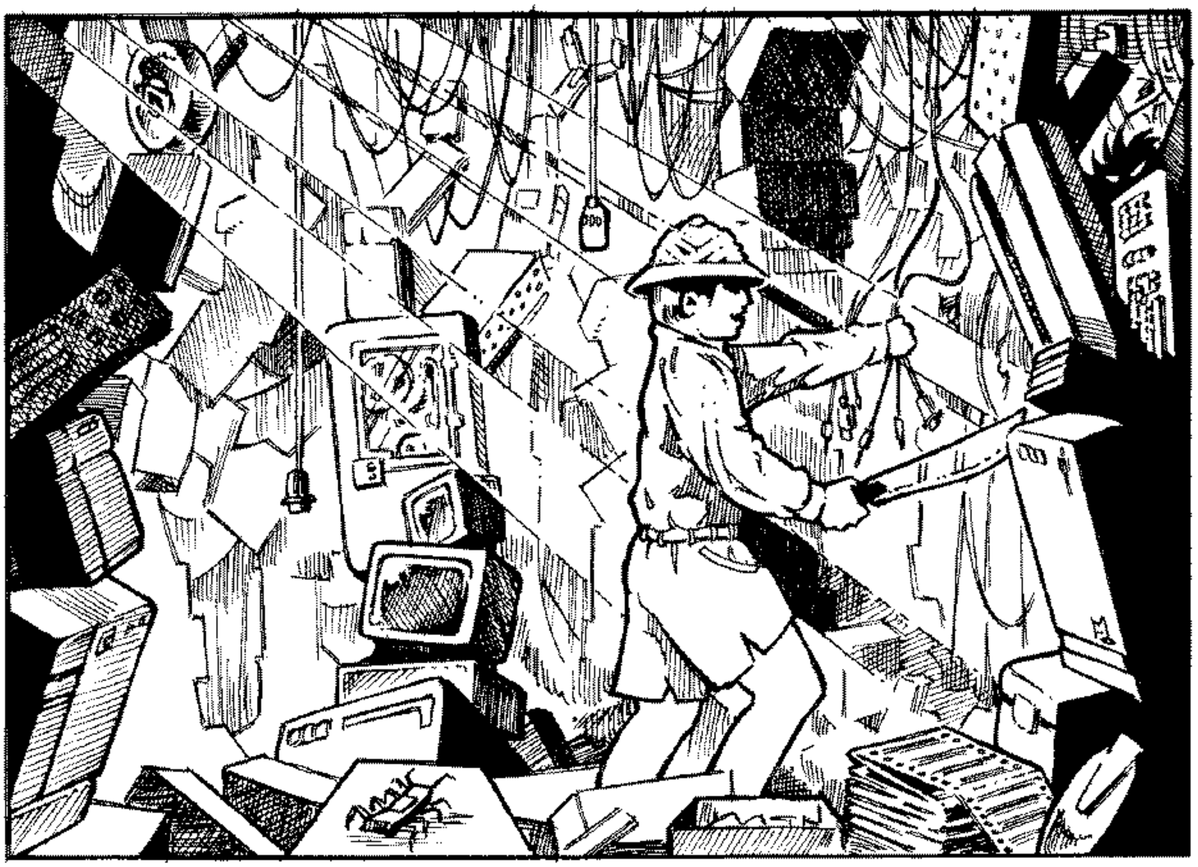
GARB

Organ för datorföreningen Lysator

1973 **20 ÅR** 1993



Augusti-93

Innehåll:

Redaktörens spalt

av Pell Pell Pell

Jaha... <tittar sig förvirrat omkring> Vart tog alla vägen? Ture är visst ny ordförande, men han är varken fysiskt eller virtuellt tillgänglig. Jens hade visst inte möjlighet att skriva en "avgående ordförandens spalt" heller. Då får väl jag skriva en "tomma skallens spalt" eftersom jag har tagit ett nattpass för att få tidningen klar och börjar känna mig virtuellt frånvarande nu när klockan närmar sig... Tre! På eftermiddagen! Usch, dags att avsluta det här medan ögonlocken hänger uppe.

