detta sant, mycket lite hårdvara är direkt felaktig eller har så låg tillförlitlighet att man bör undvika den. Det finns dock fortfarande en speciell del av en dator där kvaliten varierar betydligt: hårddisken. I stort är det ingen skillnad i kvalitet på olika datorkomponenter men när det gäller hårddiskar rekommenderas en av SCSI[4]-typ, åtminstone på servern.

SCSI-hårddiskar är tillförlitligare än andra hårddiskar. ”Andra” betyder i det här fallet numera praktiskt taget alltid hårdiskar av IDE-typ. Kvaliteten och tillförlitligheten hos IDE-hårddiskar är högre idag än vad den var, men är man intresserad av högsta tillförlitlighet[5] och prestanda är det fortfarande SCSI som gäller. Tyvärr är dessa diskar ungefär dubbelt så dyra som konventionella av IDE-typ och det kan vara svårt att motivera den extra kostnaden på klientmaskinerna. Men servrar bör i varje fall utrustas med SCSI-hårddiskar, om det finns ekonomi för det.

SCSI-hårddiskar kan inte anslutas direkt till moderkortet i datorn utan ansluts via en sk *SCSI-adapter*. En SCSI-adapter utformas som ett instickskort med ISA- eller, numera, PCI-busskontakt och finns av i huvudsak två slag, med eller utan BIOS. Om datorn ska kunna startas från SCSI-hårddisken, till exempel för att det bara finns en disk i datorn och detta är en SCSI-disk, måste SCSI-adaptern ha BIOS. Om systemet däremot startas från en IDE-disk och man först senare behöver komma åt en SCSI-enhet räcker det med ett adapterkort utan BIOS.

Ett adapterkort ger möjlighet att ansluta flera enheter till datorn. Äldre typer av adapterkort stödjer 8 SCSI-enheter, nyare upp till 16. När man räknar SCSI-enheter räknas adapterkortet själv som en enhet, ofta den sjunde, och i verkligheten kan man alltså ansluta 7 respektive 15 andra enheter till ett dylikt kort. Detta är i allmänhet fullt tillräckligt. En ytterligare anledning till att ha SCSI på servern är att många högkvalitativa serverspecifika hårdvaror, som backupstationer till exempel, kräver det.

3.1.2 Förberedelser

Inga särskilda förberedelser på datorn behöver göras. Installatören bör dock läsa den sista-minuteninformation som återfinns bland installationsdokumentationen på CD:n eller FreeBSD:s hemsida[6].

För den här installationen förutsätts att en hel hårddisk finns tillgänglig och att det är den enda hårddisken i systemet. Det är i och för sig inga större svårigheter att installera FreeBSD på en egen partition, så att operativsystemet kan samexistera med ett annat operativsystem, men för att tillåta

[4] *Small Computer Systems Interface*, en industristandard som utvecklats i flera generationer. Den senaste (160UW) hanterar överföringshastigheter på 160MB/sekund!

[5] SCSI-hårddiskar säljs ofta med 5 eller ända upp till 7 års garanti!

[6] `http://www.FreeBSD.org`

viss ”slarv-mån” vid denna förstagångsinstallation är det säkrast med en egen hårddisk. Det kan också vara, åtminstone mentalt, lättare att starta om installationen och formatera hela hårddisken på nytt utan att behöva oroa sig över eventuell förlust av data, om det skulle visa sig nödvändigt.

Det förutsätts att du har en CD att installera från, detta är det enklaste tillvägagångssättet och CD:n kommer du att ha nytta av senare också. Så antingen förser du dig med CD:n genom att tanka hem en ISO-fil från Internet[7] och bränna den eller också har du köpt en distribution från någon leverantör. I slutet av boken finns en leverantörslista.

Installationen kan påbörjas på två sätt: med disketter eller utan disketter. Om du har en distributions-CD och din dator stödjer start från CD är du nu redo att fortsätta. Om du däremot har en äldre dator som inte stödjer CD-start behöver du först skapa två sk *boot*-disketter. Instruktioner för detta finns på CD:n i filen `floppies/README.TXT`.

3.1.3 Dual boot - dubbla system

Om du ändå — trots varningen ovan — vill installera på en hårddisk med befintligt operativsystem, det må vara Windows, OS/2, Linux eller något annat, måste du själv se till att en ledig primär partition på åtminstone 300 MB finns tillgänglig. Det går att installera en minimal FreeBSD på så lite som 100 MB, men det måste fortfarande vara en primär partition. FreeBSD kan inte installeras på en utökad partition. FreeBSD kan mycket väl, efter installation, *använda* utökade partitioner, men alltså inte *installeras* på en sådan.

För att välja operativsystem vid systemstart måste dessutom en boot-manager installeras. Med FreeBSD följer boot-managern ”Booteasy” som kan starta både primära och sekundära partitioner, men du kanske redan har en befintlig manager, PartitionMagic eller LILO eller nåt, installerad? Vid installationen kommer du att få frågan om du vill installera ”Booteasy”, välj i de senare fallen alternativet ”Don’t touch the boot record”, annars kommer din nuvarande boot-manager att oåterkalleligt skrivas över.

3.1.4 Hårddisken igen

För att förstå installationsproceduren i sin helhet, och de textmeddelanden som där förekommer, behöver man veta lite om hårdisken och dess partitioner. Även om en hårddisk utåt kan se ut som en enda enhet kan den internt uppdelas, *partitioneras*, i flera separata ytor. Vardera yta kan innehålla ett eget operativsystem om man så vill.

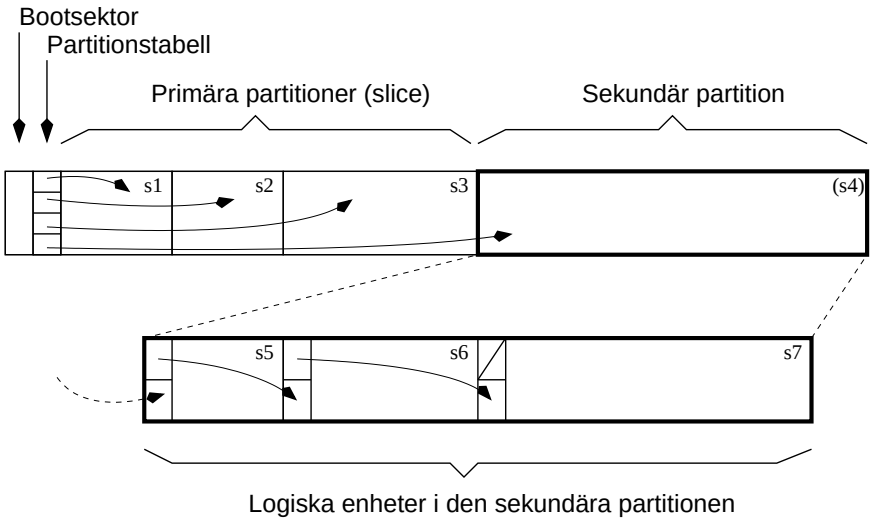
Först på en hårddisk ligger MBR, *Master Boot Record*, och diskens partitionstabell. MBR innehåller ett litet kort program som kan dirigera datorns

[7] Välj den senaste från `ftp://ftp.FreeBSD.org/releases/i386/ISO-images/`. Den här boken har använt version 4 (dvs exempelvis`4.10-install.iso`)

uppstart till en viss partition, den *aktiva partitionen*. För att datorn ska kunna starta från den aktiva partitionen krävs dessutom att denna partition innehåller ett operativsystem. I vissa fall kan MBR innehålla en bootmanager som tillåter användaren att vid start välja vilken partition som ska vara den aktiva. På så sätt kan en och samma dator innehålla flera operativsystem, även om endast ett kan köras åt gången. Man kan till exempel ha både Windows, OS/2, FreeBSD och Linux på samma dator om man så önskar[8].

Partitionstabellen innehåller information om hur disken är uppdelad i partitioner och den har plats för pekare till fyra partitioner. För att kunna ha fler än fyra partitioner på en hårddisk finns det två sorters partitioner, *primära* och *sekundära* (de senare kallas även *utökade* ibland). En primär partition är endast ”ett stycke hårddisk” medan en sekundär partition inleds med en egen partitionstabell. En sekundär partition kan dessutom innehålla flera (sekundära) partitioner, vilka var och en inleds med en egen partitionstabell. Avsikten med detta arrangemang är att låta hårddiskens partitionstabell peka ut en primär eller sekundär partition. Om den pekar ut en primär partition är allt klart för åtkomst av data på denna partition. Om den däremot pekar ut en sekundär partition pekar den i själva verket på den partitionstabell som inleder den sekundära partitionen. Först efter att den sekundära partitionstabellen analyserats är det möjligt att avgöra om data kan nås eller om ytterligare en sekundär partitionstabell ska analyseras. Vi har nu ett fall där en bild verkligen säger mer än tusen ord:

[8] Windows och FreeBSD på primära partitioner, OS/2 och Linux även på sekundära partitioner.



Figur 3. Hårddiskens uppdelning i partitioner (något förenklat). Partitionstabellen kan innehålla pekare till högst fyra partitioner. En av dessa partitioner kan utgöras av en sekundär partition som i sin tur kan innehålla flera logiska enheter. De olika partitionernas namn anges i respektive ruta.

FreeBSD kan som nämnts tidigare enbart installeras på en primär partition. I denna partition skapar sedan FreeBSD automatiskt, eller installatören manuellt, partitionerna `/usr`, `/var`, `/`, och `swap`.

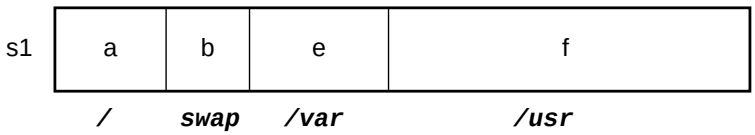
Lägg märke till att det finns två språkbruk som båda använder ordet partition, fast med olika betydelse! Eftersom vi ”pratar” FreeBSD kommer vi hädanefter att använda orden slice och partition som de används i BSD.

Partitioner är alltså det som FreeBSD skapar inom en slice. Och slice är det som DOS-sammanhang kallas partition.

Innan vi lämnar hårddisken ska vi kort beröra hur de olika hårddiskarna, slice:arna och partitionerna namnges i FreeBSD.:

- Den första SCSI-disken benämns `da0`, den andra `da1` och så vidare.
- IDE-hårddiskar kallas `ad0`, `ad1`, `...` enligt mönstret ovan.
- Inom en hårddisk numreras de olika slice:arna med 1, 2, ... Så kommer första SCSI-hårddiskens andra slice att benämnas `da0s2`. På detta sätt numreras de upp till fyra primära partitionerna. Efter dessa tar den första sekundära vid och den kallas följaktligen `da5`, följd av `da6` osv.
- Inom en slice ”numreras” partitionerna med bokstäverna `a`, `b`, `c`, `d`, `e`, `f`, `g` och `h`. En komplett specifikation kan alltså vara `ad0s2f`. En slice kan aldrig innehålla fler än åtta partitioner.

Av något vi enklast kallar ”historiska skäl” brukar FreeBSD:s partitioner ha vissa fasta betydelser: `a` innehåller vanligen ett filsystem och brukar utgöra systemets bootbara partition (`/`), `b` är en sk swap-partition (se avsnittet nedan) vars storlek i allmänhet kan väljas som två gånger datorns RAM-minne, `c` är en specialbeteckning för hela hårddisken, `e`, `f`, `g` och `h` innehåller vanliga filsystem, där `e` normalt är `/var` och `f` är `/usr`. Det är egentligen inget speciellt med dessa bokstäver, utom `c`, men om man håller sig till de vedertagna betydelserna underlättar man för sig själv och andra. Det finns heller inget krav att samtliga dessa partitioner ska finnas på en hårddisk.



Figur 4. Inom en slice tillverkar FreeBSD sina egna partitioner som betecknas med bokstäverna a — f. Här visas fyra partitioner på en hårddisks första slice, *s1*. Varje partitions normala betydelse anges under respektive partition.

FreeBSD:s filsystem heter *ffs*, Berkeley Fast File System, och är en utveckling av *ufs*, Unix File System.

3.1.5 Swap och något om minneshantering

Swap-arean på hårddisken fungerar som en sorts förlängt primärminne, RAM. För att i detalj förstå vad swap-arean används till skulle vi behöva göra en djupdykning i FreeBSD:s minneshantering, men här behandlas bara i allmänna termer det som kan vara bra att känna till.

När ett program ska köras måste det först laddas in till systemets RAM-minne. FreeBSD är smart och laddar bara in de delar av programmet som behövs för tillfället. De delar av, det kanske stora, programmet som inte har behövts än, ligger kvar på hårddisken. Detta förfaringssätt sparar RAM och möjliggör att många fler program, eller programdelar, samtidigt kan befinna sig i datorns RAM, en förutsättnig för att snabbt kunna byta mellan olika program. Det märks inte, men systemet byter program flera gånger i sekunden. Det kan tyckas vara ett onödigt slöseri med datorresurser, att ständigt hoppa mellan olika program, men genom att systemet med täta intervall byter program verkar det skenbart som om flera program körs samtidigt, vilket alltså inte är fallet. Mekanismen att byta program är en grundförutsättning för att systemet ska kunna hantera flera samtidiga användare på ett smidigt sätt.

Med tillräckligt många programdelar i minnet kan det till slut inträffa, att systemet inte kan placera ytterligare en programdel i minnet, för att minnet är fullt av andra program. FreeBSD utnyttjar vid detta tillfälle ett antal strategier som samtliga går ut på att frigöra minne. Exempelvis kan programdelar eller dataareor i programminnet som inte behövs, kastas bort eller, om de har blivit ändrade, skrivas tillbaka till hårddisken, och den nya programdelen läses in till den därvid uppkomna luckan in minnet, för vidare exekvering. Om systemet inte utan mödosamt arbete[9] kan frigöra minne, kan det program som använts mest sällan, tillfälligt kastas ut till swap:en, där programmet får ligga tills det behövs nästa gång. I det nu uppkomna hålet i primärminnet kan den tidigare programdelen från hårddisken läsas in och programmet fortsätta som om inget hade hänt.

En konsekvens av minneshanteringen man kanske inte först tänker på är den väsentliga fördelen att den möjliggör exekvering av program som är större än det tillgängliga primärminnet. Skrivning till hårddisk är ofantligt mycket mer tidskrävande än skrivning till minne (hundratusentals gånger långsammare) men kan vara försvarligt, om det sker tillräckligt sällan. Dessutom är skrivning till swap snabbare än skrivning till ett filsystem: för att skriva till ett filsystem måste systemet traversera en katalogstruktur för att hitta var datat ska skrivas på hårddisken, för swap:en har systemet bara en tabell i minnet.

Ett system med för litet minne kan komma i det hopplösa läget att det behöver swappa hela tiden, ett tillstånd som kallas *thrashing*, tröskning. Om trashing inträffar, upplevs systemet som mycket långsamt jämfört med normalt. Det hinner ju inte göra mycket annat än att ideligen swappa mot hårddisken. Enda lösningen är att öka minnesmängden i datorn, eller möjligen att minska antalet program som samtidigt exekveras, om så är möjligt.

Poängen med swap är således att fungera som en sorts extra, ehuru långsamt, minne. Det är viktigt att inte underdimensionera swap-arean då systemets prestanda annars kan bli lidande. För en mer detaljerad introduktion till

[9] FreeBSD anstränger sig mycket för att optimera denna minneshantering så att den blir så snabb som möjligt. Faktiskt är FreeBSD:s minneshantering mycket bättre än många andra operativsystems!

ovanstående komplicerade ämne hänvisas till läroböcker i Datorteknik och då speciellt de kapitel som berör virtuellt minne.

3.2 Installation

Här beskrivs bara början av installationen och de frågor som ställs som kan behöva förklaras. Övriga frågor och gången fram till färdig installation är såpass självförklarande att du själv bör förstå dem. Det är dessutom, åtminstone ur pedagogisk synvinkel, alls inget misslyckande om du inte får ett fungerande system vid första försöket[10]. Med CD:n i handen eller nygjorda installtionsdisketter är vi klara att påbörja installationen. Alltså:

Med diskett Stäng av datorn om den är på, sätt i disketten ”kern.flp” och starta datorn. Byt till disketten ”mfsroot.flp” när den ber om det. Vänta på att menyn nedan kommer fram.

Med CD Sätt CD:n i läsaren och starta datorn. På CD:n finns redan installationsdisketterna inbrända så inget byte behövs.

Om ingen av metoderna ovan fungerar kan inställningarna i datorns BIOS vara sådana att start av systemet från ett flyttbart media inte är tillåtet. Ändra isåfall denna inställning och försök igen.

Med lyckad start kommer efter en liten stund följande textmeny fram på skärmen:

BILD på menyn

Här erbjuds man möjlighet att utföra viss manuell konfiguration för att anpassa systemet till den befintliga datorn. Ofta (nästan alltid!) kan man strunta i denna möjlighet till extra konfigurering, men om man senare, under starten, upptäcker problem kan manuell konfiguration vara nödvändig.

Operativsystemet, som det levereras, innehåller stöd för en mängd olika hårdvaror; det kan gälla nätverkskort, serie- och parallellportar till exempel. När operativsystemet startar, söker den s.k. *kärnan* efter hårdvara den har konfigurerats för att känna igen, och lägger till funktioner för att kunna hantera den upptäckta hårdvaran, s.k. *device probing*. I sällsynta fall kan det inträffa att proberutinen i kärnan fallerar. Isåfall kan starten misslyckas. Det enda botemedlet i detta fall är då att stänga av probningen för den specifika hårdvaran, och det kan man göra om man väljer det övre alternativet ”`Enter visual configuration`”.

BILD, med device names

[10] Det är klart att det går att installera FreeBSD direkt, utan att misslyckas. Men för förstgångsinstallatören kan det vara intressant att prova en del andra vägar vid installationen än de uppenbara. En viss upprepning ger dessutom ett ökat förtroende med hela proceduren.

I konfigureringsmenyn kan man slå av eller på stöd för den hårdvara som kärnan förberetts för[11], men även konfigurera enstaka parametrar som *basadress* eller *irq* för de kort som kräver detta. PCI-kort är självkonfigurerande så dessa varken kan eller behöver slås på/av. Följ instruktionerna, välj sedan "`Save & exit`" för att gå vidare med uppstarten.

3.2.1 Första frågor (var lägga, devices acd0c etc)

Innan systemet installeras måste rätt disk väljas och partitioneras.

BILD på slice-menyn

Följ instruktionerna. Välj hela hårddisken om inga andra system finns eller ska finnas på den. På frågan om den skapade slicen ska vara "`True partition Entry`", svara ja. Observera att *slice-typen* måste vara "`165`" om installationen ska kunna gå vidare. Andra slicetyper är bland andra:

```
06  DOS
11  32bits DOS
```

Välj slutligen "`q`" för att komma vidare till disklabel-menyn, där tillfälle ges att partitionera den just skapade slicen. Åtminstone två partitioner måste skapas, en för `/` och en "`swap`".

BILD på disklabelmenyn

Väljer man här alternativet "`A Auto`" skapas partitionerna `/`, `/var`, `swap` och `/usr` för ett minimalt system. Bland annat är `/`-partitionen bara 50MB, det är eventuellt för litet för ett normalt system som kan behöva 60-80MB. En `/` på 100MB är i allmänhet alltid tillräcklig.

`/var` innehåller alla loggfiler och temporära utskriftsfiler, *spoolfiler*, bland annat. Om datorn ska användas som skrivarserver behöver denna partition vara speciellt stor. Det är inte orimligt att spoolfiler tar några hundratal megabytes i anspråk. När utskriften är klar raderas dock spoolfilen.

Om man inte har någon uppfattning om hur stora partitionerna bör vara kan man här enbart skapa en (1) partition, `/`, förutom `swap`. Då kommer även `/var` och `/usr` att inrymmas i denna större partition.

En vanligt tumregel för swap:ens storlek är att välja den dubbelt så stor som den mängd primärminne datorn är bestyckad med. Denna tumregel fungerar ofta rätt bra, så med 64 MB primärminne väljer vi exempelvis 128 MB swap. En swap på mer än 256 MB är antagligen aldrig nödvändig för våra syften.

När hårddisken är partitionerad skall hela operativsystemet överföras. Installationsprogrammet ger en varning, i själva verket har ingenting skrivits till hårddisken än, och man har möjlighet att avbryta hela proceduren om man så önskar:

[11] Med *omkompilering* av kärnan kan man själv lägga till eller ta bort stöd för olika hårdvaror. Men det är överkurs för tillfället!

```
Last Chance! Are you SURE you want continue the installation?
If you're running this on a disk with data you wish to save
then WE STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before
proceeding!
We can take no responsibility for lost disk contents!
```

Vi väljer förstås att fortsätta installationen och svarar alltså "YES" på frågan.

En FreeBSD-installation är uppdelad i olika programpaket, vart och ett med en egen "termometerskala" som beskriver installationsförloppet. Under denna del av installationen kan vi växla fönster med tangenterna <ALT>+F2 och <ALT>+F4. Det första fönstret visar installationsförloppet fil för fil, det andra är ett skal där man redan nu kan skriva kommandon. Observera dock att det inte är säkert att kommandot överförts från CD:n till hårddisken än. Med <ALT>+F1 växlar man tillbaka till installationsprogrammet.

Tidsåtgången för denna del av installationen beror både på hårddiskens snabbhet, CD-spelarens hastighet och datorns klockfrekvens och kan i vissa fall ta bortåt en halv timma. Under tiden kan vi läsa vidare för att förbereda oss på några av de frågor som kommer att ställas för att konfigurera systemet inför den första omstarten.

3.2.2 Frågor inför omstart (konfig efter inladdning, rootlösen)

De flesta frågor som installationsprogrammet ställer är självförklarande, men vissa kräver att du känner till hur FreeBSD fungerar. Här följer en lista på lämpliga svar på de frågor kan vara kryptiska vid en förstagångsinstallation. Samtliga inställningar kan göras om senare om det skulle behövas.

Would you like to configure any Ethernet or SLIP/PPP network devices? Svara "YES" om du vill konfigurera datorns nätverkskort. Om du inte har något nätverkskort eller inte vill konfigurera det svarar du "NO". För att kunna konfigurera ett nätverkskort krävs följande uppgifter, exempel på format ges inom parentes:

- datorns namn (pctre) och IP-adress (192.168.0.3)
- ditt subnäts namn (test.se) och nätmask (255.255.255.0)
- default gateway (130.213.3.1)

Hitta inte på egna namn och nummer om du inte vet vad du gör! Fråga nätverksansvarig först.

Will this machine be a leaf node...? Svara "YES" tills vidare.

Do you want to grant only normal users FTP access. . . ? Svara ”YES”.

Do you want to configure this machine as an NFS server? Svara ”NO”

Do you want to configure this machine as an NFS client? Svara ”NO”

Do you want to select a default security profile. . . ? Svara ”NO”. Defaultalternativet är i allmänhet tillräckligt. Men man har här möjligheten att välja hög eller låg säkerhetsprofil också.

Would you like to customize your system console settings? Svara ”YES”. Här ges möjlighet att välja tangentbords-layout och tangentmappning. Välj för Keymap alternativ 38, ”Sweden”, System Console Font till ”ISO-8859-1 Western Europe, ISO encoding” samt Alternate Screenmap till ”None”. Observera att dessa inställningar gäller för systemet och för samtliga användare. Varje användare har dock möjlighet att skräddarsy sina egna inställningar.

