1 Innehåll

2 Introduktion

2.1 Inledning

Iden bakom denna rapport var att i samband med att datorföreningen FUKT på högskolan i Karlskrona/Ronneby skulle starta upp ett nytt UNIX-system, dokumentera tillvägagångssättet och utreda vissa grundläggande begrepp.
Studenterna bakom projektet är huvudsakligen telekommunikations studenter som vill praktisera sina teoretiska kunskaper inom UNIX och Internet.
UNIX-systemet som vi valt heter Linux, en gratis UNIX-klon som huvudsakligen utvecklats av Linus Torvalds.
Linux har alla egenskaper som man kan förvänta sig av ett modernt UNIX-system, inkluderat äkta multitasking, virtuellt minne, delade bibliotek, fulländad minnes hantering och TCP/IP nätverkstöd.

Mats Karlsson & Dragos Ilie

2.2 UNIX historik.

Unix rötter går tillbaka till 1960-talets mitt, då AT&T, Honeywell, Gereral Eletric och Massachusetts Tekniska Institut gick ihop om ett massiv projekt som gick ut på att utveckla ett informations redskap. Projektet som kallades Multics (Multiplexed Information and Computing Service), var hårt sponserat av Amerikanska DARPA (Department of Defense Advanced Research Projects Agency). Den huvudskaliga delen av forskningen tog rum i Cambridge, Massachusetts på MIT.

Multics var ett modulsystem uppbyggt av banker med höghastighets processorer, minne och kommunikations utrustning. Designen var sådan att delar av datorn kunde tas ner för service utan att det påverkade andra delar av systemet. Målet var att erbjuda ett fungerande datasystem 24 timmar om dygnet, 365 dagar om året och systemet kunde få högre hastighet genom att lägga till nya delar.

Multics designades även med militär säkerhet i tankarna. Det var både resistant mot externa säkerhets attacker och skyddade användarna på systemet ifrån varandra. Multics stödde konceptet fler-lagers säkerhet (multi-level security)

Men 1969 var Multics-projektet långt efter i planeringen, utvecklingarna hade lovad mycket mer än vad de kunde leverera inom tidsramen. Projektet splittrades när AT&T bestämde sig för att dra sig ur, men kunskapsnivån och idéerna hos forskarna på AT&T levde vidare.

Redan samma år tog Ken Thompson, en AT&T forskare som hade jobbat med Multics Projektet, en oanvänd PDP-7 dator för att implementera idéerna från Multics på egen hand. Thompson fick snart sällskap av Dennis Ritchie som också hade jobbat med Multics. Brian Kernighan föreslog namnet UNIX som är var taget lite på skoj efter projektet Multics, som hade fortsatt i Cambridge. UNIX var ett fungerande system flera månader före Multics. Två år senare skrev Thompson och Ritchie om UNIX för Digitals nya PDP-11 dator.

Alltefter som de två vetenskapsmännen under 1970-talet adderade nya egenskaper till systemet, utvecklades UNIX till att bli programmerarnas dröm. Systemet var byggt på kompakta program, så kallade verktyg, som var för sig utförde en enda funktion. Genom att komponera ihop dessa verktyg kunde programmeraren utföra mycket mer komplicerade saker.

År 1973 skrev Thompson om UNIX med Ritchies nyligen uppfunna C programmerings språk. C var utvecklat till att vara ett enkelt, portabelt språk. Program skrivana i C kunde enkelt flyttas ifrån en typ av dator till en annan och ändå var de nästan lika snabba som program skrivna i maskinkod. Tanken med att kunna flytta C program till alla typer av plattformar fungerade inte riktigt, p g a olika operativsystems sätt att hantera I/O. Lösningen kom 1977 då gruppen kom på att det vore lättare att göra UNIX operativsystem flyttbart än att översätta alla I/O bibliotek. UNIX översattes först till laboratoriets Interdata 8/32, en mikrodator liknande PDP-11. År 1979 översattes operativsystemet till Digitals nya VAX minidator.

UNIX hade blivit att populärt operativsystem på många universitet och marknadsfördes redan av flera företag. Det hade vuxit till något mer en bara forskardrömmar. Thompson och Ritchie presenterade, November 1973

operativsystemet på en konferens vid Purdue universitetet. Två månader senare beställde University of Califonia at Berkeley, en kopia av operativsystemet för att köras på deras nya PDP-11/45.
1977 körde mer än 500 stationer UNIX, av dom var 125 vid universiteten.

Vid Berkeley, fick UNIX en ny skepnad. Liksom de andra skolorna hade Berkeley betalt $400 för ett band som innehöll det kompletta källkoden till operativsystemet. Men istället för att endast köra UNIX började två av Berkeleys studenter, Bill Joy och Chuck Haley, att göra modifikationer. 1977 skickade Joy ut 30 gratis kopior av ”Berkeley Software Distribution” (BSD), en samling program och modifieringar till AT&Ts UNIX-system.

Under de följande sex åren växte det så kallade BSD UNIX till ett eget operativsystem som hade signifikanta förbättringar jämfört med AT&Ts version. Till exempel, en programmerare som använde BSD UNIX kunde byta mellan flera program som körde samtidigt. AT&Ts UNIX tillät endast 14 tecken i filernas namn medan BSD tillät upp till 255 tecken. Berkeley utvecklade även mjukvara till att koppla ihop många UNIX-datorer genom höghastighets nätverk. Men den kanske viktigaste förbättringen var ”4.2 UNIX networking software” som gjorde det både möjligt och enkelt att koppla ihop UNIX-datorer till lokala nätverk. Av dessa anledningar blev BSD UNIX mycket populär i forsknings och den akademiska världen.

Idag körs UNIX på flera miljoner datorer bara i USA och mångdubbelt i övriga världen. Det finns minst en UNIX version till nästan alla datorer.

2.3 Linux historik.

Linux utvecklades av en ung finländsk student vid namn Linus Torvalds. Han hade inte som avsikt att skapa ett fulländat operativsystem. Till en början ville han endast få förståelse för de speciella task-switching kommandona hos Intels 80386-processor. Till att kompilera sitt testprogram använde han MINIX, ett pedagogiskt operativsystem av Andrew Tannenbaum som användes till undervisning i opertivsystem. MINIX hade vissa begränsningar som Torvalds snabbt stötte på, han började själv steg för steg, utveckla ett litet operativsystem (en kärna) som kördes i skyddat läge hos 80386.

Efter ”the task switcher” skrev Torvalds en enkel drivrutin för tangentbordet så att han interaktivt kunde arbeta med systemet. Än så länge var Linux fortfarande beroende av MINIX, men detta skulle snart ändras.

För att undvika utvecklingsarbetet av ett nytt filsystem bestämde han att adoptera MINIX filsystem. Han slapp inte bara en massa arbete utan fick också ett stabilt filsystem för hantering av hårddisken. Efter några månaders utvecklingsarbete ansåg han systemet vara tillräckligt moget för att presenteras för en mer generell publik.

Augusti 1991, dök för första gången den kompletta källkoden till Linux upp på Finlands största FTP server (nic.funet.fi). Det annonserades som en ”gratis distribuerad MINIX klon” och fångade endast uppmärksamheten hos en liten samling programmerare på nätet. Endast två månader senare publicerade Torvalds nästa version (0.02), som innehöll en del grundläggande UNIX kommandon och GNU kompilatorn (gcc) tillät kompilationen av ett UNIX shell (bash).

Det tidiga valet att följa POSIX, en känd standard sammanställd av IEEE (Institue of Electrcal and Electronics Engineers), spelade en viktig roll i säkrandet för översättning av standardiserade UNIX program hos dagens Linux. Men inte förrän vid slutet av 1991 började Linux få mer uppmärksamhet. Genombrottet kom den 5:e Januari, 1992, med version 0.12. Linux hade fått tillräckligt med kraft för att intressera ett större antal utvecklare. Systemet hade då fått en mekanisk swapping som gav det stora fördelar jämfört med MINIX.

Med tiden började intresserade utvecklare skicka rättelser och förslag till Finnland och därmed bidra till förbättringen av Linux systemet. Utvecklingen av Linux fortsatte att följa POSIX, egenskaper som ”Job Control” och ”switchable virtual consoles” implementerades.

Internet blev en viktig del i den snabba utvecklingen av Linux, det bidrog till att utvecklare kunde utbyta kommentarer, förbättringar och program. I början bombarderades Torvalds med 60 e-post meddelanden om dagen. Man satte upp flera diskussions grupper för att hålla utvecklings arbetet mer öppet och endast då bedarrade floden av e-post till Torvalds. I dag finns det en hel sjö av nyhetsgrupper om Linux, var av den viktigaste är comp.os.linux.announce (c.o.l.a), där nya versioner och program annonseras. Mailing listor sattes upp till Linux utvecklarna för att erbjuda informationsutbyte.

Förutom utbytet av brev och information erbjöd Internet överföring av filer, vilket bidrog till att organisera distributionen av det stora mjukvaror projket. Det snabba utbytet av information är också till fördel för användaren av Linux, som alltid kan hitta den nyaste distributionen av systemet på Internet. Om användaren stöter på problem vid installationen eller användandet av Linux finns hjälpen inte långt borta om man har tillgång till Internet.

En intressant aspekt av Linux historia är att det aldrig fanns någon auktoritet som övervakade utvecklingsarbetet. Istället drevs projektet av entusiasm hos programmerarna som fortsatte att bidraga med nya förbättringar och förslag. Dessa var ofta professionella programmerare eller anställda vid stora institutioner som ställer upp på sin fritid.

Men trots denna uppslutning av utvecklare är det huvudsakligen Linus Torvalds som tar hand om vidareutvecklingen av kärnan, en liten skara av andra kompetenta har tagit över andra areor av systemet. Sådana areor är översättningen och underhållet av GNU C kompilatorn och C biblioteken till Linux, underhållet av X Windows systemet och nätvärks utvecklingen. Andra Linux utvecklare jobbar med användare och system dokumentationen eller sätter ihop installerbara system på diskett eller CD-ROM.

Gratis källkod är tillgänglig både till kärnan och de flesta applikationer. Även om majoriteten av all mjukvara till Linux är gratis finns det flera kommersiella program till Linux. Såsom Modula-2 kompilatorer, CAD-program, flera databas system och OSF/Motif grafiska användagränssnitt som blivit standard i UNIX världen.

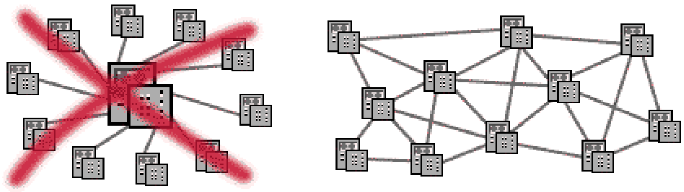
Version 1.0 var planerad till December 1992, men blev försenad - inte p g a att det inte var stabilt utan funktionen var inte riktigt enligt UNIX-standard. Torvalds sa vid ett tillfälle att den användbara versionen 0.12 borde ha fått namnet version 1.0, noll versioner har en förmåga att skrämma intresserade från att gå djupare in i systemen.

Den slutliga versionen 1.0 släpptes Mars 1994, och utvecklingen fortsatte med 1.1x serien. Numreringen av systemet ledde till står förvillelse bland användarna. Flera CD distributörer bidrog också till förvillelsen genom att sälja sina utgåvor med egna version nummer. Att fråga om version numret på CD är missvisande, istället ska man fråga efter versionen på kärnan, C biblioteken, kompilatorn eller X11.

En annan vanlig feltolkning var den att ju högre version nummer desto bättre och stabilare var versionen. Detta är inte riktigt sant. I början var kärnan 1.0 Patchlevel 9 (eller enklare 1.0.9) den enda stabila kärnan, kärnorna med numren 1.1.x innehöll många nya men inte än fullt utvecklade funktioner. Dessa var till för utvecklarna som ändrade i dom flera gånger i veckan.

2.4 Internet historik.

På 60-talet började den amerikanska militären att oroa sig för hur man skulle samordna landets dyrbara datorresurser på ett så osårbart sätt som möjligt. T.ex. ville man att datornätverket skulle överleva en kärnvapenattack. Man började diskutera idén om ett **distribuerat nätverk utan central styrning**. Alla noder i nätet skulle ges lika status, eftersom en central kommandonod snabbt skulle bli den första att slås ut.
För att öka säkerheten kom man fram till att dessa nätverk borde vara paketväxlade (varje meddelande delas upp i paket, korta snuttar som oberoende av varandra söker sin väg genom nätet, och sätts samman hos



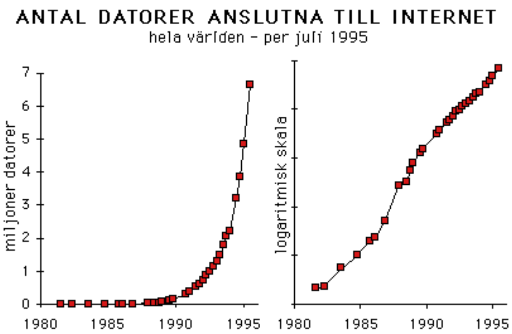
mottagaren). Om en del av nätet skulle slås ut kan pakteten hitta fram via alternativa vägar.
I slutet av 1960-talet bestämmer sig Amerikanska DARPA (Department of Defense Advanced Research Projects Agency) för att realisera idéerna, de startade ett projekt vid namn APRANET. Det experimentella nätet demonstrerades offentligt första gången 1972 och hade då ca 70 anslutna datorer. Antalet anslutna datorer

växte relativt snabbt och vid slutet av 1970-talet var ca 200 datorer anslutna. De första icke-amerikanska anslutningarna var England och Norge, etablerades 1973.

Nätverket visade sig vara mycket användbart redskap för forskarsamället. Forskarna använde inte enbart nätet för att ta del av varandras datorresurser utan började också använda nätet till kommunikation (utbyte av privata meddelanden (början till epost) och kollektiva diskussioner (början till utskickslistor)).

På det tekniska planet börjar ARPANET hitta sin form. Man baserade utvecklingen på fritt tillgängliga och öppna standarder, så att vem som helst med det tekniska kunnandet kunde sätta upp egna nät.
Inom den akademiska världen byggdes också tidigt alternativa nät ut vid sidan om ARPANET, bl.a USENET (för nyhetsgrupper, 1979) och BITNET (för e-post, 1981). Den pådrivande kraften var mindre universitet som inte hade möjlighet att delta i det dyra och hårt styrda ARPANET- projektet.

Med gemensamma standarder gavs möjligheten att knyta samman de olika delnäten till ett nätverk av nätverk, ett s.k. ”Internet”. 1982 antogs standarden TCP/IP som är en samling protokoll som bl.a. klarar översättningen mellan olika nätverkstyperna. Från 1983 blev det allt vanligare att hela nätet också kallades just INTERNET, där det ursprungliga ARPANET bara var en del (den militära delen lämnade i frustration projektet 1983 och startade ett eget). Allt fler icke-amerikanska nätverk anslöt sig - Internet blev internationellt.



Antalet anslutna datorer växte exponentiellt, det passerade tusen-strecket 1984, en miljon under 1992, och växer med ca 100% per år.

Vid sidan om Internet utvecklades många nya tekniker för datorkommunikation. Lokala nätverk (LAN) knöt samman datorer i t.ex kontor, vilket effektiviserar datorutnyttjandet hos företag och organisationer (med delade resurser som skrivare, intern epost m.m). Utvecklingen av modem gjorde det lätt för privatpersoner att ansluta sig till BBS:er eller kommersiella online-tjänster, som t.ex Compuserve och Minitel (med epost, diskussionsgrupper, filarkiv m.m).

Tillväxten av nätanvändningen ökade snabbare än man kunde finansiera och bygga ut bättre nät. Internet Society beslöt att öppna nätet för kommersiella intressen. Där med växte oro för att IP-numren skulle ta slut (anslutna datorers unika identitet/adress).

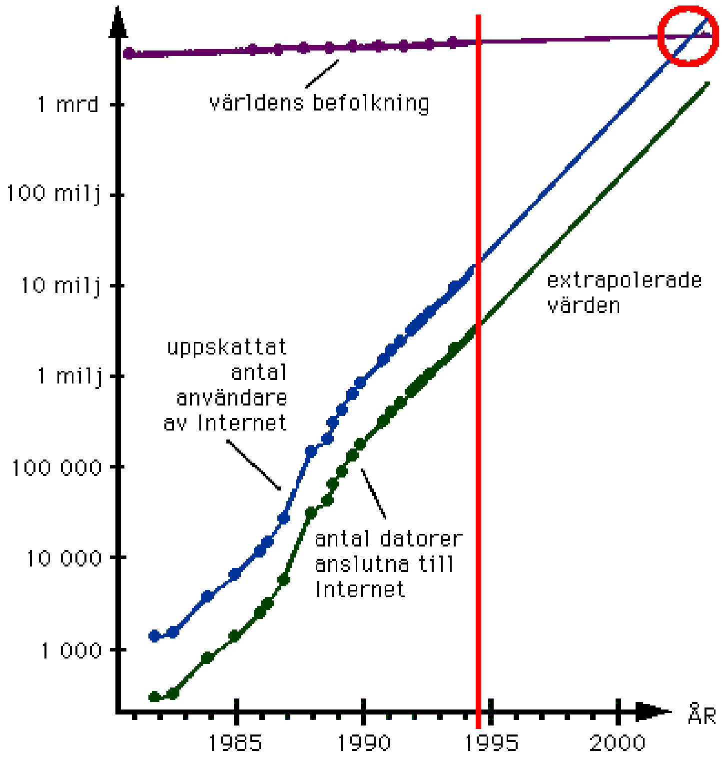
NCSA Mosaic, det första grafiska programmet till hypertext-systemet World Wide Web gjorde för första gången en del av Internet användbart och attraktivt för en bredare publik. De billiga persondatorerna och ‘hypen’ kring multimedia bidrog till webbens popularitet. Nya användarvänliga program främjade "webtop publishing".

Internet-intresset öppnade en ny hastigt växande marknad för Internet-leverantörer: företag som säljer tjänster såsom anslutning och webbhotell till företag och privatpersoner. Priskonkurrensen var hård: i princip vem som helst (med en dator) kunde nu bli ansluten utan att ruinera sig.

Hallåt kring WWW tvingade också de tidigare slutna kommersiella online-systemen att öppna sig mot Internet. Somliga (t.ex. MSN) lades t.o.m helt om från online-system till ren Internet-leverantör. I princip alla publika nätverk kunde nu kommunicera.

Det stora antalet företag som kom till webben påskyndade utvecklingen av säkerheten på Internet. I förlängningen förväntades nätet kunna kommunicera även med affärsnätverken. De nya användargrupperna (privatpersoner och företag) kommer eller har redan successivt förändrat den gamla nätkulturen.

Om man antar att Internets tillväxt kommer att fortsätta i oförminskad takt de närmaste åren kommer det en bit in på 2000-talet vara lika många Internet-användare som det finns människor på jorden... (Knappast troligt!)


1 mrd
100 milj
10 milj
1 milj
100 000
10 000
1 000
världens befolkning
uppskattat antal användare av Internet
antal datorer anslutna till Internet
extrapolerade värden
1985
1990
1995
2000
ÅR

3 UNIX grunder

3.1 Multi-user operation

När man vill utnyttja ett system (oftast en eller flera processorer med periferiutrustning) så effektivt som möjligt använder man sig av ett operativsystem(OS). Det är operativsystemet som bestämmer när och hur de uppgifter användaren ger ska utföras. Vissa operativsystem har den möjligheten att erbjuda flera användare access till systemet samtidigt. För att detta ska fungera snabbt, rättvist och säkert måste operativsystemet tillhanda hålla någon typ av accesskontroll och styrkontroll för de delade resurserna. Varje användare som vill utnyttja systemets resurser måste ha en unik användaridentifikation (UID) och lösenord[1]. Beroende på vilka accessprivilegier och rättigheter användarna har, ger operativsystemet användaren tillgång till systemets resurser.