Det här är tjockaste numret av GARB genom tiderna, vilket är väldigt roligt. Jag vill rikta ett stort tack till alla som ställde upp och skrev artiklar, trots att det blev lite jäktigt efter Bug-förvirringen i våras. Jag är lite ledsen över att jag inte fick fler synpunkter på (och artiklar om) jubileet, så hela äran för jubileumsbevakningen tillfaller den här gången Jan Lien! Däremot är jag säker på att alla som bevistade jubileet hade en underhållande dag och kväll. Det måste vi göra om, så håll föreningen levande!

Och du—flippa lugnt, va? :-)

Illustrationer:

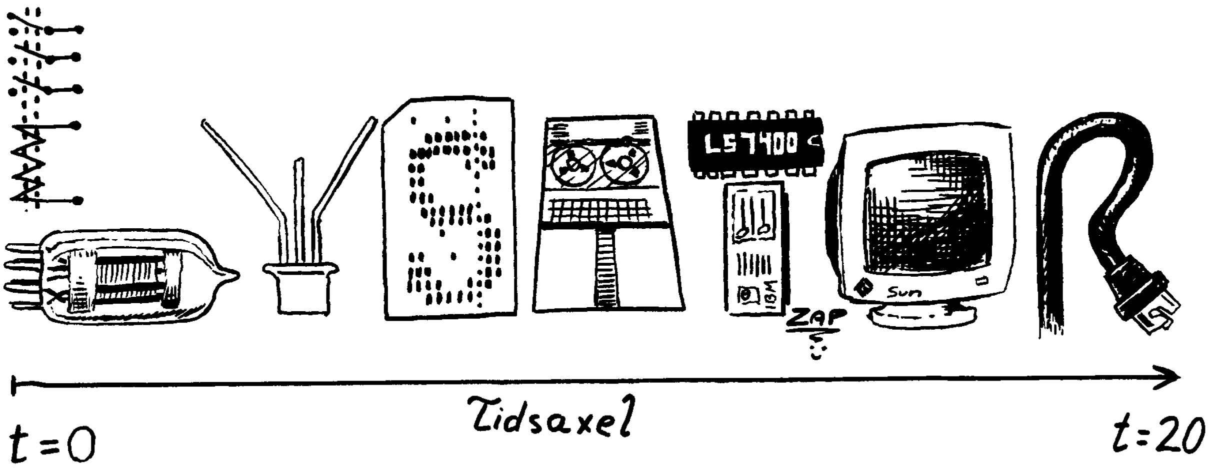
Omslag	Möhrman
Sid 3, 61	Zap
Sid 8	Bud Grace
Sid 9, 11, 36	Hepp

Foto:

Sid 4, 5	Jan Lien
Sid 25, 27, 28	Peter Eriksson

GARB är organ för datorföreningen Lysator vid Linköpings Universitet och Tekniska Högskola. Redaktör för detta nummer är Pär Emanuelsson (PPP). Detta nummer har tryckts på LinTek REPRO i 500 exemplar. Nuvarande redaktörens policy är att ta emot upp till halvsidesannonser gratis från Lysators sponsorer. För andra önskemål kontakta adressen nedan.

Lysator har postadress Lysator, Institutionen för Systemteknik, Universitetet, 581 83 Linköping. Elpost kan skickas till `Lysator@Lysator.LiU.SE`. Medlemsskap för 12 månader i Lysator kostar 192 kronor och betalas till postgiro 88 56 69 - 2.



Jubileum!

Som de flesta kanske vet firade Lysator 20-års-jubileum i mars. Jan Lien berättar här om sina intryck av tillställningen.

Den 29:e Mars fyllde Lysator, Sveriges äldsta datorförening, 20 år! Jubileet firades med inbjudna föreläsare och en fest på kvällen.

Historik

Lysator bildades 1973, efter att några studenter tiggt en DataSaab D-21 stordator. Snart ville medlemmarna ha egna datorer, som de kunde köra hemma. I föreningens regi konstruerades en för den tiden avancerad 16-bitars mikrodator, LYS-16, med 32 kB minne men utan hårddisk.