Would you like to set this machine's time zone now? Svara ”YES”. Av tradition brukar systemklockan ställas enligt Greenwich Mean Time vilket är praktiskt taget samma som UTC. Svensk normaltid är UTC+1, sommartid är UTC+2. Systemet sköter om övergången mellan sommar- och vintertid helt automatiskt.

Would you like to enable Linux binary compatibility? Skall ”linuxulatorn” installeras, dvs skall systemet kunna köra program avsedda för Linux? Svara isåfall ”YES” i annat fall ”NO”.

Does this system have a non-USB mouse attached to it? Om din dator är försedd med USB-mus svara ”YES”, med seriell mus eller ingen mus svara ”NO”.

Would you like to configure your X server at this time? Svara ”NO”. Det är säkrast att installera X separat. Om installationen påbörjas nu och inte kan fullföljas kan man behöva ominstallera hela systemet från början.

The FreeBSD package collection is a collection of thousands. . . ? Svara ”NO”. Det finns egentligen inget extra som vi vill installera för närvarande.

Would you like to add any initial user accounts to the system? Svara ”YES”. Åtminstone en användare förutom `root`, systemadministratören, skall alltid konfigureras. Ange åtminstone `Login ID` och `Password` (lösen). Det kan vara trevligt att ange `Full name` också men inte nödvändigt. Den som installerar systemet vill antagligen skriva `wheel` i `Member groups`. En användare som tillhör gruppen `wheel` kan med kommandot `su(1)` på ett enkelt sätt byta identitet till `root` på ett ögonblick utan att behöva logga in på nytt.

Now you must set the system manager's password. Ange systemadministratörens, `root`:s, lösenord. Ett bra lösenord är åtminstone 8 tecken långt och bör innehålla båda versaler och gemener och siffror slumpartat. Kan man dessutom få plats med något eller några skiljetecken har man ett bra lösenord.

Visit the general configuration menu...? Svara ”YES” om du vill korrigera något. Svara ”NO” om du är klar för omstart och redo att ta upp systemet för första gången.

3.2.3 Första start

Vid start av systemet visar sig först FreeBSD:s bootmanager som är belägen på hårddiskens början:

```
FreeBSD/i386
F1 FreeBSD
default: F1
```

Upp till fyra startbara slice:ar kan väljas med tangenterna F1 till F4. Bootmanagern visar bara de slice:ar som finns för närvarande. Väljer man inget startas defaultalternativet efter några sekunder. Defaultalternativet är det som valdes vid senaste omstart.

Nu startas operativsystemet på den valda slice:en och FreeBSD:s bootprompt, ”`boot:`”, visar sig.

Vid ”`boot:`”-prompten kan man bl.a. skriva:

- `boot` Gå vidare med starten i normal-läge, detta alternativ väljs sker automatiskt efter några sekunder.
- `boot -c` Starta ”visual config editor”, samma meny som i XXX ovan.
- `boot -v` Ge mer detaljerade startmeddelanden.
- `boot -s` Starta systemet i *single-user mode* med en s.k. *console*, konsol. Single-user mode startar ett servicetillstånd, ett minimalt system, med endast en användare och utan nätverksstöd. Denna mode kan vara praktisk vid felsökning eller om FreeBSD:s partitioner behöver

repareras. Monteringarna i `/etc/fstab` förbigås förutom `/`. Om inte ens `/` kan monteras kallas situationen *panic*, och systemet startas om. Det senare är ett allvarligt tillstånd och kan bara hävas av en kunnig systemadministratör, med rutiner som går utanför denna inledning. Vid single-user mode är alltså enbart `/` monterad. Den är dessutom monterad så att endast tillåter läsning, inte skrivning[12].

Då även en icke-root-användare kan starta ett system i single user mode via `boot -s` bör man editera filen `/etc/ttys` och ändra konsolen från `secure` till `insecure`. Efter denna ändring avkrävs `root`-lösen innan inloggning till konsolen kan ske och man kan med detta hindra obehöriga från att manipulera systemet. Speciellt viktigt är detta på klientdatorerna, en serverdator står normalt inte tillgänglig för andra än administratören.

3.2.4 `/etc/fstab`

Vid uppstart är filen `/etc/fstab` viktig. Den beskriver var och vilka av hårdiskens partitioner som ska monteras. Till skillnad från MS-DOS kan filsystem fästas, ”klistras in” var som helst i ett katalogträd. En process som benämns *mounting*, montering. `/etc/fstab` kan se ut såhär:

```
/dev/ad0s4a  /     ufs   rw  1  1
/dev/ad0s4b  none  swap  sw  0  0
/dev/ad0s4e  /var  ufs   rw  2  2
/dev/ad0s4f  /usr  ufs   rw  2  2
```

Här anger till exempel sista raden att `/dev/ad0s4f`[13] ska monteras på `/usr`. Tredje kolumnen anger att `/dev/ad0s4f` innehåller ett filsystem av *ufs*-typ. Liknande gäller för de andra partitionerna.

Liten BILD på monteringspunkt och hårddisklabels?

Filsystemet måste monteras för att man ska kunna komma åt filerna på det. Innan enheterna är monterade finns visserligen katalogen `/usr` men den är helt tom, den utgör bara en monteringspunkt.

För att monteringen ska lyckas krävs dessutom att partitionen är *clean*, ren, d.v.s. att filsystemet på den är konsistent. Ett säkert sätt att få problem är att stänga av datorn med spänningsknappen utan att först stänga av operativsystemet. Om monteringen misslyckas vid uppstart kommer systemet att avbryta starten och automatiskt gå in i single-user mode.

3.2.5 `mount(1)`

Manuellt kan montering av filsystem ske med `mount(1)`-kommandot. Det kommando som monterar `/usr` enligt ovan kan vid prompten skrivas:

[12] Med kommandot `mount -a` monteras de enheter som anges i `/etc/fstab`. Bland annat kommer då `/` att monteras ”ovanpå sig själv” men *med* skrivrättigheter.

[13] Partitionerna nämns i filen med hela sin sökväg. Eftersom `ad0s2f` befinner sig katalogen `/dev` är dess fullständinga namn `/dev/ad0s2f`.

```
# mount /dev/da0s1f /usr
```

Andra filsystem än `ufs`, hårddiskens filsystem, kan monteras[14]. I dessa fall måste man dock själv ange vilken typ filsystemet är, det kan vara bland annat dos- eller cd-rom-typ. Sålunda kan monteringar se ut som:

```
# mount -t cd9660 /dev/cd0c /cdrom
```

(eller `# mount_cd9660 /dev/cd0c /cdrom`)

```
# mount -t msdos /dev/fd0c /mntStandard(eller # mount_msdos /dev/fd0c
/mnt
# mount -t nfs ville:/export/data /mnt
```

(eller `# mount_nfs ville:/export/data /mnt`)

Alternativa kommandon anges inom parentes. För det sista kommandot gäller att `ville` är en annan dator som exporterar katalogen `/export/data`. Se vidare kapitlet om NFS.

3.2.6 `fsck(1)`

Ett filsystem innehåller alltid en liten dataarea, boot-blocket, med information om filsystemet självt. Bland annat anges här om filsystemet är *clean* eller inte. Med clean menas att filsystemet är konsistent och korrekt och inte innehåller några ”lösa” filer eller herrelösa bibliotek. Innan ett filsystem monteras kontrolleras att det är clean och om så inte skulle vara fallet tillåts inte monteringen att fortsätta.

Med programmet `fsck(1)` kan man själv testa om ett filsystem är clean eller inte. `fsck(1)` genomför ett test av en hel partition och rapporterar eventuella felaktigheter. Programmet kan också själv utföra flera korrigeringar av ett filsystem som inte är clean. Slutligen kan fsck också markera att filsystemet är clean.

Om uppstarten misslyckas på grund av att något filsystem i `/etc/fstab` inte är clean, kör systemet automatiskt `fsck(1)` på partitionen för att försöka reparera filsystemet. Om `fsck(1)` inte lyckas måste systemadministratören själv ingripa.

För att kunna köra `fsck(1)` mot ett filsystem får detta inte vara monterat. Givet att uppstarten misslyckades på grund av att `da0s1f` inte var clean och att systemet nu befinner sig i single-user mode kan systemadministratören köra `fsck(1)` manuellt:

```
# fsck /dev/da0s1f
```

[14] Normalt kan bara `root` montera enheter. `root` kan ge tillåtelse för användarna att montera enheter med kommandot `sysctl(8)`, enligt

```
# sysctl -w vfs.usermount=1
```

Behörigheter enligt kapitel XXX gäller dock fortfarande.

Utan ingående kunskaper i filsystemet torde systemadministratören bara kunna svara `y` för ”Yes” på frågorna om `fsck(1)` ska gå vidare och korrigera efter bästa förmåga eller inte. Och oftast räcker detta. Om man har ett filsystem med många fel kan det vara tröttsamt att skriva `y` många gånger. Vid start av `fsck(1)` kan man då ange att man önskar svara `y` på samtliga frågor med:

```
# fsck -y /dev/da0s1f
```

`fsck(1)` korrigerar inkrementellt dvs. den utför ändringar i flera steg. Vissa sorters fel på filsystemet kan bara korrigeras om `fsck(1)` körs flera gånger. Även om `fsck(1)` avslutades utan felmeddelande kan det alltså finnas anledning att köra det ytterligare en (1) gång för att säkerställa att filsystemet verkligen är korrekt.

3.2.7 Första inloggningen

Om uppstarten av systemet gick bra – och det borde det gjort – möts man nu av FreeBSD:s login-prompt:

```
FreeBSD/i386 (pctre) (ttyv1)
login:
```

`pctre` är det *hostname* du gav datorn tidigare, eller `Amnesiac` om du inte angav något. `ttyv1` betyder att du befinner dig vid konsolen, *console*, som är den första av flera virtuella skärmar du kan logga in på. Med <ALT>+F2 kan man byta till den andra virtuella skärmen och så vidare. Ifrån början finns åtta virtuella skärmar definierade, vilka kan nås med <ALT>+F1 ...<ALT>+F8.[15].

Man kan nu logga in som `root` eller som vanlig användare. Det rekommenderas att vara ”vanlig” så ofta det bara går. Med kommandot `su(1)` (*switch user*) kan man byta roll och temporärt erhålla `root`:s privilegier:

```
> su
Password:
#
```

Det finns inget utöver lösenordet som hindrar en användare, vilken som helst, att byta till någon annan användare. Men för att kunna byta till `root` måste man tillhöra gruppen `wheel`.

Behörigheter

Centralt för FreeBSD är begreppet behörigheter (*permissions*). Med FreeBSD kan systemadministratören tilldela resurser på användar- eller gruppnivå. Med kommandot `ls -l` listas filer och bibliotek, som med DOS-kommandot `dir`, men här med sina behörigheter:

[15] Antalet virtuella skärmar kan ändras och bestäms av innehållet i `/etc/ttys`.

```
> cd /bin
> ls -l
⋮

-r-xr-xr-x  1  root  wheel  210884  21 Sep 09:20  ps
-r-xr-xr-x  2  root  wheel   57516  21 Sep 09:20  pwd
-r-sr-xr-x  1  root  wheel  244712  21 Sep 09:20  rcp

⋮
>
```

I högra kolumnen står program- eller filnamnet. Behörigheterna anges i vänstra kolumnen. Formatet känns lite kryptiskt men är lätt att genomskåda.

Behörigheter för en fil, en katalog eller enhet kan utdelas till

- en enskild ägare
- en grupp av användare
- alla andra.

Det finns i huvudsak tre olika behörigheter

- läsning
- skrivning
- exekvering

Dessa betecknas med `r`, `w` respektive `x` efter engelskans *read*, *write* och *execute*. Man kan till exempel alltså tillåta läsning men förbjuda skrivning för en fil. Exekveringsbehörighet på en katalog ger tillåtelse att gå in i katalogen.

Övre radens behörigheter kan nu uppdelas som

katalog	ägare (*owner)*	grupp (*group*)	andra (*others*)
`-`	`r-x`	`r-x`	`r-x`

ägare (*owner*)	grupptillhörighet	storlek	datum	namn
`root`	`wheel`	`210884`	`21 Sep 09:20`	`ps`

och ger alltså `root`, medlemmar av gruppen `wheel` och alla andra läs- och exekveringsrättigheter på filen. Däremot får ingen annan än ägaren, i detta fall `root`, skriva till filen.

`root` får alltid tillgång till alla filer och bibliotek. oavsett vilka rättigheter som anges. Om resursen är en katalog inleds raden av ett `d`.

För att ändra behörigheter används kommandot `chmod(1)`. Det finns två format för att ange behörigheter: absolut eller symbolisk. Med det absoluta formatet anger man med siffror exakt vilka behörigheter som ska sättas:

```
> chmod 755 test.sh
```

Här anger `755` vilka av `rwx` som ska vara satta. `7, 5, 5` måste tolkas enligt det binära talsystemet som ”`111, 101, 101`” och man ser då att detta motsvarar `rwx, r-x, r-x` där `-` sätts in istället för nolla. En lathund kan vara på sin plats:

binärt	decimalt	rwx
000	0	---
001	1	--x
010	2	-w-
011	3	-wx
100	4	r--
101	5	r-x
110	6	rw-
111	7	rwx

Med lite övning är detta ett mycket smidigt sätt att ange behörigheter.

Med det symboliska sättet anger man behörigheterna i ordningen ”för vem och vad”. ”För vem” kan vara `u`, `g` eller `o` för *user*, *group* eller *others*. ”Vad” kan vara tillåta/förbjuda `r`, `w` eller `x`. Tillåta respektive förbjuda anges med `+` respektive `-`. För att, till exempel, slå av läsbehörigheten för *others* anges

```
> chmod o-r test.sh
```

Den tredje radens behörigheter ser lite annorlunda ut. För ägaren gäller behörigheten `r-s`. Flaggan `s` betyder att programmet skall köras SET-UID: När programmet körs får det filägarens behörigheter — i det här fallet `root`. Detta sker alltså även om det inte är `root` som startar programmet. Fler och utförligare exempel återfinns i manualsidan, skriv `man chmod`. Se vidare kapitel XXX

3.2.8 Lägga till användare

FreeBSD är ett fleranvändarsystem. Flera olika användare kan samtidigt kan ta systemets resurser i anspråk. Det är viktigt att bara legitima användare kan komma åt resurserna.

Vid installationen lämnades tillfälle att lägga till ytterligare en användare förutom root. På samma sätt kan flera (hundratals om så önskas) användare läggas till systemet. Varje användare ser ”sin” del av systemet[16] och inget

[16]Defaultbehörigheterna tillåter dock att man kan se andra användares hemkonto, men man kan fortfarande bara skriva till sitt egna. Se `umask(2)` för att förändra detta..

annat. Han har till exempel sin egen katalog, sitt *hemkonto*, för sina egna filer där han kan rumstera om enligt eget tycke.

Ofta har systemadministratören lagt till diverse initieringsprogram i användarens hemkatalog, det kan handla om styrfiler för olika skal, exempelvis ”punkt”-filer som `.cshrc` och `.login` eller styrfiler för fönsterhanterare, som `.xinitrc` exempelvis. När systemadministratören skapar en ny användare på systemet sker flera saker:

- En hemkatalog skapas, i `/home`
- *Skeletons* från `/usr/share/skel` och `/usr/local/share/skel` kopieras till hemkatalogen. Dessa styrfiler syns inte vid normal listning av kataloginnehållet, men kan listas med `ls -a`[17]. Styrfilerna heter i skelkatalogerna ovan `dot.filnamn` och efter kopiering till hemkatalogen döps de om till `.filnamn`.
- Filerna `/etc/passwd` och `/etc/master.passwd` updateras med bland annat användarnamn, (krypterat) lösenord och hemkatalog.
- En grupp med samma namn som användaren läggs till i `/etc/group`.

En ny användare kan läggas till systemet på ett antal olika sätt: med `/stand/sysinstall` eller `adduser(8)` eller `pw(8)` eller `vipw(8)`[18]. Det enklaste är kanske att använda `/stand/sysinstall` som vid installationen, men även `adduser(8)` och speciellt `pw(8)`, med tillägget `useradd`, kan användas för samma effekt. `pw(8)` är speciellt viktig då det lätt kan användas i *shell-skript*.

3.2.9 Filerna /etc/passwd och /etc/master.passwd

Den viktigaste informationen om olika användare systemet behöver ligger i filerna `/etc/passwd` och `/etc/master.passwd`. `/etc/master.passwd` är den hemligare av de två, den innehåller lösenordsinformation och måste alltid hindras från obehörig läsning. Vissa program måste ha tillgång till viss användarinformation (men definitivt inte lösenordslistan!) och för den skull finns `/etc/passwd` som är läsbar av alla. Den enda egentliga skillnaden mellan de två är att `/etc/passwd` saknar lösenordsinformation.

Med `vipw(8)` kan man göra ändringar i `/etc/master.passwd`. `vipw(8)` öppnar `/etc/master.passwd` i en editor. En rad kan se ut som:

```
mj:$1$dgby3lPy$LVRbf8t8rGapU2ygg1ZlO1:1000:1000:...
...:0:0:micke:/home/mj:/bin/tcsh
```

[17] Generellt gäller att filer som börjar med en punkt inte syns utan tillägget `-a` till `ls(1)`-kommandot.

[18] Nuförtiden använder man knappast `vipw(8)` för att lägga till en användare, men väl för att ändra användarinformation. Om man lägger till en användare med `vipw(8)` måste man själv uppdatera `/etc/group` med korrekt information.

Det finns ett antal olika användare fördefininerade för olika systemuppdrag i bakgrunden. Lokalt definierade användare, som `mj` ovan, läggs till sist i filen.

Raden innehåller flera fält avdelade med semikolon. Raden inleds med användarnamet (`mj`) följt av användarens lösenord i krypterad form.

FreeBSD:s lösenord baserar sig på MD5-algoritmen som, givet ett ord skapar ett s k *hash* för just detta ord. Det är matematiskt omöjligt att gå åt andra hållet: Med ett hash finns det ingen möjlighet att räkna ut vilket det ursprungliga ordet var. Det är dock fortfarande viktigt att lösenordet i sin krypterade form inte kommer i orätta händer då en hängiven kodknäckare kan prova sig fram med olika ”sannolika” ord ända tills han fått fram ett hash som stämmer. Det finns speciella program för detta som underlättar kodknäckarens jobb avsevärt. Med tillräckligt ”osannolika” lösenord behöver han avsätta väsentlig datorkraft och mycket tid för att prova alla ord. Ett bra lösen är minst 8 tecken långt och innehåller både siffror, bokstäver (både versaler och gemener) och skiljetecken. Det kan också vara förnuftigt att byta lösenord några gånger om året för att förhindra att ett knäckt lösenord kan komma till användning.

Om lösenordsfältet är tomt behövs inget lösen för att ta sig in i systemet! Om lösenordsfältet inleds av en ”`*`” är kontot avstängt och användaren kan inte logga in. Se till att alla lösenordsfält innehåller något!

Fälten därpå innehåller användarens *user-id* (1000), användaridentitet, även förkortad *UID* och därefter, *group-id*, grupptillhörighet, som förkortas *GID*. Såväl UID som GID kan vara mellan 0 och 65535. Bara 0 är speciellt, en användare med 0 som GID är i praktiken root på systemet. Fördefinierade användare på systemet har låga UID, under 1000, medan användare som lagts till efter installationen tilldelas ett UID med början på 1000.

GID-fältet följs av tre fält för att ange användarens *login class*, högsta tillåtna ålder på lösenordet och eventuellt datum då kontot ska upphöra. Administratören kan avända flera olika login-klasser för att hindra inloggning utanför arbetstid eller hindra en viss login-klass från att använda för mycket systemresurser med mera. Högsta tillåtna ålder på lösenordet kan användas för att tvinga användarna att byta lösen efter en viss tid och upphörandedatum kan vara praktiskt om man vill tilldela datorresursen bara under en fördefinierad tid.

De tre sista fälten innehåller användarens verkliga namn, hemkatalog och skal.

Man ser att det finns det två användare med GID 0 – `root` och `toor` (`root` baklänges). `toor` är ett hjälpkonto som kan användas för att skapa en användare med `root`-privilegier men med en annan hemkatalog. `root`:s katalog ligger ju direkt under `/` och kan inte bli så stor då man brukar hålla `/`-partitionen så liten som möjligt.

3.2.10 Katalogstruktur

De olika katalognamnen i filsystemet avslöjar något om katalogens innehåll och en liten introduktion till dem kan underlätta navigationen i filsystemet. De vanligaste beskrivs nedan[19]. Om något verkar obegripligt så läs bara vidare, det klarnar förhoppningsvis senare när sammanhangen blir tydligare.

Högst upp i filehierarkin befinner sig filsystemets rot (att skilja från användarkontot `root`), som betecknas med `/`[20]. `/` innehåller följande:

`/bin` innehåller en del av de program som är nödvändiga för att systemet ska kunna startas överhuvudtaget. De mest använda unixkommandona (`ls`, `rm`, `cp`, `sh`, ...) ligger också här. Observera att det finns ett program som heter `[`, ta inte bort det!

`/boot` innehåller de absolut första program som körs då systemet startar från vila. Dessa används bara under själva startfasen.

`/cdrom` `/cdrom` är normalt tom och används som monteringspunkt, *mount point*, för cd-skivor.

`/compat` För att kunna köra en del äldre FreeBSD-program på ett nytt system kan s.k. kompatibilitetsbibliotek behövas. Dessa installeras i sådana fall i denna katalog. ”Linuxulatorn”, som medger att de flesta program avsedda för Linux även kan köras på FreeBSD, har sin egen katalog här, om den är installerad.

`/dev` innehåller de *devices*, enheter, som systemet stöder. Dessa filer är inga normala filer utan kan beskrivas som en sorts handtag för att kunna kommunicera med datorns hårdvara. Hårddiskar är egna enheter, liksom ljudkort och annan kringutrustning.

`/dist` är ännu en tom monteringspunkt. Programmet `/stand/sysinstall` använder den för att montera distributions-cd:n.

`/etc` innehåller en mängd olika filer som inte logiskt hör hemma någon annanstans, bland annat konfigurationsfiler för systemet, som `rc.conf`. Vid start av systemet är filen `/etc/fstab` bland de viktigaste:

[19] Se även manualsidan för detta: `man hier`.
[20] Från andra operativsystem kanske du är van vid att roten betecknas \, så icke i Unix.

```
# Device     Mountpoint  FStype  Options      Dump  Pass#
/dev/ad0s4a  /           ufs     rw           1     1
/dev/ad0s2b  none        swap    sw           0     0
/dev/ad0s4f  /usr        ufs     rw           2     2
/dev/ad0s4e  /var        ufs     rw           2     2
/dev/acd0c   /cdrom      cd9660  ro, noauto   0     0
```

Denna fil innehåller information om var hårddiskens olika partitioner ska monteras. På första raden (om man bortser från kommentarraden överst) kan man läsa att `/` befinner sig på första hårddiskens fjärde slice och första partition, att den innehåller ett filsystem av typen `ufs` och att den skall monteras för både läsning och skrivning, `rw`, *read-write*. De sista siffrorna anger i vilken ordning systemet ska kopieras vid säkerhetskopiering.

`/kernel` Systemets *kärna*. Systemets kärna är den centrala del av operativsystemet som administrerar och dirigerar all data systemet hanterar. Det är också kärnan som ser till att systemet tillåter flera användare på samma gång, bland annat. Kärnan laddas in i datorns primärminne (RAM) vid systemstart och måste alltid ligga kvar där.