System med en sådan typ av resursuppdelning kallas multi-user system eller fleranvändarsystem. UNIX är ett typiskt fleranvändarsystem som tillhandahåller ett unikt UID för varje användare. UNIX erbjuder också **groups** som är grupperingar av vissa användare. En användare kan samtidigt vara medlem i flera grupper. Detta kan vara praktiskt om en användare är inblandad i flera projekt samtidigt eller behöver ha tillgång till flera olika informationskällor. Rättigheter och ägandeskap kan ges individuellt till användare och grupper, detta för att rättvist dela resurserna och samtidigt öka säkerheten.

Alla fleranvändarsystem har en privilegierad användare som administrerar systemet. Denna användare kallas för systemadministratör eller superuser. I UNIX har denna användare användarnamnet **root** med den numeriska identifikationen **0**. Denne superanvändares privilegier är obegränsade. Systemadministratören kan skapa nya användare/grupper och ändra/tilldela rättigheter. I kapitel 7 kommer systemadministratörens arbete att beskrivas mer i detalj.

Under en lång tid nu har priserna på hårdvara sjunkit medan prestanda ökat. Detta har bidragit till att fler och billigare fleranvändarsystem kunnat framställas och där med blivit mer accepterade. Istället för att ha centraliserade resurser hos varje maskin kopplas nu för tiden maskinerna ihop via nätverk som erbjuder ett decentraliserat system med delade resurser och en stora möjligheter till utbyte av information. De grafiska och användarvänliga gränssnitten erbjuder också en större och bredare användargrupp tillgång till resurserna.

Denna utveckling sker också inom UNIX. Från början var UNIX ett rent textbaserat operativsystem men har med tiden utvecklats till ett grafiskt operativsystem (X-windows). Det grafiska gränssnittet erbjuder användaren möjligheten att ett flertal gånger logga in på samma maskin för att få tillgång till flera program samtidigt, fast han endast har en monitor. Detta sker med hjälp av virtuella konsoller.

3.2 Multitasking

Nu för tiden är vanligen alla fleranvändarsystem också multitasking-system. Ett multitasking-system har förmågan att utföra flera uppgifter samtidigt[2], dvs processorn kan köra flera program samtidigt.

När det gäller UNIX delas programmen upp i mindre enheter s.k. processer som med viss prioritetsindelning får använda processorn under en viss tid. Denna tid är för oss människor mycket liten och när UNIX snabbt skiftar mellan dessa processer uppfattar vi det som om det sker samtidigt. Denna uppdelning av processorkraft styrs av en s.k. **scheduler**, en speciell process som ser till att processorns kraft rättvist delas upp mellan systemets processer. Dessa processer som exekveras parallellt kan antingen vara program som körs av olika användare eller vara program som alltid körs i bakgrunden s.k. **deamons**.

Det finns flera olika strategier som en **scheduler** kan använda för att bestämma vilken process som ska köras efter nästa byte. En mycket enkel strategi är **round robin** som lägger processer som vill använda processorn i en cirkulär kö och låter en efter en köras i en viss tid (t ex 50 ms). Om en processen inte hinner färdigt under denna tid sparas dennes tillstånd och läggs sist i kön för att nästa gång fortsätta där den blev avbruten. En

[1] Lösenord är inget krav för att det ska vara ett Multi-user system utan mer en säkerhetsfråga.
[2] Som vissa filosofer påstår existerar inte samtidighet. Detta gäller även (oftast) i operativsystems sammanhang.

annan strategi är att ge varje process en prioritet och de processer som har högre prioritet får använda processorn en längre tid.
I UNIX finns något som heter **nice levels**, som markerar processernas prioritet. Detta ger användaren möjlighet att lägga program (t ex simuleringar) i bakgrunden så att de inte drar ner det övriga systemets hastighet. Likaså kan systemadministratören sätta högre prioritet på viktiga processer.

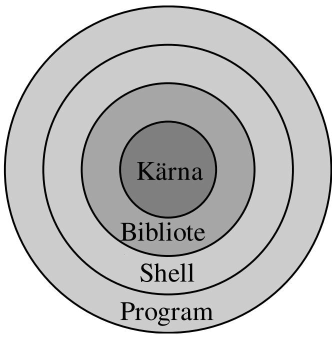
Eftersom ett program kan bestå av flera processer måste dessa kommunicera med varandra för att kunna synkronisera sina uppgifter, detta sker med hjälp av **IPC (InterProcess Communication).** IPC är en förutsättning för att multi-taskingen ska fungera.

3.3 Minnes hantering

Minnes hanteringen hos dagens UNIX-system skiljer sig kraftigt från enklare operativsystem. T ex kan UNIX tillhandahålla mer huvudminne till programmen än vad som egentligen finns tillgängligt. Detta kallas för **virtuellt minne** och använder sig av metoden **paging**. Med hjälp av tabeller mappar operativsystemet en större logisk minnesarea till mindre fysiska minnesareor. När en process kräver mer minne än vad som finns tillgängligt läggs individuella segment av logiskt minne, som inte använts på ett tag, ut på hårddisken och mer minne blir ledigt. När sedan en annan process vill komma åt en logisk adress, som finns på hårddisken, laddas respektive minnes segment (kallas för en **page**) tillbaka in i huvudminnet medan ett annat segment måste läggas på hårddisken, för att ge plats. Eftersom accesstiden hos hårddisken är mycket 'sämre' än hos huvudminnet kostar detta utökade minne exekveringstid, men risken för att systemets minne ska ta slut minskar kraftigt.

För att kunna använda hårddisken för **virtuell minnehantering** måste det finnas **swap filer** eller en **swap partition** på hårddisken. Det är till denna fil eller partition som minneshanteraren **swappar** ut minnessegmenten. Utan sådana filer eller partitioner är huvudminnet begränsat till sin fysiska storlek.

3.4 Skalmodellen



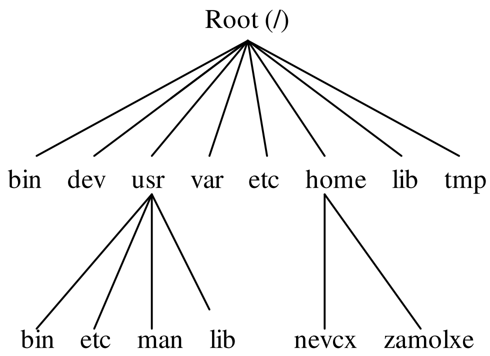
UNIX struktur brukar ofta avbildas som en **skalmodell** (se figure). Kärnan i UNIX-systemet innehåller komponenter som **scheduler**, minneshanterare och drivrutiner. Drivrutinerna sköter accessen mot den interna hårdvaran och periferiutrustningen.

Processerna i kärnan skiljer sig ifrån processerna som körs i skalet utanför kärnan. Processerna som körs utanför kärnan kan när som helst stoppas. De har dessutom individuellt allokerat minne och om de försöker komma åt minnet utanför sitt eget avbryts dom med felmeddelandet "**segmentation fault**". Processerna som tillhör kärnan har däremot obegränsad tillgång till datorn resurser och kan inte stoppas. Därför delar man in processer i två olika klasser, de som finns utanför kärnan (körs i **user mode)** och de som tillhör kärnan (körs i **kernel mode**).

Mellan det yttre skalet och kärnan finns biblioteken som ger tillgång till kärnans drivrutiner och diverse biblioteksfunktioner (oftast skrivna i C). Dessa bibliotek är normalt länkade till program under kompileringen för att ge programmen access till datorns resurser.

Program som inte använder sig av dessa bibliotek (**statiskt länkade**) kräver mycket mer minne, därför används oftast **delade bibliotek**, vilka består av två delar. En mindre del som endast innehåller referenser till biblioteken och en större del som är själva biblioteket och som betraktas som en del av operativsystemet. Detta tillåter att flera program samtidigt kan använda drivrutinerna i de delade biblioteken. Program som använder delade bibliotek kräver mindre minne och man kan dessutom uppgradera biblioteken utan att behöva länka om alla program som använder dom.

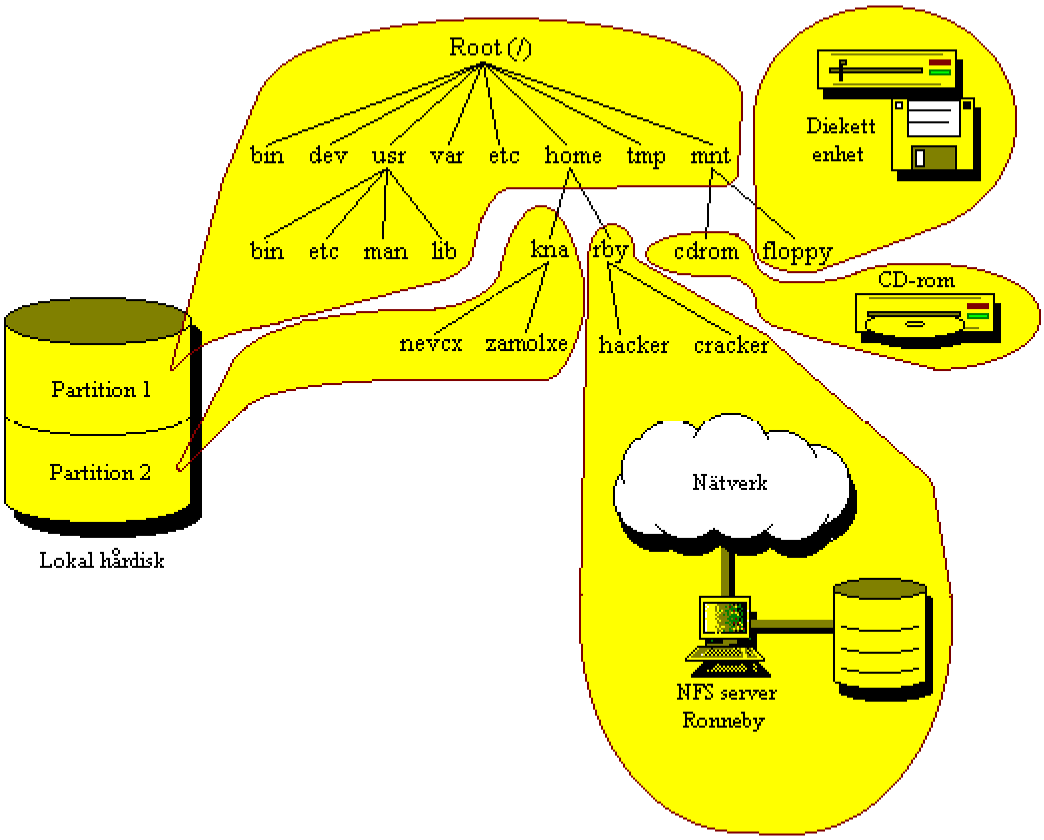
3.5 Filsystemet



Ett filsystem har hand om lagringen av data på hårdisken. Dagens moderna filsystem har en hierarkisk struktur (se figur) som erbjuder en s.k. biblioteksträd med roten i toppen (**root (/)**). Användaren kan lagra filer i olika bibliotek beroende på vilken typ av fil det rör sig om, detta gör det enklare att underhålla filsystemet. För att komma åt filerna i de olika biblioteken anger användaren vägen (**the path**) till filen. I UNIX används **slash (/)** för att beteckna vägen mellan de olika biblioteken. T ex när användaren **nevcx** vill komma åt filen tentor.txt som ligger i hans bibliotek skriver kan **/home/nevcx/tentor.txt** .

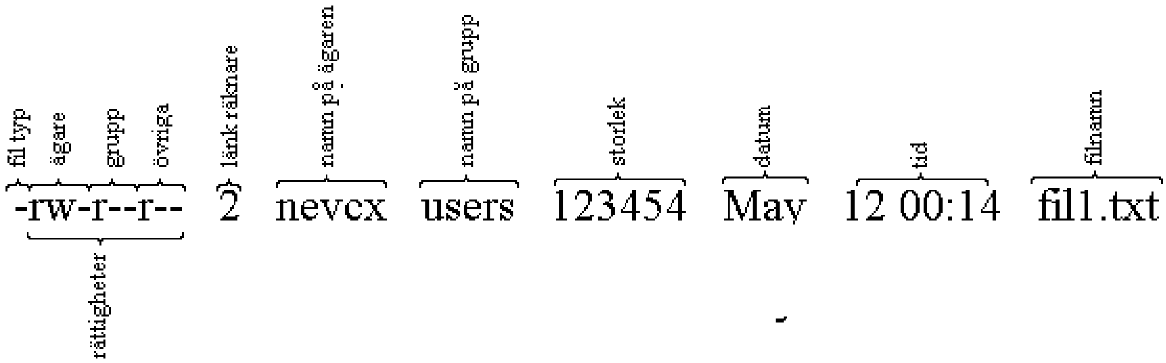
I UNIX kan vägen till en fil eller ett bibliotek anges absolut (som exemplet ovan) eller relativt till det bibliotek som man för tillfället befinner sig i (börja med **./** som ersätter vägen upp till det nuvarande biblioteket). Användarnas bibliotek är speciellt viktiga här, eftersom det är där användaren lagrar alla sina personliga filer och det är där användaren hamnar när han loggar in.

Till skillnad från t ex DOS bakar UNIX in alla lagringsenheter i filsystemet, detta innebär att man inte kan se på vilken hårdisk eller partition man jobbar. Filsystemet verkar som en enda stor virtuell hårddisk. Detta kan vara problematiskt när man vill använda flyttbara lagringsenheter som t ex disketter. Innan man kan använda en diskett måste diskettenheten först länkas till filsystemet med kommandot **mount**, vilket normalt endast systemadministratören kan göra.



Själva underhållet av filerna skiljer sig också ifrån DOS. I DOS skapas en filallokeringstabell (FAT) på varje enhet, som innehåller alla fria och allokerade sektorer. Biblioteken innehåller både namnen på filerna och deras attribut, som storlek och datum.
UNIX allokerar istället en **i-node** för varje fil i vilken de viktigaste attributen lagras, så som namn, rättigheter och start block. Biblioteken där emot innehåller endast referenser till de respektive i-noderna.
Eftersom metoden med i-noder är både mer ekonomisk och effektiv i avseende på access, passar den bättre till underhållet av större filsystem än metoden med FAT.

3.5.1 Rättigheter



När en fil skapas i UNIX lagrar operativsystemet inte bara filnamn och datum utan också användaridentiteten (UID) av den som skapade filen plus skaparens gruppidentitet (GID). För att kunna skydda filen ifrån oönskad access lagras även filens rättigheter. Rättigheterna för en fil kan sättas för användaren, gruppen och generell åtkomst (övriga). Dessa rättigheter sätts med kommandot **chmod** och kan ses med kommandot **ls -l** (listar filerna i det aktuella biblioteket). Rättigheterna för en fil sätts med bokstäverna **r** (**r**ead), **w** (**w**rite), **x** (e**x**ecute). I exemplet ovan (se figur) har ägaren nevcx läs och skriv rättigheter medan gruppen users och de övriga i systemet endast har läsrättigheter.

3.5.2 Länkar (links)

En annan trevlig egenskap hos UNIX filsystem är länkar (links). Ibland kan det vara praktiskt att ha en fil på flera ställen samtidigt i filsystemet, men i normala fall skulle detta betyda att man slösar med lagringsutrymme. Men UNIX erbjuder genom länkar ett mer praktiskt alternativ.

En länk till en fil kan antingen vara symbolisk eller hård. En hård länk är en extra referens från ett bibliotek som pekar på filen eller dess i-nod. Länkräknaren håller reda på antalet hårda länkar. Om man tar bort en fil med flera länkar minskar länkräknaren antalet länkar med ett. Endast då länkräknaren kommer till värdet ett kan filen tas bort fysiskt. Eftersom antalet i-noder endast gäller inom ett fysiskt filsystem kan inte länkar göras mellan olika enheter eller partitioner. Då måste man använda sig av symboliska länkar vilka kan referera till godtycklig fil eller bibliotek. Om den refererade filen finns eller inte spelar ingen roll när man skapar symboliska länkar.
Med hjälp av kommandot **ls -l** kan man se skillnaden mellan de olika typerna av länkar.

```
Linux> ls -l smb.conf localtime
lrwxrwxrwx   1  root    root        36  Mar 13 19:03  localtime -> ../usr/lib/zoneinfo/Europe/Stockholm
-rw-r--r--   2  root    root       840 Apr  30 08:16 smb.conf
```

Symboliska länkar markeras representeras med ett **l** i kolumnen med filtyp medan en hårdlänk endast känns igen på den ökade länkräknaren i den andra kolumnen.

3.5.3 Virtuella filsystem

För att kunna stödja olika typer av filsystem erbjuder UNIX ett extra lager, det virtuella filsystemet, mellan kärnan och drivrutinerna till filsystemen. Det virtuella filsystemet definierar ett antal rutiner som måste finnas i alla filsystem, t ex öppna fil, läs fil, skriv till fil och stänga fil. Sedan översätter detta lager mellan kärnan och filsystemen. Detta innebär t ex att man kan komma åt ett DOS filsystem som ligger på hårddisken.

3.6 Tillbehör (Devices)

UNIX mappar hårddiskar så väl som terminaler och annan periferiutrustning i en speciell katalog (**/dev**) i filsystemet. Detta för att programmeraren ska kunna komma åt enheterna som om de vore filer.
Detta ger också användaren vissa praktiska möjligheter. T ex om användaren vill att innehållet i en fil ska skrivas ut på skrivaren istället för på skärmen, kan han omdirigera datan till skrivarens enhetsfil i katalogen **/dev**.

$ cat textfil.txt > /dev/lp1

Likaså är diskettenheten (**/dev/fd0**), musen (**/dev/mouse**) och hårdiskarna (**/dev/sda)** representerade som filer i katalogen **/dev** (se tabell).

/dev/console	system konsolen
/dev/mouse	seriell mus
/dev/hda	första AT-buss hårddisken
/dev/hda1	första partitionen på första AT-buss hårddisken
/dev/hda2	andra partitionen på första AT-buss hårddisken
/dev/hdb	andra AT-buss hårddisken
/dev/hdb1	första partitionen på andra AT-buss hårddisken
/dev/sda	första SCSI hårddisken
/dev/sdb	andra SCSI hårddisken
/dev/lp0	första skrivaren (LPT1)§
/dev/null	null enhet (all utdata kastas)
/dev/ttyN	virtuell konsol
/dev/ptyN	pseoudoterminal för access via nätverk
/dev/cuaN	seriell port

Filerna i katalogen **/dev** och respektive enhet är associerade med två nummer, det större enhetsnumret och de mindre enhetsnumret. Det är dessa nummer som bildar gränssnittet mot kärnan. Det finns också två typer av enheter, teckenenheter (character) och block enheter. Teckenenheter används främst till terminaler och seriell portar medan blockenheterna används för att överföra större datablock till hårddiskar och andra lagringsenheter. Enheternas nummer kan fås genom att köra kommandot **ls -l** i katalogen **/dev**. Det större enhetsnumret (kolumn 5) specificerar enhetens typ och har ofta en drivrutin i kärnan associerad med den. Det mindre enhetsnumret (kolumn 6) används för att skilja på enheter av samma typ.

```
Linux:/dev> ls -l
                                  …
brw-r-----   1  root   root    3,   0   May 12  12:26  hda
brw-r-----   1  root   root    3,   0   May 12  12:26  hda1
brw-r-----   1  root   root    3,   0   May 12  12:26  hda2
                                  …
crw-rw-rw-   1  root   root    4,   0   May 12  15:15  tty0
crw-rw-rw-   1  nevcx  users   4,   1   May 12  17:50  tty1
                                  …
```

3.7 Kommando hanterare (Shells)

Kommando hanteraren erbjuder ett interaktivt gränssnitt mellan operativsystemet och användaren. Även om man nu via de grafiska gränssnitten kan utföra det mesta, finns det förfarande UNIX-användare som föredrar den textbaserade kommando hanteraren.