Senare gick föreningens verksamhet i ide, och stordatorerna stod stilla. Till slut tog några intresserade tag i föreningen och verksamheten återuppstod. D-21:orna åkte ut, föreningen flyttade ned i källaren under ingång 29, och man fick mindre datorer.

Verksamheten utvecklades och man fick många aktiva medlemmar, då man fick veta att Lysator måste flytta ut ur källarlokalen, bland annat för att elinstallationerna inte kunde godkännas. Några i föreningen satsade mycket av sin tid på att tigga ett helt nytt hus med tillbehör till föreningen, och det är i det huset föreningen idag bor, det s.k. Q-huset bakom E-huset.

Till jubileet hade man bjudit in fyra talare, som höll föreläsningar om sin verksamhet. På kvällen anordnades en bankett i Ryds Herrgård. Innan föreläsningarna skapade jag och Jonas Melin en linux-drink, som senare postades till `comp.os.linux`. Den som vill veta ingredienserna kan fråga mig.

Alpha - het dator

Förste talare var Pelle Delby från Digital Equipment, som berättade en del om deras nya Alpha-maskin.

- Nästan alla arkitekturer har dött p.g.a. tekniska problem. Vi räknar med 20-30 års livslängd för Alpha. Nu räknar vi med att fördubbla prestanda var 18:e månad under ett tag framåt.

- Vi har vissa problem med vår PC och Windows NT — den är klassad som superdator och kräver exportlicens. Men skruvar man ner klockfrekvensen och levererar den utan FDDI-kort klassas den inte som superdator, och kan levereras direkt!

Pelle fick frågan om Digital utvecklar större maskiner bara för att man skall kunna köra tyngre och långsammare operativsystem, som Windows NT. Digitals representant undvek skickligt att ta ställning för ena eller andra sidan. Många synpunkter framkom om detta, och talaren fick nästan stå kvar betydligt längre än avsett.

Linux - fri UNIX

Näste talare var Linus Torvalds, skaparen av Linux, en fri unix för 386-datorer. Lysator kör linux på en av sina datorer, och det finns flera medlemmar som själva kör linux hemma. Han berättade lite om hur linux uppstod.

- Jag satt med minix, och tyckte att det där skulle väl jag kunna göra bättre själv. För att sätta igång med ett operativsystem krävs tre egenskaper:

- arrogans - jag kan bättre
- ignorans - jag visste inte vilka problemen var
- envishet, samt inspiration - jag har valt fel, men aldrig så fel att det inte gått att rätta.

Linus Torvalds — skapare av LINUX

- Jag visste inte att det var ett operativsystem jag börjat skriva, från början förbättrade jag ju bara minix. För ett och ett halvt år sedan var linux ganska hjälplöst och med många buggar. Det är många som hjälpt mig och kommit med goda kommentarer.

Linus beskrev de viktigaste punkterna i linux, bland annat att det är portabelt, samt att källkoden är fri och gratis. Det är inte vackert, utan snabbt och effektivt.

- Det är många olika personer som hjälpt till, och eftersom utvecklingen sker öppet på nätet går det snabbt. Sen finns det folk som hjälpt till utan att veta det. En som klagade fick virtuellt minne, en viktig milstolpe för att kunna köra gcc.

Någon frågade hur mycket tid Linus Torvalds lägger ned på arbetet med linux.

- När jag inte hade sommarjobb höll jag väl på 10 timmar per dag. Nu har det blivit mindre programmering, det tar nu minst 2 timmar per dag att svara på mail och läsa news. Jag har väl arbetat 5-6 timmar per dag i två års tid, men inspirationen kommer inte alltid när man vill. Jag kompilerar om kernel 10-20 gånger per dag.

- Jag får en massa dumma frågor, de flesta är dock trevliga, men en del ignoreras.

En annan fråga var vilka egenskaper Linus Torvalds själv var nöjd eller missnöjd med.