På datorer med mycket lite minne (4-8MB) kan kärnan behöva *trimmas*, d.v.s. man avlägsnar funktioner som inte behövs: Om man exempelvis inte har några SCSI-enheter anslutna kan SCSI-stödet avlägsnas[21]. Kärnan tar då mindre plats och det härmed frigjorda minnet kan användas av övriga program.

En felaktigt konfigurerad kärna kan innebära att systemet inte kan startas. Som extra säkerhetsåtgärd finns därför alltid den ursprungliga, generiska, kärnan, `/kernel.GENERIC`, kvar som säkerhetsåtgärd. Man kan vara säker på att systemet i varje fall går att starta med denna.

`/mnt` En allmän monteringspunkt. Medan /cdrom och /dist har sin förutbestämda funktion kan /mnt användas fritt.

`/modules`[22] För att hålla nere storleken på kärnan pågår en utveckling av kärnmoduler. Tanken är att den generiska kärnan bara skall innehålla ett absolut minimum av funktionalitet och övriga funktioner ska kunna läggas till när de behövs. Denna katalog innehåller dessa kärnmoduler. Se också `kldstat(2)` och `kldload(8)`/`kldunload(8)`.

`/proc` Det så kallade processfilsystemet innehåller information om varje enskild s.k. process. En process är i allmänhet detsamma som ett program som körs. Skillnaden är ovesäntlig för oss. Man kan titta på en del filer i

[21] Genom omkompilering av kärnan. Fortfarande överkurs!

[22] Efter omkompilering av kärnan och dess moduler sparas de gamla modulerna i `/modules.old` som säkerhetsåtgärd. Är det ont om plats på `/` kan man i allmänhet ta bort `/modules.old`.

processfilsystemet men skriv aldrig till dem. Det är bara `root` som skulle kunna skriva till dem men lockas inte till det. Med kommandot `df` kan man bland annat se hur mycket ledig plats de anslutna filsystemen har. `/proc` tar alltid 100% kapacitet i anspråk, detta är helt normalt och inte något tecken på minnesbrist.

`/root` Superanvändaren `root`:s personliga hemkatalog. Den används normalt enbart vid systemunderhåll eller i single-user mode. `/root` skall normalt inte innehålla några större mängder data. Om användaren `root` behöver en större katalog kan `toor` aktiveras och den senares hemkatalog, `/home/toor`, användas.

`/sbin` Ytterligare filer, vilka likt de i `/bin`, är nödvändiga för systemstart men med inriktining på systemunderhåll. Här återfinns exempelvis monteringskommandot `mount(1)` och diverse nätverkskommandon.

`/usr` innehåller filer som är tänkta för normal drift. Filerna är inte nödvändiga för systemstart, men behövs för att det ska bli ett komplett körbart system.

/`usr` innehåller bl.a. följande underkataloger:

`/usr/home` Här ligger alla användares hemkataloger. Ofta kan man komma åt `/usr/home` genom länken `/home`. Om den länken inte skulle finnas kan man skapa den med `ln(1)`:

`ln -s /usr/home/ /home` eller, enklare `ln -s /usr/home /`

`/usr/local` Den kanske mest använda katalogen under `/usr`. Här hamnar alla körbara filer och en del annat smått och gott som installerats *efter* systeminstallationen. Speciellt kommer de filer som installeras genom ports/packages-mekanismen att hamna under `/usr/local/bin`.

`/usr/ports` kan innehålla det s.k. *ports*trädet. Mer om det senare. `/ports` kan också vara tom beroende på vad som valdes vid installationen.

`/usr/X11R6` innehåller alla de filer som fönstersystemet X11 behöver. Katalogen innehåller en mängd underkataloger med namn och funktioner liknande de som återfinns under `/`. `/usr/X11R6/bin` innehåller program som kräver fönstersystem för att fungera och andra program för XFree.

`/tmp` innehåller temporärfiler. Inga av dessa är nödvändiga mellan systemstarter. Med ändringar i filer i `/etc/periodic/daily`[23] kan systemet med automatik tömma `/tmp` på onödiga filer. Kan vara nyttigt då ett system mycket sällan behöver omstartas och `/tmp` kan innehålla en del ”glömda” filer.

`/var` innehåller bland annat systemloggar (`/var/log/`) genererade av `syslogd(1)`, men även databasen för ports/packages (`/var/db`)om de installerats. Spool-filer lagras här inför utskrift och kan uppgå till åtskilliga tiotals megabyte.

Om den ursprungligen tilldelade platsen för `/tmp` och `/var` inte längre räcker till är ett knep att montera ytterligare hårddiskyta hit med hjälp av `mount(1)`-kommandot. Man kan även låta `/tmp` och `/var` vara länkar till motsvarande bibliotek placerade i `/usr`, där det brukar finnas plats:

```
# mkdir /usr/tmp
# ln -s /usr/tmp /tmp
# mkdir /usr/var
# ln -s /usr/var /var
```

... eller undvik problemet helt och hållet genom att vid installationen bara ange en enda (stor) partition för hela systemet.

3.2.11 Kommandon

Det finns en översvallande mängd kommandon i FreeBSD. För att snabbt komma igång med systemet behöver man dock inte kunna mer än en handfull. De som man oftast använder är listade nedan tillsammans med sin ungefärliga MS-DOS-motsvarighet:

[23] Närmare bestämt i filen `110.clean-tmps`.

FreeBSD	MS-DOS	Förklaring
ls	dir	visa innehållet i en katalog
cd /bin	cd \bin	gå till katalog /bin
mv	move, rename	flytta filer/kataloger
cp	copy	kopiera filer
cp -R		kopiera rekursivt
		d. v. s. även underkataloger
rm	del	ta bort fil
rm -r	deltree	ta bort (tomma) kataloger
rm -rf	deltree	ta bort (även icke-tomma) kataloger!
df	chkdsk	visa diskutnyttjande
du		lista filhierarki och storlek
du -d 1		som ovan men med djupet 1
top		visa systembelastning m.m.
man		sök in manual-sidorna
mkdir	mkdir	skapa katalog
more		visa filinnehåll sidvis
pwd		visa nuvarande katalog
XXX		

3.3 Packages och ports

Som tidigare nämnts innehåller `/usr/ports` det s.k. *ports*trädet, en bortåt 80 MB stor biblioteksstruktur med vars hjälp man kan installera över 5000 program med bara några enkla knapptryckningar. Programmen är sorterade i ett antal bibliotek uppdelade i olika kategorier som print, mail och www bland många andra.

Som exempel på packagesmekanismen ska vi installera programmet `bash`(1). `bash(1)` är ett s.k. *skal* och beskrivs närmare nedan, men för att kunna prova det måste det först installeras. Installationen kan ske som antingen som *package* eller *port:*

- Package innebär att filerna som ska installeras redan är kompilerade och sedan komprimerade till en enhet, ett package.
- Ports innebär att filerna levereras som okompilerad källkod och måste kompileras innan installation.

Det senare alternativet kan vid första anblicken verka opraktiskt men möjliggör att ändringar genomförs innan kompilering och installation. Man kan särskilja packages från ports på filnamnen i och med att packagesfilernas namn avslutas med `.tgz` och ports avslutas med `.tar.gz`.

```
bash-xx.tgz
bash-xxx.tar.gz
```

Åter till `bash(1)`. Med **ports** installeras bash genom:

```
# cd /usr/ports/shells/bash                    ← välj rätt bibliotek
# make                                ← hämta källkoden och kompilera den
# make install                          ← installera de resulterande filerna
```

Om `bash-XX.tar.gz` inte återfinns på datorn i `/usr/ports/distfiles` eller inte återfinns på distributions-cd:n försöker `make` att hämta det från Internet innan kompilering. Med `make install` installeras de i förra steget kompilerade filerna. `make`[24] är intelligent nog att förstå vad som avses om man bara anger `make install` – även i detta fall försöker den hämta källkoden först, om det skulle behövas.

Med `make clean` kan man efter avslutad kompilering ta bort källkoden och den arbetskatalog som använts. Arbetskatalogen behövs inte längre och tar bara upp plats på hårddisken. Källkoden finns dock kvar i komprimerad form i `/usr/ports/distfiles`. Vill man även ta bort den komprimerade källkoden används kommandot `make distclean`.

Med **packages** sker installationen lättast med programmet `/stand/sysinstall`. Sätt i CD:n, kör `/stand/sysinstall`, välj Post Installation Configuration⇒-Precompiled programs⇒CD-rom. Välj sedan ***`shells`*** och markera `bash`-raden, gå ur och svara Yes på frågan om du vill installera `bash`. Inom några sekunder är `bash` installerat och kan användas direkt. Gå slutligen ur `/stand/sysinstall`.

Efter installation av ett program kan man inte nå programmet om man inte anger programmet med full sökväg. Vid inloggning sammanställs alla körbara program i en lokal databas, individuell för varje användare. Program som tillkommit efter inloggningen ingår inte i denna databas. Finessen med databasen är att systemet mycket snabbt kan ta reda på om ett önskat program finns tillgängligt för exekvering utan att behöva gå ut på hårddisken och söka igenom den. Åtminstone om man har skalet `tcsh(1)`[25] kan man uppdatera databasen med kommandot `rehash`. I andra fall kanske man måste logga in på nytt för att kunna nå det installerade programmet, om man inte anger exakt sökväg.

På liknande sätt installerar man övriga program i ports. För närvarande (2001) finns över 5600[26] program att välja på och antalet ökar hela tiden. Tänk på att program tar plats på hårddisken och det kan vara viktigt att inte bara installera program utan också vara noggrann med att ta bort sådana som inte används längre. Vill man rensa hela ports-trädet från gamla kompileringskataloger kan man köra kommandot `make clean` direkt i `/usr/ports`.

[24] Med `make` kan man även söka efter program: `make search key=`***`editor`*** presenterar alla program som har något med *editor* att göra. Detta fungerar enbart om man befinner sig i `/usr/ports`.

[25] `tcsh(1)` ingår i grundsystemet från och med FreeBSD 3.0 och behöver alltså inte installeras efteråt.

[26] Se `http://www.freshports.org` för ”dagsnoteringen”!

`/stand/sysinstall` använder programmet `pkg_add(1)` för att genomföra uppackning och installation av packages (alltså inte ports!). Det finns inget som hindrar att man använder `pkg_add(1)` direkt på kommandoraden om man så önskar:

```
# pkg_add bash-XX
```

I `/var/db/pkg` finns en katalog för varje installerad ports/package. Dessa katalognamn kan även användas för att ta bort programmet. För att ta bort `bash(1)` som installerades ovan ser vi att `/var/db/pkg` innehåller katalogen `bash-XXX`. Använd kommandot `pkg_delete(1)` för att ta bort det från systemet:

```
# pkg_delete bash-XXX
```

Man kan även hitta rätt `bash(1)`-program med kommandot

```
# pkg_info | grep bash
```

som söker igenom samtliga installerade packages och ports.

Programmet kan inte tas bort om det är en förutsättning för något annat programs funktion, ett beroende har uppstått, en s.k. *depedency*. I sådana fall måste man ta bort det andra programmet först.

3.4 Skal

Ett *skal* i unixvärlden är detsamma som ett kommandofönster i DOS. Det är i ett skal man skriver kommandon till datorn och det är i samma skal man kan läsa vad datorn skriver tillbaka. I ports finns ett antal olika skal att välja mellan. I ett grundsystem kan man vara säker på att skalet `sh(1)` (sh för *shell*, skal) finns. Det är det äldsta skalet och det som är mest standardiserat mellan olika sorters unix och används numera mest för att skriva portabla skript-program.

Under årens lopp har flera olika skal utvecklats, `bash(1)` enligt ovan, men även korn-shell, `ksh(1)` och `zsh(1)` och andra. Här beskriver jag bara min personliga favorit `tcsh(1)`, som dessutom numera ingår i FreeBSD:s grundsystem. Låt inte mitt val hindra dig från att studera andra skal i `/usr/ports/shells`.

3.4.1 `tcsh(1)`

`tcsh(1)` har en imponerande manualsida. Den är nästan ogenomtränglig på grund av sin storlek men innehåller också allt man någonsin kan behöva veta om `tcsh(1)`. Här presenteras ett litet urval.

För att förflytta sig på kommandoraden kan man ha nytta av följande lilla hjälptabell:

`ctrl-a`	Hoppa längst till vänster på raden, till början
`ctrl-e`	Hoppa längst till höger på raden, till slutet
`ctrl-f`	Stega ett steg höger. Även piltangenten → fungerar.
`ctrl-b`	Stega ett steg vänster. Även piltangenten ←fungerar.
`ctrl-w`	Ta bort allt till vänster om prompten. Lagra undan det.
`ctrl-y`	Sätt in det undanlagrade.
ctrl-u	Radera hela raden.
ctrl-k	Radera allt till höger om prompten.
ctrl-d	Ta bort teckent sprompten står på.
ctrl-h	Ta bort tecknet till vänster

Desutom kan <TAB>-tangenten användas för att komplettera kommandon:

```
> ls /u<TAB>
```

ger

```
> ls /usr/
```

då det, i det här fallet, inte var något tvivel om bibliotekets, eller kommandots, namn. `tcsh(1)` expanderar så långt det kan. `<ctrl>-d` kan användas för att ta reda vilka möjliga alternativ som föreligger:

```
> ls /b<TAB>(inget händer)<ctrl>-d
bin/ boot/
> ls /bo<TAB>
> ls /boot/
```

`<ctrl>-d` används också för att logga ut, om prompten är i början på en tom rad. Detta kan vara något överraskande om man råkar trycka vid fel tidpunkt...

Till varje program som startas från kommandoraden knyts två speciella in- och utenheter, `stdin` och `stdout`. `stdin` är varifrån programmet tar sina indata och `stdout` dit programmet skickar utdata. Vanligen är då tangentbordet kopplat till `stdin` och skärmen till `stdout` men båda kan lika gärna vara filer eller någon *device*. Ett enkelt exempel kan konstrueras med de båda kommadona `cat(1)` och `more(1)` . `cat` *`filnamn`* skickar ut hela filens innehåll till `stdout` vilket innebär att filen syns på skärmen,

```
> cat /COPYRIGHT
:
(en massa text)
:
University of California, Berkeley
>
```

`more(1)` tar det som anländer på `stdin` och skickar ut det på `stdout`, en sida i taget. För att bläddra framåt en sida i `more(1)` trycker man på mellanslagstangenten. Med returtangenten kan man bläddra en rad åt gången,

```
> more /COPYRIGHT
:
(en massa text, men bara en sida i taget)
:
>
```

Med operatorn `|` (*pipe*, uttalas ”pajp”) kan man knyta ihop `stdout` med `stdin`. Pipe:n kan liknas vid en mellanlagringsstation. `cat(1)` levererar text in i pipen och `more(1)` läser från den och skickar sedan ut texten till sin `stdout`, skärmen:

```
> cat /COPYRIGHT | more
```

Inte så upphetsande kan tyckas, men en otroligt användbar mekanism och ett måste för att kunna utnyttja skalet effektivt. Det är enkelt att sortera raderna innan `more(1)`:

```
> cat /COPYRIGHT | sort | more
```

eller använda programmet `sed(1)` för att avlägsna tomrader och sedan räkna antalet rader med wc(1)[27]:

```
> cat /COPYRIGHT | sed '/^$/d' | wc -l
```

Det finns dussintals små program[28] i FreeBSD vilka var för sig inte gör något speciellt imponerande men vilka tillsammans på en kommadorad utgör ett mycket kraftfullt verktyg. Nästan ett programmeringsspråk i miniatyr. Överhuvudtaget är det en filosofi som genomsyrar FreeBSD: Hellre flera små program som kan kombineras ihop än ett stort program som kan allt. Kombinationsmöjligheterna och den flexibilitet man uppnår resulterar i en mycket kraftfull kommandorad.

Är man inte intresserad av programmets utdata kan dessa styras över till *the bit-bucket*, en samlingsplats för alla ointressanta bits och bytes, `/dev/null`. Exempel:

```
# cd /usr/ports
# make clean > /dev/null
```

Här används `>`-operatorn som styr över utdata till en fil och skapar filen om den inte finns[29]. I detta fall är filen en speciell device som bara kan kasta bort det den får via sin `stdin`, men `>` kan lika gärna spara undan utdata i en fil om man så vill. På så sätt kan man spara en logg-fil att studera vid senare tillfälle.

En finess som ofta används är att köra ett program i bakrunden. Med `<ctrl>-z` kan ett program som körs, stoppas och läggas som ett *jobb* i

[27] Försök göra det här genom att ”peka och klicka”...

[28] För textbehandling är `sort(1)`, `uniq(1)`, `join(1)`, `split(1)`, `fmt(1)`, `wc(1)` och små enraders program skrivna i `sed(1)` eller `awk(1)` mycket användbara. `perl(1)` är ett programmeringsspråk som också är mycket lämpat för dessa uppgifter.

[29] `>>`-operatorn kan användas för att lägga till text till en befintlig fil.

bakgrunden. Genom att skriva `bg` (*background*) kan man få programmet att köra igång igen, *i bakgrunden*. Det körs, men man har fortfarande ett fungerande skal att ge nya kommandon i. Med kommandot `fg` (*foreground*)kan programmet ”lyftas upp” till förgrunden igen. Flera program kan startas och sedan läggas i bakgrunden om man så vill. Förutom kommandona `fg` och `bg` är `jobs` praktiskt. Med `jobs` listas vilka program som för närvarande finns i bakgrunden och deras status. `<ctrl>-c` slutligen avbryter det jobb som för närvarande befinner sig i förgrunden:

`> cd tmp`	Gå till ett ”slask”-bibliotek
`> cp -R /usr/src .`	Kopiera något stort hit
`<ctrl>-z`	Stoppa körningen och lägg i bakgrunden
`Suspended`	
`> bg`	Kör det i bakgrunden
`[1] cp -R /usr/src . &`	
`> jobs`	Se efter vilka jobb som finns
`[1] Running cp -R /usr/src .`	
`> fg %1`	Flytta det till förgrunden
`cp -R /usr/src .`	(Det kyckades igen!)
`<ctrl>-c`	Avbryt programmet i förgrunden
`>`	Klar för nya kommandon

Redan vid programstart kan man lägga programmet i bakrgrundeb genom att avsluta kommandoraden med ett &-tecken:

```
> cd tmp
> cp -R /usr/src . &
>
```

Om ett kommando eller program genererar mycket utskrift kan man vilja hejda utskriften så att det går att läsa. Då använder man:

`<ctrl>-s`	Stoppa utskriften
`<ctrl>-q`	Fortsätt utskriften

Det speciella `tcsh(1)`-kommandot `!$` hämtar förra kommandots sista ord och sätter in det:

```
> cd /usr/local/bin
> ls !$
```

Kommandot `!!` används för att hämta senaste kommandot en gång till men ↑ duger lika bra.

Vill man utföra ett kommando många gånger kan det `tcsh(1)`-specifika kommandot `repeat` användas, dvs

```
> repeat 4 ls
```

är samma som

```
> ls
> ls
> ls
> ls
```

En gedigen kunskap om något skal underlättar väsentligt en systemadministratörs arbete. Det är väl värt mödan att lära sig.

`tcsh(1)` är omfattande och kan konfigureras i många avseenden, kommandon kan köras vid vissa tider, `tcsh(1)` kan korrigera felskrivna kommandon och mycket mer. Läs man-sidan.

3.4.2 `sh(1)`

`sh(1)` är det ursprungliga skalet på alla unix-system. Alla system har inte `tcsh(1)` eller några andra skal installerade, men man kan lita på att `sh(1)` finns. Nuförtiden används knappt `sh(1)` direkt på kommandoraden, men det lever kvar i åtskilliga program, *scripts*. Ett script är unix' motsvarighet till BAT-filer i DOS, dvs ett körbart program som består av rena skalkommandon. `sh(1)` är däremot väsentligt mycket kraftfullare än BAT-filer. FreeBSD använder shell-scripts i stor omfattning speciellt vid systemstart[30]. Ofta används också shell-scripts för att få installationsprogram att vara portabla mellan flera olika unixar.

Enklast är ett script några kommandorader är hopsatta i en fil till ett kommando. Tag kommandoraden

```
> cat /COPYRIGHT | wc -l
```

Om denna lagras i en fil, kanske `cat.sh`, kan scriptet exekveras genom:

```
> sh ./cat.sh
```

Som säger åt `sh(1)` att i tur och ordning utföra det kommandon som finns i filen. Vill man undvika att skriva `sh(1)` först kan filen `cat.sh` göras till ett eget körbart program. Om första raden i en fil inleds med tecknen `#!` väntar sig systemet att resten av raden är sökvägen till ett program och för att filen skall vara körbar måste den ha exekveringsrättigheter. Alltså:

```
> echo "#/bin/sh" >cat.sh
> echo "cat /COPYRIGHT | wc -l" >> cat.sh
> chmod 755 cat.sh
```

scriptet kan nu exekveras med

```
> ./cat.sh
```

som vilket annat program som helst.

`sh(1)` har normalt inte kommandohistorik med piltangenterna men efter kommandot `set -o emacs` fungerar piltangenterna som man förväntar sig.

[30] Studera till exempel `/etc/rc`, som är ett av det större start-skripten.

Man kan även skriva scripts i andra skal. Men av tradition och för portabilitet brukar man inte göra det.

Många scripts skrivs numera också i andra programspråk, `sed(1)`, `awk(1)` och framförallt `perl(1)` är vanliga.

3.4.3 Miljövariabler

Följande *environment variables*, miljövariabler, tolkas av de flesta (alla?) skal och kan vara bra att känna till:

LC_CTYPE

LC_ALL

LANG (dess betydelse för vi och lyx, datum och tidsformat)

echo $LANG

3.5 Editorer

De flesta inställningar sker genom att ändra i systemets konfigureringsfiler. Det finns grafiska konfigureringsverktyg[31] för att utföra detta, men i allmänhet är filerna såpass enkla att ändra i, att det är mycket snabbare att ändra i konfigurationsfilerna direkt med en editor.

Vilken editor skall då systemadministratören välja? Detta är ett kärt debattämne på epostlistor och annorstädes. Det finns historiskt i huvudsak två falanger: De som använder editorn `vi(1)` och de som använder editorn `emacs(1)`. Båda har sina för- och nackdelar.

För `vi(1)` talar att den alltid finns installerad med grundsystemet. `vi(1)` har en lång historia och har utvecklats av rena programmerare, på gott och ont. Den är kraftfull och kan användas till alla sorters texteditoeringar men med dess många kommandon kan den verka förvirrande till en början. Men `vi(1)` tar relativt liten plats och är snabb att starta. Den kan utan tvekan användas till allt det en administratör vill.

För `emacs(1)` talar att den är visuellt enklare än `vi(1)` och kanske mer intuitiv. `emacs(1)` installeras dock inte med automatik i ett grundsystem utan måste läggas till efteråt. `emacs(1)` är dessutom väldigt stor och tar plats både på hårddisken och senare i minnet när den används. Men med `emacs(1)` kan man utföra allt man kan önska sig och den kan dessutom konfigureras i oändlighet genom ett lisp-liknande programspråk. Av denna anledning kan `emacs(1)` inte bara användas som en editor utan också som news-läsare och e-postklient bland annat. `emacs(1)` finns i huvudsak i två varianter, en som kräver XFree, det grafiska fönstersystemet, och en som inte kräver det, utan kan köras direkt i ett terminalfönster.

[31] Webmin är mycket populär, http://www.webmin.org???

Det finns inget lätt sätt att säga vad just du ska välja. Prova båda och välj den du tycker bäst om efter några timmars övning. Låt dina förväntningar (och tålamod...) fälla avgörandet. Fråga inte på epostlistor bara! Då kommer de olika lägren att försvara sina ställningar till sista blodsdroppen och du kommer fortfarande inte att vara ett dugg klokare.