3.7.1 Standardiserade kommando hanterare

Vanligen erbjuder UNIX tre modeller av kommando hanteraren: **Bourne shell (sh)**, **Korn shell (ksh)**, och **C shell (csh)**. Bourne shell var det första kommando hanteraren till UNIX, Korn shell är en utökning av Bourne och C shell är en kommando hanterare med C liknande syntax.

Som en regel använder Linux ingen av de ovanstående kommando hanterarna utan erbjuder utökade varianter av dessa. De kommando hanterare som Linux använder finns att hämta gratis till nästan alla UNIX system och ger bättre tolkning av gamla kommando hanterare.
The Bourne Again shell (bash) har ersatt gamla Bourne och Korn shell, och **the tcsh** har ersatt C shell. Systemadministratörer använder sig oftast av Bourne shell för de flesta administrativa script kräver det. De följande exemplen bygger huvudsakligen på denna kommando hanterare.

3.7.2 Interaktiv användning

UNIX-kommandon som **ls**, **cd** och **man**, vilka man skriver på den vanliga kommandoraden, exekveras inte av själva operativsystemet utan är själva program som vanligtvis ligger i katalogerna **/bin** och **/usr/bin/**. En användare som skriver sådana kommandon, är inte i direkt kontakt med UNIX operativsystem, utan med en kommando hanterare (ett **shell**) som i sin tur kommunicerar med operativsystemet.

När en användare loggar in på ett UNIX system hamnar han oftast i ett sådant här kommando hanterare. Kommando hanteraren känner endast till ett fåtal egna kommandon, de flesta kommandona består av mindre program som startas när användaren skriver in namnet på dom. Kommando hanteraren gör egentligen inte så mycket förutom att läsa av det användaren skriver från tangentbordet och startar respektive program. Den enhet (oftast tangentbordet) som kommando hanteraren läser ifrån, kallas **standard input**.
När en användare vill få upp en lista på vilka filer som finns i en katalog, används kommandot **ls** som startar programmet **/bin/ls**. Den information som ls ger skrivs till **standard output**, som oftast är konsolen. För att kommandona inte ska bli för många kan de flesta kommandon ta emot parametrar, som gör att de kan utföra flera olika saker. T ex kan kommandot **ls** ta emot ett flertal olika parametrar, ett exempel är parametern **-l** som får **ls** att skriva ut en mer detaljerad lista över filerna i katalogen.

```
snuggles:/home/kna/t94mk> ls
addresses.dat  dead.letter    mail          www

snuggles:/home/kna/t94mk> ls -l
total 4
-rw-r--r--    1 t94mk    users        786  Apr    8  16:54 addresses.dat
-rw-------    1 t94mk    users        284  May   4  23:10 dead.letter
drwx------   2 t94mk    users       1024 May   4  22:37 mail
drwxr-xr-x  2 t94mk    users       1024 Mar 18  00:52 www
snuggles:/home/kna/t94mk>
```

3.7.3 Omgivningsvariabler (enviroment)

Ett annat sätt att överföra data till program är genom omgivningsvariabler (environment). Dessa omgivningsvariabler skickas till programmet varje gång det startar. När man vill lista dessa variabler använder man sig av kommandot **set** i Bourne shell.

```
snuggles:/home/kna/t94mk> set
PS1=$HOST:$PWD>
PS2=>
PATH=/bin:/usr/bin:/usr/local/bin:.
PWD=/home/kna/t94mk
TERM=vt100
UID=24012
snuggles:/home/kna/t94mk>
```

Omgivningsvariabler kan när som helst skapas, tas bort eller ändras. T ex om man vill skapa variabeln **AUTO** och sätta den **VW**, använder man kommandot export.

```
snuggles:/home/kna/t94mk> export AUTO=VW
snuggles:/home/kna/t94mk> set
PS1=$HOST:$PWD>
PS2=>
PATH=/bin:/usr/bin:/usr/local/bin:.
PWD=/home/kna/t94mk
TERM=vt100
UID=24012
AUTO=VW
snuggles:/home/kna/t94mk>
```

3.7.4 Omdirigering

I UNIX används något som kallas för **standard input (stdin)** och **standard output (stdout)**. I normala fall är stdin associerat med tangentbordet och stdout med konsolen. Enkla kommandon som **ls**, skriver till stdout(konsolen) och kommando hanteraren läser ifrån stdin(tangentbordet). Kommando hanteraren i UNIX tillåter att man omdirigerar stdout och stdin utan att påverka kommandona. För att göra detta används operatorerna **>** och **<**.

```
snuggles:/home/kna/t94mk> ls > list.txt
snuggles:/home/kna/t94mk>
```

Exemplet ovan visar att om man dirigerar om kommandot **ls** utdata till filen **list.txt**, visas inget på konsolen. Informationen som **ls** ger skrivs istället till filen. För att lista innehållet i filen **list.txt** använder man kommandot **cat** med filnamnet som parameter.

```
snuggles:/home/kna/t94mk> cat list.txt
addresses.dat
dead.letter
mail
www
snuggles:/home/kna/t94mk>
```

Kommandon som läser data ifrån stdin kan också omdirigeras till att läsa ifrån annan enhet eller fil. Ett exempel på ett sådant kommando är **cat** som när det används utan parametrar läser ifrån stdin.

```
snuggles:/home/kna/t94mk> cat < list.txt
addresses.dat
dead.letter
mail
www
snuggles:/home/kna/t94mk>
```

Att omdirigera indatat till **cat** ifrån en fil ger som synes samma resultat som att ge filnamnet som parameter. Men istället för att öppna filen läser **cat** filens innehåll ifrån stdin.

Förutom stdin och stdout finns det en tredje kanal som är länkad med konsolen. Eftersom den används till att visa felmeddelanden på konsolen, heter kanalen **standard error (stderr)**. Även om användaren omdirigerar både stdin och stdout, kommer felmeddelandena fortfarande att visas i kosolen. Det olika kanalerna kan adresseras enligt tabellen nedan.

Kanal	Förkortning	Adressering	Standard enhet
Standard input	stdin	0	tangentbordet
Standard output	stdout	1	konsolen
Standard error	stderr	2	konsolen

T ex, för att dirigera om alla felmeddelanden till filen **error.txt**, när man använder **cat** kan följande rad skrivas.

```
snuggles:/home/kna/t94mk> cat list.txt 2>error.txt
```

Det går även att dirigera om både stdout och stderr till en fil.

```
snuggles:/home/kna/t94mk> cat list.txt 2>&1> output.txt
```

I exemplet ovan dirigeras först stderr om till filen output.txt sedan dirigeras stdout om till samma fil.

En anna typ av omdirigering är att använda ett kommandos utdata som indata till ett annat. Detta kan uppnås med hjälp av operatorn **| (pipe)**.

```
snuggles:/home/kna/t94mk> ls | wc
  8     8     63
snuggles:/home/kna/t94mk>
```

Här räknar kommandot **wc** antalet ord, linjer och tecken som den får ifrån stdin, i vårt fall räknas antalet filer istället för ord. Detta ger möjligheten att skapa nya kommandon av flera mindre kommandon.

3.7.5 Filändelser

Ibland kan det vara praktisk att kunna överföra flera filer samtidigt till att kommando, istället för att skriva in dom en efter en. Detta kan göras med s.k. **wildcards**, som fungerar som en joker när man spelar kort. Om ett kommando innehåller ett wildcard, kommer kommando hanteraren att skicka alla filer som passar till kommandot.

```
snuggles:/home/kna/t94mk> cat *.txt
```

I exemplet skickas alla filer som slutar på **.txt** till kommandot **cat**. Tabellen nedan visar de viktigaste wildcard.

Tecken	Funktion
*	Ersätter godtyckliga tecken (eller ingen).
?	Ersätter ett godtyckligt tecken.
abc…]	Ersätter ett tecken ifrån definierad mängd
[a-z]	Mängder kan def. med bindestreck (-).
[!abc…]	Ersätter alla tecken utom dom i mängden.

3.8 Daemons

Daemons är speciella processer som körs i backgrunden. Dessa processer utför oftast viktiga uppgifter åt UNIX systemet. Genom att använda daemons kan stora delar av UNIX köras som oberoende program. Detta gör att operativsystemets kärna håller sig relativt liten. Dessutom kan enstaka daemons aktiveras eller startas om efter uppdateringar eller konfigurationsändringar utan att påverka resten av systemet.
Här följer några exempel på daemons.

Line printer daemon (lpd)
Med regelbundet intervall kollar **lpd** i katalogen **/usr/spool/** efter nya utskriftsjobb. För att starta ett utskriftsjobb används kommandot **lpr**.

Cron daemon (crond)
Om en användare vill exekvera ett program vid speciella tider eller med ett regelbundet intervall, kan **crond** användas. För varje användare tillhanda håller crond en tabell över kommandon (och tider) som ska köras.

Syslog daemon
Eftersom en daemon körs i bakgrunden är dess stdout inte länkat med konsolen. Därför erbjuder **syslog daemon** ett protokoll som tar hand om utdata och felmeddelanden ifrån andra daemons. Datan som skickas till syslod daemon kan antingen skrivas på konsolen, skickas som e-mail eller skrivas till en fil.

3.9 Översikt av kommandon

Här följer en lista över de viktigaste UNIX kommandona med en kort förklaring. För att få reda på mer information om dessa kommandon och deras användning kan manualerna listas med kommandot **man** på konsolen. T ex om man vill veta mer om **ls** skriver man:

```
snuggles:/home/kna/t94mk> man ls
```

- **ls** - Skriver ut en lista över filer och kataloger. Man kan använda ls med parametern **-l**, som gör att ls även skriver ut filernas storleken, rättigheter, ägare osv.
- **cd** - Ändrar den aktuella katalogen. Om inga parametrar är specificerade ändras den aktuella katalogen till hemkatalogen.
- **cp** - Kopierar den specificerade filen till en annan katalog eller fil. Med parametrar kan hela träd av kataloger kopieras.
- **mv** - Flyttar en fil inom ett filsystem. Kan även användas till att ändra namn på en fil eller katalog.
- **rm** - Tar bort en fil. Med parametrar kan kataloger eller hela filträd tas bort.
- **mkdir** - Skapar en ny katalog.
- **rmdir** - Tar bort en tom katalog.
- **exit** - Går ur den aktuella kommando hanteraren.
- **more** - Visar en fils innehåll, sida för sida på konsolen.
- **man** - Visar manualsidorna till kommandon.
- **cat** - Används till textfiler. Kan t ex lista innehållet av en fil.
- **grep** - Söker efter vissa mönster eller ord i angivna filer.
- **passwd** - Ändrar användarens lösenord.
- **ps** - Ger en lista över processer och deras process ID.
- **kill** - Avslutar angiven process.
- **su** - ändrar tillfälligt UID (eng. *User Identity* - användaridentitet).

4 Egenskaper hos Linux

4.1 Virtuella konsoler

Med hjälp av virtuella konsoler kan en eller flera olika användare logga in på en och samma konsol samtidigt. För att skifta mellan de olika sessionerna används **<Alt>** plus en funktionstangent. Det maximala antalet virtuella konsoler är bestämt i kärnan.

När man kör det grafiska gränssnittet X11 är **<Alt>** tangenten reserverad för applikationerna. Så när man kör X11 används istället kombinationen <Ctrl-Alt> för att byta virtuell konsol. Detta ger användaren möjligheten att skifta mellan det grafiska X11 och det textbaserade gränssnittet hos den virtuella konsolen.
Man till och med starta upp flera X11-processer (s.k. X-servers), men detta är inte att rekommendera eftersom det slösar med minnet. Istället kan man köra en X windows hanterare (t ex **olvwm** eller **fvwm**) vilka också erbjuder att öppna flera virtuella konsoler.

4.2 Linux filsystem

Linux stödjer flera olika typer av filsystem. För att se vilka filsystem kärnan stödjer kan man lista filen **/proc/filesystems**.

```
snuggles:/> cat /proc/filesystems
        ext2
        minix
        umsdos
        msdos
        vfat
nodev   proc
nodev   nfs
nodev   smbfs
        iso9660
snuggles:/>
```

Här följer en kort beskrivning av de vanligaste filsystemen hos Linux.

MINIX file system
Den första linux versionen stödde endast en typ av filsystem, och det var mycket beroende av MINIX filsystem. MINIX erbjöd ett stabilt filsystem under utvecklingen av Linux. Men MINIX filsystem har sina nackdelar, filnamn kan inte vara längre än 14 tecken och partitionernas maximala storlek är 64 MB.

Extended file system (ext)
För att bli av med MINIX nackdelar skapade, en fransman vid namn Remy Card, det första alternativa filsystemet. Hans filsystem fick namnet Extended File System (ext) och stödde filnamn upp till 255 tecken och partitioner upp till 2 GB.
Men även detta filsystem hade sina svagheter. Fria block och i-noder finns inte lagrade i en bit vektor utan i en länkad lista. Detta ger efter en längre tids användning dyr fragmentering av skivminnet, vilket i sin tur ger sämre accesstid.

Exteneded2 file system (etx2)
Från det tidigare ext utvecklades ext2, vilket är det mest använda filsystemet under Linux. Fragmenterings problemet är löst och gränsen på 2 GB stora partitioner har tagits bort. Förutom detta har etx2 en mekanism som sparar alla förlorade sektorer i en speciell katalog med namnet **lost+found**. Om systemet av någon anledning skulle krascha, kan filsystemet sökas igenom (av programmet **e2fsck)** efter skadade filer och reparera dessa.

ISO 9660/HighSierra file system
För att erbjuda åtkomst till CD-rom stödjer Linux både filsystemet ISO 9660 och High Sierra. Här har även **Rockridge Extensions** implementerats för att stödja långa filnamn.

Proc file system
Detta filsystem håller inte reda på fysiska filer utan ger endast access till information i kärnan och till processer som körs i systemet. Vid uppstarten av systemet läggs detta filsystem under katalogen **/proc**. För varje process som körs skapas en underkatalog med samma namn som process ID. Filerna i dessa underkataloger innehåller process-specifik information som kan listas med kommandot cat.

Övriga filsystem
Filsystemet **msdos** gör att Linux kan läsa och skriva till DOS disketter eller partitioner, även OS/2 FAT partitioner.
För att komma åt Xenic, System V och OS/2-HPFS partitioner, finns det speciella filsystem, av vilka de flesta fortfarande är under utveckling.

För att kunna använda sig av andra partitoner eller hårddiskar någon annan typ av filsystem, måste dessa bakas in i det virtuella filsystemet. Detta görs med kommandot **mount**. Ett vanligt fall när det gäller Linux är att man har en partition eller hårddisk med DOS installerat i sin PC.

```
snuggles:/> mkdir msdos
snuggles:/> mount -t msdos /dev/hda2 /msdos
snuggles:/> cd msdos
snuggles:/msdos > ls -a
./              command.com*format.com*         tools/
../             config.sys*           io.sys*              windows/
autoexec.bat*   dos/                  msdos.sys*
snuggles:/msdos > cd dos
snuggles:/msdos/dos >
```

Exemplet ovan skapar katalogen **/msdos** och använder sedan kommandot **mount** för att lägga ett DOS filsystem som finns på enheten **/dev/hda2**, i denna katalog.

4.3 Laddningsbara moduler

Klassiska UNIX system har en monolitisk kärna. Hela kärnan måste länkas om när man vill installera en ny drivrutin. Linux kärna är lik denna i princip men erbjuder också något som kallas laddningsbara moduler.
En laddningsbar modul är oftast en drivrutin som kan laddas in eller tas bort under tiden systemet körs. Ett flertal drivrutiner är redan tillgängliga som moduler.
Följande kommandon används vid administration av modulerna.

Insmod <module>	Integrerar modulen i systemet
depmod	Listar modulens resurser
modprobe	Testar en modul
rmmod <module>	Tar bort modulen ur systemet
lsmod	Listar alla moduler som är integrerade i systemet

5 Nätverk

”Så som ett fiskenät är uppbyggt av serier av knutar, så är allting i välden sammankopplat i knutar. Om någon tror att en maska i ett nät är en självständighet, en isolerad del, så misstar han sig. Det kallas ett nät för att det är uppbyggt av serier av hopkopplade maskor, och varje maska har sin plats och sitt ansvar till de andra maskorna.”

--Buddha

5.1 Introduktion

Kommunikationsnätverk indelas i två huvudområden: växlade nätverk (eng. *switched networks*) och nätverk av typen *broadcast*[1]. Växlade nätverk överför data från källa till källa till destination genom en rad mellanliggande noder. De indelas i sin tur i två kategorier: kretskopplade nätverk (eng. *circuit switched networks*) och paketförmedlande nätverk (eng. *packet switching networks*).

I ett kretskopplad nätverk använder sig två noder som kommunicerar med varandra av en virtuell väg[2] (eng. *virtual path*) genom nätverket som inte är tillgänglig till andra stationer under den tid som kommunikationen pågår. Resurser allokeras till den virtuella vägen och frias inte förrän en av noderna stänger den. Alla noder får använda flera virtuella vägar samtidigt. Ett bra exempel på kretskopplade nätverk är telefonin: så fort två personer lyckas koppla upp sig med varandra blockeras andra personer som ringer till samma nummer.

Paketförmedlande nätverk använder sig av en annan strategi för att få informationen att komma fram. Informationen som består av data, vanligen i binär format, delas upp i mindre delar, s.k. paket. Varje paket skickas sedan ut till första noden som ligger längs vägen till destination. Sedan skickas den till efterföljande nod också vidare. Finessen är den att varje nod kan, med ledning av information om rådande förhållanden inom nätverket och på väg till destinationen, välja en optimal väg för varje enskild paket. Detta innebär att även om paket härstammar från samma data, de behöver inte följa samma väg och komma vid ankomsten i samma ordning som de hade vid avfärden. Till skillnad från kretskopplade nätverk, resurser allokeras bara för att skicka en packet och frias omedelbart därefter. På det viset är paketförmedlande nätverk resurssnålare än kretskopplade nätverk. Paketförmedling är tekniken som används inom Internet.

Vad är Internet då? För att kunna ge svar till den här frågan en måste analysera nätverkshierarkin. Längst ner i hierarkin befinner sig användaren (eng. *user*). Användaren brukar vara en människa vid en terminal men den kan också vara ett program. På nästa nivån finns värden (eng. *host*). Värden är en dator som tjänstgör under en eller flera användare. Flera värdar bygger upp ett nätverk, alltså flera datorer som kommunicerar med varandra. När dessa datorer befinner sig i samma rum eller byggnad (högst ett par kilometer avstånd) kallas de för ett LAN (eng. *Local Area Network*). Nätverk som täcker delar av eller en hel stad kallas för MAN:s (eng. *Metropolitan Area Networks*). Högst upp finns WAN (eng. *Wide Area Network*) som kopplar ihop datorer från olika städer och länder. Två nätverk som kopplats ihop formar en internet. Det som brukar kallas Internet (observera versalen!) är mängden av alla nätverk, LAN, MAN och WAN, som har kopplats tillsammans i ett världsomspännande internet. Som läsaren säkert märker det finns inga klara gränser mellan de olika områden inom hierarkin. Detta beror på Internets växt de senaste åren som har fått många av definitionerna att bli inaktuella. Trots det, de som jobbar inom området föredrar att behålla de gamla definitionerna och försöka anpassa dem till rådande förhållanden. En organisation som kämpar med detta är Internet Task Force.