- När folk talar om micro-kernel design får man skämmas och gömma sig i någon vrå.

- Jag är nöjd med hur processer sköts och fungerar. Linux använder inte klassisk sleep/wakeup, och detta är delvis inspirerat av mig själv. I vanliga unixar förekommer sleep/wakeup, och det kan leda till racevillkor. Linux har inte sleep, utan processer har en viss state, som de själva sätter. De kan till exempel sätta sleep, men fortsätter att köra. De kallar schedulern när de verkligen vill sova. Man får sleep-test-scheduler. Det fungerar bra med select, man `sover', testar sedan alla villkor, och man blir aldrig sovande om man inte borde bli det.

- Jag är också nöjd med att ha undvikit att använda longjmp[1] — djävulens danssteg. Jag skall inte använda detta i linux.

Linus Torvalds avslutade med att säga att det *går* att göra linux Sys-V-kompatibelt, men ingen har erbjudit sig göra det.

Det avslutande föredraget om KOM-systemets tillblivelse hölls i A-huset, så alla åhörarna marscherade över från C-huset.

KOM-system och KOM-makare

Jacob Palme och Torgny Tholerus berättade om hur det svenska KOM-systemet växte fram. Det ursprungliga svensk-KOM kördes under TOPS-10, och sedemera även TOPS-20. Kanske medlemmarna minns det livaktiga KOM på Linus, eller KOM på Kicki i Stockholm.

Förste talare var Jacob Palme. Han berättade koncentrerat, högt tempo, mycket fakta, inga pauser, snabbt, engagerat, få eller inga upprepningar, rasande tempo ... Puh!

- En amerikan fick 1969 uppdraget att ta fram ett datorbaserat system för att hantera plötsliga kriser. Resultatet blev EMISARIO konferenssystem.

- Jag var 1972 huvudansvarig för inköp av Sveriges första DEC-10 system, ODEN. Det var en investering i infrastruktur och basmiljö. Vi visste då inte att konferenssystem fanns. Senare fick vi prova EMISARIO och ett annat system som kallades FORUM/planet.

- När FOA skulle decentraliseras undrade man hur kommunikation skulle kunna ske mellan enheterna. FORUM/planet tillät maximalt 256 inlägg per möte, och vi undrade om vi inte kunde utveckla ett bättre konferenssystem.

- Det var startskottet för KOM. Vi var påverkade av de olika system vi sett. En av de negativa sidorna var submenyer — för att återse ett inlägg skulle man t.ex. först välja meny 5, sedan undermeny 3 och sedan val 2 i den menyn. För att snabba upp det kunde man göra 5:3:2 för att `återse nästa inlägg'. Vi bestämde att KOM inte skulle ha siffersystem, och inte fixa förkortningar, utan istället vanliga ord som kommandon, och de skulle kunna förkortas.

Datainspektionen krånglar

- 1973 kom datalagen, och datorer blev övervakade och kontrollerade. 1978 förbjöd Datainspektionen KOM, som var stoppat i 4 månader. Därefter kunde systemet starta med förödmjukande begränsningar, som inga politiska uppgifter, inga uppgifter om sjukdomar eller om religion. Efter 2 år gav Datainspektionen upp.

Jacob Palme — KOMskapare

- Vi anser att man skall få skriva vad som helst i ett inlägg. Man skall särskilt skydda medborgarnas rätt att meddela sig med varandra.

- Eskil Block skrev 1982 ett av de mest kända inläggen i KOM. Maj Wechselmann stämde Eskil Block (som arbetade på FOA) för förtal. (Eskil Block skrev ungefär att Maj Wechselmann var betald av ryssarna) Eskil

1. Avancerad spaghettiprogrammering - Red.

Block blev fälld, och ålades att publicera domstolens utslag i KOM.

DEC-10 KOM användes på flera platser, och såldes även till åtskilliga installationer utomlands. Det var populärt att lägga in de svenska menytexterna till 1:e April hos installationerna i USA.