Om du bara vill ha en snabb, enkel editor för XFree och vill att den ska fungera utan överraskningar kan det vara idé att prova `nedit(1)` från ports. Den är snabb att komma igång med och kan rekommenderas till användare med windowsvana.

Bland ports kan du finna ytterligare ett stort antal editorer (bland annat ”förbättrade” `vi(1)`-varianter som bland annat `vim(1)`) för olika ändamål.

3.6 Mer information

Att skriva böcker om datorsystem är vanskligt. Utvecklingen går mycket fort och det som var högsta mode för tre år sedan kan vara hopplöst ute idag. Att skriva en bok om ett så rörligt mål är naturligtvis problematiskt. Alla ledtider för tryckning och distribution innebär att mycket datorlitteratur är något omodern redan när den givs ut. Den enda rimliga lösningen på problemet är att lagra dokumentationen elektroniskt och skicka med dokumentationen med programmen — och det är precis så man gör. I ett installerat grundsystem finns mycket dokumentation gömd i manualsidor, GNU:s info-sidor med mera. Dokumentation förekommer ofta i html-format men även andra format förekommer.

3.6.1 Man-sidorna

Först och främst brukar manualsidorna, ”man-sidorna”, användas. De fungerar som ett referenskort över ett visst kommando, någon programmeringsfunktion eller annan aspekt av systemet. Alla manualsidor har ett standardiserat utseende för att man lätt ska kunna hitta eftersökt information. Man kommer åt man-sidorna med kommandot `man(1)` (vad annars?) följt av det kommando man önskar veta mer om, exempelvis ’`man ls`’:

```
          LS(1)  FreeBSD General Commands Manual  LS(1)

NAME

     ls - list directory contents

SYNOPSIS

     ls [-ABCFGHLPRTWabcdfgiklnoqrstu1] [file ...]

DESCRIPTION

For each operand that names a file of a type other than directory, ls
```

```
displays its name as well as any requested, associated information. For
each operand that names a file of type directory, ls displays the names
of files contained within that directory, as well as any requested, asso-
ciated information.
If no operands are given, the contents of the current directory are dis-
played. If more than one operand is given, non-directory operands are
displayed first; directory and non-directory operands are sorted sepa-
rately and in lexicographical order.
The following options are available:
     -A List all entries except for '.' and '..'. Always set for the
     super-user.
     -B Force printing of non-graphic characters in file names as \xxx,
     where xxx is the numeric value of the character in octal.
⋮
⋮
⋮
COMPATIBILITY
The group field is now automatically included in the long listing for
files in order to be compatible with the IEEE Std1003.2 (■POSIX.2■)
specification.
SEE ALSO
chflags(1), chmod(1), xterm(1), termcap(5), symlink(7), sticky(8)
HISTORY
An ls command appeared in Version 1 AT&T UNIX.
STANDARDS
The ls function is expected to be a superset of the IEEE Std1003.2
(■POSIX.2■) specification.
BUGS
To maintain backward compatibility, the relationships between the many
options is quite complex.
BSD July 29, 1994 5
```

Lägg speciellt märke till rubrikerna `NAME` som ger en kort beskrivning av kommandot, `SYNOPSIS` som visar hur kommandot används, `DESCRIPTION` som kan vara en ganska lång beskrivning av kommandot samt `SEE ALSO` som räknar upp närliggande kommandon med egna man-sidor som ofta

är av intresse för att förstå kommandots hela omfattning och funktion i systemet.

Det finns många kommandon och många manualsidor. För att lättare kunna hitta bland dem är manualsidorna uppdelade i kapitel. Totalt finns det nio kapitel som vart och ett innehåller kommandon av en viss typ. Man brukar anges kapitelnumret i parentes bakom kommandot, exvis `ls(1)`. Kapitel 1 innehåller *user commands*, användarkommandon, och är kanske det mest använda kapitlet. Kapitlen är indelade enligt

1	User Commands	Användarkommandon
2	System Calls	Systemanrop
3	Subroutines	Subrutiner
4	Devices	Enheter
5	File Formats	Olika Filformat
6	Games	Spel
7	Miscellaneous	Diverse
8	System Administration	Systemadministration
9	Kernel Interfaces	Anrop till kärnan

`man(1)`-kommandot letar igenom manualsidorna i kapitelordning och slutar då den första matchningen hittats. Om kommandot råkar finnas i flera kapitel kan man ange kapitlet i argumentlistan:

```
> man intro
> man 1 intro
> man 3 intro
```

Om man inte vet exakt vad kommandot heter kan man ofta ändå hitta det med kommandot apropos,

```
> apropos editor
```

`Apropos(1)`-kommandot är detsamma som `man(1)`-kommandot med `k`-flaggan satt,

```
> man -k editor
```

Naturligtvis finns även `man(1)`-kommandot beskrivet i `man(1)`-sidorna,

```
> man man
```

Har du XFree installerat finns ett trevligt program, `xman(1)`, för att titta på manualsidorna.

3.6.2 Infosidor

Förutom manualsidorna finns det separat information om de program från *Free Software Foundation* som utgör en del av en FreeBSD-distribution, i `info(1)`-sidorna. Man kommer åt infosidorna med kommandot '`info`',

eller '`info ämne`'. I infosidorna tar man sig sedan runt med piltangenterna för att flytta sig uppåt och nedåt, returtangenten för att välja ämne och tangenten `u` kan man komma upp till föregående nivå.

Tyvärr finns det inget säkert sätt att avgöra om man ska leta i `info(1)`-sidorna eller i `man(1)`-sidorna.

3.6.3 Handbok och FAQ

I ditt nyligen installerade system finns dessutom FreeBSD-projektets handbok. Den skrivs parallellt med utvecklingen av FreeBSD och återfinns i `/usr/share/doc/handbook`. Handboken är ett massivt dokument på flera hundra sidor och den levereras i ett par varianter lämpade för olika användning.

Om du har en laserskrivare, tid (och papper!) kan du skriva ut hela handboken, `/usr/share/doc/handbook/handbook.pdf`, men troligare är att du läser den med din webbrowser, `/usr/share/doc/handbook/handbook_toc.html` eller tittar på den rena textfilen, `/usr/share/doc/handbook/handbook.txt`.

FAQ:n, en sammanställning av ofta frågade frågor (Frequently Asked Questions) och svar, är inte fullt så omfattande som handboken men kan även den vara för stor för att skrivas ut. Dess varianter återfinns under `/usr/share/doc/FAQ`.

Av de två är FAQ:n det levande dokumentet. Efter hand arbetas FAQ:n in i handboken, men ändringar införs i regel först i FAQ och sedan i handboken.

Tyvärr tycker många att det är lättare att fråga på en epost-lista eller annat forum än att läsa den medföljande dokumentationen. Försök hindra den första impulsen att fråga i ett allmänt forum utan att först kontrollerat om frågan inte återfinns i FAQ eller handboken. Många prenumeranter på epostlistor har bara modem och att behöva betala i tid och pengar för att få se samma gamla frågor besvaras med samma gamla svar är inte bara dåligt utnyttjande av resurser utan även att tära på prenumeranternas förtroende.

3.6.4 Övrig dokumentation

Förutom den mängd information som återfinns i manual- och infosidor och handbok och FAQ finns en hel del smått och gott även i övriga bibliotek i `/usr/share/doc`. Speciellt kan nämnas biblioteken `psd`, *Programmers Supplementary Documentation*, `smm`, *System Managers Manual* och `usd`, *Users Supplementary Documentation*. Informationen här är ofta av äldre datum men tillräckligt tidlös för att fortfarande vara läsvärd. För att inte ta stor plats är dokumenten komprimerade och måste packas upp innan de kan läsas. Om du exempelvis vill veta mer om FreeBSD:s filsystem kan du:

```
> cd /usr/share/doc/smm/05.fastfs
> zcat paper.ascii.gz | more
```

3.6.5 Övrig övrig dokumentation

Nätet! Här finns oöverskådliga mängder dokumentation. Bortsett från samtliga ovan nämnda dokument kan man hitta åtskilliga hemsidor med beskrivningar och lösningar av specifika problem. Flera är mycket ambitiösa och många innehåller information som inte ens hunnit komma in i FAQ:n

Se i appendix för bara några av dessa länkar.

3.7 Systemövervakning

Administratören arbetsbeskrivning är enkel: Se till att datorsystemet fungerar så att användarna kan utföra sina arbetsuppgifter. Det låter enkelt men kan innebära de mest skiftande arbetsuppgifter, som att läsa logg-filer för att kontrollera att systemet fungerar som det är tänkt, att planera systemunderhåll så att driftstörninngar minimeras, att se till att datorsystemets resurser utnyttjas på bästa sätt, att identifera och eliminera uppkomna flaskhalsar oavsett om dessa beror på hårdvara (processorkraft, nätverksbelastning och vad det vara må) eller mjukvara (minneskrävande program exempelvis). Dessutom måste systemadministratören numera hålla sig à jour med de senaste händelserna på datorsäkerhetsfronten och vara beredd att installera säkerhetsfixar med kort varsel. Att prenumerera på epostlistor och relevanta nyhetsbrev hjälper också till att hålla systemadministratören *up-to-date*.

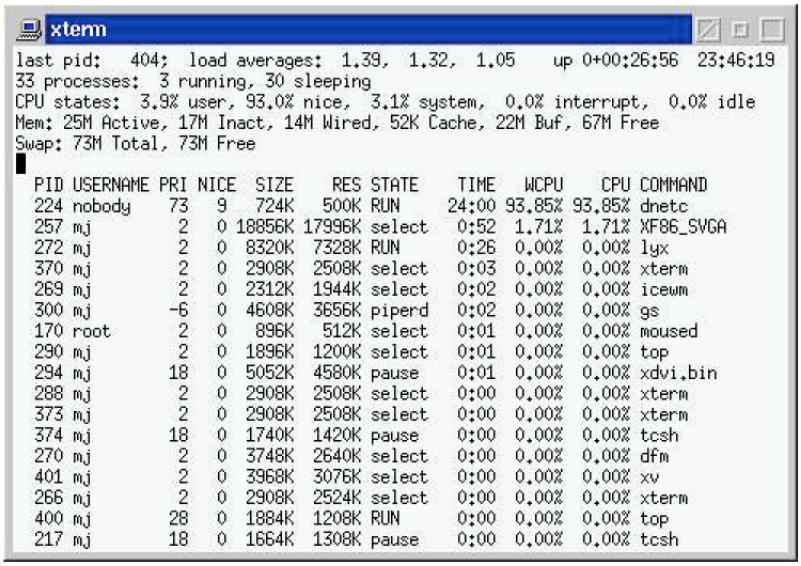
Ofta är det systemadministratören som, kanske i samråd med användarna och andra, avgör vilka program som ska användas i systemet och i vissa fall är det även hans lott att utbilda användare och ge support på programvaran.

Oavsett datorsystem är hänger det totala systemets effektivitet, prestanda och åtkomlighet mycket på hur väl administratören lyckas i sin uppgift. I fallet med FreeBSD underlättas uppgiften eftersom FreeBSD redan från första installationen erbjuder höga prestanda med moderata till låga hårdvarukrav. Administratören kan dessutom kontrollera hela sitt nät med samtliga klientmaskiner från en enda dator, inte nödvändigtvis samma som servern. Med programvaran `ssh(1)` (som beskrivs senare) kan administratören även logga in på vilken som helst av datorerna för att kontrollera funktion eller felsöka.

I och med att samtliga datorer är hopknutna genom ett nätverk ger detta administratören en unik möjlighet att studera, inte bara serverns prestanda, utan varje i nätet ingående maskins parametrar, det kan gälla upptider och belastning med mera. Här redovisas ett antal kommandon administratören kan använda för att analysera en enskild maskins belastning och status, i kapitel XXX beskrivs sedan några speciella nätverkskommandon man kan använda för att analysera nätet som helhet.

På en enskild dator är kommandot `top(1)` mycket användbart. Med `top(1)` avslöjas löpande, sekund för sekund, flera av de parametrar som berör da-

torns status: antalet processer, vilka de är, vad de håller på med i huvudsak, swaputnyttjande och systemets belastning till exempel.



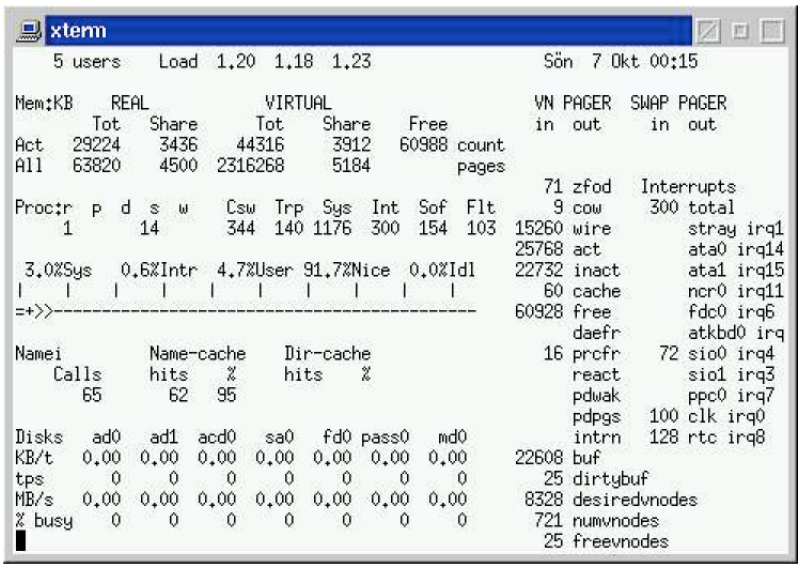
Resultat från `top(1)` kan ses i bilden ovan. De viktigaste parametrarna är *load averages*, medelvärden av systemets belastning under den senaste minuten, 5 minuterna och 15 minuterna, *processes* som talar om hur många processer (program) systemet kör, *swap*, hur mycket swaparea som används för närvarande samt kolumnerna *SIZE* och *COMMAND*.

- Systembelastningen, *load averages*, beräknas enligt en formel som tar hänsyn till hur olika delar av datorn används. En ”normal” belastning är under 4-5. Däröver är datorn så lastad att svarstiderna kan bli lidande. Det är dock inget alarmerande fel i detta. FreeBSD uppför sig mycket elegant även under tung belastning och en hårt belastad filserver kan mycket väl ha en högre siffra utan att klientmaskinerna märker av det.

- Fler processer igång samtidigt betyder bara att datorn har mer att göra. Det betyder inte nödvändigtvis att systemets belastning är hög. Processer kan ligga och vänta på att få tillgång till, exempelvis, hårddisken och under den tiden ingår de inte i systemets totala belastning.

- Swap-användningen slutligen, kan vara en indikation på om datorns RAM-minne är tillräckligt stort eller inte. Program som inte används för stunden kan skickas ut till swaparean för att frigöra RAM. Så länge de ligger där, och inte används, påverkar de inte systemets prestanda men om systemet ständigt måste nyttja swap-arean betyder detta att RAM-minnet inte räcker till. Speciellt allvarligt är det om de översta programmen i `top(1)`:s lista har en sammanlagd *SIZE* som överstiger installerat minne, då kan man var rätt säker på att systemet måste swappa för att köra dessa program. Lösningen är att minska antalet program som körs eller att bestycka datorn med mer RAM.

Tungviktaren bland de program som används för att analysera systemets status är utan tvekan `systat(1)`. Med `systat(1)` kan man erhålla en mycket ingående beskrivning av vad systemet håller på med. Kommandot beskriver all tillgänglig information vad gäller swap, dataflöde till/från hårddiskar, nätverksutnyttjande och mycket mer. I sin enklaste skepnad visar `systat(1)` med termometerskalor systembelastningen och de program som körs. Mer (mycket mer!) information ges om `systat(1)` startas med växeln `-vm`. Då visas en tablå över systemets minneshantering och denna innehåller oftast den information man är ute efter:



`man(1)`-sidan innehåller en detaljerad redogörelse för de olika fälten, en del kan vara nog så kryptiska om man inte äger ingående kunskaper i virtuell minneshantering. I huvudsak kan man extrahera följande: antal inloggade användare, systembelastning, minnets användning, swapens utnyttjande och dataflöde till/från yttre enheter som exempelvis hårddiskar, cd-spelare och bandstationer.

I kolumnen *Interrupts* kan man också avläsa hur många avbrott systemet har behövt göra sedan senaste uppdatering av skärmen. Även i vila sker åtminstone 100 avbrott per sekund, nämligen de avbrott som håller ordning på systemklockan. Det är inte orimligt med över tusen avbrott i sekunden!

Medan de två ovan beskrivna kommandona löpande redovisar systemet visar `df(1)`-kommandot en ögonblicksbild av vilka lagringsenheter som är anslutna till systemet och deras nyttjandegrad:

```
> df
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/ad0s4a     79359    45363    27648    62%    /
/dev/ad0s4f   2099112  1482758   448426    77%    /usr
/dev/ad0s4e     29751     7443    19928    27%    /var
/dev/ad0s1     851200   758336    92864    89%    /mnt/c
procfs              4        4        0   100%    /proc
```

Man ser att en hårddisk med två partitioner är monterade, `/dev/ad0s1` och `/dev/ad0s4`. Man kan vidare se att rot-filsystemet `/dev/ad0s4a` är cirka 80MB stort varav 62% används och så vidare.

4 Fönstersystemet XFree86

4.1 Fönsterhanterare

Det finns numera flera mycket kompetenta fönsterhanterare att välja mellan. Vilket som väljs beror mycket på vad användaren förväntar sig av sin dator. Om man färdigställer datorer för andra än sig själv är det fullt möjligt att installera flera fönstersystem och låta den slutlige användaren avgöra vilket han gillar bäst. Här visas bara tre stycken: KDE, GNOME, icewm.

KDE

GNOME

ICEWM

Valet är ditt. Personligen tycker jag att KDE och GNOME är för utrymmes- och resurskrävande och jag använder icewm som både är liten och snabb. Men det är bara mitt val, du, eller dina användare kanske ser fördelar med KDE, GNOME eller något annat.

4.2 XDM.

5 Nätverk

I verkligheten är det egentligen bara en typ av nät som kommer ifråga för att koppla ihop ett mindre antal datorer med FreeBSD: TCP/IP. TCP/IP är det i särklass vanligaste protokollet för att skapa mindre och större[1] nätverk.

På senare tid har fiberoptik blivit intressant ut kostnadsperspektiv, speciellt för fasta nät i bostäder, kontor och liknande. En fiber erbjuder mycket hög bandbredd och vill man vara förutseende, eller om man vet att nätet kommer att användas för bandbreddskrävande tillämpningar i framtiden, kan det vara värt att undersöka fiberalternativet. FreeBSD stöder även fiber om det skulle behövas. Fram till den enskilda datorn är troligen TP fortfarande rimligt, men om man drar kabel i ett hus kan det vara idé att redan från början dra fiber. Observera att en del nödvändig hårdvara, som hubbar och switchar, är väsentligt dyrare för fiber än för TP. Prisbilden ändras dock säkert med tiden och om några år kan fiber vara det enda rimliga alternativet.

Nätverkskort

Samtliga moderna nätverkskort är s.k. PCI-kort. FreeBSD identifierar dem automatiskt utan extra inställningar. Äldre datorer använder den s.k. ISA-bussen (*Industry Systems Architecture*) och kräver därmed särskilda nätverkskort. Ofta måste FreeBSD informeras om dessa nätverkskorts interupptnummer (IRQ) och address. Kontrollera med dokumentationen om FreeBSD kan hantera ett viss kort innan du köper det. I en del fall säljs samma kort under flera olika namn. I sådana fall kan det vara svårt att veta om kortet stöds.

5.1 Kablage

För den fysiska kopplingen mellan datorerna finns för närvarande i huvudsak två möjligheter:

- Koaxialkabel. Koaxialkabeln används knappast vid nyinstallationer men om datorernas nätverkskort är av en äldre modell kanske de endast kan kopplas till koaxialkabel. Kontaktdonet är av BNC-typ.

[1] Hela Internet går i princip på TCP/IP.

- FTP/UTP-kabel. Den för närvarade mest använda kabeln. Kabeln finns i två varianter beroende på om datatakten i dem ska vara 10MBPS eller 100MBPS. Den förra kallas 10BaseT och den senare 100BaseT. Kontaktdonet är en s.k. modularplugg av typen RJ-45.

5.1.1 Koaxialkabel

Koaxialkabel är den äldre av de båda metoderna och nya nätverkskort brukar inte ha kontakt för det längre, men för bara några år sedan var koaxialkabel det enda som erbjöds. Övertar man äldre datorer ska man vara beredd på att de kanske endast har kontakter för koaxialkabel.

En koaxialkabel är en kabel som består av en central ledare omgiven av en strumpa som utgör den andra ledaren. Den vanligast sorten heter RG-58/U och har en impedans på 50Ω. Det är ytterst viktigt att aldrig använda kabel med annan impedans. Det finns en till utseendet liknande koaxialkabel med impedansen 75Ω! Koaxialkabelnät har en högsta hastighet på 10MBPS.

Kabeln måste dras så att den går från dator till dator och avslutas med en *terminering* på 50Ω i ändarna. Ett brott på kabeln eller vanligare, en plötslig knyck på den, kan ge sådana impedansreflektioner att signalen distorderas totalt eller, värre, ger egendomliga och mycket svårfunna felsymptom. Den är för övrigt enkel att dra och kräver inte normalt (i ett subnät) ytterligare hårdvara.

Figur X. Koaxialkontaktdon. Från vänster till höger ser vi BNC-kontakt på koaxialkabel, skarvdon s. k. *T-stycke* och termineringsdon.

5.1.2 10/100BaseT

Efter att ha slagits med koaxialkabelnätverk i några år kom BaseT som en befriare. Den är mycket enklare att dra och fungerar i allmänhet omedelbart utan några som helst problem. I sin enklaste form kan två datorer kopplas ihop med bara en TP-kabel. Ingen terminering, som för koaxialkabeln, behövs. Däremot måste kabeln vara *korsad* för att kommunikationen ska kunna upprättas.

Om flera datorer ska kopplas samman skall en *hub*, nav, eller *switch*, växel, användas.

En hub eller nav fungerar som en förgrening, med en ingång och kanske 2, 5, 8, 20 eller fler utgångar. Kabeln mellan ingången och en dator måste vara korsad, medan utgångarnas kablage skall vara raka. Ofta finns ett speciellt markerat uttag på hubben som redan internt i hubben är korsad, i sådana fall behövs ingen korsad kabel alls. Moderna hubbar och switchar kan automatiskt känna av om en ingång ska vara korsad eller rak.

Switchen är den mer sofistikerade av de två. Medan ett anrop in till en hub till en viss dator går ut till alla datorer vilka själva får avgöra om de är den rätta, lär sig en switch vilken eller vilka datorer som sitter på en viss utgång och låter bara anropet gå ut i rätt kabel. Detta är fördelaktigt då antalet kollisioner på så sätt nedbringas till ett minimum.

Figur X. Hub av fabrikatet D-link. Denna hub har 16 portar, men även 8 portar är vanligt förekommande. Uttaget längst till vänster är korsat internt. En switch ser i huvudsak likadan ut.