Den senaste tiden ett nytt begrepp har växt fram: Intranet. Intranet är interna Internet, dvs. bara behöriga personer, t.ex. anställda på ett företag, får använda det. Intranet är oftast inte kopplade till Internet utan är bara ett sätt att sprida information mellan de behöriga användare. Anledningen till att Intranet har blivit uppskattad är att den har bara en enda användargränsnitt: webbläsaren. En annan anledning är att systemsäkerheten kan bestämmas på webbsida nivå genom att använda lösenord på varje länk som är känslig ur säkerhetssynpunkt.

[1] Den engelska termen *broadcast networks* betecknar nätverk som till varje station (nod) har en sändare/mottagare som kommunicerar över ett medium delad med många andra stationer (noder)

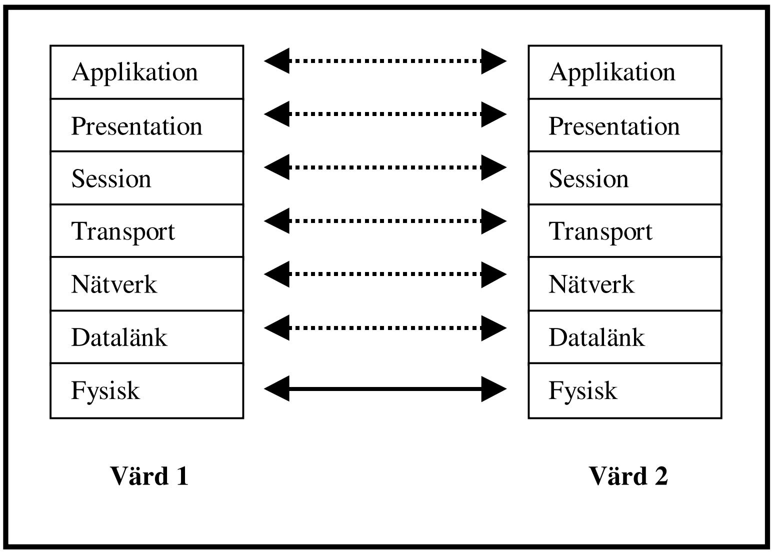
[2] En virtuell väg genom nätverket kan vara samma väg som används för att skicka paket. Den kan sträcka sig över flera nätverk. Skillnaden är att den virtuella vägen består under hela kommunikationens gång, medan paketens förbindelse måste kopplas upp varje gång. Kort sagt, den virtuella vägen simulerar en rak end-to-end förbindelse.

5.2 Öppna system

Internet är en hopsampling av nätverk som befinner sig runt omkring i världen. Det finns ingen centraldator eller server som övervakar kommunikationen utan varje dator sänder sina meddelanden (datapaket) enligt en samling väl bestämda regler. Dessa väl bestämda regler kallas tillsammans för en protokoll. Processer[1] skickar också meddelanden till andra processer på den egna datorn eller på en annan dator. De använder egna protokoll. Då flera protokoll samlas för att uppnå det önskade funktionaliteten i ett nätverk kallas dem för en protokoll familj (eng. *protocol suite*). Eftersom de flesta nätverksmodeller består av flera skikt kan protokollsamlingen också kallas för en protokollstack.

Omkring 1970-talet många företag hade byggt upp sina egna nätverk. Alla använde sig av olika tekniker som inte var lätt tillgängliga till de andra. En standard behövdes. Vid slutet på 1970-talet lade ISO (eng. *International Organization for Standardization*) fram ett förslag till standardisering. Förslaget hette *Open Systems Interconnection Reference Model* eller som det heter på svenska - OSI-modellen. Standarden skulle vara tillgänglig till alla som ville ta del av, därtill namnet *Öppna system*.

OSI-modellen består av sju skikt. Nedan presenteras de tillsammans med deras huvuduppgift.



OSI - modellen

1. **Det fysiska skiktet:** Tar hand om kommunikationen över ett medium (koaxial kabel, optiska fiber, radio etc.). Här omvandlas bitarna till elektriska eller elektromagnetiska signaler.
2. **Datalänkskiktet:** På den här nivån skickas enkla datapaket. Skiktet har som uppgift att förhindra att data förstörs i det fysiska skiktet.
3. **Nätverksskiktet:** Väljer väg genom nätverket för enskilda paket.
4. **Transportskiktet:** Sköter om meddelanden[2] som kommer från olika processer på värddatorn eller från andra datorer. Tillhandahåller felfri kommunikation.
5. **Sessionsskiktet:** Tar hand om sessioner[3]. Ser till att kompatibilitet råder mellan system som kopplar sig ihop.

[1] Skillnaden mellan processer och program är att program ligger på hårddisken eller på annat lagringsutrymme som "överlever" då strömmen slås av och processer är instanser av programmet som befinner sig i datorns primära minne. Ett program blir en process då den laddas i det primära minnet och börjar exekveras.

[2] Meddelanden från en process delas upp vanligen i mindre datapaket för att tillfredsställa bandbredd kraven inom ett nätverk. Vid destination, datapaketen sätts ihop till det meddelande som processen skickade ursprungligen.

[3] En session är tidsintervallet från det att en användare (människa eller program) loggar in på ett system till dess han/hon/det loggar ut från systemet.

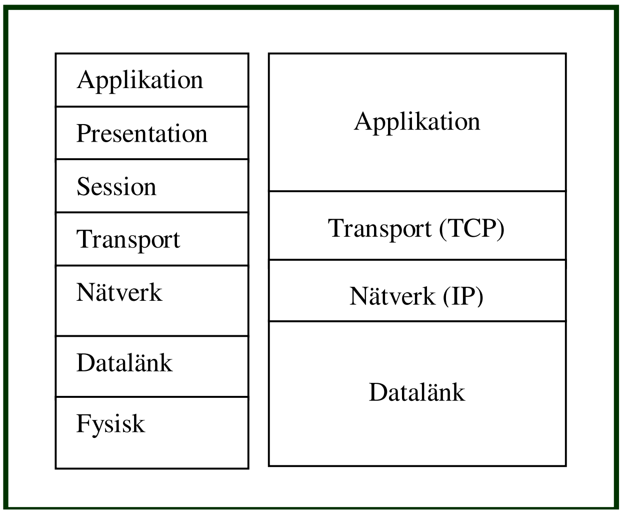
6. **Presentationsskiktet:** Löser skillnader i dataformat mellan olika system.
7. **Applikationsskiktet:** Är användarens gränssnitt mot nätverket. På det nivån finns e-post och FTP programmet.

Det fysiska skiktet är det enda skiktet som verkligen kommunicerar direkt med en annan fysiskt skikt på andra värdar. Alla andra skikt använder sig av en virtuell väg, dvs. de luras tro att det finns en direkt förbindelse allokerat för deras kommunikation med motsvarande skikt på den andra värden, trots att allt data måste passera de undre skikten på båda värdarna.

De sju skiktens uppgift är att strukturera alla funktioner inom ett nätverk för att lättare få ett överblick och underlätta vidareutveckling. Det är dock viktig att minnas att OSI-modellen är bara ett förslag på hur ett nätverk bör vara konstruerad. De flesta nätverksteknologierna idag har hämtat mer eller mindre funktioner ifrån OSI-modellen, men har försökt på olika sätt att förbättra original konceptet.

5.3 TCP/IP

Trots att OSI-modellen är nyare och används som referensram inom nätverkskonstruktion finns det en annan modell som är äldre och dessutom verkar ha blivit på sistone en *de-facto standard* (det vill säga har adopterats frivilligt av branschen utan att vara ämnad som generell standard). TCP/IP-stacken, som den brukar kallas, är en vidareutveckling från protokollen som fanns i ARPANET (se kapitel 2.4 *Internet historik*) och DDN (*Defense Data Network*).



TCP/IP-modellen mappad mot OSI-modellen

Skillnaden mellan TCP/IP och OSI-modellen är att TCP/IP har en mycket friare syn på skikten i modellen. Inom OSI-modellen en applikation måste skicka data genom alla sju skikten medan TCP/IP tillåter applikationen att starta på vilken nivå som helst.

En annan skillnad är att TCP/IP har bara fyra skikt. Länkskiktet motsvarar det fysiska skiktet och datalänkskiktet i OSI-modellen. IP-skiktet och TCP-skiktet motsvarar nätverksskiktet resp. transportskiktet. Slutligen, applikationsskiktet inkluderar sessionsskiktet, presentationsskiktet och applikationsskiktet från OSI-modellen. Dessutom, för att komplicera det hela, brukar TCP-skiktet halka in på OSI-sessionskitet och länkskiktet in på OSI-nätverksskiktet.

TCP (eng. *Transmission Control Protocol*) erbjuder en kopplingsorienterad, tillförlitlig, full-duplex, byte-ström tjänst. Tillförlitligheten uppnås genom att använda glidande fönster (eng. *sliding windows*) tillsammans med mottagningsbekräftelser. Tekniken med glidande fönster gör att TCP-protokollet kan även optimera bandbredden i TCP-uppkopplingen genom att förhandla om storleken på den glidande fönstret och dataöverföringshastigheten. En bieffekt av fönstertekniken är att byte-strömmen delas i TCP-segment. Detta märks inte givetvis av användaren på applikationsskitet.

Välkända portar är en subjekt som är stark kopplad till TCP-protokollet. På en datorvärd rullar flera nätverksapplikationer, oftast samtidigt. Det gör att det behövs en väldefinierad procedur att komma i kontakt med rätt program. För att lösa detta, varje viktig Internet tjänst tilldelas en s.k. välkänd port som alla applikationer känner till, t.ex. **FTP** har tilldelats port 21. När en applikation vill kontakta FTP-servern, den anropar port 21 på serverns värd. Anropet innehåller den anropande applikationens port, som är ett vanligt slumptal. Den behövs för att servern skall kunna hitta programmet som utförde anropet. Välkända portar kallas ofta också för välkända socklar (eng. *well-known sockets*). Socklar är en samling funktioner som används för att underlätta nätverkskommunikation programmering. Varje sockel representerar en ändpunkt i nätverket. RFC 1700 är en detaljerad beskrivning på de välkända portarna.

Nedan finns ett utdrag från filen **/etc/services** som beskriver kopplingen mellan portar och applikationer. Första fältet är applikationens namn. Det andra fältet beskriver portnummer och protokoll som applikationen stödjer. Det tredje fältet är andra namn som tjänsten får ha. Allt efter tecknet # är kommentarer.

```
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single
# well-known port number for both TCP and UDP; hence, most entries here
# have two entries even if the protocol doesn't support UDP operations.
#
# Updated from RFC 1340, ``Assigned Numbers'' (July 1992).  Not all ports
# are included, only the more common ones.
#
#         from: @(#)services    5.8 (Berkeley) 5/9/91
#         $Id: services,v 1.9 1993/11/08 19:49:15 cgd Exp $
#
tcpmux          1/tcp               # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp    sink null
discard         9/udp    sink null
systat          11/tcp   users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp   quote
msp             18/tcp              # message send protocol
msp             18/udp              # message send protocol
chargen         19/tcp   ttytst source
chargen         19/udp   ttytst source
ftp             21/tcp
# 22 - unassigned
telnet          23/tcp
# 24 - private
smtp            25/tcp   mail
# 26 - unassigned
time            37/tcp   timserver
time            37/udp   timserver
rlp             39/udp   resource   # resource location
nameserver      42/tcp   name       # IEN 116
whois           43/tcp   nicname
domain          53/tcp   nameserver # name-domain server
domain          53/udp   nameserver
mtp             57/tcp              # deprecated
bootps          67/tcp              # BOOTP server
bootps          67/udp
bootpc          68/tcp              # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp              # Internet Gopher
gopher          70/udp
rje             77/tcp   netrjs
finger          79/tcp
www             80/tcp   http       # WorldWideWeb HTTP
www             80/udp              # HyperText Transfer Protocol
link            87/tcp   ttylink
kerberos        88/tcp   krb5       # Kerberos v5
kerberos        88/udp
supdup          95/tcp
# 100 - reserved
hostnames       101/tcp  hostname   # usually from sri-nic
iso-tsap        102/tcp  tsap       # part of ISODE.
csnet-ns        105/tcp  cso-ns     # also used by CSO name server
csnet-ns        105/udp  cso-ns
rtelnet         107/tcp             # Remote Telnet
```

```
rtelnet                107/udp
pop2                   109/tcp    postoffice # POP version 2
pop2                   109/udp
pop3                   110/tcp               # POP version 3
pop3                   110/udp
sunrpc                 111/tcp
sunrpc                 111/udp
auth                   113/tcp    tap ident authentication
sftp                   115/tcp
uucp-path  117/tcp
nntp                   119/tcp    readnews untp          # USENET News -
                                                         # Transfer Protocol
ntp                    123/tcp
ntp                    123/udp                           #    Network    Time
Protocol
netbios-ns             137/tcp                           #    NETBIOS    Name
Service
netbios-ns             137/udp
netbios-dgm            138/tcp                           #   NETBIOS   Datagram
Service
netbios-dgm            138/udp
netbios-ssn            139/tcp                           # NETBIOS session
                                                         # service
netbios-ssn            139/udp
imap2                  143/tcp                           # Interim Mail Access
Proto v2
imap2                  143/udp
snmp                   161/udp                           # Simple Net Mgmt
                                                         # Proto
snmp-trap              162/udp    snmptrap               # Traps for SNMP
cmip-man               163/tcp                           # ISO mgmt over IP
                                                         # (CMOT)
cmip-man               163/udp
cmip-agent             164/tcp
cmip-agent             164/udp
xdmcp                  177/tcp                           # X Display Mgr.
                                                         # Control Proto
xdmcp                  177/udp
nextstep               178/tcp    NeXTStep NextStep      # NeXTStep window
nextstep               178/udp    NeXTStep NextStep      # server
bgp                    179/tcp                           # Border Gateway
                                                         # Proto.
bgp                    179/udp
prospero               191/tcp                           # Cliff Neuman's
                                                         # Prospero
prospero               191/udp
irc                    194/tcp                           # Internet Relay Chat
irc                    194/udp
smux                   199/tcp                           # SNMP Unix
                                                         # Multiplexer
smux                   199/udp
at-rtmp                201/tcp                           # AppleTalk routing
at-rtmp                201/udp
at-nbp                 202/tcp                           #   AppleTalk   name
binding
at-nbp                 202/udp
at-echo                204/tcp                           # AppleTalk echo
at-echo                204/udp
at-zis                 206/tcp                           # AppleTalk zone
                                                         # information
at-zis                 206/udp
z3950                  210/tcp    wais                   #     NISO     Z39.50
database
z3950                  210/udp    wais
```

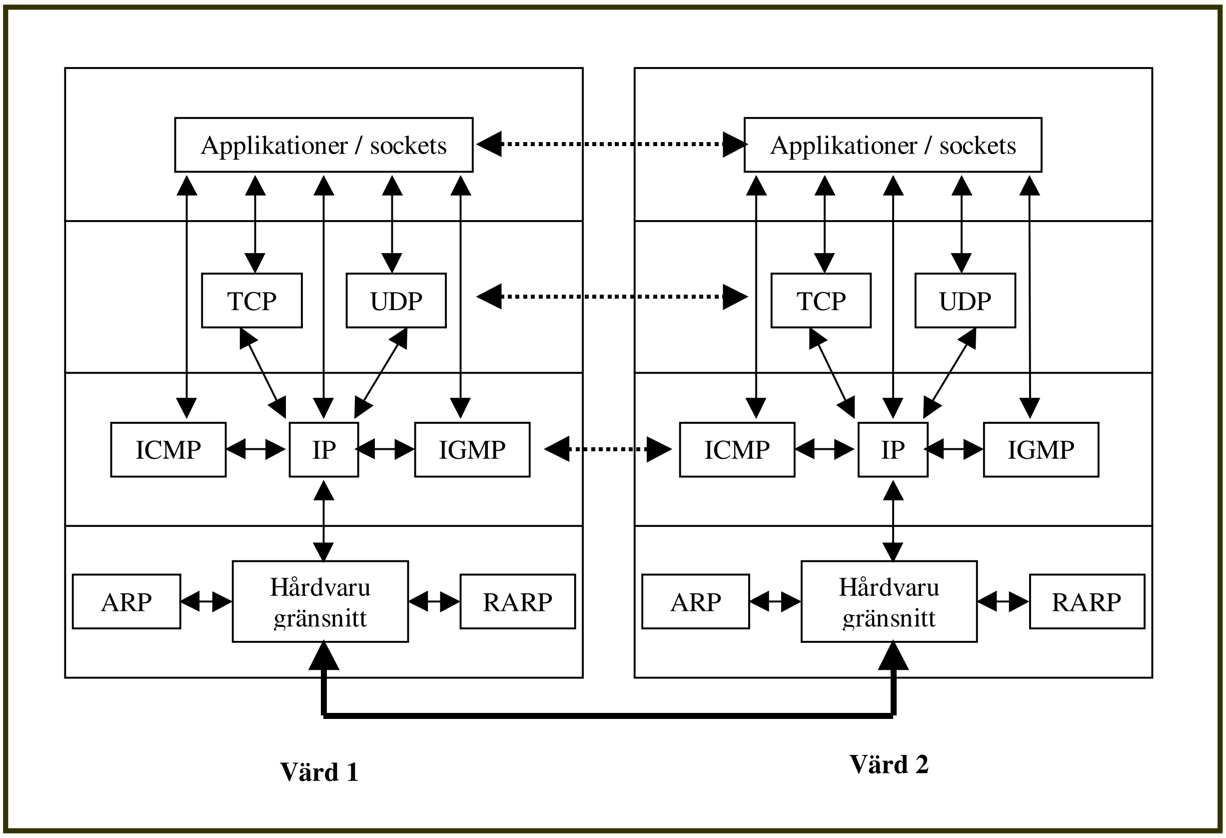
```
ipx                  213/tcp                           # IPX
ipx                  213/udp
imap3                220/tcp                           # Interactive Mail
                                                       # Access
imap3                220/udp                           # Protocol v3
ulistserv            372/tcp                           # UNIX Listserv
ulistserv            372/udp
#
# UNIX specific services
#
exec                 512/tcp
biff                 512/udp    comsat
login                513/tcp
who                  513/udp    whod
shell                514/tcp    cmd                    # no passwords used
syslog               514/udp
printer              515/tcp    spooler                #     line     printer
spooler
talk                 517/udp
ntalk                518/udp
route                520/udp    router routed          # RIP
timed                525/udp    timeserver
tempo                526/tcp    newdate
courier              530/tcp    rpc
conference 531/tcp              chat
netnews              532/tcp    readnews
netwall              533/udp                           # -for emergency
                                                       # broadcasts
uucp                 540/tcp    uucpd                  # uucp daemon
remotefs             556/tcp    rfs_server rfs         # Brunhoff remote
                                                       # filesystem
klogin               543/tcp                           # Kerberized `rlogin'
                                                       # (v5)
kshell               544/tcp                           # Kerberized `rsh'
                                                       #(v5)
kerberos-adm         749/tcp                           # Kerberos `kadmin'
                                                       # (v5)

webster              765/tcp                           # Network dictionary
webster              765/udp
#
# From ``Assigned Numbers'':
#
#> The Registered Ports are not controlled by the IANA and on most systems
#> can be used by ordinary user processes or programs executed by ordinary
#> users.
#
#> Ports are used in the TCP [45,106] to name the ends of logical
#> connections which carry long term conversations.  For the purpose of
#> providing services to unknown callers, a service contact port is
#> defined.  This list specifies the port used by the server process as its
#> contact port.  While the IANA can not control uses of these ports it
#> does register or list uses of these ports as a convienence to the
#> community.
#
ingreslock           1524/tcp
ingreslock           1524/udp
prospero-np          1525/tcp             # Prospero non-privileged
prospero-np          1525/udp
rfe                  5002/tcp             # Radio Free Ethernet
rfe                  5002/udp             # Actually uses UDP only
#
#
# Kerberos (Project Athena/MIT) services
```

```
# Note that these are for Kerberos v4, and are unofficial.  Sites running
# v4 should uncomment these and comment out the v5 entries above.
#
#kerberos           750/udp    kdc        # Kerberos (server) udp
#kerberos           750/tcp    kdc        # Kerberos (server) tcp
krbupdate           760/tcp    kreg       # Kerberos registration
kpasswd             761/tcp    kpwd       # Kerberos "passwd"
#klogin             543/tcp               # Kerberos rlogin
eklogin             2105/tcp              # Kerberos encrypted rlogin
#kshell             544/tcp    krcmd      # Kerberos remote shell
#
# Unofficial but necessary (for NetBSD) services
#
supfilesrv          871/tcp               # SUP server
supfiledbg          1127/tcp              # SUP debugging
```

Det finns ett alternativ till TCP. Det kallas UDP (eng. *User Datagram Protocol*). UDP är ett opålitigt, kopplingslöst protokoll som används för små datamängder som inte kräver stor tillförlitlighet. UDP delar datameddelanden i datagram som skickas ut på nätet oberoende av varandra.