SuperKOM - ny generation

- Nu har vi utvecklat SuperKOM som varit i drift ett par år. Skillnaderna är att SuperKOM har en distribuerad databas, och liknar usenet news. Man kan kopiera över presentation av möten, inlägg i möten, med mera. Allt sker automagiskt. Alla möten annonseras i annonseringsmöten.

- Det är positivt att det tekniska fungerar bra. Det negativa är problem med uppdatering, ändra namn på möte eller användare fungerar inte alltid, det gamla namnet ligger kvar.

- Man måste ha en parallell nedladdning av nyheter medan användaren läser. Man kan ha kommunikation mellan ett möte och en newsgrupp. Vi stoppade loopar genom att skapa externmedlemmar. Kommentarer är länkade till huvudinlägget, och inte till konferensen. Man behöver då bara flytta en länk för att flytta kommentarerna.

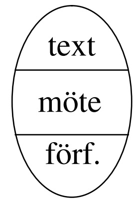
KOMs inre väsen

Efter Jacob Palme tog Torgny Tholerus över. Han berättade i ett lugnare tempo, och inte fullt så faktaspäckat.

- KOM började egentligen 1967 när jag blev ordförande i syndikalistiska studentföreningen. Vi publicerade en stencilerad tidning som kallades `Anarkisten'. Där tog jag in allt som folk sände in, det fanns inga gränser.

- Min filosofi är att alla hushåll skall kunna använda KOM. Kanske man kan genomföra folkomröstningar via KOM i framtiden.

- SuperKOM är skrivet i C. Man har objekt som kan representeras som behållare med olika fack. Sedan kopplar man slangar mellan dem, som informationen rinner i. Behållarna är uppdelade i tre rum:



Genom att lägga slangar till olika delar av behållarna kan man uppnå olika effekter. Alla objekt är representerade på detta vis, från enskilda texter upp till hela databaser. Databaserna är emellertid dolda för användaren och information flyter automatiskt mellan dem som tidigare nämnts.

Nu avbröt Lars Aronsson för att ställa en mycket komplicerad fråga, som handlade om den i hans tycke användbara utvidgningen att kunna låta användare kommentera en hel databas och vad som då skulle kunna hända. Efter den långa komplicerade frågan tittade Torgny Tholerus på Jacob Palme och frågade `Vad säger du?'

Jacob Palme svarade utan tvekan `Det kommer i nästa version!'.

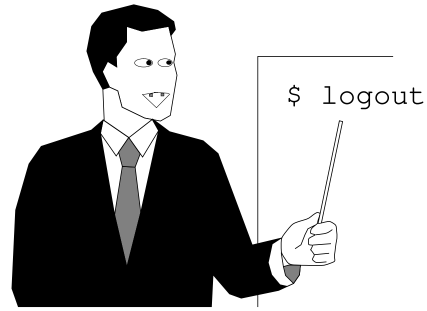
Torgny Tholerus fortsatte att berätta. Man har ett radorienterat och ett skärmorienterat gränssnitt. Ett API-gränssnitt kommer. Man måste även ha ett lika bra mailsystem i KOM som de bästa externa. Han nämnde att SuperKOM har ungefär 200 användare, medan QZ-KOM användes av 1600 användare per månad.

Torgny och Jacob arbetar nu på Stockholms Universitet, och de har finansiering för utveckling i 3 år, om än i något begränsad omfattning.

Efter föreläsningarna gick folk tillbaka till Q-huset för att beundra de presenter som inkommit från några företag. På kvällen vidtog banketten i Ryds Herrgård, dit givetvis alla föreläsarna var inbjudna.

/ Lien

Kurser!



Då och då ger medlemmar i Lysator kurser för andra medlemmar. Det brukar vara mycket uppskattat och den här hösten finns (minst) följande kurser att välja på:

Emacslispkurs

- Emacs programmeringskurs - TLYS 43 0,0 poäng.

Fö 10 Lab 10 Projekt 7

Version 2 av Lysators, dvs min (Linus Tolke `<linus@lysator.liu.se>`), bejublade emacslisp-kurs kommer att gå av stapeln under period 1.