6 Internetprotokollet

6.1 Allmänt

Ett internet består av ett antal elektroniskt hopkopplade datorer. Det behöver inte vara anslutet till Internet som är det stora världsomspännande datornätverket, det räcker med i princip två datorer för att ett internet ska uppstå. Dataöverföringen sker i vårt fall enligt *internetprotokollet*, en samling regler som definierar hur sändare och mottagaren skall uppträda. Data överförs i små bitar, *paket*, från en sändare till en eller möjligen flera mottagare, ofta bara cirka 1500 byte per paket. Om man ska överföra stora mängder data sköter datorn om att paketera datat i lämpliga storlekar innan de slussas ut på nätet, iväg mot sin destination

Överföringskanalen tillåter, oavsett om mediet är koaxial- eller TP-kabel, bara en sändare åt gången. Om två sändare skickar iväg var sitt paket vid exakt samma tillfälle *kolliderar* paketen och informationen i dem går förlorad. Internetprotokollet har en mekanism för att känna av kollisioner och försöker då sända om paketen. Omsändning sker inte vid exakt samma tidpunkt för alla datorer, den exakta tidpunkten styrs av funktioner i internetprotokollet.

Omsändning är inte på något sätt fatalt utan utgör en naturlig del av vardagen för den s.k. *TCP/IP-stacken* som handhar dataöverföringen. För mycket kollisioner medför förstås till slut att den effektiva bandbredden minskar och det patologiska fallet inträffar då samtliga datorer ägnar sig åt omsändning och ingen egentlig data överförs.

6.2 IP-adresser

Oavsett storleken på internetet måste varje dator ha en address, s.k. *internetadress* eller *IP-adress* eller ofta bara *IP-nummer*. IP-adressen utgörs av ett 32-bitars tal. Med 32 bitar kan man numrera datorerna från 0 till $2^{32} - 1$, det vill säga cirka 4 miljarder. Ett 32-bitars tal ser enklast ut som exempelvis

```
01001111111100000101010110100101
```

Ett sådant tal är svårt att memorera (eller?), och det är brukligt att dela upp det i 4 oktetter, grupper om 8 bitar, med en punkt som separator:

```
01001111.11110000.01010101.10100101
```

Man brukar slutligen översätta varje oktett till decimal form och sålunda kan 32-bitars adressen skrivas

```
92.123.12.9
```

Det lägsta tänkbara och det högsta tänkbara decimalt tecknade IP-adresserna är alltså `0.0.0.0` respektive `255.255.255.255`.

6.2.1 Subnet

Om alla de runt 4 miljarder datorer som IP-adressrymden ger möjlighet till, satt ihop med varann direkt skulle det behövas en enorm central växel för att koppla ihop varje dator som vill skicka något med den, eller de, som vill ta emot meddelandet. Detta skulle omedelbart resultera i kraftiga ”trafikstockningar” och har även andra nackdelar. Man har löst problemet genom att indela adressrymden i s.k. *subnät*. Ett subnät fungerar så att alla ipadresser inom det ’ser’ och kan nå varann omedelbart. Det behövs ingen växel utan alla kan direkt ta kontakt med varje annan adress i samma subnät.

RFC1918[1] definierar ett antal subnät som inte får användas allmänt (de routas i allmänhet aldrig mellan subnät) utan som speciellt är skapade för lokala, privata, nät. RFC1918 fastslår följande ipnummerområden för privata nät:

```
10.0.0.0      -  10.255.255.255
172.16.0.0    -  172.16.255.255
192.168.0.0   -  192.168.255.255
```

`192.168.0.0` är ett vanligt subnät som sträcker sig från `192.168.0.0` till `192.168.255.255` och ger alltså plats för drygt 65000 datorer, även detta ett groteskt stort antal datorer i samma subnät. För att undvika stockningar och kollisioner är det lämpligt att dela upp även detta subnät i mindre delar. Det är exempelvis rimligt att låta ett lokalt nätverk definieras av subnätet `192.168.0.0` till `192.168.0.255`.

För att beteckna storleken på det lokala nätet använder man en *nätmask*. Nätmasken anger hur stor del av en IP-adress som är fix för subnätet (*host part*) och hur stor del av adressen som är rörlig (*client part*). Subnätet som sträcker sig från `192.168.0.0` till `192.168.0.255` innehåller de möjliga adresserna `192.168.0.1`, `192.168.0.2`, `192.168.0.3` och så vidare, ända till `192.168.0.255`. Den fixa delen av adressen är tydligen `192.168.0`. Om man skriver `?` för de bitar som är rörliga kan man skriva subnätets IP-adresser i binär form, som

IP-adress	`11100000.10101000.00000000.????????`	`192.168.0.?`
nätmask	`11111111.11111111.11111111.00000000`	`255.255.255.0`

[1] Request For Comment. De flesta standarder för internet började sitt liv som en RFC, ett förslag. Ofta implementerades de och blev på så sätt *de facto*-standarder innan RFC:n var färdigbehandlad.

där nätmasken alltså utförs med med ettor så länge IP-adressen är fix och med nollor för övrigt.

Subnätet ovan skrivs ofta även som `192.168.0.0/24`, då nätmasken inleds med 24 ettor, eller kortare som `192.168.0`. I det senare fallet underförstås att endast *host*-delen angetts. Samma subnät kan alltså beskrivas på flera olika sätt.

Om man vill, kan man skräddarsy sitt subnät utefter de antal IP-adresser man behöver, men det är ingen teknisk nödvändighet att göra det. Outnyttjade IP-nummer eller IP-nummerserier med luckor eller dylikt påverkar inte prestanda på något sätt. Enklast är ofta att ta subnätet `192.168.0/24` och om man behöver flera subnät kan man enkelt välja dessa som `192.168.1/24`, `192.168.2/24` osv.

Det är alltså inget fel att skapa subnätet `192.168.0/30`, om man nödvändigtvis vill det. Däremot är det inte bra att låta för många datorer samsas i samma subnät eftersom risken för kollisioner och omsändningar ökar med antalet datorer. Det exakta antalet avgörs från fall till fall, beroende på, bland annat, nätverkshastighet och trafikmängd.

6.2.2 Gateway och routing

Inom ett subnät kan, som beskrivits ovan, alla datorer fritt kommunicera med varann. Ofta vill man att subnätet även ska ha tillgång till Internet eller kanske ett annat subnät. En dator i ett subnät har från början ingen kännedom om datorer i andra subnät. Om en dator i nätet `192.168.0/24` vill sända ett meddelande till `136.138.1.12` blir den rådvill. Vad ska göras med denna helt okända adress? *Routingtabellen* har svaret! En routingtabell innehåller den information datorn behöver för att leverera ett IP-paket. Den kan exempelvis för datorn `192.168.0.101` se ut som[2]:

```
Destination  Gateway    Flags  Refs  Use  Netif   Expire
127.0.0.1    127.0.0.1  UH     1     64   lo0
192.168      link#1     UC     3     0    xl0 =>
```

Manualsidan för `netstat(1)`-kommandot beskriver de olika fälten i detalj. De viktigaste är `Destination`, `Gateway` och `Flags`:

`Destination och Gateway` Första raden är självklar. För att sända till `127.0.0.1` — skicka paketet till `127.0.0.1`. `127.0.0.1` är en pseudonym för den egna datorn. Denna adress finns alltid på varje dator och betyder alltid den lokala datorn, den s.k. *localhost*-adressen. FreeBSD använder detta som en genväg för en del funktioner som skickar paket internt i datorn.

Den andra raden är intressantare. Av den kan vi utläsa att för att komma till `192.168` ska datorn skicka paketen till `link#1` som är ett nätverkskort.

[2]Kommandot `netstat -rn` visar routing-tabellen.

Tydligen är detta nätverkskort anslutet till nätet `192.168`. Sista kolumnen, `Netif`, anger vilket nätverkskort, *network interface*, som avses, i det här fallet `xl0`.

`Flags` `Flags`-fältet anger en del egenskaper för respektive rad. Här betyder `U` (*Usable*) att routen är aktiv och `H` (*Host*)att den leder till en enskild dator och `C` att den pekar ut en väg till ett subnät.

Ett paket till det egna subnätet levereras enligt sista raden i routingtabellen ovan. Paketadressen stämmer med `192.168.0/24` och vi överlåter åt den egna maskinen att ta kontakt med rätt mottagare. Så långt inga problem. Men ett paket till, säg, `136.168.1.12` stämmer inte med något i *Destination*-kolumnen, det tillhör ju varken `192.168` eller `127.0.0.1`. I detta fall kommer datorn inte kunna leverera paketet och det kastas helt sonika bort. Paketet kommer aldrig ut från nätverkskortet!

Vad som saknas är en *gateway* för de paket som inte matchar *Destination*-kolumnen, en slags utkorg för sånt vi inte kan reda ut själva. Denna ”utkorg” kallas *default gateway* och är en IP-adress till en maskin som (vi antar) vet mer, än vi själva, om addressaten till de paket vi skickar till den.[3] En routingtabell med default gateway, är då:

```
Destination  Gateway         Flags  Refs  Use  Netif   Expire
127.0.0.1    127.0.0.1       UH     1     64   lo0
192.168      link#1          UC     3     0    xl0 =>
default      192.168.0.251   UGSc   0     0    xl0
```

Paket som vi inte kan leverera själva — det vill säga lokalt inom det egna subnätet — skickas till datorn `192.168.0.251` och sedan är den saken ur världen. Flaggan `G` betyder att denna route behöver redas ut ytterligare av någon annan, i detta fall överlåter vi åt `192.168.0.251` att reda ut vart paketet ska skickas.

En komplett routing-tabell innehåller åtminstone en rad till, nämligen *broadcast*-adressen. Vi återkommer till denna teknikalitet i avsnitt 6.2.3 som beskriver `arp(8)`.

`192.168.0.251` är en *router*, en dator vars uppgift är att reda ut vart inkommande paket ska ta vägen, och vidarebefordra dem. FreeBSD kan även användas som router om så önskas. En router har (minst) två nätverkskort, ett per nät den betjänar, och har i och med detta även (minst) två IP-adresser. Även routern har en routingtabell för att klara av jobbet, men innehållet är förstås helt annat än den hos klienten. Man kan tänka sig följande routingtabell på routern:

[3] Man anger normalt *default gateway* i samband med att man konfigurerar datorns IP-nummer.

```
Destination     Gateway           Flags   Refs   Use      Netif    Expire
default         95.12.3.200       UGSc     83   2102668  fxp0
127.0.0.1       127.0.0.1         UH        7   1994250  lo0
95.12.4/21      link#2            UC       14   0        fxp0 =>
95.12.4.101     0:a0:c9:44:2b:9   UHLW      0   112681   lo0
192.168.0       link#1            UC       49   0        xl0 =>
192.168.0.251   0:1:2:9b:12:2f    UHLW      0   16970    lo0
```

Ur denna tabell kan man utläsa att

- routerns inre namn är `192.168.0.251`, för att paket adresserade till denna adress skall levereras genom `lo0`. Routerns yttre namn är av samma skäl `95.12.4.101`. Då `192.168` är ett privat nät måste detta nät vara det lokala.
- routerns inre nät är `192.168.0`, där nätmasken `255.255.255.0` är underförstådd. Routerns yttre nät är `95.12.4/21`, dvs dess nätmask är `255.255.248.0`.
- routerns inre nätverkskort är `xl0` med Ethernetadressen `0:1:2:9b:12:2f`, dess yttre nätverkskort är `fxp0` och har Ethernetadressen `0:a0:c9:44:2b:9`. Ethernetadressen avhandlas i avsnitt 6.2.3.
- routerns default gateway är `95.12.3.200`.

HÄR BEHÖVS EN BILD!

6.2.3 `arp(8)`

Ytterligare ett protokoll figurerar i bakgrunden för att få nätverket att fungera, nämligen `arp(8)`, *Address Resolution Protocol*. Att en dator (egentligen ett nätverkskort) skulle identifieras med sin IP-adress är att tänja lite på sanningen. I själva verket känner nätverkskortet bara till sin egen *Ethernetadress*, ibland även kallad *MAC*-adress (*Media Access Control*).

Redan vid tillverkningen av ett nätverkskort förses det med ett unikt serienummer i form av sin Ethernetaddress. Det får inte förekomma två kort med samma adress. En Ethernetaddress kan se ut som `00:00:C0:F2:7D:30`, och ska inte förväxlas med kortets IP-adress. Oftast behöver man inte bry sig om vad Ethernetadressen är, det är bara `arp(8)` som intresserar sig för det.

För att kunna skicka ett paket måste sändaren veta vilken Ethernetadress paketet ska till, men vi har ju tidigare visat att IP-adressen är tillräcklig. Så vad händer egentligen?

Om vi har ett paket till `192.168.0.20`, och detta är det första paketet sedan den egna datorn startades, måste datorn på något sätt få reda på Ethernetadressen till mottagaren för att kunna adressera paketet rätt, och

det är precis vad `arp(8)` gör för oss. IP-adressen `192.168.0.20` är okänd och `arp(8)` skickar ut en allmän förfrågan, en *broadcast*, till hela subnätet: ”Vem är `192.168.0.20`?”. Broadcast är en form av massanrop på nätverket och sker på en speciell IP-adress. Alla mottagare inom ett subnät lyssnar på denna broadcast-adress från varje annan dator i subnätet.

I detta fall antar vi att någon svarar: ”Jag, med Ethernetadressen `00:00:C0:F2:7D:30` har IP-adressen `192.168.0.20`”. Och saken är klar, vi har fått kopplingen mellan Ethernetadress och IP-adress. En `arp(8)`-fråga kan även routas om frågan gäller en IP-adress utanför det egna subnätet.

För att inte belasta nätet med ideliga förfrågningar av denna sort kommer `arp(8)` ihåg (cachar) denna koppling en stund, oftast 1200 sekunder, varefter en ny förfrågan måste göras. På köpet har alla andra på subnätet tjuvlyssnat och även dom uppdaterat sina cachar med denna information. Det räcker alltså med att en enda dator skickar en `arp(8)`-fråga och får svar för att alla ska dra nytta av frågan. Hur lång tid en adress har kvar i cachen innan den raderas anges i routing-tabellen, i kolumnen `Expire`.

Med kommandot `arp(8)` kan man själv se vad cachen innehåller för tillfället:

```
>arp -a
pc101.bsd.lab   (192.168.0.101)  at  0:10:5a:b1:f6:13  permanent  [ethernet]
server.bsd.lab  (192.168.0.251)  at   0:1:2:9b:12:2f              [ethernet]
?               (192.168.0.255)  at   ff:ff:...:ff:ff  permanent  [ethernet]
```

Man ser att pc101 känner till sig själv, servern i subnätet och broadcastadressen.

Slutligen, vilken är då IP-broadcastadressen? Den är alltid den högsta adress som subnätet tillåter (Ethernetadressen för broadcast är alltid `ff:ff:ff:ff:ff:ff`) och utgående från egen IP-adress och nätmask kan man lätt räkna ut broadcastadressen genom att fylla på med ettor de ställen där den egna adressen har nollor i nätmasken:

Egen IP-adress	`192.168.000.003`
Nätmask	`255.255.255.000`
Broadcast-adress	`192.168.000.255`

För tydlighetens skull har extra nollor lagts till ovan. Gör inte det vid konfiguration av nätverkskortets IP-adress! Tal som anges med inledande nolla tolkas av systemet efter det oktala talsystemet. Och det är oftast inte vad vi vill. Det är lätt att inse att man inte ska låta någon dator i sitt nätverk ha denna broadcast IP-adress som sitt eget IP-nummer.

6.2.4 Kommandon

Nätverksinställningarna görs vanligen med programmet `/stand/sysinstall` en gång för alla vid installationen av FreeBSD. Även efterkonfigurationer

är lättast att verkställa med `/stand/sysinstall`-programmet. Efter att inställningarna gjorts i det grafiska interfacet anropar sedan programmet de kommmandon som behövs.

Men man kan mycket väl konfigurera upp nätverkskortet för hand om man vill. Och även om man använder `/stand/sysinstall` är det lärorikt att veta vilka kommandon som *egentligen* används. Som vanligt finns många relaterade kommandon men med kunskap om bara några få kan man utföra både konfiguration och även felsökning. Följande kommandon tillhör de mest använda:

- `hostname(1)` används för att tilldela datorn sitt namn. Man kan ange datornamnet antingen som enbart datornamnet (den s.k. *host*-delen) eller också som ett FQDN, då både host- och nät-delen anges. Vilket som väljs avgör hur *resolvern* fungerar. Se mer om detta i avsnittet om DNS, 6.3.

- `ifconfig(1)` används för att konfigurera nätverkskortet. Man kan här ange kortets IP-adress och nätmask bland annat. Ett typiskt kommando kan vara `ifconfig xl0 192.168.0.101 netmask 255.255.255.0`. Om `ifconfig(1)` används utan parametrar, eller med argumentet `-a`, visas nuvarande konfiguration.

- Med kommandot `route(1)` manipulerar man routingtabellen och definierar även default gateway. Oftast sköter systemet detta automatiskt men vid felsökning kan det vara praktiskt att manuellt lägga till eller radera enskilda router. Med `route add -host 192.168.0.200 192.168.0.101` kommer en specifik route till datorn `192.168.0.200` med `192.168.0.101` som gateway att läggas till routingtabellen. På liknande sätt kan man lägga till hela nät om man så vill. Default gateway kan definieras som `route add 192.168.0.251 default`.

- `netstat(1)` är en verklig arbetshäst i sammanhanget. Med `netstat(1)` kan man bland mycket annat studera routingtabellen (`netstat -rn`) och visa förbindelsestatistik (`netstat -I` *interfacenamn* `-w 1`). I det senare fallet visas antal sända respektive mottagna paket och antal kollisioner, sekund för sekund. Antalet kollisioner är en indikation på nätverkets belastning. Kollisioner medför alltid att parterna utför omsändning som i sin tur betyder att nätverkets bandbredd utnyttjas ineffektivt. Om andelen kollisioner är mindre än, säg, ungefär 5 procent behöver man inte oroa sig, men om nätet kontinuerligt har över 10 procent kollisioner kan man behöva vidta åtgärder. Enstaka toppar med över 10 procent kollisioner är ingen anledning till oro, det anger bara att nätet används flitigt och det är ju det det är till för. Om värdena i kolumnen `errs` är högt — de ska normalt vara noll — kan detta tyda på felaktigheter i kableringen.

- `arp(1)` används mest vid felsökning och då för att studera arp-cachen (`arp -a`). Arp-cachen innehåller de aktuella kopplingarna mellan IP-adress och Ethernetadress.

- `ping(1)` är felsökningsverktyget framför alla. Antagligen mycket för att det är så enkelt att använda. Med `ping(1)` skickar man ett paket till en adressat angiven med IP-adress eller namn och om paketet kommer fram skickar denna paketet tillbaka till sändaren igen. Den sändande datorn mäter tiden det tog för paketet att ”studsa” och komma tillbaka, tider som mäts i millisekunder över ett direktkopplat TCP/IP-nätverk och kanske upp till något hundratal millisekunder över modem. I normalfallet sänder `ping(1)` tills man bryter det med <CTRL>-C och visar då hur många paket som skickats, hur många procent av dom som inte kom tillbaks och lite statistik över ping-tiderna. Normalt ska mängden förlorade paket vara noll, men momentant kan paket förloras genom kollisioner med annan trafik, eller genom att den mottagande datorn inte hinner med att besvara alla. Även felaktig kablering kan ge upphov till höga procentsiffror här.

Vi har nu gått igenom hur datorns grundläggande nätverkskonfiguration utförs. Det är ett rätt omfattande stoff och är definitivt grunden för fortsatta studier. Det kan vara lämpligt att befästa kunskaperna med någon träning innan vi går vidare. Använd alltså kommandona ovan för att etablera kontakt mellan några datorer, och prova att ändra routingtabellen, IP-adress osv. Experimentera!

6.3 DNS

De allra flesta adresser på Internet ser inte ut som adresserna ovan. De kan se ut som `www.freebsd.org` eller `www.yahoo.com`. En sådan adress kallas *FQDN*, Fully Qualified Domain Name, om den förutom ett datornamn (`www`) också innehåller ett komplett domännamn (`freebsd.org`). Även om en adress anges med sitt FQDN och inte med sin IP-adress hittar paketen fram. Hur? Hemligheten ligger i *DNS*, Domain Name System, och programmet `named(8)` som ingår i *bind*, Berkeley Internet Name Daemon. På liknande sätt som `arp(8)` kopplar ihop IP-adresser med Ethernetadresser, kopplar bind ihop en FQDN med rätt IP-adress. Detta sker genom en *namnuppslagning* och systemet för detta benämns DNS.

Fördelarna med DNS är flera. Det är uppenbart lättare att komma ihåg en adress som går att läsa, än en som består av en radda siffror med punkter emellan. En mer subtil fördel är att DNS tillåter att IP-adressen ändras! Om `www.freebsd.org` byter nätverksleverantör skulle med all säkerhet IP-adressen ändras, men namnet `www.freebsd.org` behåller sin giltighet och alla som söker efter `www.freebsd.org` skulle fortfarande automatiskt hitta fram till rätt IP-adress.

I sin enklaste form innehåller filen `/etc/hosts` all information för namnuppslagning. Filen måste i sådana fall finnas lokalt på varje dator som behöver namnuppslagning[4]. Detta blir snart otympligt både på grund av

[4]Eller göras tillgänglig på annat sätt, till exempel via NIS. Se avsnitt 7.6.

filens storlek, om alla Internets adresser skulle ligga i den, och på grund av det rent administrativa problemet att byta samtliga filer — på alla datorer — så fort någon post i den ändrats. I Internets barndom gjorde man faktiskt på detta sätt, men i och med att fler och fler datorer anslöts till Internet blev hanteringen med tiden otymplig och vid till 1980-talets mitt var det nödvändigt att lösa problemet. och komma bort från en allt större `/etc/hosts`-fil.

Den bekväma vägen, och i de långa loppet enda rimliga vägen, är att upprätta en *namnserver*. En namnserver är en automat som ger svar på frågor av typen ”vad har datorn `pc101.bsd.lab` för IP-adress?” (*framåtuppslagning*) och ”vem har IP-adressen `192.168.0.164`?” (*bakåtuppslagning*). FreeBSD:s namnservertjänst bind har sina konfigurationsfiler i katalogen `/etc/namedb/`.

Fullt utbyggd är namnuppslagningstjänsten en komplicerad apparat. Här kommer bara nämnas det som är av intresse för att få igång en lokal namnserver[5] vars uppgift är att utföra namnuppslagning inom ett subnät.

Mekanismen för DNS styrs av filen `/etc/host.conf` på varje dator. Den innehåller bara några få rader:

```
hosts
bind
```

Denna fil anger i vilken ordning de olika namnuppslagningsmetoderna kommer att användas. Med innehållet ovan kommer filen /etc/hosts frågas först, och om den inte hittar något konsulteras namnservertjänsten bind. Sökordningen kan omordnas om så önskas.

För den egentillverkade domänen `bsd.lab` definierar följande filer namnuppslagningen:

`/etc/namedb/bsd.lab`

För framåtuppslagningen används informationen i denna fil. Bortsett från de rader som definierar kopplingen mellan namn och IP-adress inleds filen med *options* och en s.k. *SOA-record*, Start of Authority:

```
$TTL      3600
bsd.lab.  IN    SOA  server.bsd.lab.  root.server.bsd.lab  (
                       20010808         ; serial
                       3600             ; refresh
                       900              ; retry
                       3600000          ; expire
                       600 )            ; minimum
```

[5]Strängt taget kanske det är lite överdrivet att ha en egen lokal namnservertjänst i sitt interna nät men om nätet överstiger ett halvdussin datorer tjänar man i allmänhet på att ha det. Det är dock ingen nödvändighet, kör med enbart `/etc/hosts`, om det räcker. Om uppkoppling till Internet ska ske är det dock alltid en fördel med en lokal namnservertjänst.