IP (eng. *Internet Protocol*) är ett kopplingslöst, icke-tillförlitlig protokoll. Det garanterar ej att IP-datagram kommer fram utan dess enda uppgift är att skicka eller ta emot datagram i nätverket. Tillförlitligheten måste tillhandahållas av högre skikt dvs. TCP. IP har till sin hjälp två andra protokoll: ICMP (eng. *Internet Control Message Protocol*) och IGMP (eng. *Internet Group Message Protocol*). ICMP används av IP för att skicka felmeddelanden, kontrollmeddelanden och informationsmeddelanden ut till IP-skiktet på andra värdar. IP och ICMP är nära släkt med varandra: IP använder ICMP för att kunna styra dataflödet mellan två värdar på IP-nivå och ICMP använder IP för att kunna skicka sina meddelanden. IGMP kopplas samman med multicasting. Multicasting är en teknik att skicka samma data till en grupp värdar. Grupptillhörigheten är dynamisk dvs. värdar väljer själva när och hur länge de skall tillhöra en grupp. Interaktiva konferenser är en tillämpning där IGMP brukar användas.



TCP/IP-modellen

IP-protokollet är stark relaterad till en värds IP-adress. IP-adressen är ingeting annat än en 32-bitar stort tal som är unik för varje värd inom Internet och används för att kunna identifiera värden. Den delas upp i fyra fält som är 8 bitar stora (1 byte eller 1 oktett), där varje fält skrivs som ett decimalt tal, t.ex. 194.47.130.1. Första talet i IP-adressen räknas som högsta byten och den sista talet är den lägsta byten. Alla IP-adresser kombinerar ett nätverksnummer och ett värdnummer. Med små undantag, alla fält kan innehålla tal mellan 1 och 254. Talet 255 är reserverad för broadcast dvs. IP-datagram som använder en sådan adress (t.ex. 194.47.130.255) går till alla värdar inom ett LAN. En IP-adress där värdnumret är 0 preciserar den lokala (nuvarande) värd. Om istället nätverksnumret är 0 menas det lokala (nuvarande) nätverk.

Fyra IP-adresser är reserverade. Nätverksadress 0.0.0.0 är den förvalda vägen genom nätverket. Alla paket som skall till en för nuvarande nätverk okänd adress skickas till adress 0.0.0.0 som är kopplad till en gateway. Nätverksadressen 127.0.0.0 är reserverad för lokal trafik på en värd. Vanligen brukar adress 127.0.0.1 kopplas tillsammans med en enhet som kallas för *loopback interface*. Alla paket som kommer från övre skikt och skall till 127.0.0.0 kommer att returneras precis som om de precis hade kommit utifrån från nätverket. Loopback gränssnittet används för att testa nätverket.

Det finns fem klasser av IP-adresser.

Klass A nätverksadresser (**A-net**) sätter första biten i den högsta byten till 0. Första byte används därmed till att beteckna nätverksnummer. Det kan finnas bara 125 (128 stycken minus nätverksnummer 0, 127 och 255) klass A nätverk. Varje sådan nätverk teoretisk kan ha upp till 2^{24} värdar (där alltså värdar som har har nummer 0 eller 255 i någon av värdfälten räknas bort).

Klass B nätverksadresser (**B-net**) sätter de första två bitarna i den högsta byten till 10. De första två byte används till att identifiera nätverksnummer. Detta gör att det kan finnas upp till 16 384 - 5 (2^{14} - 5 reserverade nummer) sådana nätverk. Varje nätverk kan ha upp till 65 534 värdar.

Klass C nätverksadresser (**C-net**) sätter de första tre bitarna till 110. För nätverksnummer används 21 bitar och för värdar 8 bitar vilket resulterar i 2^{21} - 3 nätverk med 254 värdar var.

Klass D nätverksadresser används för multicast adresser.

Klass E nätverksadresser är reserverade för framtida användning.

Den intresserade läsaren kan hitta en fullständigt beskrivning av ovanstående protokoll i RFC 791 (IP), RFC 792 (ICMP), RFC 1054 (IGMP), RFC 793 (TCP) och RFC 768 (UDP).

RFC (eng. *Request for Comments*) dokument är Internet protokollstandarder. Vilken person som helst som skapar ett nytt protokoll eller bygger på ett nuvarande kan skicka in sitt förslag. Det får ett nummer och sedan finns tillgängligt för alla andra som vill läsa det. När ett sådant protokoll hunnit mogna blir den accepterad som en Internet standard. En lista med RFC som tas upp i denna dokument finns i kapitel 6 *RFC-lista*.

5.3.1 Datalänkskiktet

Alla datorer kopplade inom ett LAN, med undantag för de som använder SLIP/CSLIP/PPP/PLIP som skall förklaras senare, har minst ett nätverkskort. Den senaste tiden, nätverkskort av *Ethernet* typ har blivit mycket populära. De är relativt billiga, stöds väl av TCP/IP-protokollet och arbetar med hastigheter mellan 10 och 100 Mbit/s. Alla Ethernetkort har en egen 48-bitars adress, s.k. Ethernetadress. Adressen är unik, två kort får absolut inte ha samma adress. Den skrivs som en sekvens av två siffriga tal t.ex. **a7:32:ff:5c**.

Det finns även andra typer av nätverkshårdvara, t.ex. Token Ring, Token Bus, FDDI (eng. *Fiber Distributed Data Interface*) och DQDB (eng. *Distributed Queue Dual Bus*) men det är huvudsakligen Ethernet som tas upp här.

Datalänkskiktet innehåller förutom en hårdvarugränssnitt mot nätverkshårdvaran, två protokoll: ARP och RARP.

ARP (eng. *Address Resolution Protocol*) används för att mappa en IP-address mot motsvarande Ethernetadress. För att göra det används ett Ethernet *broadcast* meddelande som innehåller IP-addresen som skall översättas till en Ethernetadress plus det egna Ethernetadressen som gör möjligt för de andra värdarna att skicka ett svar tillbacka. Broadcast tekniken innebär att alla värdar inom ett LAN måste läsa alla Ethernetramar markerade som broadcast. Värdarna jämför IP-adressen i ramen med deras egen och om de är identiska skickas ett svar som innehåler värdens Ethernetadress tillbacka till värden som frågade. Ethernetadressen lagras i ett cache minne för att kunna återanvändas senare. Cache minnet rensas med jämna mellanrum för att få informationen uppdaterad ifall någon värd har t.ex. bytts ut.

RARP (eng. *Reverse Address Resolution Protocol*) fungerar precis tvärtom. Värden vet sin egen Ethernetadress (det gör den alltid!) men inte IP-adressen. För att ta reda på den, värden skickar ut en broadcast meddelande som innehåller Ethernetadressen tillsammans med en förfråga om att få den översatt till en IP-adress. De andra värdarna inom det lokala nätverket läser meddelandet och om de hittar en post i sin cache minne som motsvarar Ethernetadressen, de skickar ut svarsmeddelanden som innehåller IP-adressen till värden som frågade. RARP-protokollet används speciellt då en disklös maskin skall boota sitt operativsystem. Hur det görs beskrivs av BOOTP-protokollet och kommer ej att beskrivas här.

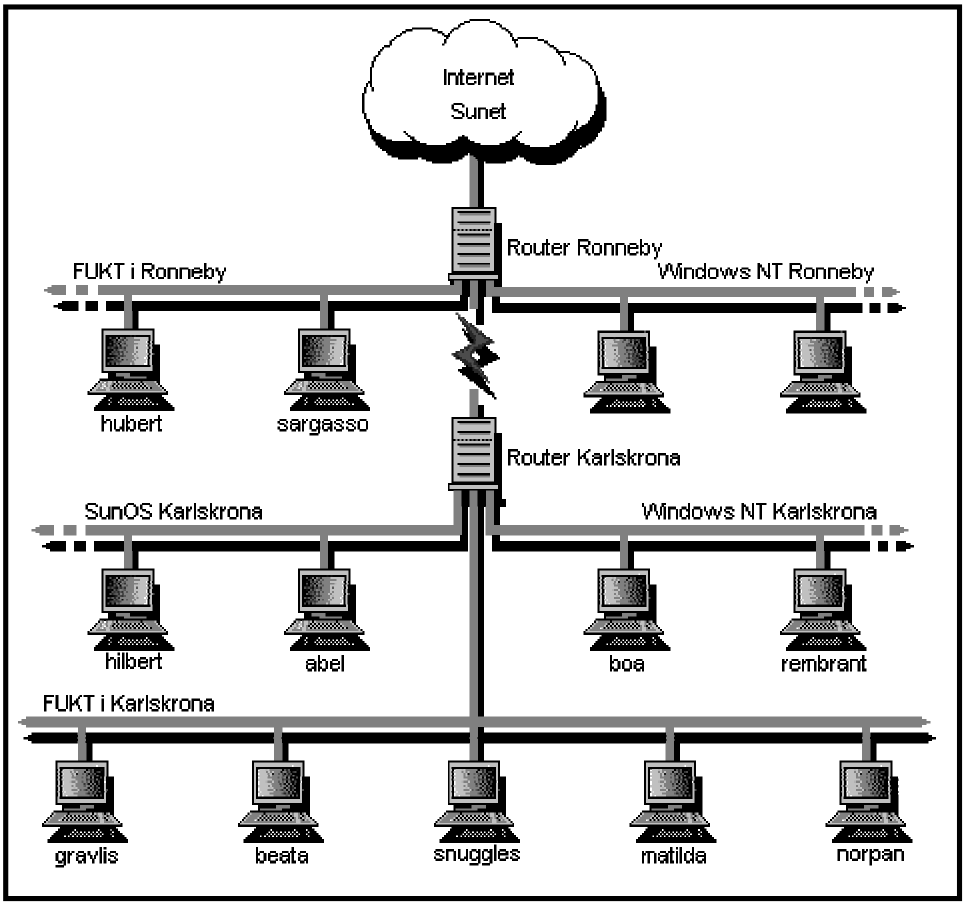
Specifikationer till ovanstående protokoll hittas i RFC 826 (ARP), RFC 903 (RARP) och RFC 1497 (BOOTP).

5.3.2 Att hitta väg genom Internet

En fråga som många Internet användare ställer sig är hur deras kommunikationsprogram tar reda på vägen fram till en datorvärd. Svaret är DNS (eng. *Domain Name System*).

En datorvärd (från ett nätverk) som vill ansluta sig till Internet behöver veta egentligen bara tre saker: sin egen IP-adress, IP-addressen till en gateway[1] och IP-adressen till en DNS-server (också kallad namnserver). DNS-serverns IP-adress är egentligen inte absolut nödvändigt. DNS-servern används bara att översätta till och från IP-adresser till namn.

Antag nu att FUKT:s server i Karlskrona som heter **snuggles.fukt.hk-r.s**e vill koppla sig mot en UNIX-arbetsstation i Ronneby som heter **mir.rby.hk-r.se** och hämta en fil. Observera nu att punkterna som skiljer de olika orden i namnen har absolut ingeting att göra med IP-adressen utan de enbart skiljer åt olika element. Varje element kallas en etikett (eng. *label*) och ger lite information om värden och dess nätverk. Exempelvis är **snuggles** maskinnamnet, **fukt** anger att den tillhör FUKT:s domän (nätverk) som i sin tur tillhör Högskolan i Karlskrona-Ronneby, **.hk-r**, som finns i Sverige, **.se**. Arbetsstationen i Ronneby heter **mir**, den finns i Ronneby, **.rby**, tillhör skolans nät **.hk-r** och den finns i Sverige, **.se**. Standarden säger att det fullständiga namnet **bör** se ut på följande sätt: *maskin.institution.typ.land*. Land fältet är namnet på landet i ISO 3166 notation (t.ex. .se för Sverige, .no för Norge o.s.v.). Vidare kan maskin fältet uttryckas som: *dator.avdelning*. Trots standarden finns ett stort mängd avvikelser.



Nätverkstopologi på Högskolan i Karlskorna / Ronneby

[1] En gateway är en värd som tilldelats flera giltiga IP-adresser och används för att skicka packet mellan olika typer av nätverksteknologier, t.ex. mellan ett Ethernetnät och ett FDDI-nät. Definitionen missbrukas ofta när den används generellt i sammanhang då det pratas om maskiner med två eller fler nätverkskort, men vilka binder ihop nätverk av samma typ. Dessa heter vanligen bryggor eller routrar.

För att **snuggles.fukt.hk-r.se** skall kunna påbörja kommunikation med **mir.rby.hk-r.se** behövs IP-adressen till **mir**-datorn. **Snuggles** använder DNS-serverns IP-adress och skickar denne en förfrågan efter IP-adressen för **mir.rby.hk-r.se**. Tillsammans med förfrågan skickar **snuggles** sin egen IP-adress. När IP-datagrammet hamnar i **snuggles** datalänkskikt, Ethernet kortet upptäcker att det finns ingen Ethernetadress i sin cache som motsvarar destination IP-adressen. Ethernet kortet använder sig då av ARP-protokollet och skickar ut en broadcast förfråga om att få IP-adressen översatt. Förfrågan är en vanligt Ethernetram där vissa fält fyllts med speciella värden. Alla Ethernet kort som kan översätta IP-adressen till en Ethernetadress, dvs. de har den i sin cache, svarar. Under förutsättningen att **snuggles** saknar även egen IP-adress (vilket skulle tyda på en extrem slarvig system administrator) kan RARP-protokollet användas. När Ethernetkortet på **snuggles** har fått Ethernetadressen till DNS-serverns nätverkskort sänder den sin förfråga. Namnservern letar i sin egen databas efter IP-adressen till **mir**. Hur den gör det tas upp lite längre fram. Om den hittar ingen post för **mir** ett felmeddelande skickas med hjälp av ICMP. Annars svarar den på **snuggles** förfrågan och sänder tillbacks **mir**:s IP-adress.

Nu kan **snuggles** kontakta **mir**. Det återstår bara ett problem. Vi förutsätter ju att **snuggles** vet ingeting om nätverket förutom de tre viktiga IP-adresserna. Det betyder att **snuggles** har ingen aning om **mir** ligger inom samma nätverk eller ej. Om **mir** finns inte inom samma nätverk kan den ju inte nås av ett Ethernetram utan det är nödvändigt att en gateway tar hand om IP-datagramen och skickar dem till rätt nätverk. **Snuggles** beslutar därför att skicka IP-datagramet med förfrågan om uppkoppling direkt till sin gateway. Den kommer att få hela ansvaret att data kommer fram till destination. Men **snuggles** Ethernetkort kan givetvis inte Ethernetadressen till **mir**. Därför kommer det att skicka ut en broadcastförfrågan precis som ovan. När slutligen gateway datorn får IP-datagramet skickar den vidare till en gateway i Ronneby. Den sänder sedan vidare datagramet till **mir**. IP-skiket på **mir** läser datagrammets huvud för att ta reda på källa och destination och den upptäcker att datagramet är adresserad till sig själv. Den tar bort då datagrammets huvud och skickar resten till TCP-skiktet. Mjukvaran där läser TCP-huvudet och ser att det är port 21 (ftp) som skall få tillgång till data. FTP-servern använder sig sedan av en liknande procedur för att bekräfta att förbindelsen är öppen.

Det här exemplet visar att TCP/IP-system kan klara sig med väldigt lite information. Dock väljer de flesta systemansvariga att lägga till mer information i form av vägval tabeller (eng. *routing tables*), netmask och annat information som ger en robustare system och minskar onödigt trafik och fel i nätverket. Dessutom kräver många program för att kunna fungera tillfredsställande att denna information har definierats.

5.3.3 Lägga till stöd i kärnan för nätverkskommunikation

Alla program som använder TCP/IP-kommunikation anropar färdiga funktioner, mestadels genom sockel gränssnittet. Sockel gränssnittet i sin tur anropar funktioner som finns programmerade in i kärnan. Men UNIX system låter sina användare som har root rättigheter att konfigurera vilka funktioner, eller rättare sagt vilket hårdvarustöd kärnan skall innehålla. Det enklaste är väl då alla funktioner är med, men då skulle kärnan växa onödigt stort. En kärna som innehåller stöd för alldeles för mycket hårdvara är mycket sårbar för fel. Procedurer som styr hårdvara kan börja reagera med varandra på ett oönskad sätt. Därför bör bara de nödvändiga funktionerna väljas. Det ger inte bara en säkrare kärna, utan en snabbare och minnessnålare.

Här kommer kärnkompilering inte att gås genom i detalj. Det är tradition att det skall finnas en README fil i källkodens katalog som förklarar hur kärnan skall kompileras. Däremot kommer kapitlet att koncentrera sig på de olika valmöjligheter för kommunikation som kommer upp vid kärnkonfigurering.

Innan kompilering läs noga README filerna. Oftast när något inte fungerar som det skall finns svaret i README filerna.

Vid kärnkompilering, i stort sett, behöver du bara bytta katalog till katalogen som innehåller källkoden och där köra i följande ordning:

make clear
make config
make zImage
mv /vmlinuz /vmlinuz-old
cp arch/i386/boot/zImage /vmlinuz

När **make config** körs ställs olika parametrar in som avgör vilka filer som tas med vid kompileringen. Nedan kommer **bara** att tas upp konfigurationsparametrarna för nätverkskommunikation.

Networking support (CONFIG_NET) [y]

Den här frågan avgör om överhuvudtaget något stöd för nätverk skall installeras. Svaret bör vara positivt, **y**, vilket är också det förvalda svaret, den mellan hakparanteser.

Lite senare, under rubriken **Networking options** kommer följande frågor upp:

TCP/IP networking (CONFIG_INET) [y]

Här frågas om stöd för TCP/IP skall läggas till. Självklart svaras ja till den här frågan om Internet kommunikation önskas.

IP forwarding/gatewaying (CONFIG_IP_FORWARD) [n]

Om maskinen kommer att användas som brygga[1]/router[2]/gateway måste den här funktionen finnas med i kärnan.