Förkunskapskrav: Kunskap om emacs (minst väl genomgången TUTORIAL). Kunskap om lisp (syntax, cons-cell, atom) Kunskap om unix (vad är en fil, en process)

Anmälan till linus@lysator.liu.se.

Program med dagar/tider/platser kommer att distribueras via email. Instuderingsmaterial/labbmaterial kommer att distribueras efter hand (som jag skriver det) till dem som är anmälda i kursen.

Projekt skall genomföras för att få godkänt.

Kursprogram:

Träff 1: Dagens motto: Välkommen till emacslisp.

- Emacs historia: rms mm.
- Grunderna i lisp: atom, cons-cell, funktionsdefinitioner (interactive nämns), macron, variabelbindningar (med let eftersom det är fråga om emacslisp)
- Hur evaluera lisp i emacs? ESC ESC, scratch-bufferten (lisp-interaction-mode), eval-defun, eval-last-sexp.
- Hur man laddar lisp-program i emacs. load-library (load-path), load-file, eval-current-buffer, autoload.
- Hur kommer man åt documentation: describe-function, describe-variable, info

Övningsexempel på car, cdr, list, quote, while, let, mapcar, apply, defun, defmacro.

Träff 2: Motto: Varför gör vi på detta viset?

- Emacs speciella typer: vector, string Buffrar, marker window, window-configuration process keymap obarray (symboler) (har jag glömt någon?)
- Emacs speciella grejor för funktioner: doc-strängar, hur de skall skrivas, för variabler—interactive konstruktionen, alla alternativ som finns

Övningsexempel: lokala variabler, interactive, mapatoms, set-window-configuration.

Träff 3: Motto: Nu har jag fått den jag vill ha.

- Emacs configurering (detta är mindre emacslisp och mer emacs): moder, mode-map, require/provide hookar, define-key syntax-table (abbrev) defvar/ defconst

Vi tittar lite närmare på någon mode.

Träff 4: Motto: Går det att göra i emacslisp?

- rekursiv/iterativ programmering, fördelar/nackdelar
- regexp, regexp-sökningar, byten
- processer, open-network-stream
- felsignalering, fångande av fel
- filer/streams.

Övningsexempel: max-lisp-eval-depth, string-match, match-data, signal, condition-case, set-process-filter, set-process-sentinel, find-file-noselect, with-output-to-temp-buffer, read.

Träff 5: Motto: Nu är vi klara!

- Hjälpmedel: elib, debug/edebug
- Optimeringsmöjligheter: byte-compilering
- Hur man skriver dokumentation: info-sidor

UNIX-för nybörjare

dvs för dem som inte får gå ODDen.

Kursledare:

Linus Tolke <linus@lysator.liu.se>

Kväll 1: Start tisdagen den 7 september klockan 19.15 med samling i/hänvisning från Lysators fikahörna.

- logga in, ändra lösen, logga ut.
- filträdet, gömda filer.
- shellen
- processer

Kväll 2: Vi kommer överrens om tid/plats under Kväll 1.

- nätverkande: telnet(1), rsh(1), rlogin(1), ftp(1)
- världsperspektiv: mail(1), news.

Eventuellt överkurs:

- programutveckling: make
- världperspektiv: mud, irc, lyskom.

Ernie och kaffebryggaren—Favorit i repris

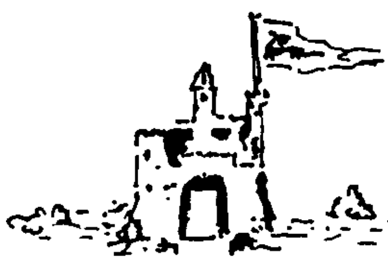
MUD!

SvenskMUD-träff

av Linus skaparen (gud)

Historiens första SvenskMUD-träff gick av stapeln söndagen den 28 mars. Målet med träffen var att lära känna olika magiker, ansikte mot ansikte och också samordna verksamheten med världsbyggandet.

Närvarande var några magiker från SvenskMUD, vissa hade rest till från andra delar av vårt långa land men de allra flesta kom lokalt från Linköping.