Låt inte denna något kryptiska inledning förfära. Det mesta är standardinställningar och det viktigaste är lätt att sammanfatta: `bsd.lab` är namnet på den domän filen har information om. `server.bsd.lab` är namet på den dator som filen ligger på och `root.server.bsd.lab` betyder att eventuell epost till anvarig ska skickas till `root@server.bsd.lab`. Observera att man ska sätta in `@` istället för `.` för att få epost-adressen. Observera också att `bsd.lab` måste följas av en punkt (`.`). En avslutande punkt i ett namn betyder att namnet är komplett och utan punkten skulle datorn försöka lägga till sitt eget domännamn.

Härnäst i `test.lab` följer raden

```
bsd.lab.  NS  server.bsd.lab
```

som beskriver att datorn `server.bsd.lab` är namnserver för domänen `bsd.lab`.

Med alla formalia sålunda ifyllda följer nu namnserverns innehåll, kopplingarna mellan hostname och IP-adress:

```
; samtliga datorer i lab-domänen
pc101                           A  192.168.0.101
pc102                           A  192.168.0.102
pc103.bsd.lab.                  A  192.168.0.103
```

Filen kan fyllas på efter behov då nya datorer eller nätverksskrivare och dylikt ska inlemmas i domänen. Datornamnen i vänstra kolumnen kan som synes anges på två olika sätt. Om datornamnet anges utan avslutande punkt läggs domännamnet automatiskt till innan uppslagning sker. På så sätt kommer de två första datornamnen att expanderas till `pc101.bsd.lab` och `pc102.bsd.lab`. Datornamnet på den tredje raden avslutas med en punkt och betraktas därmed som redan expanderat och klart innan namnuppslagning sker. Det domännamn som läggs till är det som står först i SOA-fältet ovan.

/etc/namedb/bsd.lab.rev

För bakåtuppslagning används filen `bsd.lab.rev` som också måste inledas med *options* och *SOA-record* på samma sätt som ovan:

```
$TTL      3600
bsd.lab.  IN    SOA  server.bsd.lab.  root.bsd.lab.  (
                       20010808         ; Serial
                       3600             ; Refresh
                       900              ; Retry
                       3600000          ; Expire
                       600 )            ; Minimum
```

Efter denna anges vilken dator som är namnserver för denna fil:

```
NS  server.bsd.lab.
```

och sedan anges kopplingarna för bakåtuppslagning:

```
101  PTR  pc101.bsd.lab.
102  PTR  pc102.bsd.lab.
103  PTR  pc103.bsd.lab.
```

Observera här den avslutande punkten i högerledet. Denna medför som vanligt att namnet är angivet komplett och färdigt.

`/etc/namedb/localhost.rev`

För att kunna slå upp den speciella adressen 127.0.0.1 behövs ytterligare en fil, `/etc/namedb/localhost.rev`. De utförs som man kan förvänta sig med de båda andra zon-filerna i minnet:

```
$TTL      3600
bsd.lab.  IN    SOA  server.bsd.lab      root.bsd.lab. (
                       20010808           ; Serial
                       3600               ; Refresh
                       900                ; Retry
                       3600000            ; Expire
                       3600 )             ; Minimum
          IN    NS   server.bsd.lab.
1         IN    PTR  localhost.bsd.lab.
```

`/etc/namedb/named.conf`

`named.conf` innehåller övergripande information om namnservern. I denna fil anges vilka filer som ska användas och det finns en hel del andra konfigurationer man kan göra i den. Som vanligt visas här en så enkel variant som möjligt:

```
options  {
         directory                    ■/etc/namebd■;
//       forwarders                   {
//                                     95.12.3.1;
//       };
};

zone     ■.■ {
         type                         hint;
         file                         ■named.root■;
};

zone     ■0.0.127.IN-ADDR.ARPA■ {
         type                         master;
         file                         ■localhost.rev■;
};

zone     ■bsd.lab■ {
         type                         master;
         file                         ■bsd.lab■;
};

zone     ■0.168.192.IN-ADDR.ARPA■ {
         type                         master;
         file                         ■bsd.lab.rev■;
};
```

Det första `options`-fältet anger var namnserverns konfigurationsfiler är belägna. Om denna namnserver också ska kunna hantera yttre adresser kan man här även ange vilken namnserver som ska användas för dess yttre adresser. De tre rader som gör detta är bortkommenterade med `//` i filen ovan. Ändra IP-adressen till den aktuella namnserverns.

Därefter kommer ett antal s.k. zon-deklarationer som dels anger att denna namnserver är auktoritativ (`type master`) för de data den innehåller och dels pekar ut namnet på respektive zon-fil.

Samtliga hittills nämnda filer kan innehålla mycket mer information[6] men det som behandlats är tillräckligt för ett lokalt nät.

Namnservern startas manuellt med kommandot `named(8)` beläget i `/usr/sbin/`. Efter ändringar i någon av de tre filerna ovan måste namnservern läsa in informationen igen. För detta finns kommandot `named.reload(8)`. Kommandor `named.restart(8)` kan användas för att starta om hela namnuppslagningstjänsten. Med lämpliga ändringar i `/etc/rc.conf` kan namnservern fås att startas vid datorns start.

[6] Den definitiva referensen när det gäller namnserverkonfiguration är DNS and BIND av Paul Albitz och Cricket Liu. Boken är mycket omfattande (>450 sidor) men beskriver också allt man någonsin kan behöva veta om namnservrar och deras konfiguration.

6.3.1 /etc/resolv.conf

En avslutande namnserver-relaterad konfiguration återfinns i filen `/etc/resolv.conf`. I denna fil anges vilken domän datorn tillhör och vilken namnserver som ska konsulteras för namnuppslagning. Upp till tre namnservrar kan anges för säkerhets skull, om någon av de första skulle vara nere. Till skillnad från konfigirationsfilerna till `named(8)` är `/etc/resolv.conf` föredömligt kort:

```
domain bsd.lab

nameserver 192.168.0.254
```

Det domännamn som anges i `/etc/resolv.conf` läggs till alla hostname som inte innehåller egen domännamn, det utgör ett s.k. *domänsuffix*.

Kommandot `nslookup pc1` expanderas till `nslookup pc1.bsd.lab` innan det utförs. På liknande sätt expanderas `pc1.test` till `pc1.test.bsd.lab` vilket uppenbarligen är fel. Men `pc1.test.lab` är redan ett komplett namn (FQDN) och kommer inte att expanderas innan namnuppslagningen sker.

I själva verket behöver domännamnet inte anges i `/etc/resolv.conf`. Om det inte är konfigurerat där används det domännamn som användes då datorns hostname definierades. Om datorns hostname konfigureras till `pc2.bsd.lab` sätts domänsuffixet till `bsd.lab`, men om hostname sätts till enbart `pc2` måste `/etc/resolv.conf` innehålla resten av den information som krävs för att lägga till domänsuffixet.

6.3.2 Kommandon

Med `named(8)` konfigurerad och funktionerlig kan det vara tid att se lite på hur namnuppslagning egentligen går till. En namnuppslagning inom domänen `test.lab` kommer alltid att svaras av ”vår” namnserver. En namnuppslagning på ett annat FQDN, exempelvis `ftp.freebsd.org` kommer att skickas vidare enligt figuren nedan tills svar erhållits eller sökningen misslyckades.

BILD på hur namnuppslagningen funkar. Se till att domänindelningen av namnspacet framgår.

Som vanligt finns det flera olika kommandon för att ananlysera och felsöka namnservrar. Det vanligaste är nslookup. nslookup har både en interaktiv och en kommandoradsmod. Om `nslookup` körs utan argument startas kommandot i interaktiv mod, men vanligast är att man anger ett IP-nummer eller hostname som argument:

```
> nslookup 192.168.0.101
Server: server.bsd.lab
Address: 192.168.0.251
Name: pc101.bsd.lab
Address: 192.168.0.101
```

Man ser att kommandot inte bara svarar på frågan utan också avslöjar vem som visste svaret.

6.3.3 Uppslagning

6.4 DHCP

Varje dator i vårt subnät måste tilldelas en unik IP-adress. Ett paket kan inte sändas till två adressater med olika IP-adress.[7]Det är möjligt att sätta IP-adressen en gång för alla vid installationen på varje dator men det visar sig vara onödigt om man använder DHCP. DHCP är ett protokoll för automatisk tilldelning av IP-adresser. Om klientens IP-adress sätts till strängen "`DHCP` vid installationen kommer den att försöka erhålla sin IP-adress, nätmask, default gateway och eventuellt också hostname från en DHCP-server. DHCP är ingen tung tjänst och kan med lätthet även den husera på det interna nätets server.

DHCP tillhandahåller IP-adresser inom en domän. DHCP-servern förfogar då över en mängd IP-adresser, till exempel `192.168.0.1` till `192.168.0.254`, och ser själv till att nya anslutande datorer får en unik IP-adress. Detta kan vara mycket praktiskt om man inom nätet ständigt ansluter till exempel bärbara datorer. Vill man ha mer ordning i sin datorpark kan också DHCP konfigureras så att den knyter ett och endast ett IP-nummer till en viss Ethernetadress. Detta förfarande hindrar utomstående från att "kidnappa" ett IP-nummer — om den utomstående inte lyckas byta Ethernetadress på sin dator förstås.

Liksom med DNS ovan går det att framgångsrikt administrera ett nätverk utan DHCP. I vissa fall kan det dock vara mycket praktiskt. Låt din egen situation avgöra. Tänk på att möjligheten att "kidnappa" ett IP-nummer kan vara oacceptabel i vissa miljöer. Det är definitivt inte fel att inte använda DHCP. Men det kan underlätta mycket om det implementeras på rätt sätt i rätt miljö.

I ett datorlab med 16 numrerade arbetsplatser kan det underlätta om dator på plats 1 heter `pc1`, på plats 2 heter `pc2` osv. Om ett fel uppträder på `pc12` är det då lätt att hitta datorn. Använde man fri tilldelning av IP-adresser kan vilken som helst av de 16 för tillfället heta `pc12` - och det är inte säkert att datorn heter samma sak vid varje uppstart. Ett fel på `pc12` igår kanske ska repareras på `pc4` idag!

DHCP-server ingår inte i FreeBSD utan måste läggas till via ports[8]. För isc-dhcpd ser konfigurationsfilen `/usr/local/etc/dhcpd.conf` ut som (med kommentarer inspängda):

```
option domain-name "bsd.lab";
```

[7]Såvida man inte använder det speciella nätet `224.0.0.0`, som tillåter s. k. *broadcasting* till flera mottagare samtidigt, men det är överkurs just nu.

[8]`> cd /usr/ports/net/isc-dhcpdXXX; make install clean`

```
option domain-name-servers server.bsd.lab;
ddns-update-style ad-hoc;
default-lease-time 600;
max-lease-time 7200;
```

Här definieras servernamn och det lokala domännamnet. De andra inställningarna behöver inte ändras. Läs i manualsidorna för `dhcp(8)` om dessa — och fler.

```
authoritative;
```

Det är den här DHCP-servern som har ansvar för alla adresser inom domänen.

```
log-facility local7;
```

Med ändringar i `/etc/syslogd.conf` kan alla eventuella meddelanden från `dhcpd(8)` fås att hamna i en loggfil[9].

```
subnet 192.168.0.0 netmask 255.255.255.0
{
range 192.168.0.180 192.168.0.185;
option domain-name-servers server.bsd.lab;
option domain-name "bsd.lab";
option nis-domain "bsd.lab";
option broadcast-address 192.168.0.255;
default-lease-time 600;
default-lease-time 600;
max-lease-time 7200;
}
```

Avsnittet ovan deklarerar fri tilldelning av IP-adresserna `192.168.0.180` till `192.168.0.185`. Om konfigurationsfilen slutar här kommer servern att tilldela dessa sex IP-adresser till de datorer som efterfrågar DHCP. En sjunde dator kommer inte att få någon adress om inte någon annan kopplar ur sig.

```
host pc101.bsd.lab
{
hardware ethernet 0:1:2:1e:46:a1;
fixed-address pc101.bsd.lab;
option host-name "pc101.bsd.lab";
}
```

[9] Lägg till dessa rader i `/etc/syslogd.conf` och skicka `-HUP` till `syslogd(8)`så att den läser in den nya konfiguratione:

```
!local7
*.* <TAB>/var/log/dhcpd.log
```

samt

```
# kill -HUP ‘cat /var/run/syslog.pid ‘
```

Resten av konfigurationsfilen kan bestå av poster liknande den ovan. I detta avsnitt deklareras att den dator som ansluter sig med Ethernetadressen `0:1:2:1e:46:a1` alltid ska tilldelas namnet `pc101.bsd.lab`.

Se till slut till att de fasta och fria IP-numren i DHCP-poolen inte överlappar varann. `dhcpd(8)` kan inte startas om med signalen `-HUP` som är brukligt utan måste verkligen stängas av först.

7 Daemoner

En daemon är en typ av program som kännetecknas av att det körs i bakgrunden. Det finns flera olika sorters daemoner. Vissa daemoner körs alltid och är nödvändiga för att operativsystemet överhuvudtaget ska kunna starta, andra körs bara när de behövs. Gemensamt för dem alla är dock att de körs fråkopplade från någon terminal och stdin respektive stdout är i allmänhet oåtkomliga.

Man ska inte förväxla daemon med demon (observera stavningen). En *daemon* är i den äldsta grekiska mytologin en allmän beteckning på andeväsen.

7.1 Portar

Innan vi går in på daemonerna i BSD måste vi först behandla lite grundläggande nätverkskommunikation.

Det räcker inte bara att känna till IP-adressen för en dator för att kunna etablera kontakt med den. Man måste även veta exakt vilken *port* man ska kontakta. Man kan dra likhet med att det inte räckar att känna till gatunamnet om man ska besöka någon, husnumret är också nödvändigt.

Om man ”knackar på” på en port med ett visst portnummer kan antingen någon svara eller inte. Om det svarar ligger ett program, ofta just en daemon, och lyssnar på just ”sin” port och är beredd att starta kommunikation. De flesta portar har dock ingen daemon knuten till sig. I ett TCP/IP-nätverk finns 65535 ”husnummer” att välja mellan på varje dator. Somliga portnummer är specificerade från början, andra aktiveras efter hand. I korthet gäller att portnummer mellan 1 och 1024 är speciella, sk priviligierade, och kan bara användas av root-användaren. Alla vanliga standarddeamoner har portnummer mindre än 1024. Övriga 1025-65535 kan användas även av vanliga användare. Under årens lopp har flera portnummer standardiserats och alla dessa well known services har sina portnummer inskrivna i filen /etc/services. Men även om en port är specificerad i /etc/services behöver den inte vara utnyttjad på en viss dator. Som root kan du slå av alla daemoner.

BILD på IP-koppel och portnummer

Att ha en port öppen (dvs att låta ett program lyssna på en port) är en potentiell säkerhetsrisk. Om daemonen har en bugg, en oavsiktlig felaktighet, kan denna utnyttjas för att i vissa fall ta sig in i datorn utifrån via nätet. En

illvillig inkräktare kan ställa till mycket oreda. Det är förnuftigt att inte ha fler portar öppna än man behöver, speciellt om datorn är uppkopplad mot Internet. Med särskilda program kan potentiella inkräktare söka av din dator (skanna) och ta reda på vilket operativsystem du använder. Med denna kunskap kan dom sedan, i vissa fall, utnyttja kända säkerhetshål och bereda sig åtkomst av datorn. Genom sin konstruktion är FreeBSD besparad från många säkerhetshål och om det upptäcks några tar det bara några dagar innan läckan kan tätas. Se vidare kapitlet om omkompilering av systemet.

Med en brandvägg kan man isolera det inre nätet från det yttre. Det betyder att man spärrar möjligheten för yttre trafik till vissa yttre portar medan man tillåter samma portar att utnyttjas på det inre nätet. Man kan tillexempel göra, säg, port 23 tillgänglig utifrån och spärra av allt annat. Konfigurationsmöjligheterna är enorma och det är fullt möjligt att spärra trafik som kommer utifrån till en viss port, medan samma port kan utnyttjas för trafik från det interna nätet utåt. Att bygga brandväggar är relativt komplicerat och kräver noggrannhet och ingående kunskap om nätverkskommunikation. Den intresserade läsaren rekommenderas Building Internet Firewalls, Zwicky et.al.

Det kan vara sunt att undersöka sin egen sårbarhet dvs. analysera vilka portar som faktiskt är öppna på de egna maskinerna. Det är lätt att glömma någon daemon påslagen här och var, och det enklaste sättet att utröna vilka portar som är aktiva är att själv skanna av sina datorer. Det finns flera program för detta och ett av de enklast är nmap. Med nmap kan man skanna av vissa eller alla portar på en viss dator, men nmap kan också gissa vilket operativsystem datorn använder.

Nmap ger också en del kommentarer angående säkerheten på den skannade datorn. Tro inte på de meddelandena. Det verkar som att om bara datorn är identifierad som linux av något slag ger nmap kopiösa överord om säkerheten, medan om systemet är av MS-Windows-typ, säkerheten alltid döms ut som ett skämt. Detta inte sant i någondera fallen och om du använder nmap så glöm nmaps åsikter om säkerheten. Däremot anmäler den de portar som är öppna helt riktigt.

Innan vi går vidare mot några daemoner, som ju kapitlet egentligen handlar om, måste vi också nämna programmet portsentry. Det är ett program som ligger som en daemon och lyssnar på många kända portar, kända för att dom ofta används i diverse cracknings- och fjärrstyrningsprogram. Om någon yttre dator försöker etablera kontakt med en sådan port registreras detta i en logg-fil och porten stängs av för att förhindra ytterligare hemsökningar.

7.2 Standarddaemoner

7.2.1 r-tjänster (the r-services)

Som enklaste exempel på daemoner kan man ta de s. k. r-tjänsterna (*the r-services*). De heter så eftersom deras namn alltid inleds med ett ”r” (som i *remote*, på distans). De har samtliga alltid följt med sedan unix först försågs med nätverksfunktioner och är ett antal grundläggande funktioner för nätverk. Det är inte säkert att du vill använda dem, men de tjänar som enkla exempel på daemoner och på hur några nätverkstjänster fungerar.

Observera att dessa r-tjänster kom till i en värld fri från nutidens portskanningar och crackers. Bland annat överförs användarens namn och lösen i klartext på nätverket[1]! Om du litar på dina användare och inte är direkt ansluten till Internet kan du dock sannolikt använda dem utan problem. I annat fall rekommenderas STARKT att tjänsterna förblir avstängda och `ssh(1)`/`scp(1)`, som beskrivs nedan, används om möjligt.

7.2.2 `rlogin(1)` och `rsh(1)`

`rlogin(1)` och `rsh(1)` är närbesläktade, och båda tillåter att man kör program på en annan maskin över nätverket. Den enda skillanden är att `rlogin(1)` öppnar ett terminalfönster medan `rsh(1)` inte gör det. Terminalfönstret ser likadant ut som om man satt vid den andra datorn och man har samma möjligheter att köra program och så vidare. `rsh(1)` betyder *remote shell*, och är ett sätt att köra ett program eller kommando på den andra datorn utan att behöva öppna ett fönster på den.

Både `rlogin(1)` och `rsh(1)` är mycket praktiska funktioner för en systemadministratör. Man kan tänka sig situationen att administratören behöver starta ett program eller köra ett kommando på en klientdator. Med dessa kommandon kan han göra det utan att behöva bege sig datorn ifråga, hela proceduren kan ske från vilken dator som helst. Om nätet är anslutet till Internet kan hela nätet övervakas och fjärrstyras utifrån, vilket kan vara till stor hjälp om systemadministratören är borta av någon anledning. Hela nätet kan underhållas från andra sidan jordklotet om det skulle behövas!

`rsh(1)` och `rlogin(1)` fungerar i princip på samma sätt, här beskrivs av utrymmesskäl bara `rsh(1)`: antag att en användare på `dator1`, den anropande datorn, vill utföra ett kommando på `dator2`, den anropade datorn. För att `dator2` ska kunna känna av `rsh(1)`-anropet krävs att `rsh(1)`:s daemon, `rshd(8)`, är startad (för rlogin heter daemonen `rlogind(8)`). Enligt `/etc/services` är port 222 tilldelad `rshd(8)`.

Om man, från `dator1`, utför kommandot

```
> rsh -l mj dator2 ls
```

[1] Jodå, du kan själv kontrollera detta med programmet XXX i ports.

loggas man först in som användaren `mj` på `dator2` och där utförs sedan kommandot `ls(1)`. Resultatet av kommandot presenteras på den egna skärmen och man loggas sedan direkt ut från `dator2`.

För att detta ska fungera krävs

- att användaren, här `mj`, existerar på `dator2`, så att användaren kan logga in lokalt på `dator2`,
- att det i användarens hemkatalog (`/home/mj` i exemplet) på `dator2` finns en fil med namnet `.rhosts` som innehåller raden `dator1 mj`
- att `.rhosts` ägs av användaren och bara är skrivbar för användaren och inte läsbar för någon annan (-rw——-)
- att rshd körs på `dator2`.

7.2.3 `rcp(1)`

Med `rcp(1)` kan man kopiera en fil mellan `dator1` och `dator2`. Syntaxen är som man förvänta sig och samma autenticieringskrav som för `rsh(1)` gäller:

```
> rcp -l mj dator2 testfil
```

sänder `testfil` till användaren `mj`:s hemkatalog på `dator2`.

7.2.4 `ruptime(8)`

För systemadministratören är det viktigt att lätt kunna se om alla datorer i nätverket fungerar som de ska. `ruptime(8)` är i det fallet en stor hjälp. Precis som kommandot `uptime(1)` visar den lokala datorns belastning och tiden sedan senaste start, visar `ruptime(1)` detta för alla datorer i nätverket:

```
> ruptime
pc101      up   12+21:18,   5 users,  load 4.03, 2.56, 1.54
pc102      up   39+02:46,  12 users,  load 2.01, 1.58, 2.12
pc103      up   56+01:55,   8 users,  load 2.27, 2.71, 2.12
server     up  188+23:34,   9 users,  load 7.93, 8.46, 4.58
```

Det är lätt att se om någon dator är avstängd eller har en ovanligt hög belastning. Tiden sedan senaste omstart anges i dygn och timmar:minuter.

Motsvarande daemon, som måste vara startad på samtliga maskiner, heter `rwhod(8)`. Den startas lämpligen från `/etc/rc.conf`.

7.2.5 rwho(1)

rwho(1) och dess daemon rwhod(8) ger användaren möjlighet att se vilka som är inloggade i nätverket:

```
>rwho
mj      pc101:ttyp0   11 Okt 14:01
andni   server:ttyp2  11 Okt 12:55
```

I vänsterkolumnen anges användarnamnet, sedan följer datornamn och inloggande terminal, ttyp*X* är inloggning i terminalfönster, ttyv*X* inloggning i virtuell terminal, exempelvis via rlogin(1) eller fönster i X. Här anges också tidpunkten när personen ifråga loggade in.

7.2.6 telnet(1)/telnetd(8)

En klassiker! Den hittills mest använda metoden att logga in över nätet. Man får samma möjligheter som om man vore lokalt inloggad på den andra datorn. Användaridentiteten avgörs direkt med den anropade datorns lösenordsfil och ingen .rhosts behövs. Motsvarande daemon heter telnetd(8) och tar rätt mycket primärminne av systemet. Av den anledningen brukar telnetd(8) bara startas när den ska användas och sedan stängas av. Telnet skickar lösenord i klartext över nätet och bör numera användas med stor försiktighet och helst ersättas helt med ssh(1).