IP multicasting (CONFIG_IP_MULTICAST) [n]

Den här parameterns skall väljas om värden kommer att användas som server för tillämpningar som använder multicasting, t.ex. interaktiva konferenser.

IP firewalling (CONFIG_IP_FIREWALL) [n]

En *firewall* är en värd som kopplar ihop två nätverk. Den låter användare på den lokala nätverket komma ut på Internet men blockerar alla användare från Internet att logga in på systemet. Används ofta på företag som skydd mot hackers. Den här parametern bör inte användas tillsammans med **IP forwarding/gatewaying** eftersom risken för säkerhetsläckor ökar dramatisk då.

IP accounting (CONFIG_IP_ACCT) [n]

När **IP accounting** används, allt IP-kommunikation övervakas och loggas. Sedan kan speciella program behandla informationen och dra slutsatser om nätvekts egenskaper och ställa upp statistik. Den här parametern bör användas bara om nätverksanalys önskas.

PC/TCP compatibility mode (CONFIG_INET_PCTCP) [n]

PC/TCP är en kommersiell implementation av TCP/IP för DOS-maskiner. Tyvärr är den inte helt kompatibel med UNIX TCP/IP protokollet. Om den här parameter väljs kommer Linux fortfarande att kunna kommunicera

[1] En brygga är en maskin som agerar som förbindelse mellan två nätverk av samma typ. Den gör ingeting annat förutom att skyffla över IP-datagram till den andra nätverket.
[2] Till skillnad från en brygga, routern måste ta egna beslut om en IP-datagrams väg genom nätverket. Den baserar sina beslut på en mängd olika faktorer såsom nätverksbelastning, väglängd och tjänstkvalitè.

med andra UN*X[1] maskiner, men dess uppförande med slöa uppkopplingar kan skadas. Välj den här parametern bara om problem uppstår vid kommunikation med DOS-miljöer.

Reverse ARP (CONFIG_INET_RARP) [n]

Den här parametern aktiverar RARP-protokollet. Använd den här funktionen bara om disklösa klienter eller X-terminaler kommer att användas.

Assume subnets are local (CONFIG_INET_SNARL) [y]

Antag att ett företag är glad innehavare till ett C-nät med nätverksadress 194.47.147.0. Företaget består av tre olika avdelningar som var och en har sitt eget rum. Det finns två möjligheter att koppla ihop värdarna till ett nätverk: antingen kopplas allihopa direkt till C-nätet eller flera subnät kan skapas. Subnät skapas genom att använda en eller fler bitar av värdadressen till interna nätnummer. Antag nu att ovannämda företag bestämmer sig för att skapa tre subnät, en för varje avdelning. Det betyder att den system ansvarige tar de första två mest signifikanta bitarna i värdadressen och bestämmer sig att avdelning 1 får subnet 0 eller nätverksadress 194.47.147.0, avdelning 2 får subnet 1 med nätverksadress 194.47.147.64 och avdelning 3 får subnet 2 med nätverksadress 194.47.147.128. Kvar finns nätverksadress 194.47.147.192 som kan sparas till framtida ändamål. Varje subnet kan koppla upp till 63 värdar. Värdar utanför nätverk 194.47.147.0 vet ingeting om den interna indelningen. Det är gateway-maskinen på företagets nätverk som måste se till att varje datapaket kommer till rätt C-net.

När alla subnet förutses vara lokala kommer kärnan att anta att alla subnet kan nås med Ethernetramar och således använda största möjliga storlek som tillåts av Ethernet på datapaketerna. Om SNARL variabeln aktiveras ej, **n**, då kommer kärnan att anta att alla subnet som den inte har ett gränsnitt till är inte lokala och storleken på datapaketen kan komma att sänkas.

Disable NAGLE algorithm (normally enabled) (CONFIG_TCP_NAGLE_OFF) [n]

Telnet program och dylika har en tendens att skapa något som kallas för *tinygrams*. De är datapaket som innhåller bara ett litet antal tecken. På länkar med liten bandbredd detta kan bli lätt slösaktig. Nagles algorithm försöker lösa det genom att samla ett större antal tecken i en buffer minne och först därefter sända dem över till den andra värden.

The IPX protocol (CONFIG_IPX) [n]

IPX protokollet är Novells transport protokoll. Används ofta av spel som kan spelas över nätverk.

Strax därefter skall nätverksenheterna konfigureras.

*****
*** Network device support**
*****

Network device support? (CONFIG_NETDEVICES) [y]

Givetvis skall svaras **y** här om nätverksenheter kommer att användas.

Dummy netdriver support (CONFIG_DUMMY) [y]

Dummy netdriver är ett loopback gränssnitt som är ganska användar. Bör alltid tas med.

[1] UN*X system är operativsystem som fungerar ungefär som riktiga UNIX system men kan eller vill ej betala licens avgifter för att kalla sig UNIX™ system.

SLIP (serial line) support (CONFIG_SLIP) [n]

SLIP (Serial Line Internet Protocol) är ett protokoll som används då användare kopplar sig med modem till ett nätverk och vill använda TCP/IP. Användara får en IP-adress som är giltig bara så länge som användaren är kopplad till nätverket.

CSLIP compressed headers (CONFIG_SLIP_COMPRESSED) [y]

CSLIP är en version av SLIP som snabbar upp överföringen genom att komprimera huvudet på SLIP-paketen. CSLIP bör väljas framför SLIP då nätvervek man kopplar upp sig mot stödjer CSLIP.

16 channels instead of 4 (SL_SLIP_LOTS) [n]

CSLIP använder samtidigt bara 4 kanaler under normala förhållanden. Kanalantalet kan ökas till 16 men då föreligger risk för trafikstockning.

PPP (point-to-point) support (CONFIG_PPP) [n]

PPP (Point-to-Point Protocol) är ett protokoll som liknar (C)SLIP men avsevärt bättre. Den största fördelen med PPP är att den kan kapsla in andra pakettyper som Novells IPX och Appletalk. PPP bör väljas framför (C)SLIP.

PLIP (parallel port) support (CONFIG_PLIP) [n]

PLIP är nästan samma protokoll som SLIP med undantaget att PLIP använder sig av parallelporten istället för serieporten.

Kvar finns att välja stöd för nätverkskortet. Make-filen kommer att låta användaren välja sin/sina kort ur en lista på alla kort som stöds av kärnan.

Något som användaren inte skall glömma är att välja stöd för **/proc** filsystemet. Många nätverksapplikationer kräver /proc filsystemet för att kunna fungera alls.

5.4 DNS

5.4.1 Introduktion

När datorer pratar med varandra vill de gärna tilltala varandra med nummer. På Internet är det numret 32 bitar långt, eller mellan ett och (sisådär) fyra miljarder. Det är lättare för en dator att lagra siffror än namn bestående av bokstäver, men så är det inte för människor. T.ex. är det både lättare och trevligare att komma ihåg www.fukt.hk-r.se än IP-numret 194.47.147.34 som är adressen till FUKTs web-server i Karlskrona.
Därför utvecklades DNS (Domain Name System), som mappar mellan värddatorers namn som vi människor lättare kommer ihåg och internet adresser som datorn använder.
DNS är standard på Internet för att erhålla information om värddatorer, inte bara adresserna. DNS används av alla internetbaserade program, såsom e-post, telnet och ftp. DNS bygger på ett distribuerat databas system som har förmågan att hitta information på flera olika ställen. DNS ger namnservarna intelligensen att navigera genom databaserna och hitta data om godtycklig domän.

5.4.2 Historik

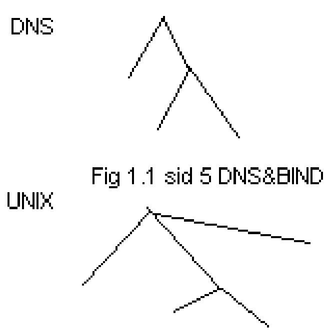
Under 1970-talet var ARPANET ett litet trevligt nätverkssamhälle med några hundra värddatorer. En ensam fil, HOSTS.TXT, innehöll all information som man behövde veta om dess värddatorer. Den innehöll en namn-till-adress översättning för varje värddator på ARPANET, filen hette HOSTS.TXT.

Filen underhölls av SRI’s Network Information Center (NIC) och distribuerades ifrån en ensam dator, SRI-NIC. ARPANETs administratörer skickade helt enkelt e-post med ändringar och använde ftp för att erhålla den aktuella HOSTS.TXT. Men allteftersom ARPANET växte blev detta sätt oeffektivt och till slut omöjligt. Storleken på HOSTS.TXT-filen växte proportionellt med antalet värddatorer. Arbetet med att hålla filen HOST.TXT uppdaterad blev mycket tidskrävande. Flera fel började inträffa, namnkollision, driftavbrott osv. När ARPANET sedan gick över till att använda TCP/IP-protokollet exploderade populariteten och ARPANETs styrelse tvingades tillsätta en grupp som skulle utveckla ett system som kunde ersätta HOST-TXT. Resultatet blev DNS.

5.4.3 DNS: Domain Name System

Domän Namn Systemet är en distribuerat databas. Det tillåter lokal kontroll av segmenten i den heltäckande databasen, data ifrån varje segment kan erhållas via ett klient-server schema. Stabilitet och ökad effektivitet uppnås med hjälp av spegling och caching.

DNS-strukturen är mycket lik strukturen av UNIXs filsystem. Databasen brukar liknas vid ett inverterat träd, med roten som topp. I UNIX betecknas roten med ett slash ”/”, i DNS betecknas roten med null (””), men när



man beskriver DNS används ofta punkt ”.”. Sid 5 DNS & BIND

5.5 NIS

5.5.1 Introduktion

NIS står för 'Network Information Service' och har tagits fram av SUN™. Först hette det Yellow Pages (YP) men namnet 'Yellow Pages' är ett registrerat varumärke hos 'the United Kingdom of British Telecom' och fick inte användas utan tillåtelse. Så SUN™ bytte namn till NIS.
Den nyaste versionen (v.3) heter NIS+ och har en hierarkisk struktur med förbättrad säkerhet. Om man inte har några speciella krav på extra hög säkerhet är valet mellan NIS och NIS+ enkelt, nämligen NIS. För NIS+ är mycket svårare och mer problematiskt för administratören att konfigurera. Dessutom är stödet för NIS+ hos Linux fortfarande under utveckling.

Syftet med NIS är att distribuera information som är viktig för hela nätverket. För ett av de stora problemen när man ska administrera ett UNIX-nätverk är underhållet av alla maskinernas konfigurationsfiler. NIS erbjuder ett distribuerat databassystem som låter flera datorer dela på lösenordsfiler, gruppfiler och andra globalt viktiga filer genom nätverket. Ur användarens synvinkel verkar det som om filerna är lokalt lagrade på varje maskin men i själva verket finns dom endast hos en maskin, en så kallad 'NIS master server' eller NIS huvudserver. De övriga datorerna i nätverket som använder NIS kallas NIS klienter och kan använda databaserna som är lagrade hos huvudservern som om de vore lokalt lagrade. Exempel på filer som kan distribueras av NIS är **/etc/passwd**, **/etc/group** och **/etc/hosts**. Genom att använda NIS kan en användare logga in på valfri maskin i nätverket och alltid känna igen sig. NIS ser till så att användaren hamnar i sin egen katalog med sina egna konfigurationsfiler. Och när någon ändring ska göras behövs endast huvudserverns konfigurationsfiler ändras.

Både NFS och NIS använder sig av protokollet ´Remote Procedure Call´ (**RPC**). Med RPC kan en maskin utföra ett processanrop och uppfatta det som om det utförs lokalt, men i själva verket utförs det på en annan maskin i nätverket. Vanligtvis saknar den maskin som utför anropet en resurs och tvingas därför att använda en annan maskins resurser. Denna distribution av tjänster skapar en klient/server relation mellan de två maskinerna. Maskinen som tillhanda håller resurserna kallas för server och den anropande som behöver resursen kallas klient. Resursen kan vara en central konfigurationsfil (NIS) eller ett delat filsystem (NFS).

Alla konfigurationsfiler som ska distribueras av en NIS-server lagras i databasfiler (s.k. mappfiler).
I ett nätverk kan det finnas flera NIS-servrar, men det är endast en som innehåller de ursprungliga datafilerna. Denna server är huvudservern och det är hos den mappfilerna genereras. Sedan kan det också finnas ´Slave NIS servers´ eller slavar vars enda uppgift är att serva klienternas anrop. Varje gång en NIS mapp är generad för att inkludera en ändring , distribueras den nya mappfilen till alla slavarna.

Om nu en maskin i nätverket skulle vilja använda filen **/etc/hosts** frågar klienten hos denna maskin en NIS-server om informationen istället för att använda den lokala filen. Det spelar ingen roll om klienten frågar huvudservern eller någon av slavarna, för alla servrar har en kopia av den korrekta mappfilen.

Om man vill installera en ny maskin i nätverket är det bara att uppdatera **/etc/hosts** filen hos huvudervern och generera en ny mappfil. Huvudservern skickar sedan ut den uppdaterade mappfilen till alla slavarna och när klienterna vill använda **/etc/hosts** erhåller de den aktuella informationen.

En NIS-server kan distribuera ett flertal typer av konfigurations och informationsfiler, man kan även definiera egna filtyper. Det är inte alltid önskvärt att alla maskiner i nätverket ska använda samma filer. Därför erbjuder NIS något som heter NIS-domäner. Det är viktigt att notera att en NIS-domän inte är samma sak som en Internetdomänen, använd därför inte samma namn på dessa. Ett nätverk kan ha flera NIS-domäner, det som skiljer dom är att de har olika uppsättningar av mappfiler. Alla maskiner som behöver samma mappfiler inkluderas i samma domän.

5.5.2 Installation

Det första man behöver göra innan man börjar med installation och konfiguration av NIS är att bestämma vilken maskin som ska vara NIS huvudserver och vilka som ska vara slavar. Eftersom klienterna hämtar nästan alla sina viktiga konfigurationsfiler från NIS, måste både huvudservern och slavarna ha hög åtkomst i avseendet bandbredd och svarstid. Man måste också tänka på att välja huvudservern så att den är lätt att komma åt att konfigurera. För det är ju hos huvudservern som alla konfigurationsfiler ligger som original. Det är också viktigt att välja maskiner som är stabila, för om en klient inte får svar på ett NIS-anrop är risken stor att den hänger sig. Det kan vara en bra ide att på huvudservern ha någon typ av administrationsprogram installerat.
NIS-versionen som vi valt att använda heter NYS och är en gratis variant av NIS, utvecklad av Peter Eriksson <pen@lysator.liu.se>. NYS är utvecklat för Linux men kan även köras under SunOS 4.1.x, Solaris 2.x, AIX, HP-UX, IRIS, Ultrix och OSF1 (alpha).
Programvaran för NYS kan hämtas från datorföreningen lysators ftp.
Hämta hem följande filer.

ftp://ftp.lysator.liu.se/pub/NYS/servers/ypserv-1.1.5.tar.gz
ftp://ftp.lysator.liu.se/pub/NYS/clients/yp-clients.tar.gz

För att snabbt komma igång följer här en guide för installationen[1].

Installation av programvara för NIS-servrarna.

- Packa upp serverpaketet.
  gzip -dc ypserv-1.1.5.tar.gz | tar -vxf -
  cd ypserv
- Läs **README** och **README.secure** filerna för mer information om vilka bibliotek som behövs. Läs även filen **INSTALL** för en mer detaljerad beskrining av installationen.
- Kör scriptet **./configure** som kontrollerar kompileringsmiljön och skapar en **Makefile.**
  Kolla igenom **Makefilen** och gör nödvändiga modifikationer.
- Kör **make**, som kompilerar NYS-servern.
- Kör **make install** som kopierar programmen till de rätta ställena (**ypserv** till **/usr/sbin/**).
- Gå till **./etc** och editera filerna **ypserv.conf** och **securnets** (serverkonfiguration och säkerhetsinställningar)**.**
  Kopiera **ypserv.conf** till **/etc** och **securnets** till **/var/yp/**
- Gör modifieringar i filen **/var/yp/Makefile**
- Sätt ditt NIS-domännamn**:**
  domainname infoserv (**infoserv** är ett förslag och kan väljas fritt)
- Generera NIS mappfiler**.**
  cd /var/yp
  make
- Om du tänker använda NIS-slavar i ditt nätverk kör:
  /usr/lib/yp/ypinit -m Endast hos NIS huvudserver.
  /usr/lib/yp/ypinit -s masterserver Endast hos NIS slavarna.
  Där **masterserver** är huvudservern.
  Du måste ta bort raden ‘ **NOPUSH = ”True”** ‘ i **/var/yp/Makefile** hos huvudservern, annars kommer huvudservern aldrig att skicka mappfilerna till slavarna. Se i filen **INSTALL** för mer info.
- Konfigurera servrarnas rc-filer så att både NIS domännamnen sätts och att serverprogrammet **ypserv** startar. Denna konfiguration skiljer sig mellan olika UNIX-distubtioner, men oftast gäller det rc-filer under katalogen **/etc/rc*.** Lägg till följande rader i lämplig **/etc/rc**-fil. Oftast **rc.local** (**/etc/rc.d/rc.local** i Linux).

```
domainname infoserv
if [ -f /usr/sbin/ypserv -a -d /var/yp/infoserv ]; then
        ypserv;        echo -n ´ ypserv ´
fi
```

[1] Denna guide är långt ifrån fullständig. Läs noga igenom dokumentationen som medföljer programpaketen.

Installation av programvara för NIS-klienterna.

- Packa upp klient-packetet.
  gzip -dc yp-clients.tar.gz | tar -vxf -
  cd yp.bin
- Läs igenom filen **README**, där står det lite mer utförligt om installation och konfiguration.
- Kompilera genom att köra **make**.
  Då kompileras progammen **ypbind**, **ypcat**, **ypmatch**, **yppoll**, **yppasswd**, **ypset** och **ypwhich**. Sedan kopieras de till sina rätta ställen (**/usr/bin** och **/usr/sbin**)
- Skapa filen **/etc/domainname** som innehåller namnet på NIS-domänen som ska användas.
  echo -n infoserv > /etc/domainname
- Konfigurera **/etc/rc*** filerna så att klienten startar upp vid boot. Lägg till följande rader i lämplig **/etc/rc**-fil. Notera att om du tänker köra klienten på huvudservern måste raderna exekveras efter det att **ypserv** har startat. Oftast gäller det filen **rc.local** (**/etc/rc.d/rc.local** i Linux).
  ```
  if [ -r /etc/domainname ] ; then
          domainname ´cat /etc/domainname´
  fi
  if [ -d /var/yp ] ; then
          /usr/sbin/ypbind ; echo -n ´ ypbind´
  fi
  ```
  Det är viktigt att domainname sätts innan du anropar **ypbind**.
- Inkludera även raden
  +:*:0:0:::
  i slutet av **/etc/passwd**, och
  +:*:0:
  i slutet av **/etc/group**.
- Glöm inte heller att skapa katalogen **/var/yp** (Används av **ypbind** för temporär lagring). Kontrollera även att **portmap** körs (används av RPC).
- För att klienten ska använda NIS när den letar efter maskiner i nätverket, lägger du till **nis** i filen **/etc/host.conf**.
  order hosts,nis,bind
  multi on
  Först kommer maskinen att kolla den lokala **/etc/hosts** sedan NIS och tillslut fråga DNS.
- Nu är det dags att leta upp NIS-servern. Om NIS-servern finns på det lokala nätet är det bara att starta **ypbind**. Den kommer att skicka ett ‘broadcast’ för att hitta en server.
  Du kan leta efter NIS servrar genom att skriva kommandot:
  rcpinfo -b 100007 2
  NIS-servrar kan sättas i filen **/etc/yp.conf** med raden
  ypserver masterserver slaveserver
  Där **masterserver** och **slaveserver** är namn på maskiner som kör **ypserv** (kan både vara huvudervern och slavar).
  ypbind kommer att testa de angivna servrarna och använda den server som svarar först.
  Om ingen av serverna svarar kommer den att börja skicka broadcast för att leta upp någon en annan server.