Schemat för dagen var inte riktigt väl genomtänkt, det fanns allvarliga brister på flera punkter - mest tids och lokalmässigt men vi hann med vad vi ville och det viktiga är väl att man får något gjort, inte att planeringen stämmer.

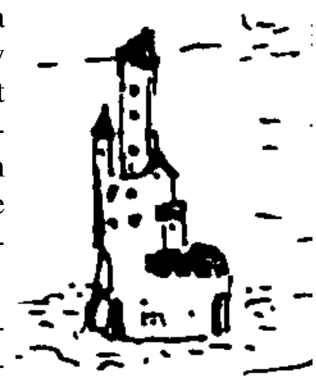
Första officiella delen handlade om SvenskMUDs historia, se tidigare nummer av Garb, och dessutom lite om hur objekt byggs upp och samarbetar. Just detta samarbete är den svåraste delen i hela programmeringsprocessen.

Det finns nämligen ingen bra dokumentation om hur det görs och vad som egentligen händer. Det enda som finns är `/doc/lfun/*`. Dessutom är koden som sköter de grundläggande objekten inte riktigt bra, den är gammal, ursprungligen skriven för en helt annan driver.

Under punkten erfarenhetsutbyte om kodning blev det inte så mycket prat. Det verkar som om det inte är vidare lämpligt att diskutera sådana frågeställningar - det är alldeles för små detaljer för att det skall bli diskussion av det hela.

Nu var det dags för det obligatoriska Snoddaskorvenbesöket. Oturligt nog hade någon magiker i verkliga livet klonat fram en massa snö över oss precis när vi hade kommit dit. Det blev lite kyligt men härdade magiker står ut.

Den intressantaste delen av diskussionen var den som handlade om hur världen ser ut nu och hur den skall se ut i fortsättningen. För närvarande är världen konsistent så tillvida att man ganska lätt kan rita en karta och pricka in områden såsom skog, sjö, hed, berg mm och det stämmer någotsånär. Med lite vilja kan det vara en svensk by med lite skog och annat ikring i något svenskt landskap. En massa problem löstes eftersom vi lätt kunde prata med varandra om olika saker.



Det visade sig i diskussionen att det största problemet med kodandet i spelet och med att behålla en bra och konsistent värld är ett kommunikationsproblem. Man behöver förmedla olika ideer till de nya magikerna för att de skall kunna ta aktiv del i världsskapandet och det finns ganska dåliga möjligheter för sådant.

Avslutningsvis samlades vi för den officiella banketten som vi avnjöt på pizzeria-kebab Liz. Diskussionen under middagen drev iväg ganska långt från SvenskMUD men det var i alla fall en trevlig avslutning på en trevlig dag.

Deltagarlista:
Aronsson, St, Gunnar, Povin, Nordgrimm, Hades, Meta, Testola, Storken, (Snöskred)

Försäljning av SvenskMUDprylar!

Den speciella SvenskMUDknappen finns till försäljning hos Linus skaparen (gud). Texten "Jag kör SvenskMUD" skriven i rött på vit bakgrund. Kan uppvisas på begäran!

Kostnad 15 daler. Om du vill så skickar jag knapparna om du sätter in pengarna på mitt postgirokonto 483 11 40 - 1 eller skickar dem till mig på annat sätt.

För alla SvenskMUDmagiker finns den speciella Handbok för SvenskMUDmagiker som tar upp en del av det som en magiker i SvenskMUD bör veta. Boken innehåller dock ingen LPC utan bara saker som har med själva byggandet i just SvenskMUD att göra. Denna bok kan du hämta med anonym FTP från Lysator som texinfo-dokument eller infoiserat i directoriet `/pub/svenskmud`. Jag har även en tryckt upplaga i behändigt A5-format som du kan få mot 30 daler enligt ovan.

Mer än 15000 rum i NannyMUD!

av Padrone

Lysators äldsta och största Mud växer över alla bräddar. Prova själv (`"telnet mud.lysator.liu.se 2000"`)!