7.2.7 syslogd(8)

syslogd(8) är den daemon som skriver till loggfilerna i /var/log. Den är konfigurerbar via /etc/syslogd.conf om det skulle behövas.

syslogd(8) kan även logga meddelanden från andra datorer i nätverket. Detta är ofta praktiskt eftersom man då inte behöver gå runt till, eller logga in på, klientmaskinerna och granska deras lokala loggfiler dator för dator. För att inte få /var/log fylld med loggfiler använder systemet kommandot newsyslog(8) med konfigureringsfilen /etc/newsyslog.conf för att då och då komprimera loggfilerna och efter ett tag kasta bort dom. Man kan till exempel välja att spara högst 10 loggfiler. newsyslog(8) lägger då till den elfte genom att ta bort den äldsta för att få plats, s. k. *loggfilsrotation*.

7.2.8 ftp(1)/ftpd(8)

ftp(1), File Transfer Protocol, är bland de äldsta sätten att överföra filer mellan datorer över nätet. I det enklaste fallet använs ftp(1) för att logga in på sitt hemkonto från någon annan dator. Man kan då flytta filer till eller från hemkatalogen och även skapa och radera bibliotek om man så vill.

BILD hel ftp-session för att hämta package

passiv/aktiv

(inklusive ändringar i /etc/inetd.conf ftpd -S -l -l)

7.2.9 ssh/scp/sshd

keygen, inloggning, automatisk inloggning... långt kapitel det där.

7.2.10 `inetd(8)`

En dator med många nätverkstjänster enligt ovan igång kräver mycket primärminne om alla daemoner ska köras samtidigt. Detta är uppenbart slöseri om de inte nyttjas flitigt och man använder nästan alltid `inetd(8)`, en ”superdaemon” som lyssnar på vissa portar och, om anrop till någon av dessa sker, automatiskt startar rätt daemon för att ta hand om den vidare kommunikationen. De program som `inetd(8)` ska kunna starta anges i dess konfigurationsfil `/etc/inetd.conf(8)`.

För att `telnet` ska fungera krävs antingen att `telnetd(8)` är igång på målmaskinen alltid eller att `inetd(8)` är tillsagd att starta den så fort någon försöker prata med `telnet`-porten (port 23, enligt filen `/etc/services`). För telnettjänsten måste `/etc/inetd.conf` då innehålla raden:

```
telnet stream tcp nowait root /usr/libexec/telnetd telnetd
```

Filen `/etc/inetd.conf` innehåller exempel på många tjänster. De som inte är aktiva är bortkommenterade och föregås av ett `#`-tecken. Det är dock inte tillräckligt att ändra i `/etc/inetd.conf` enbart, för att starta eller stoppa nya tjänster. Programmet `inetd(8)` måste också göras uppmärksam på att konfigurationen har ändrats och ska läsas in på nytt. Det mest korrekta sättet att signalera detta till `inetd(8)` är

```
# kill -1 ‘cat /var/run/inetd.pid‘
```

där `/var/run/inetd.pid` innehåller processnumret för den `inetd(8)` som just körs. Men man kan lika gärna göra:

```
# ps    ax |   grep     inetd
89      ?? Is  0:00.04  inetd -wW
1129    p0 S+  0:00.01  grep inetd
# kill  -1 89
```

om man skulle vilja det.

7.3 Starta/stoppa daemoner

En del daemoner med vissa grundläggande funktioner måste vara igång för att operativsystemet överhuvudtaget ska kunna fungera och startas i och med datorn startar operativsystemet. Andra kan man själv få datorn att

starta vid systemstart genom att lägga ett startprogram i `/usr/local/etc/rc.d/`. För att starta programmet `foo`[2] vid start av operativsystemet kan man lägga följande skript i denna katalog, se `rc(5)`:

```
#!/bin/sh
#
# initialization/shutdown script for foobar package
case "$1" in
start)
/usr/local/sbin/foo -d && echo -n ' foo'
;;
*)
echo "unknown option: $1 - should be 'start'" >&2
;;
esac
```

Kalla skriptet för `foo.sh`. För att starta programmet `foo` skriver man sedan :

```
# sh /usr/local/etc/rc.d/foo.sh start
```

eller om det är exekverbart (se `chmod(1)` och kapitel XXX), bara

```
# /usr/local/etc/rc.d/foo.sh start
```

Vissa skript har även en avstängningsfunktion och anropas som ovan men med argumentet stop istället för start. Ytterligare andra skript tillåter flera andra argument, som restart eller reload etc. Vid systemstart eller avstängning används dock endast start repektive stop.

En del daemoner kan fås att läsa om sina konfigurationsfiler om man med kommadot `kill(1)` skickar dom signalen[3] `-HUP`. Använd isåfall `ps`(1)-kommandot först för att idfentifiera processnumret, PID.

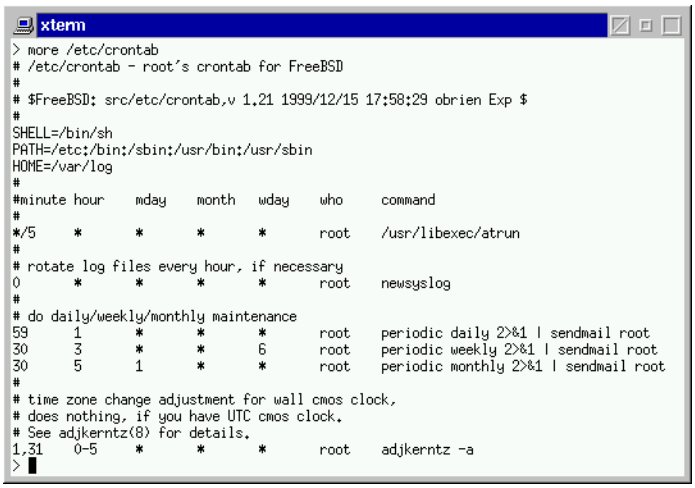
7.3.1 cron

Ofta vill man utföra samma uppgift flera gånger och kanske vid en bestämd tid. Det kan handla om sammanställningar av loggfiler eller andra typer av uppgifter som tar mycket av datorns resurser i anspråk och som man kanske hellre vill köra utanför kontorstid. För att åstadkomma detta finns en daemon `cron` (efter guden chronos i den grekiska mytologin). `cron` styrs av filen `/etc/crontab` som den läser varje minut, på minuten. Om `/etc/crontab`

[2]För den språkintresserade: *foo* och dess kompanjon *bar* används ofta som namn på olika saker när man beskriver Unix. Orden kommer ur uttrycket fubar vilket sägs vara en förkortning av Fucked Up Beyond All Recognition, som jag inte ens tänker försöka översätta.

[3]Istället för `kill -HUP` kan man använda det enklare skrivsättet `kill -1`. Var dock försiktig och kontrollera kommandot innan du slår på returntangenten. Förvåningen var stor, i alla fall hos mig, när jag trodde jag skrev `kill -1 1` men råkade ha skrivit `kill 1 -1`. Prova gärna men gör det inte när något viktigt program är igång...

innehåller ett matchande klockslag kan ett kommando köras. Formatet beskrivs i `crontab(5)` men är såpass enkelt att texten i filen är tillräcklig:



De olika raderna som kör kommandot `periodic(1)` innebär till exempel att `periodic daily` körs klockan 01.30 varje dag, `periodic weekly` klockan 03.30 veckodag 6 (dvs lördag) och `periodic monthly` 05.30 den 1:a varje månad.

Om systemadministratören tillåtit det, har även användarna möjlighet att använda `cron`. Om ett användarnamn förekommer i filen `/var/cron/allow` kan denna användare skapa sin egen `crontab`. Och genom att lägga till användarnamnet i `/var/cron/deny` kan systemadministratören utestänga användare från denna möjlighet. Användarna använder kommandot `crontab -e` för att editera sin egen `cron`-tabell. Alla användaren `cron`-tabeller ligger samlade i `/var/cron/`.

Det glöms ofta bort att även `root` har en personlig crontab, som inte är samma som `/etc/crontab`, även denna editeras med `crontab -e`.

7.4 Icke standarddaemoner

Det finns många daemoner som inte följer med installationen av FreeBSD. Här presenteras två vanligare, apache och natd. Den förstnämnda är den mest populära webservern på Internet. Över 60% av alla webservrar som körs använder Apache. Naturligtvis kan även FreeBSD använda den och den är naturligtvis helt gratis.

Natd är en praktisk daemon som utför s. k. *Name Address Translation*. Praktiskt betyder det att vi kan ansluta vårt `192.168`-nät till Internet utan att dessa IP-nummer syns utåt. Det krävs dock att vi har tillgång till en yttre IP-adress från en internetleverantör.

7.4.1 apache

Apache[4] är som de flesta program till FreeBSD mycket enkelt att installera och som vanligt kan man kompilera hela källkoden själv eller välja att installera det färdigkompilerade paketet. Väljer du det senare använder du `/stand/sysinstall` som förut. Här väljer jag att installera från ports och kompilera upp det på nytt.

```
# cd /usr/ports/www/apacheXXX
# make install
```

Och det är alltsammans! Nu har du omvandlat din dator till en webserver med bara några få tangenttryckningar! Om du inte installerat någon webläsare kan du passa på att göra det nu, så att du kan surfa till din dator. I `ports` kan du bland andra välja bland mozilla[5] eller Netscape[6]. Netscape är väl känd sedan gammalt, mozilla är en nyare webläsare. Båda är mycket kompetenta.

Om du nu startar din webläsare och anger din egen dators adress (eller, om du inte har något nätverkskort installerat och adress tilldelad, adressen `http://127.0.0.1`) ser du Apaches egen välkomstsida som tecken på att installationen lyckades. Antagligen vill du byta ut denna sida mot någon med ditt eget innehåll och det gör du i biblioteket `/usr/local/www/data`. Välkomstsidan du just såg heter `index.html` och du kan använda den här katalogen som du vill för dina websidor.

Konfigurationsfilerna befinner sig i katalogen `/usr/local/etc/apache` och de flesta inställningarna görs i filen `httpd.conf`. Det vanliga är att inte röra filen `httpd.conf.default` utan kopiera denna till `httpd.conf` och göra alla ändringar i den senare. Då kan du alltid byta tillbaks om du skulle behöva.

Ytterligare information finns i katalogen `/usr/local/doc/apache`.

7.4.2 natd(8)

Det vore ju synd om vårt nätverk inte skulle kunna ansluta sig till något annat nät. Man kan hyra en IP-nummerserie med en IP-adress per dator i nätet om man behöver, men det finns klara fördelar att bara hyra *en* IP-adress och låta hela sitt nät hållas bakom detta enda nummer. Bland annat är det billigare att bara behöva en adress, men det är också säkrare — datorerna syns inte från det yttre nätet och man kan alltså tillåta sig en lägre säkerhetsnivå på det inre. För yttre intrång räcker det att skydda den dator som har direktkontakt med det yttre nätet, *routern*, där `natd(8)` körs.

[4] Uttalas som ”A Patchy” webserver, en hoplappad webserver, vilket var det tidiga arbetsnamnet.

[5] `www/mozilla`

[6] www/Netscape-Communicator-4.72

Yttre datorer kan inte nå de inre datorerna direkt, all kommunikation måste gå genom routern och det är de inre datorerna som måste initiera kommunikationen. Man kan alltså surfa obehindrat från det inre nätet ut mot det yttre men omvändningen är inte möjlig.

BILD

För att använda `natd(8)` måste FreeBSD:s kärna vara förberedd med brandväggsfunktionalitet. FreeBSD version 4.4 har redan detta stöd inkompilerat men äldre variant kanske inte har det och då kan du behöva göra denna kompilering själv. Omkompilering av kärnan ingår inte i denna grundläggande text men i slutet av boken finns i varje fall några korta instruktioner om hur du lägger till brandväggsstöd.

Vi förutsätter att du har en kärna med brandväggsfunktionalitet inbyggd. Då är `natd(8)` mycket enkel att sätta igång, det kräver bara enkla ändringar i filen i `/etc/rc.conf`:

```
natd_program="/sbin/natd"   # path to natd, if you want a different one.
natd_enable="YES"           # Enable natd (if firewall_enable == YES).
natd_interface="fxp0"       # Public interface or IPaddress to use.
natd_flags=""               # Additional flags for natd.
```

Ändringarna är markerade i kursiv stil. Byt `NO` mot `YES` och ange namnet på det nätverkskort som är anslutet till det yttre nätet. Byt `fxp0` till det som gäller för din router.

Dessa ändringar styr den relevanta sektionen under i `/etc/rc.firewall`:

```
############ # These rules are required for using natd. All packets are
passed to
# natd before they encounter your remaining rules. The firewall rules
# will then be run again on each packet after translation by natd,
# minus any divert rules (see natd(8)).
if [ "X${natd_enable}" = X"YES" -a "X${natd_interface}" != X"" ]; then
$fwcmd add divert natd log all from any to any via ${natd_interface}
fi
```

För att få ändringarna att slå igenom är det i det här fallet enklast att starta om datorn. Man behöver dock inte göra det (om brandväggsfunktionen redan var inbyggt i kärnan) man *kan* manuellt starta `/sbin/natd` och lägga till brandväggsreglerna med kommandot `ipfw(8)` enligt `/etc/rc.firewall` ovan. För att vara säker på att datorn faktiskt är helt rätt konfigurerad är det trots detta bäst att boota om.

Oavsett vilken metod du väljer för att få ändringarna att slå igenom ska du nu kunna komma åt det yttre nätet från routern och om klienterna har denna som default gateway ska även dessa kunna nå det yttre nätet.

7.5 NFS

7.5.1 Introduktion

Via NFS, *Network File System*, kan en dator lämna ut, *exportera*, sitt filsystem till en eller flera andra datorer. Rätt använt ger detta väsentliga administrativa fördelar: centrala filarkiv kan exporteras så att lokala kopior av filer och kataloger undviks. Säkerhetskopiering underlättas om det som ska kopieras befinner sig på ett ställe centralt och inte är utspritt på flera datorer.

Till exempel kan klientdatorerna ha en standardinstallation av FreeBSD och via NFS montera de ytterligare filsystem som behövs. Monteringsmekanismen tillåter s. k. *övermontering*, en NFS-exporterad `/usr` kan monteras ovanpå och ersätta den lokala `/usr`.

Ett extremfall innebär s. k. *diskless boot*, där klientdatorn inte ens har en hårddisk utan via nätet laddar ett minimalt operativsystem direkt till minnet och sedan via NFS monterar till exempel både `/usr` och `/var` och eventuella andra filsystem.

7.5.2 Förutsättningar

Det finns en mängd krav som omgärdar NFS-montering och exportering. Det kan vara värt att noggrannt studera man-sidan för `mount(1)` och `exports(5)`! I huvudsak är dock innebörden denna:

- På den *exporterande* datorn måste programmen `mountd(8)` och `nfsd(8)` vara igång. Dessutom måste filen `/etc/exports` innehålla information om vad som skall exporteras och vilken/vilka datorer som får montera det exporterade. Om `/etc/exports` innehåller felaktig eller motstridig information kommer bara de rader som är korrekta att utföras. Eventuella felmeddelanden skrivs till `/var/log/messages` som vanligt. Efter att ändringar införts i `/etc/exports` måste signalen `-HUP` sändas till `mountd(8)` för att ändringarna ska slå igenom. Detta görs säkrast med `kill -1 ‘cat /var/run/mountd.pid‘`[7].
- På den *monterande* datorn räcker det att ska programmet `nfsiod(8)` körs.

Ändringarna ovan kan genomföras och permanentas via `/stand/sysinstall` eller genom att flytta över och editera relevanta rader ur filen `/etc/defaults/rc.conf` till `/etc/rc.conf`[8]. Man kan naturligtvis starta programmen manuellt men inställningarna försvinner då vid nästa bootning.

[7] Observera att de ‘ som används är grav accent som fås, på svenskt tangentbord, genom att hålla ner skift-tangenten och accent-tangenten bredvid ”?”.

[8] Man kan ändra direkt i `/etc/defaults/rc.conf` men det rekommenderas att inte göra det med tanke på eventuella framtida uppgraderingar då denna fil med stor sannolikhet byts ut. Då är det bra att alla avvikelser finns lagrade på annat ställe, i `/etc/rc.conf`.

Det är inte möjligt att exportera vad som helst. Man kan inte exportera både en katalog och en dess underkataloger bland annat. Däremot kan man exportera en katalog med samtliga underkataloger (tillägget `-alldirs`). Följande krav ställs på de exporterade filsystemen:

- Det exporterade får inte ingå i ett redan exporterat filsystem, på samma hårddisk. Man kan således inte exportera både `/usr` och `/usr/local` såvida inte endera ligger på en annan hårddisk.
- Man kan bara exportera lokala filsystem. Det är alltså inte möjligt att vidareexportera ett monterat filsystem.

Filen `/etc/exports` beskrivs i `exports(5)` och kan se ut ungefär så här:

```
/usr/X11R6 /usr/export  -maproot=root  -network 192.168.0 -mask 255.255.255.0
/home                   -maproot=root  -network 192.168.0 -mask 255.255.255.0
/disk2                  -maproot=root  pc101.bsd.lab pc7.bsd.lab
/cdrom                                 -network 192.168.0 -mask 255.255.255.0
```

Första raden delar ut `/usr/X11R6` och `/usr/export` till alla datorer i nätverket `192.168.0/24`. Andra raden gör samma med `/home`. Av reglerna ovan framgår att `/home` här måste vara ett separat filsystem för att exporten ska kunna genomföras, och inte `/usr/home` som är standardinställningen. Raden därunder delar ut `/disk2` till två namngivna datorer, dessa datorer behöver inte nödvändigtvis befinna sig på det lokala nätet, även om just de i exemplet gör det. Slutligen delas serverns `/cdrom` ut till alla klienter i nätet. Detta är ofta praktiskt ty dels behöver inte klienterna ha var sin CDROM-spelare och dels är det ett lätt sätt att komma åt FreeBSD:s installations-CD om man skulle behöva den.

Med tillägget `-maproot=root` åstadkommer man att en `root`-användare på en monterande dator erhåller `root`-behörighet även på det utdelade filsystemet. Om detta inte är önskvärt — det kanske är lite väl ”aggressivt” i vissa omgivningar — kan `-maproot` även användas till att begränsa `root`:s behörighet till någon annan användares, kanske `nobody`. Det är lämpligt om man är säker på att `root` inte ska kunna komma åt filsystemet. Andra användare än `root` berörs inte av `-maproot`-tillägget. Med tillägget `-mapall` kan alla användare (alltså även root) ges samma behörigheter.

Ett ofta förekommande fel, i en i övrigt rätt tänkt `/etc/exports`, är att man delar ut olika filsystem till samma dator på olika rader:

```
/usr 192.168.0.23 -netmask 255.255.255.0
/var 192.168.0.23 -netmask 255.255.255.0 < FEL!
```

Detta är fel! En rätt rad ser ut så här:

```
/usr /var 192.168.0.23 -netmask 255.255.255.0
```

Återigen: Läs man-sidan för `exports(5)` om just din `/etc/exports` trilskas!

7.6 NIS

NIS, *Network Information System*, är en metod att via nätverket distribuera information från en central server till ett antal klienter. Det innebär inte att själva filerna med informationen överförs på nätverket utan NIS är en mekanism för klienter, i samma *nisdomän*, att få tillgång till den centralt lagrade informationen. Till skillnad från NFS ovan saknar NIS ett filsystem och dessutom innebär NIS enbart läsrättigheter. NIS-domänen har inget med TCP/IP-domänen att göra, speciellt behöver inte NIS-domänen sammanfalla med TCP/IP-domänen.

NIS kallades förr YP, *Yellow Pages*, dvs Gula Sidorna, men då YP var ett registrerat varumärke tvingades man byta namn. YP finns dock fortfarande kvar som del av namn på vissa NIS-relaterade program som `yppasswd(1)`, `ypcat(1)` med flera.

NIS' funktion förklaras kanske bäst med ett exempel: En fil som typiskt distribueras via NIS är `/etc/passwd`. På så sätt kan en hel NIS-domän samsas om en enda centralt placerad `/etc/passwd`-fil. Denna fil skall då vara placerad på NIS-servern. Att placera gemensamma filer under NIS' kontroll underlättar administrationen av nätverket väsentligt. Till exempel behöver en ändring i `/etc/passwd` inte kopieras ut till samtliga klientdatorer utan en ändring i den centralt placerade `/etc/passwd` räcker[9].

NIS kan distribuera vilka (data)filer som helst, vi har nytta av att endast distribuera `/etc/passwd`, `/etc/master.passwd` och `/etc/groups`. Det är då lätt att centralisera användarhanteringen i hela nätverket. Om man vill distribuera fler filer än de här behandlade eller upprätta en större NIS-domän med slavservrar för större redundans och ytterligare möjligheter rekommenderas boken *NIS and NIS+* av Liu.

7.6.1 NIS-servern

För att upprätta en NIS-domän måste servern konfigureras. Det är en rätt enkel process då ett kommando gör det mesta av jobbet. Vi måste dock "rensa manegen" lite innan vi släpper loss kommandot. Här nedan antas att vi ska skapa en master-server som ska betjäna NIS-domänen `bsd`. Först måste vi placera oss i den katalog där NIS-domänen ska hålla hus:

```
# cd /var/yp
```

Vi vill att NIS ska distribuera lösenord åt oss och därför måste vi kopiera `/etc/master.passwd` hit:

```
# cp /etc/master.passwd .
```

Med detta klart gör kommandot `ypinit(8)` resten. Installationen kräver att vi svarar på några inledande frågor:

[9] Följt av att man inte glömmer att uppdatera NIS-databasen (`cd /var/yp; make`).

```
# ypinit -m bsd
:
Do you want this procedure to quit on non-fatal errors? [y/n:
n] y
```

Vi svarar `y` för ”YES”, och går vidare,

```
next host to add:
```

Eftersom dettta är den enda NIS-servern i domänen avbryter vi här genom tangentkombinationen <CTRL>-D, som tar oss vidare till den avslutande frågan,

```
The current list of NIS servers looks like this:
pc101
Is this correct? [y/n: y] y
```

Efter denna fråga bygger `ypinit(8)`-kommandot upp de databasfiler som innehåller den information NIS-domänen behöver.

```
:
Building /var/yp/bsd/ypservers...
:
:
NIS Map update completed.
pc101 has been setup as an YP master server without any er-
rors.
#
```

Som framgått av ovanstående används inte `/etc/master.passwd` direkt utan NIS använder den kopia vi kopierade till `/var/yp`. Detta är helt i sin ordning men får en konsekvens som måste påpekas. Om en lokal användare ändrar sitt lösenord med kommandot `passwd(1)`, och NIS-domännamn är definierat, genomförs ändringen bara i den lokala filen `/etc/master.passwd` om användaren är definierad där. Om användaren däremot enbart är definierad via NIS, dvs i `/var/yp/master.passwd` på NIS-servern sker ändringen enbart i NIS-databasen. Detta fungerar i allmähet som man kan förvänta sig så länge inte användaren försöker logga in på servern efter att ha ändrat sitt lösenord. Servern ingår inte som klient i sin egen domän och förväntar sig således det lösen som angivits i `/etc/master.passwd` och har ingen kunskap om att ett nytt lösen definierats i NIS. Inloggningen misslyckas alltså.