5.5.3 Konfiguration

I detta avsnitt ges en mer djupare beskrivning av konfigurationen. Låt oss säga att vi endast behöver en NIS-domän[1], vilken vi döper till **infoserv[2].**
Efter det att du har installerat NIS sätter du NIS-domännamnet genom att använda kommandot **domainname**.

 # domainname infoserv

NIS-domännamet sätts i någon av maskinens **rc**-filer som man kan hitta under katalogen **/etc**, dessa körs när systemet startas upp. I Linux finns raden: **domainname 'cat /etc/domainname'[3]** i filen **/etc/rc.d/rc.local**, men detta kan variera från system till system. Ta reda på var NIS-domännamnet sätts och sätt det exempelvis till **infoserv**. Ofta är domännamnet redan satt, detta är ofarligt om NIS inte körs.
Efter att ha satt NIS-domännamnet är det dags att gå igenom alla administrativa system filer, kontrollera att de bara innehåller den information du vill distribuera (varken mer eller mindre). Det är viktigt att systemet startar upp med korrekt information i mappfilerna. Vilka administrativa systemfiler NIS distribuerar varierar, de vanligaste visas i tabellen nedan.

Fil	Innehåll
/etc/bootparams	Information om disklösa maskiner.
/etc/ethers	Ethernet nummer (MAC adresser).
/etc/group	Användar grupper.
/etc/hosts	Maskinnamn och IP-aresser.
/etc/aliases	Alias och mailinglistor till e-post systemet.
/etc/netgroup	Nätgrupps definitioner (Används av NIS)
/etc/netmasks	Nätverks ”maskar”.
/etc/networks	Nätverksadresser.
/etc/passwd	Användarnamn, användarID och lösenord.
/etc/protocols	Nätverksprotokolls namn och nummer.
/etc/rpc	Remote procedure call program och nummer
/etc/services	Portnummer och tjäntenamn för nätverket.

Förutom **netgroup**, är alla dessa filer standard i UNIX. När sedan NIS körs kommer dessa att antingen ersättas eller kompleteras hos klienten. Men eftersom alla inte används är det inte säkert att de existerar.

Kontrollera så din **/etc/passwd** fil hos huvudservern **inte** innehåller raden:

+::0:0::

Denna rad används hos NIS-klienterna för att indikera att de vill inkludera information ifrån huvudserverns mappfiler i sina lösenords filer. Det går även att köra huvudservern som en klient och då måste denna rad finnas i **/etc/passwd**. Om så är fallet sätt då en asterisk (*) i lösenordsfältet så att denna ”+ användare” inte kan logga in.

+:*:0:0::

[1] Att skapa flera domäner skiljer sig inte mycket ifrån att endast ha en. Det enda man behöver komma ihåg är vilket maskiner som hör till vilken domän.
[2] Kom ihåg att NIS-domännamnet inte är samma sak som Internetdomämen.
[3] Initierar domännamnet genom att skriva följande: **echo -n infoserv > /etc/domainname**

När du har satt domänamnet och kontrollerat filerna som du vill distribuera är det dags att initiera huvudservern. Detta görs med verktyget **ypinit**. Se till att du är **superuser** och anropa **ypinit** med **-m** parametern.

newmaster# /usr/etc/yp/ypinit -m

ypinit skapar nu domänkatalogen för det domännamn du valt. I vårat exempel skapar **ypinit** katalogen **/var/yp/infoserv**, sedan genereras alla NIS mappfilerna i denna katalog. Den första mappen som genereras är **ypservers** mappen, **ypinit** kommer att fråga dig om de maskiner som kommer att vara NIS-slavar. Maskinerna som anges måste inte köra NIS för tillfället, men de bör göra det innan första modifikationen av mappfilerna görs.
Se till att det endast finns en NIS huvudserver per NIS-domän. Det finns inget som hindrar att det finns flera men det är inte att rekomendera. Det kan skapa förvirring när NIS ska uppdatera vissa mappfiler, t.ex. lösenords filen **passwd** hos huvudservern.
När **ypinit** är klart är det bara att köra igång NIS-servern, detta görs antingen manuellt eller vid uppstarten av systemet. För att **ypserv** ska starta vid boot, lägg förslagvis till följande rader i **/rc.local** underkatalogen **/etc/rc.d** (gäller både huvudservern och slavarna).

```
domainname infoserv
if [ -f /usr/sbin/ypserv -a -d /var/yp/infoserv ]; then
        ypserv;         echo -n ´ ypserv ´
fi
```

Notera att dessa rader kanske redan står i konfigurations filerna under katalogen **/etc/rc***, villkorssatsen **if** ser till så att NIS inte startar om katalogen **/var/yp/infoserv** inte existerar. Ett annat möjligt fall är att dessa rader är ´bortkommenterade’ med tecknet **#**, då är det bara att ta bort tecknet **#** som står framför dessa rader.

Nu är det dags att lägga till eventuella NIS-slavar, detta görs också med kommandot **ypinit**. Fast nu används parametern **-s**. Kör följande rad hos klienterna.

newslave# domainname infoserv
newslave# /usr/etc/yp/ypinit -s master

Där **master** är namnet på den maskin som är huvudserver för domänen. Innan kommandot körs ska NIS-slaven ha sitt domännamn satt med **domainname** och huvudservern för domänen måste vara igång.

När NIS-slaven är initierad kommer den att överföra mappfilerna från huvudservern och sedan skapa egna kopior i katalogen **/var/yp/infoserv**. Det är viktigt att komma ihåg att slavens mappfiler har genererats av informationen hos huvudservern, så om det finns viktig information i någon av slavens konfigurationsfiler måste denna information också finnas i huvudserverns konfigurationsfiler.

Man kan när som helst lägga till nya NIS-slavar, detta görs genom att uppdatera huvudserverns **ypservers**-mappfil.

master# ypcat -k ypservers > /tmp/ypservers
Lägg till den nya NIS-slaven i filen **/tmp/ypservers** och generera den nya mappfilen.
master# cd /var/yp
master# cat /tmp/ypservers | makedbm - /var/yp/infoserv/ypservers

Den nya slaven kommer nu att få alla mappfiler från huvudservern.
När huvudserverns **ypservers**-fil har blivit uppdaterad kan du köra **ypinit -s** på den nya slaven och sedan **ypserv** (för att starta NIS-tjänsten).

newslave# /usr/etc/yp/ypinit -s master
newslave# ypserv

Där **master** är namnet på huvudservern.

Nu när en eller flera NIS-servrar kör **ypserv** är det dags att få igång NIS-klienterna. Innan du börjar med klient-initeringen måste det minst finnas en fungerande NIS-server i nätverket, annars kommer klienterna att hänga sig.

När NIS körs kommer distributionen av filer att ske på två olika sätt.

- NIS byter ut vissa filer.
  De lokala kopiorna av följande filer kommer att bytas ut mot informationen i NIS-serverns mappfiler: **ethers, hosts, netmasks, netgroups, networks, protocols, rpc** och **services**.

- En del filer kommer att kompletteras.
  När maskinen vill ha information ifrån dessa filer kommer den först att kontrollera den lokala filen och sedan om den inte hittade informationen fråga NIS.
  T.ex. om en användare vill logga in på maskinen kommer den först att kontrollera den lokala lösenordsfilen **/etc/passwd** och om användaren inte finns där kommer den att kolla i NIS-serverns mappfil.

Bara för att NIS byter ut vissa filer kan man inte ta bort dessa hos klienterna, för maskinen använder vissa av dessa filer vi uppstart. T.ex. filen **/etc/hosts** används ända fram till det att **ypbind** startar och först då kommer den att ersättas av NIS.

För att sätta igång NIS-klienterna behöver man göra tre saker.

1. Försäkra dig om att maskinens konfigurations filer är satta till att använda NIS.
   När NIS kompletterar lokala konfigrationsfiler används plustecknet (+) som indikation. Det visar var NIS ska lägga in den kompilerade informationen.

/etc/group;	**ps-staff:*:20:stern,julie,peter** **sysadm:*:stren,johnc** **+:*:***
/etc/aliases:	**# end of local alias list** **last-local-alias george@bigfeet** **#insert NIS entries** **+**
/etc/bootparams	**# end of local bootparams list** **+**
/etc/passwd:	**root:si4NojF9Q8JqE:0:1:System Adm:/root:/bin/bash** **deamon:*:1:1::/:** **sys:*:2:2::/:/bin/csh** **bin:*:3:3::/bin:** **+:*:0:0:::**

   För att klienten ska använda NIS när den letar efter maskiner i nätverket, lägger du till **nis** i filen **/etc/host.conf**.

   order hosts,nis,bind
   multi on

   Först kommer maskinen att kolla den lokala **/etc/hosts** sedan NIS och tillslut fråga DNS

2. Sätt vilket domän maskinen ska tillhöra med kommandot **domainname**.
   Detta görs enklast genom att editera någon av maskinens **/etc/rc***-filer.
   Börja med att skapa filen **/etc/domainname**

   echo -n infoserv > /etc/domainname

   Lägg sedan till följande rader i lämplig **/etc/rc***-fil.

   ```
   if [ -r /etc/domainname ] ; then
           domainname ´cat /etc/domainname´
   fi
   ```

   Oftast gäller det filen **rc.local** (**/etc/rc.d/rc.local** i Linux).

3. Starta **ypbind** som ansvarar för lokalisering och uppkoppling mot NIS-servern.
   Lägg till följande rader i samma **/etc/rc***-fil som domännamnet sätts.

   ```
   if [ -d /var/yp ] ; then
           /usr/sbin/ypbind ; echo -n ´ ypbind´
   fi
   ```

   Observera att det är viktigt att **ypbind** startar efter det att domännamnet är satt och om huvudservern ska köras som klient måste **ypbind** startas efter **ypserv**.

   Om man vill specificera vilka servrar **ypbind** ska testa först lägger man till följande rad i filen **/etc/yp.conf**.

   ypserver masterserver slaveserver

   Där **masterserver** och **slaveserver** är namn på maskiner som kör **ypserv**.

   ypbind kommer att testa de angivna servrarna och använda den server som svarar först.

   Om man inte specificera några servrar eller om ingen av serverna svarar skickar **ypbind** ‘broadcast’ och använder sig av den svarar först.

Nu är frågan om NIS huvudserver också ska vara en klient. Ibland vill man kunna begränsa accessen till denna maskin. Men i den mesta litteratur om NIS rekommenderas att även köra huvudservern som en NIS-klient.
Flera typer av komplikationer kan inträffa om man inte kör huvudservern som klient. Bl.a. kan mail-systemet bli förvirrat om vissa användare inte finns på alla datorer. En god tumregel i detta fall är ‘Det kostar mer i komplexitet och flexibilitet än vad man tjänar i säkerhet’.

Mappfilerna som NIS distribuerar är lagrade i databaser med formatet DBM (DataBase Management).
När man genererar mappfilerna körs programmet **makedbm** som konverterar ASCII-filer till DBM. När man ska generera mappfilerna är det enklast att använda Makefilen som följer med serverpaketet, **/var/yp/Makefile**. Gör nödvändiga ändringar i denna och kör sedan **make**. (Observera att det är endast hos huvudservern som mappfilerna ska genereras.)

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

Nu genereras mappfilerna i katalogen som hör till domänen under **/var/yp/***. I vårat fall läggs mappfilerna i katalogen **/var/yp/infoserv**. Beroende på hur informationen söks används det flera olika mappfiler per informationsfil. T.ex. av filen **/etc/passwd** bildas både **passwd.byname** och **passwd.byuid** i katalogen **/var/yp/infoserv** i DBM-format. *.byname används för sökning när användarnamnet är känt och *.byuid när UID (användare ID) är känt.

Nu när både servrarna och klienterna är initierade är det dags att starta om maskinerna för att vara säker på att programmen startar korrekt, starta i ordningen: huvudserver, slavar, klienter.
Om maskinerna startade om utan felmeddelanden kan du hos klienter testa NIS med följande kommadon.

Listar passwd-filen ifrån huvudservern

clientmachine# ypcat passwd
user1:Iw53hKj8e3mw:500:500:User no one:/home/user1:/bin/bash
user2:Uif4fSfuOjd3e:500:500:User no two:/home/user2:/bin/bash

Visar maskinnamnet på NIS-servern

clientmachine# ypwhich
masterserver

Letar upp **user1** i passwd.byname

clientmachine# ypmatch user1 passwd
user1:Iw53hKj8e3mw:500:500:User no one:/home/user1:/bin/bash

Letar upp maskinen **hilbert** i hosts.byname

clientmachine# ypmatch hilbert hosts
194.47.141.22 hilbert hilbert.kna.hk-r.se

5.5.4 Säkerhet, nätgrupper och administration

Hos maskiner som använder NIS (klienter) konfigureras systemfilerna med ett plustecken (+). Plustecknet berättar för programmen som letar i systemfilerna att fråga NIS-servern efter resten av filen. T.ex. filen **/etc/passwd** hos en klient kan se ut på följande sätt:

root:si4NojF9Q8JqE:0:1:System Adm:/root:/bin/bash
+:*:0:0:::

När ett program hos klienten vill använda filen **/etc/passwd** för att verifiera ett lösenord kommer de att, om informationen inte finns lokalt, få lösenordet ifrån NIS.
Om du inte kör huvudservern som klient gäller att var noga med plustecknet endast finns hos klienterna. För hos maskiner som inte kör NIS betyder plustecknet inget speciellt utan kan tolkas som ett användarnamn. Försäkra dig om att rader som börjar med plus- eller minustecken inte finns i **/etc/passwd** filen hos maskiner som inte kör NIS.

Om raden ”**+::0:0:::”** finns i din **/etc/passwd** fil hos maskiner som inte kör NIS kan vem som helst logga in på din dator genom att skriva ett plus tecken vid login: promten. Du kan minimera denna säkerhetsrisk genom att alltid inkludera ett lösenord för ”plusanvändaren”. Använd följande rad hos klienterna (även hos servern om den körs som klient).

+:*:0:0:::

För om NIS servern skulle misslyckas att distribuera lösenordfilen finns risken att någon kan logga in genom att helt enkelt använda (+) som användarnamn. Därför bör du testa att logga in med användarnamnet ‘+’ för att vara säker på att systemet är rätt konfigurerat.
Om du följer följande exempel bör det inte vara några problem:

login: +
password: ***något***
Login incorrect

Om följande skulle inträffa har du ett problem:

login: +
Last login: Sat Mar 27 16:15 32 on ttya
#

Netgroups, är en del av NIS, som är till för att klassificera användare i ett NIS nätverk. Det liknar UNIXs användargrupper men är mycket mer komplicerat. Man kan använda **netgroups** ihop med NIS eller NFS för att specificera vem som får använda vissa tjänster. Genom att korrekt specificera nätgrupper kan man öka säkerheten hos systemet genom att begränsa individer och maskiners tillgång till kritiska resurser.
Databasen som innehåller nätgrupperna finns hos NIS servern i filen **/etc/netgroup** eller **/usr/etc/netgroup**.
Filen **netgroup** kan innehålla en eller flera rader som har syntaxen:

gruppnamn medlem1 medlem2 …

Medlemmar i varje grupp kan vara en värddator, en användare eller/och en domän:

(värddator, användarnamn, domännamn)

Om man inte inkluderar användarnamn blir alla användare hos den specificerade värddatorn medlemmar i gruppen. Om inte domännamnet är inkluderat antas den aktuella domänen.

OBSERVERA

NIS domännamnet är inte det samma som Internets domännamn.
Det är ett administrativt namn som används för gruppering. Du kan använda ditt Internet domännamn, men detta leda till säkerhetsrisker och bidra till komplikationer med vissa versioner av ***sendmail.***

T.ex. för att skapa en nätgrupp med namnet **Profs**, vilken består av användarna **nevcx** och **zamolxe** på maskinen **snuggles** i NIS domänen **infoserv** kan följande rad användas i filen **netgroup**.

Profs (snuggles,nevcx,infoserv) (snuggles,zamolxe,infoserv)

För att skapa en nätgrupp med namnet **Servers**, som inkluderar alla användarna på maskinerna **norpan**, **beata** och **gravlis**, kan följande rad användas.

Servers (norpan,,) (beata,,) (gravlis,,)

För att skapa en nätgrupp med namnet **SysAdm** som inkluderar alla användare med användarnamnet **adm** på alla maskiner, används raden.

SysAdm (,adm,)

För att skapa nätgruppen **Universal** som inkluderar alla användare på alla maskiner används raden.

Universal (,,,)

Ibland vill man begränsa accessen för vissa användare. Då kan det vara bra att ha en nätgrupp med dessa användare. Nätgruppen **Dangerous-users** innehåller alla användare med användarnamnen **hacker** och **cracker** på alla maskiner.

Dangerous-users (,hacker,) (,cracker,)

Ett annat sätt att konfigurera nätgrupper är att skapa grupper för varje avdelning eller institution. Följande rader visar hur man skapar en s.k. 'master group' med namnet **Science**.

Math (mathserve,,) (math1,,) (math2,,) (math3,,)
Chemistry (chemserve1,,) (chemserve2,,) (chem3,,) (chem1,,) (chem2,,) (chem3,,)
Biology (bioserve1,,) (bio1,,) (bio2,,) (bio3,,)
Science Math Chemistry Biology

Nätgrupper är viktiga för systemets säkerhet, för man kan använda dem till att begränsa användarnas och värddatorernas resurser.

Förutom nätgrupper kan access styras med plus- och minustecken framför enskilda användare.
Här följer några exempel på hur man med NIS kan reglerar access.

Lägger till hela NIS mappfilen för passwd.
/etc/passwd: **root:si4NojF9Q8JqE:0:1:System Adm:/root:/bin/bash**
+:*:0:0:::

Lägger endast till användarna **nevcx** och **zamolxe** från NIS-passwd
/etc/passwd: **+nevcx:*:**
+zamolxe:*:

Lösenord, **UID** och **GID** tas ifrån NIS mappfil medan " **President - Guest**" byts ut mot det som står i mappfilen.
Nollorna används som 'wildcards'.
+clinton:*:0:0:President - Guest:::

Inkluderar användarna i nätverkgruppen **SysAdm** till **/etc/passwd**.
+@SysAdm

Tar bort användaren **guest** och nätgruppen **Dangerous-users** från den lokala **/etc/passwd**.
-guest:*:523:500::
-@Dangerous-users

Om man vill ta bort användare och nätgrupper måste detta ske före alla tillägg.
T.ex. man vill ta bort användaren **johnc** som finns med i nätgruppen **trusted-users** men samtidigt tillåta access till resten av nätgruppen, måste användaren **johnc** tas bort först.
-johnc:*:12389:20::
+@trusted-users
+:*:0:0:::

När man använder **-användare** indikationen måste både **UID** och **GID** vara med.
-johnc:*:12389:20::
-@dangerous-users
+:*:0:0:::

Det enda sättet att få ett användarnamn att använda ett visst **UID** och **GID** istället för det i NIS-mappen är att inkludera en fullständig passwd-rad i den lokala **/etc/passwd**.
Denna rad sätter **sterns UID** till **1234** och **GID** till **10** även om annat anges i mappfilen. Obs ordningen!
stern:JdKs4RfsefshJ:1234:10:Hal Stern:/home/stren:/bin/csh
+:*:0:0:::

5.5.5 Underhåll

När man ska göra ändringar i nätverket är det bara att göra dessa i huvudserverns konfigurations filer och sedan generera nya mappfiler. Observera att detta måste göras hos huvudservern.
T.ex. om man vill lägga till en maskin, editerar men filen **/etc/hosts** och kör sedan **make** i katalogen **/var/yp**. Då generaras de nya mappfilerna och skickas sedan till eventuella slavar.