En lösning på detta problem är att inte kopiera `/etc/master.passwd` till `/var/yp`, som vi gjorde ovan, utan att modifiera `/var/yp/Makefile` så att den läser `/etc/master.passwd` direkt — då kommer alla ändringar av lösenord via NIS att direkt påverka `/etc/master.passwd`. Det visar sig att man inte behöver ändra i Makefile, det räcker att ange sökvägen till `/etc/master.passwd` när man kör `make(1)`,

```
# make MASTER_PASSWD=/etc/master.passwd
```

I och med detta är NIS-serverns konfiguration klar och NIS-domänen `bsd` definierad. Det serverprogram som läser konfigurationsfilerna heter `ypserv(8)` och startas i allmänhet ifrån `/etc/rc.conf` vid uppstart av datorn, men kan naturligtvis startas manuellt redan nu. Återstår bara att få klientdatorerna uppmärksamma på NIS-serverns existens.

7.6.2 NIS-klienterna

Klientdatorerna måste göras uppmärksamma på NIS-serverns existens och förberedas för att kunna samverka med NIS-servern. Dels måste vi tala om för den att den ska lysnna på en master-server i domänen `bsd` och dels måste de filer vi vill distribuera via NIS prepareras. De filer det handlar om för vår del är `/etc/passwd`, `/etc/master.passwd` och `/etc/group` som måste ”förlängas” med information via NIS.

Här anges först de manuella åtgärder man behöver göra för att definiera klienten. För att permanenta inställningarna visas sedan vilka rader i `/etc/rc.conf` som måste läggas till.

- Med kommandot `domainname(8)` anger man för klienten vilken NIS-domän den tillhör. Namnet *domainname* är något illa valt då det lätt kan blandas ihop med TCP/IP-domänen vilket är en helt annan domän.. Här anger vi alltså `domainname bsd`, och vi kan kontrollera vilken domän datorn tror den tillhör genom att i kommandot utesluta argumentet `bsd`.
- Programmet `ypbind(8)` knyter upp klienten mot servern och måste alltså köras för att kontakten skall kunna upprättas.

I `/etc/rc.conf` införs ändringarna ovan som man kan förvänta sig:

```
nisdomainname="NO" # Set to NIS domain if using NIS (or NO).
```

samt

```
nis_client_enable="NO" # We're an NIS client (or NO).
```

De filer som ska ersättas med information från NIS måste nu modifieras. Ett ’+’-tecken i slutet i respektive fil är en indikation på att filen fortsätter i NIS-databasen. Vi måste alltså modifiera de tre filerna med ett avslutande ’+’-tecken. Dessutom, visar det sig, måste filernas format upprätthållas. I våra filer innebär detta att antalet separatorer ’:’ måste vara lika många i den sista raden som i de övriga. Med den sista raden i respektive fil enligt nedan, kommer klienten att först läsa igenom filen och sedan efterfåga ytterligare information från NIS-servern. Man kan låta filerna bestå av enbart denna sista rad men det är praktiskt att åtminstone ha en `root`-användare kvar i `passwd`-filen

`/etc/passwd` och `/etc/master.passwd` avslutas med '`+:*:::::`' enligt[10]

```
mj:*:1000:1000:micke:/home/mj:/usr/local/bin/tcsh
+:*:::::
```

`/etc/group` avslutas som

```
nobody:*:65534:
+:::
```

Som test för att se om domänen är korrekt konfigurerad kan man nu logga in, som `root`, på en klientmaskin. I och med att `root`-kontot är definierat i den lokala `/etc/passwd` på respektive dator kan `root` logga in även om NIS-domänen inte skulle vara i funktion, på så sätt kan felsökning utföras. Med en fungerande NIS-domän kan man med kommandot `ypcat(1)` nu se det NIS-modifierade filinnehållet:

```
# ypcat /etc/groups
:
:
```

Under förutsättning att NIS fungerar enligt ovan kan man nu logga in som vilken annan användare som helst, givet att denne användare kan logga in på NIS-servern lokalt. Denne användares lösen, default shell och hemkatalog läses nu via NIS före användning på klienten:

XXX

7.6.3 Underhåll

Om någon uppgift, i de filer som distribueras via NIS, skulle ändras lokalt på servern[11] måste NIS-databasen skapas på nytt:

```
# cd /var/yp
# make
```

Om man lägger till en användare med `pw(8)` kan man med växeln `-Y` automatiskt uppdatera NIS-databasen.

7.6.4 Hemkatalogen

Vid inloggning får man än så länge felmeddelandet `Using / for home`. I `/etc/passwd`-filen står hemkatalogen angiven, ofta som `/home/`*`<användarnamn>`*. På klientdatorn är ännu inte någon sådan katalog skapad och systemet tillåter då bara att `/` används som hemkatalog.

Man kan lösa den uppkomna situationen på två sätt.

Man *kan* naturligtvi helt enkelt lägga till ett `/home/mj`-bibliotek på klienten

[10] Stjärnan '*' behövs för att förhindra att användaren '+' ska kunna logga in utan att ange lösen!

[11] Gäller alltså inte om filerna ändras via NIS, till exempel med `passwd(1)`-kommandot.

```
# mkdir /home/mj
# chown mj:mj /home/mj
```

och logga in på nytt. Eftersom det nu finns ett hembibliotek på den lokala klienten kommer inloggningen att ske utan felmeddelande.

Problemet är nu att användarens hembibliotek ligger lokalt på just den dator det skapades på. Det innebär att användaren måste befinna sig vid just denna dator för att vid ett senare tillfälle komma åt sina filer[12]. I vissa fall kan detta vara acceptabelt, men ur en administrativ synvinkel ställer det till en del problem. Hur ska administratören till exempel kunna ta backup på användarnas hemkonton om de är spridda på flera datorer?

Ett bättre sätt är att låta varje användare ha sin hemkatalog på en central server och montera denna servers `/home` på varje klient. Med lämpliga ändringar i `/etc/fstab` på klienten kan detta ske automatiskt vid klientens start, ungefär med en rad som denna[13]:

```
server:/home /home nfs rw 0 0
```

För att NFS-servern skall exportera `/home` måste exempelvis följande ändringar göras i serverns `/etc/fstab`:

```
# echo "/home -network 192.168.0" > /etc/exports fil
# kill -HUP ‘cat /var/run/mountd.pid‘
```

Man kan lätt kontrollera om ändringen blivit den önskade enligt,

```
# showmount -e
```

Med NIS konfigurerat enligt ovan och hemkatalogen ansluten via NFS är systemet mer lättanvänt för användarna och mer lättadministrerat för systemadinistratören, bland annat:

- Nya användare behöver bara läggas till på servern och de kommer med automatik att omedelbart kunna logga in vid vilken klientdator som helst.
- Användaren behöver inte längre logga in på samma maskin för att kunna komma åt sina filer.
- Backup av användarnas filer är enkelt, det räcker att göra backup på serverns `/home` så är allt klart.

7.7 amd(8)

Här skulle vi kunna vara nöjda, vi har skapat ett utmärkt stabilt, lättadministrerat och pålitligt nätverk, men vi kan öka graden av förfining ett

[12] Där användaren ursprungligen skapades har systemet också skickat med en rad filer med inställningar, `.cshrc`, `.login` och `.profile` bland andra.
[13] Se `fstab(5)` för ytterligare möjligheter.

sista snäpp: Strängt taget är det ju onödigt att alla användares hemkataloger ligger monterade på samtliga klienter hela tiden. Det är i och för sig inget fel i det, men kan medföra en stor belastning på servern när samtliga klienter startar om, till exempel vid ett strömavbrott. När strömmen kommer tillbaka efter strömavbrott och alla klienter bootar om kommer de att vid ungefär samma tidpunkt vilja återansluta sin `/home` från servern, det sker en s. k. *mount storm*. Om servern då inte hinner att hantera alla klienter kan någon klient bli utan `/home`.

Det borde vara tillräckligt att endast den användare som loggar in får sin hemkatalog monterad och det just när inloggningen sker! Programmet `amd(8)` gör precis detta.

`amd`, (*4.4 BSD AutoMounter Daemon*) är det program som automatiskt monterar ett filsystem som utdelats från en annan dator eller görs tillgänglig på annat sätt[14]. Monteringen sker precis vid det tillfälle resursen efterfrågas (till exempel då man önskar lista filerna i ett bibliotek) och resursen kommer att avmonteras då den inte använts på en viss tid, ofta 300 sekunder.

`amd(8)`:s dokumentation är omfattande och här kommer enbart beröras de delar som är av intresse för att kunna automounta NFS-exporterade filesystem. För en komplett dokumentation hänvisas till dokumentationen i *Am-utils - The 4.4BSD automounter* vilken ingår i `ports` under `net/am-utils`.

`amd(8)` styrs av minst två filer, `/etc/amd.conf` och `/etc/amdmaps/amd.*`, där `/etc/amd.conf` innehåller systermövergripande konfiguration:

I `/etc/amd.conf` kan man exempelvis ändra den ovan nämnda time-out:en, `time_out = 300`, men även andra konfigureringar. Man kan också ange om resurserna ska vara synliga för kommandot `ls(1)`.

Med resursen osynlig (`browsable_directory=no`):

```
> cd /home
> ls
>
```

Man kan alltså inte se namnen på de kataloger eller filer som ingår i dessa kataloger. Vet man däremot det exakta namnet kan man lätt komma åt innehållet:

```
> cd /home
> ls mj
:
>
```

[14] Med `amd(8)` kan även andra resurser som floppydiskar eller CDROM automatmonteras.

Med resursen synlig (`browsable_directory=full`):

```
> cd /home
> ls
mj toor
>
```

`/etc/amd.conf` avslutas med en eller flera s. k. *amd maps*. Dessa innehåller information om hur själva monteringen ska gå till, vilket filsystem som ska anslutas, om det ska monteras enbart för läsning mm. I vårt fall, vi ska ju enbart automounta `/home`, avslutas `/etc/amd.conf` med:

```
[ /home ]
map_name = amd.home
```

Inom hakparentes står monteringspunkten på klientdatorn och på raden under anges sökvägen till den fil som innehåller den närmare beskrivningen, `/etc/amdmaps/amd.home`, där `/etc/amdmaps` är underförstått eftersom det angivits i `/etc/amd.conf` (`search_path = /etc/amdmaps`):

```
/defaults opts:=rw,grpid,resvport,vers=2,proto=udp,nosuid,nodev
* sublink:=${key};type:=nfs;rhost:=server;rfs:=/home
```

Av intresse här är nu

- rhost, *remote host*, den exporterande datorn, dvs. servern med `/home` på sig
- rfs, *remote file system*, det externa filsystemets namn, som det anges på den exporterande datorn.

`amd(8)` startas vanligen direkt från `/etc/rc.conf` vis datorns start men programmet `ctl-amd(8)`[15] kan även användas interaktivt:

```
# ctl-amd start
#
:
# ctl-amd stop
#
:
# ctl-amd restart
#
:
```

I och med detta sista moment kan vi anse att nätverket, med server och klientdatorer, är färdigt för användning.

[15] `ctl-amd(8)` ingår i `/ports/net/am-utils`.

8 Backup

8.1 Inledning

Med konfigureringen klar återstår för administratören den riktigt stora utmaningen: att hålla systemet i drift. Detta innebär bland annat som tidigare nämnts att hålla sig a jour med händelserna på säkerhetsfronten, att lyssna på användarna och anpassa systemet och program efter deras krav. Men inte minst att förbereda för det värsta: En serverkrasch.

Vi känner vid det här laget igen ett lättadministrerat system på att åtskilliga funktioner är centraliserade, filserver, skrivarserver, DHCP med mera. Medan denna centralisering medför en avsevärt enklare administration utsätter den dock hela systemet för en central svaghet, servern. Om servern skulle råka ut för ett missöde i form av en diskkrasch, minnesfel, brand eller helt enkelt att systemadministratören skriver fel kommando, är det av högsta vikt att hårddiskens innehåll finns lagrat på annat medium och att denna kopia inte förvaras tillsammans med servern utan i annat rum eller kanske till och med i en annan byggnad[1].

Säkerhetskopiering av data, *backup*, är en mycket viktig och ofta underskattad eller helt bortglömd procedur. Därför avslutar vi med att kort beskriva hur man lättast utför en backup av en FreeBSD-server.

En tillförlitlig backup är en väsentlig del av ett fungerande datornätverk. Hårddiskar med SCSI-anslutning medför från början högre tillförlitlighet och minskar risken att backup:en skall behöva användas. En annan fördel med SCSI-anslutning är de högkvalitativa backuplösningar som erbjuds. Oftast innebär dessa bandstationer med SCSI-interface.

8.2 Backuplösningar

Flera olika backuplösningar finns. Ofta används bandspelare av något slag vilka lagrar data på band. Flera olika bandstandarder är etablerade: DAT, QIC, DLT med flera. Det finns både analoga och digitala bandstationer, de digitala är dyrare men har högre kapacitet per band. De analoga kostar mindre men de lagrar å andra sidan data med inte fullt lika hög kapacitet.

[1] Det kan vara förnuftigt att införskaffa ett brandsäkert skåp för att lagra kopian i. På ett företag jag känner till tar en sekreterare med sig veckobackupen hem. Det fungerar alldeles utmärkt.

I samtliga fall kan man tänka sig att informationen på bandet lagras seriellt[2]. Man kan se bandet som en enda lång rad av bytes. För att hitta ett visst data, det kan vara en fil eller katalog, måste bandstationen söka igenom bandet från början ända tills den hittar datat. Det medför att sökning på band är en mycket långsam process.

Bandstorleken mäts numera[3] i gigabyte och band finns i storlekar från någon GB till några hundra GB.

Samtliga SCSI-bandstationer är pålitliga och driftsäkra emellertid verkar QIC- och DLT-band något bättre i det avseendet än övriga typer.

Nuförtiden är hårddiskar såpass billiga och finns i tillräckliga storlekar att man även kan överväga att använda en hårddisk som backup-medium. Hårddiskar kan göras lätt löstagbara med speciella slädar om så önskas. På så sätt kan backupen göras och sedan disken avlägsnas från backupdatorn för vidare befordran till säkrare förvaring.

Vid val av bandstation ska man i allmänhet inte låsa sig vid vad själva bandstationen kostar. Bandkostnaden kan bli en väsentlig del av backupkostnaden över tiden. Även prestanda vid backup, bandkapacitet och lagringshastighet, är av väsentlig vikt.

En typisk enklare bandstation (QIC-typ, Tandberg SLR-5) kan ha band i storleksordningen 4/8[4] GB och en datalagringstakt på cirka 300 KB per sekund. Med denna bandstation tar det alltså cirka 4-8 timmar att fylla ett band. Om data inte får plats måste man vara närvarande och byta band efter cirka 4-8 timmar. Har du råd/tid med det? Med hårddiskar i storleksordningen 20-100 GB är denna bandstation uppenbarligen otillräcklig, det går åt för många band.

Partionering av systemets hårddisk i mindre partitioner och uppdelning av hårddiskens innehåll kan nedbringa antalet band. Systemfiler ändras relativt sällan och behöver inte säkerhetskopieras lika ofta som användarnas filer som bör kopieras varje dygn. Man kanske kan acceptera flera band för systemet och ett band för `/home`?

8.2.1 `mt(1)`

För styrning av bandstationen finns kommandot `mt(1)`. Med `mt(1)` kan man bland annat ställa in hårdvarukompression om bandaren stödjer detta, stega över tidigare inlagda data och spola bandet fram och åter innan första användning, s. k. *retension*.

[2] Själva lagringstekniken kan variera, men ur användarsynvinkel upplevs det alltid som om datat är lagrat seriellt, efter vartannat.

[3] 2004.

[4] Bandstationen har kretsar från IBM för att automatiskt komprimera informationen innan den läggs på band. Därav beteckningen 4/8, den innebär att bandet klarar av 4 GB men med den inbyggd komprimeringsalgoritmen kan upp till 8 GB data lagras. Text och annan ”regelbunden” data komprimeras väl, medan körbara kompilerade filer, med mer oregelbundna data, knappt komprimeras alls.

SCSI-bandstationen, `/dev/sa0`, kan nås genom två olika enheter, `/dev/rsa0` och `/dev/nrsa0`. `r` står för *rewinding*, återspolande och `nr` för *non-rewinding*, icke återspolande enhet. Om man kommer åt bandaren med det första, `/dev/rsa0`, kommer den att återspola bandet vid avslutat kommando, medan en åtkomst via `/dev/nrsa0` låter bandet stå kvar där det är. Detta senare kan utnyttjas för att fylla ett band med flera, efter varann följande, backuper.

I detta sammanhang måste nämnas hur bandaren skiljer på en backup och en annan på samma band. Efter en backup placeras ett `EOF`, *End-Of-File*, filslutstecken som separator och efter samtliga backuper placeras två `EOF`. Genom att studera hur många `EOF` det finns på bandet kan `mt(1)` avgöra om den spolat framåt/tillbaka en fil eller om sista backupen på bandet nåtts. Direkt på bandet, ofta i form av små hål i bandet, finns dessutom en annan markör för *End-Of-Tape*, slut-på-bandet, `EOT`. `EOT` används bland annat av `dump(1)` (se nedan) för att anmoda användaren att sätta i ett nytt band.

8.2.2 Program

Backup kan göras med i huvudsak två kommandon[5]: `tar(1)` eller `dump(1)`. Vilket kommando man väljer beror på vilket syfte man har med backupen. Om syftet är att spara enbart filer och katalogträd är `tar(1)` att rekommendera. Med `dump(1)` kan man bara spara undan hela partitioner, även bootbara, vilket inte `tar(1)` kan.

Observera att man behöver inte nödvändigtvis en bandstation för att använda `tar(1)` och `dump(1)`, båda kan spara till en fil istället. För att återställa en backup gjord med `tar(1)` används `tar(1)` medan en backup gjord med `dump(1)` måste återställas med `restore(1)`.

8.2.3 tar(1)

`tar(1)` är en förkortning för *tape archive*, bandarkiv. Kommandot har under årens lopp dragit på sig en hel del växlar, som kan verka förbryllande till en början. Emellertid används `tar(1)` oftast med bara några få av dem.

I sin enklaste form används tar för att lägga ihop flera filer, eller kataloger, till en enda s. k. tarboll:

```
> cd /
> tar cf /tmp/utfil.tar /home
>
```

Kommandot skapar (`c`, *create*) filen `/tmp/utfil.tar` som innehåller hela `/home`.

Omvändningen, att återställa katalogen /home från filen utförs som,

[5]Det finns fler, till exempel `cpio(1)` eller `pax(1)`, men `tar(1)` och `dump(1)` är definitivt de mest använda.

```
> cd /tmp
> tar xf /tmp/utfil.tar
>
```

Växeln `x` innbär extract, extrahera, innehållet i den angivna filen. Om `home` inte finns i `/tmp` kommer den att skapas innan uppackning påbörjas.

Stora filer ger stora arkiv och för att minska storleken på arkivet kan tar använda `gzip(1)` för att komprimera de filer som ingår i arkivet. Växeln `z` används för att markera att tarbollen även är komprimerad:

```
> cd /
> tar czf /tmp/utfil.tar.gz /home
>
```

Och uppackning är det omvända:

```
> cd /tmp
> tar xzf /tmp/utfil.tar.gz
>
```

Lägg märke till filändelsen på `utfil`. Man brukar ange att en fil är ett filarkiv genom filändelsen `.tar` och om arkivet dessutom är *"g-zippat"* brukar filen avslutas med `.tar.gz` eller kortare `.tgz`.

För att bara lista filinnehållet i ett arkiv anropas `tar(1)` med växeln `t`:

```
> tar tf /tmp/utfil.tar
```

eller

```
> tar tzf /tmp/utfil.tgz
```

om arkivet skulle vara komprimerat.

För att skriva till en bandstation kan dennas enhetsnamn, exempelvis `/dev/rsa2`, anges. För att slippa behöva göra detta kan miljövariabeln `TAPE` sättas till lämplig enhet. `tar(1)` blir i sådana fall enklare att använda,

```
# tar cz /home
```

i och med att TAPE-variabelns värde används.

8.2.4 dump

Det säkraste sättet att utföra backup är med `dump(1)`. Eftersom `dump(1)` bara kan göra backup på hela partitioner är det rätt utrymmes- och därmed tidskrävande. För att undvika detta kan backupen utföras på flera nivåer, *levels*. En *nivå 0*-backup backar upp *allt* i hela partitionen. En *nivå 1*-backup backar bara upp de filer eller kataloger vars tidsstämpel är senare än senast utförda *nivå 0*-backup. En *nivå 2*-backup backar upp bara de filer eller kataloger vars tidsstämpel är senare än senaste *nivå 1*-backup osv. Upp till *nivå 9*-backuper kan göras.

Medan detta system minskar tidsåtgången för backuperna, ökar det tiden för återläsning av dem, eftersom återläsningen också måste ske i ordningen *nivå 0*, *nivå 1*, ..., *nivå 9*, om man nu använde alla nivåerna[6]. Det kan alltså vara förnuftigt att reda ut om man verkligen behöver mer än, säg, nivåerna 0 och 1.

Avslutningsvis redovisas här två skript som används för att backa upp en hel dator i nivå 0 och nivå 1. Här antas att systemet i sig inte ändras ofta och bara kräver nivå 0-backuper. Informationen på /home ändras däremot ofta och utöver en nivå 0-backup som utförs en gång per vecka sker också dygns-backuper på nivå 1.

Nivå 0-backup, systemet Skriptet startas manuellt och kräver vid stora backupvolymer också bandbyte. En log över hela backupprocessen lagras för senare inspektion.

```
#!/bin/sh
echo 'Total dump av systemet på server: / /usr och /var'
mt comp on
dump -0uab 64 -f /dev/nrsa0 /dev/da0s1a 2>&1 | tee /root/ADM/sysdump0.log
dump -0uab 64 -f /dev/nrsa0 /dev/da1s1e 2>&1 | tee -a /root/ADM/sysdump0.log
dump -0uab 64 -f /dev/rsa0 /dev/da0s1e 2>&1 | tee -a /root/ADM/sysdump0.log
mt offline
```

Nivå 0-backup, `/home` Här visas också hur man i shell-skript definierar en miljövariabel, `TAPE` i detta fall.

```
#!/bin/sh
set TAPE=/dev/nrsa0
echo 'Total dump av /home på server'
mt comp on
dump -0uab 64 -f /dev/nrsa0 /dev/da2s1e 2>&1 | tee /root/ADM/dump0_home.log
mt offline
```

Nivå 1-backup, `/home` Istället för att ange miljövariabeln `TAPE` kan man explicit ange vilken enhet som `mt(1)` ska använda. Observera frånvaron av `mt offline`-raden för att hindra återspolning.

```
#!/bin/sh
echo 'Inkrementell dump av /home på server'
mt -f /dev/nrsa0 comp on
dump -1uab 64 -f /dev/nrsa0 /dev/da2s1e 2>&1 | tee /root/ADM/dump1_home.log
```

[6] I manualsidan för `dump(1)` finns förslag på en effektivare strategi.

Slutord

Jag har i boken försökt ge en inblick i hur FreeBSD kan underlätta flera uppgifter man använder en dator till vare sig det gäller jobb, skola eller hemma. Läsaren bör nu vara så förtrogen med FreeBSD att han kan på egen hand fortsätta utforskningarna. Under detta jobb kommer han säkert att träffa på situationer som inte berörts i denna bok.

Det är först då administratören får visa sin egen skicklighet och kreativitet. Men för att kunna lösa alla uppkomliga situationer måste han först fått en ordentlig förståelse för grundelementen i FreeBSD. Det har hela tiden varit min avsikt att inte så mycket beskriva specialfall utan i huvudsak presentera en helhet, en helhet som är avgörande för hur systemet kan anpassas till varje unik situation.

9 Referenser

9.1 Böcker

9.2 Internet-länkar

9.3 Tips

9.4 Register