 # vi /etc/hosts
Lägg till nya maskiner
 # cd /var/yp
 # make

Överföringen av mappfilerna mellan huvudservern och slavarna sker med programmen **yppush** och **ypxfr**.
yppush finns hos huvudservern och används vid överföringa av mappfiler till slavarna.
ypxfr finns hos slavarna och hämtar mappfiler från huvuservern.

För att vara säker att alla slavarna fick alla mappfiler vid den senaste ändringen körs vissa script regelbundet som automatiskt uppdaterar mappfilerna. Scripten heter **ypxfr_1perhour** och **ypxfr_1perday**. Dessa läggs in i slavarnas **cron**-tabell. Programmet **crond** är standard i UNIX och erbjuder schemalagd exekvering av program. Scripten kan exempelvis se ut så här:

ypxfr_1perhour:
/usr/etc/yp/ypxfr passwd.byuid
/usr/etc/yp/ypxfr passwd.byname
/usr/etc/yp/ypxfr aliases.byname
/usr/etc/yp/ypxfr mail.aliases

ypxfr_1perday:
/usr/etc/yp/ypxfr services.byname
/usr/etc/yp/ypxfr protocols.byname

Och för att slavarna regelbundet ska hämta hem de aktuella mappfilerna läggs följande rader till i **/etc/crontab.**

0 * * * * ypxfr_1perhour (exekveras varje timme)
0 0 * * * ypxfr_1perday (exekveras varje dag)

För att kontrollera att slaven har hämtat mappfilerna regelbundet loggar **ypxfr** överföringarna i filen **/var/yp/ypxfr.log**

Om man vill kan man även schemalägga **yppush** så att huvudservern regelbundet skickar ut mappfilerna. Men detta är oftast onödigt eftersom det vanligaste felet vid distributionen av mappfilerna till slavarna är att slaven inte var tillgänglig vid tillfället då huvudservern skickade ut mappfilerna. Då är det bättre att slaven själv får ta hem dom när den kan.

En annan viktig detalj när man kör NIS är den när användaren själv vill ändra sitt lösenord. När man inte kör NIS sker detta med programmet **passwd.** Men **passwd** ändrar innehållet i filen **/etc/passwd** och om man då kör NIS är det inte säkert att användaren finns i den filen **/etc/passwd**. Därför använder sig NIS av programmen **yppasswdd**(server) och **yppasswd**(klient).
När en användare vill ändra sitt lösenord på en maskin som körs som NIS-klient använder han **yppasswd** som anropar programmet **yppasswdd** hos huvudservern. **yppasswdd** i sin tur uppdaterar både mappfilerna och **/etc/passwd** hos huvuservern och användaren ändrar sitt lösenord för hela NIS-domänen.

yppasswd ingår i klientpaketet för NYS, medan serverprogrammet yppasswdd kan hämtas från datorföreningen lysators ftp.

ftp://ftp.lysator.liu.se/pub/NYS/servers/yppasswdd-0.7.tar.gz

Följ instruktionerna som följer med programmetpaketet. Glöm inte att se till så att **yppasswdd** startat vi boot hos huvudservern. Förslagsvis startas programmet i **rc.local.**

5.6 NFS

NFS, står för Network File System, är ett transparent filsystem som ger access till hårddiskar genom ett nätverk. Precis som NIS underlättar NFS administrationsarbetet. Med hjälp av NFS behöver man endast installera och uppdatera program på en maskin, en s.k. NFS-server. De övriga maskinerna i nätverket kopplar upp sig mot denna server och använder den på samma sätt som en lokal hårddisk. NFS erbjuder också användarna att alltid hamna i sina egna hembibliotek.

NFS är precis som NIS uppbyggt av klienter och servrar, och använder sig också av protokollet ´Remote Procedure Call´ (**RPC**). En NFS-server är en maskin som har ett eller flera filsystem och gör dom tillgängliga i nätverket. En NFS-klient använder sedan kommandot **mount** för att komma åt de distribuerade filsystemen.

5.6.1 Konfiguration

Denna konfiguration av NFS är huvudsakligen baserad på Linux, men principen är den samma för de flesta UNIX-system. Det första man måste göra är att få igång en NFS server, för detta måste ett antal filer konfigureras och bakgrundsprogram startas.

<u>Konfiguration av NFS server</u>

- **The RPC Portmapper**

För att NFS ska kunna använda protokollet RPC måste programmet **/usr/sbin/rpc.portmap** startas. Vanligast är denna konfiguration redan gjord i någon av Linux-systemets **rc***-filer. Ibland kan raderna vara kommenterade, då är det bara att ta bort #-tecknet framför raderna och starta om systemet.
RPC portmapper är ett program som konverterar RPC programnummer till TCP/IP (eller UDP/IP) protokollets port nummer. Det måste köras för att RPC-anrop ska kunna utföras. När en server som använder RPC startar, berättar den för portmapper vilken port han avlyssnar och portmapper ser till att anropen kommer fram.

- **/etc/exports**

Nu är det god tid att bestämma vilka delar av filsystemet som ska distribueras. Denna konfiguration görs i filen **/etc/exports**. Låt oss säga att vi vill göra katalogen **/usr/X11R6/,** som finns på maskinen **snuggles,** åtkomlig ifrån maskinen **norpan**. Detta betyder att maskinen **snuggles** kommer agera server medan maskinen **norpan** är klient. Hos servern lägger vi till följande rad i **/etc/exports.**

```
#/etc/exports
/usr/X11R6/     norpan(rw)
```

Raden ovan ger maskinen **norpan** läs och skrivrättigheter i katalogen **/usr/X11R6/** hos servern **snuggles.** Om man inte vill att klienten endast ska ha läsrättigheter kan man byte ut **rw** (read/write) mot **ro** (read only).
Om man vill att fler maskiner ska få tillgång till denna katalog är det bara att skriva dom efter varandra i **/etc/exports**.

```
#/etc/exports
/usr/X11R6/     norpan(rw)     matilda(ro)     gravlis(rw)
```

Denna rad ger maskinerna **norpan** och **gravlis** läs och skrivrättigheter medan **matilda** endast får läsrättighet.
Det finns även andra parametrar än rw och ro, dessa står i **man**-filen för **exports.** Om man kör NIS kan man också använda **netgroups** istället för att skriva ut varje maskin.

- **Mountd** och **nfsd**

Nu är allt klart för att starta programmen **mountd** och **nfsd** (har ibland namnen **rpc.mountd** respektive **rpc.nfsd**). Båda kommer att läsa **/etc/exports**. Varje gång man gör ändringar i filen **/etc/exports** måste man se till både **mountd** och **nfsd** får reda på att filen har ändrats. Detta görs antingen men kommandot **exportfs** eller genom att starta om programmen **mountd** och **nfsd**.

För att vara säker på att **mountd** och **nfsd** är igång kan man använda kommandot **rcpinfo -p**.

```
snuggles: > rcpinfo -p
   program vers proto   port
   100000    2   tcp     111   portmapper
   100000    2   udp     111   portmapper
   100005    1   udp     776   mountd
   100005    1   tcp     778   mountd
   100003    2   udp    2049  nfs
   100003    2   tcp    2049  nfs
   100004    2   udp    813   ypserv
   100004    1   udp    813   ypserv
   100004    2   tcp    816    ypserv
   100009    1   udp    821   yppasswdd
   100007    2   udp     957  ypbind
   100007    2   tcp    959   ypbind
snuggles: >
```

Som man kan se, så körs **portmapper**, **mountd** och **nfsd** och är där med redo att serva RPC-anropen.
Det kan vara bra att konfigurera systemets **rc**-filer så att **mountd** och **nfsd** startas vid boot.
Nu är servern konfigurerad och det är dags att få igång klienterna.

Konfiguration av NFS klienter

Det första man måste ha är en kärna som har stöd för NFS filsystem, antingen inkluderat vi kompilationen eller som en laddningsbar modul. Om kärnan inte stödjer NFS måste den konfigureras om, läs 'kernel HOWTO' för mer info (**ftp.funet.fi/pub/Linux/doc/HOWTO Kernel-HOWTO**).

Om kärnan stödjer NFS är det bara logga in som **root**, och med kommandot **mount** koppla upp sig mot servern.
Vi kan ta exemplet med **norpan** (klient) och **snuggles** (server).

```
norpan:/ > ls -l /mnt
norpan:/ > mount   snuggles:/usr/X11R6   /mnt
norpan:/ > ls -l /mnt
total 9
drwxr-xr-x   2 root     root        4096 Apr 14 17:01 bin
drwxr-xr-x   2 root     root        1024 Feb  6  1996 doc
drwxr-xr-x   3 root     root        1024 Mar 13 18:25 include
drwxr-xr-x   4 root     root        2048 Mar 13 19:00 lib
drwxr-xr-x   6 root     root        1024 May 18 02:02 man
norpan:/ >
```

Som exemplet visar, finns filerna i **snuggles** katalog **/usr/X11R6** också hos **norpan** i katalogen **/mnt**. Att använda ett NFS filsystem går inte riktigt lika snabbt som att använda ett på den lokala hårddisken men man sparar både administrativt arbete och lagringsutrymme.

Om följande felmeddelanden skulle visas har du antingen glömt att ändra filen **/etc/exports** eller inte startat om **rpc.mountd** och **rpc.nfsd** hos servern.

```
norpan:/ > mount   snuggles:/usr/X11R6   /mnt
mount: snuggles:/usr/X11R6 failed, reason given by server: Permission denied
norpan:/ >
```

Eller om följande felmeddelande visas körs inte **mountd** eller **nfsd** på servern.

```
norpan:/ > mount   snuggles:/usr/X11R6   /mnt
mount clntudp_create: RPC: Program not registered
norpan:/ >
```

För att ta bort ett NFS filsystem används kommandot **umount.**

```
norpan:/ > umount /mnt
norpan:/ > ls /mnt
norpan:/ >
```

Om man vill att ett NFS filsystemet ska inkluderas vi boot lägger man till följande rader i filen **/etc/fstab**.

```
. . .
# device              mountpoint     fs-type    option     dump    fsckorder
snuggles:/usr/X11R6   /mnt           nfs                   0       0
```

Storleken på de block som NFS skickar via nätverket är normalt 4096 eller 8192 bytes. Men ibland kan vissa kombinationer av Linux kärnor och nätverkskort inte fungera eller fungerar slött med dessa storlekar. Därför tar kommandot **mount** parametrar med vilka man kan sätta **rsize** (lässtorlek) och **wsize** (skrivstorlek). Genom att experimentera med dessa värden kan man finna den optimala konfigurationen av **mount**.
Här följer ett exempel på rsize=1024 och wsize=1024.

```
norpan:/ > mount  -o  rsize=1024,wsize=1024   snuggles:/usr/X11R6   /mnt
norpan:/ >
```

När man hittat den konfiguration man vill ha kan dessa parametrar även läggas till i filen **/etc/fstab.**

```
. . .
# device              mountpoint     fs-type    option                   dump    fsckorder
snuggles:/usr/X11R6   /mnt           nfs        rsize=1024,wsize=1024    0       0
```

<u>Vanliga parametrar till kommandot **mount** när man använder NFS</u>

rw/ro — **rw** binder filsystemet med både läs och skrivrättigheter. **ro** gör filsystemet endast läsbart. Om inget anges använder **mount** parametern **rw** automatiskt.

hard/soft — Dessa parametrar används för att specificera vad som ska ske om servern inte svarar. Om inget anges används **hard** automatiskt, vilket gör att klienten försöker anropa servern tills den svarar. Om parametern **soft** är satt, skickas en felsignal till processen som ville komma åt filen och fler försök att komma åt filen görs ej.

retrans/timeo — Dessa parametrar sätter hur många försök respektive hur lång timeout varje försök att komma åt en fil hos servern får. Det är **retrans** som sätter antalet försök medan **timeo** sätter tiden i tiondels sekunder.

intr — Normalt kan man inte avbryta NFS, utan NFS fortsätter tills den får ett RPC felmeddelande eller tills anropet är fullbordat. Med parametern **intr** satt kan användaren avbryta ett försök att komma åt en fil.

5.6.2 Säkerhet

Att med hjälp av NFS öppna en maskins hårddiskar för andra i ett nätverk, innebär också en stor säkerhetsrisk. Dessa säkerhetsrisker är speciellt farliga om nätverket är uppkopplat mot större publika nätverk. Det räcker med att en maskin i nätverket har ett modem installerat för att ett inbrott ska kunna ske. Använder man då också NFS eller NIS kan detta betyda att hela nätverket är i fara.

En svaghet som NFS har är att klienten alltid litar på servern och även omvänt. Detta kan vara farligt. Om serverns root access kan knäckas är det ganska enkelt att få root access hos klienterna också, gäller även tvärtom.

För att hålla ett NFS-nätverk säkert bör man regelbundet läsa CERTs råd om NFS. Dessa finns att hämta ifrån deras ftp. Hämta hem **ftp.cert.org/01-README** som är en lista över '**CERT advisories**'. Här är några exempel.

```
CA-94:15.NFS.Vulnerabilities                    12/19/94
    This advisory describes security measures to guard against several
    vulnerabilities in the Network File System (NFS). The advisory was
    prompted by an increase in root compromises by intruders using tools
    to exploit the vulnerabilities.

CA-96.08.pcnfsd                                 04/18/96
    This advisory describes a vulnerability in the pcnfsd program (also
    known as rpc.pcnfsd). A patch is included.
```

Säkerhet hos klienten

Klienten kan man sätta till att inte lita på severn för mycket. T ex kan man förbjuda **suid**[1]-program att fungera när de kör ifrån ett NFS-filsystem. Detta görs genom att använda parametern **nosuid** med kommandot **mount**. Denna parameter bör vara satt hos alla NFS-filsystem i **/etc/fstab**. Om man tillåter suid-program att exekveras ifrån ett NFS-filsystem, kan en användare logga in som vanligt och sedan använda **suid**-programmet för att bli **root** på klienten också. Om man vill vara riktigt säker på att det inte finns farliga program på NFS-filsystemet kan man förbjuda all exekvering med parametern **noexec**.

Säkerhet hos servern

Hos servern kan man bestämma att man inte vill lita på klientens **root**-användare. Detta görs genom att använda parametern **root_squash** i **/etc/exports**.

```
#/etc/exports
/usr/X11R6/     norpan(rw,root_squash)
```

Om en användare med UID 0 hos klienten försöker att komma åt det exporterade filsystemet byts denna användares UID ut mot 'nobody'. Detta betyder att **root** hos klienten inte kan komma åt eller ändra i filer som endast **root** hos servern kan komma åt eller ändra. Men **root** hos klienten kan fortfarande använda kommandot **su** för att bli någon annan användare och där med komma alla andra användares filer utom de filer som ägs av **root**. Därför bör alla viktiga filer i ett UNIX system ägas av **root** och inte av **bin**, **daemon** eller någon annan användare som inte har UID 0. I **man**-filen för **nfsd** finns det fler **squash** parametrar med vilka man kan styra access.

En annan säkerhetslucka är den att **nfsd** inte alltid kontrollerar att anropen kommer ifrån rätt port. Om så är fallet kan en användare köra ett program (som är lätt att få tag i på Internet) och lura NFS att tro att han är den han påstår att han är. Linux **nfsd** kontrollerar automatiskt att anropen kommer ifrån rätt port, hos vissa andra operativsystem måste man ge denna parameter själv.
En annan viktig sak är att aldrig exportera ett filsystem till '**localhost** eller IP: **127.0.0.1**

Den vanliga **portmapper** kombinerad med **nfsd** ger svagheten att man kan komma åt filer utan vara den man säger att man är. Linux **portmapper** är relativt säker mot denna attack, och kan bli ännu säkrare genom att konfigurera filerna **/etc/hosts.deny** och **/etc/hosts.allow.**
Filen **/etc/hosts.deny** bör innehålla raden.

```
portmap: ALL
```

Denna rad förbjuder access ifrån alla. Detta är kanske lite drastiskt, så vi öppnar accessen igen genom att lägga till följande rader i filen **/etc/hosts.allow**.

```
portmap: 194.47.147.0/255.255.255.0
```

Denna rad ger alla på subnet: 194.47.147.0 med netmask: 255.255.255.0 access till **portmapper.**

[1] **suid**-program exekveras som en annan användare, oftast som **root**.

Mer information om filerna **/etc/hosts.deny** och **/etc/hosts.allow** finns i respektive **man**-filer med samma namn.

6 RFC lista

Listan innehåller alla RFC som har tagits upp. Första fältet är RFC numret, andra fältet är författarens namn, det tredje fältet är RFC's namn och det sista fältet är datumet den komm ut angiven i följande format: månad/dag/år. RFC dokument uppdateras ständigt och många föråldras till det nivån att de inte längre används. Därför bör de som önskar läsa de nyaste RFC dokument titta genom RFC-index filen. Den brukar finnas i samma katalog som resten av RFC dokumenten.

RFC 768	J. Postel, "User Datagram Protocol", 08/28/1980
RFC 791	J. Postel, "Internet Protocol", 09/01/1981
RFC 792	J. Postel, "Internet Control Message Protocol", 09/01/1981
RFC 793	J. Postel, "Transmission Control Protocol", 09/01/1981
RFC 826	D. Plummer, "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", 11/01/1982
RFC 882	P. Mockapetris, "Domain names: Concepts and facilities", 11/01/1983
RFC 883	P. Mockapetris, "Domain names: Implementation specification", 11/01/1983
RFC 903	R. Finlayson, T. Mann, J. Mogul, M. Theimer, "Reverse Address Resolution Protocol", 06/01/1984
RFC 1054	S. Deering, "Host extensions for IP multicasting", 05/01/1988
RFC 1497	J. Reynolds, "BOOTP Vendor Information Extensions", 08/04/1993
RFC 1700	J. Reynolds, J. Postel, "ASSIGNED NUMBERS", 10/20/1994

7 Källförteckning

Operating System Concepts	Abraham Silberschartz, Peter B. Galvin
A Practical Guide to the UNIX System	Mark G. Sobell
Linux, Unleashing the Workstation in Your PC	Stefan Strobel, Thomas Uhl
Practical UNIX security	Simson Garfinkel, Gene Spafford
The Hitchhikers Guide to the Internet	E. Krol, University of Illinois Urbana
Managing NFS and NIS	Hal Stern
Internet programmering	Kris Jamsa, Ken Cope
UNIX network programming	W. Richard Stevens
Data and Computer Communications	William Stallings
Communication Networks: A first course	Jean Walrand
The Linux Network Administrators' Guide	Olaf Kirch
Linux Installation and Getting Started	Matt Welsh
Linux System Administrator's Guide	Lars Wirzenius
The Linux Kernel Hackers' Guide	Michael K. Johnson
Introduction to the Internet Protocols	Charles L. Hedrick, Rutgers University
Internet Protocol (RFC 791)	Jon Postel
Transmission Control Protocol (RFC 793)	Jon Postel

UR INTERNETS HISTORIA	http://grub01.physto.se/internet/inet/historia.html
LINUX HOWTOS	ftp://ftp.funet.fi/pub/Linux/doc/howto/