Enligt den senaste uppskattningen finns det 15147 rum i NannyMUD. Av dessa rum ingår 155 i "/room-arean", dvs byn och andra gemensamma områden, och 14992 är byggda av olika wizards.

Uppskattningarna kommer från ett program som går igenom alla filerna med beskrivningar av mud-objekt, och som försöker avgöra vilka av dessa filer som beskriver rum och vilka som beskriver andra objekt (vapen, monster mm).

Så här ser utmatningen ut från programmet som räknar rum, när det räknar wizards-byggda rum:

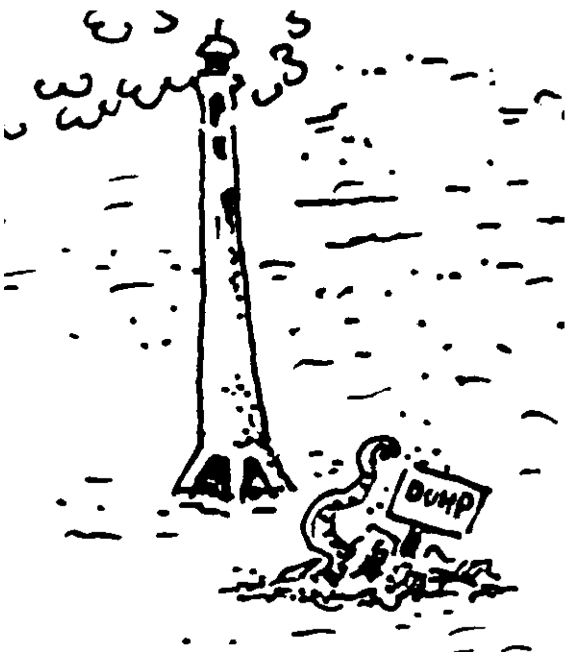
```
Statistics from leta-rum2, started Sun Jul 18 12:39:16 1993
Finished Sun Jul 18 14:46:36 1993
I went through the tree starting at '/usr/lpmud/lib/players'.
Searched 2249 directories, plus 3 that I couldn't open.
Ignored 79 symbolic links.
Couldn't stat 0 files.
Checked 39523 files, and 25693 of these ended with '.c'.
Couldn't open 0 of these '.c' files.
Result:
I found 15512 probable rooms:
   2063 containing #includes of (some) std.h,
    552 containing #includes of (some) room.h,
  11475 inheriting (some) room, and
   1422 containing move_player.
Of these 15512 probable rooms, 520 had a file name containing
one of the strings 'old', 'obsolete', 'unused' etc.
Suggested number of rooms in use: 14992
```

Dessutom finns det alltså, enligt samma program, 155 rum ("suggested number of rooms in use") i "/room-arean", vilket totalt ger 15147 rum!

Lite annan statistik om NannyMUD (18 juli):

Total antal spelare: 2776
Antal wizards: 329(varav 152 aktiva, dvs som har varit inloggade senaste månaden)
Antal dödliga spelare: 2447 (varav 625 aktiva)
Antal dödliga spelare högre än nivå 1: 1318 (varav 372 aktiva)

-- Padrone, 18 juli 1993



Kobayashi's Tower och Village Dump

Padrones Walking Castle

Shinto-Temple of Chrisp

LysKOM

LysKOM är ett konferenssystem utveckat av Lysator-medlemmar. Det arbetar enligt klient-server-modellen och är för tillfället i drift på flera ställen i Sverige. För att koppla upp sig mot en server behöver man alltså en klient och det finns flera att välja mellan. Dessutom är ännu fler under utveckling, vilket bl.a. är ämnet för följande artiklar.

LysKOM — the saga continues

eller

Buzzwords, buzzwords, buzzwords

av

Per Cederqvist och Inge Wallin

LysKOM kan beskrivas med följande ord:

- client-server
- öppet system
- groupware
- ANSI-C
- POSIX (nästan, i alla fall)

Snart kan ytterligare några buzzwords läggas till listan:

- objektorientering
- X-windows
- menystyrt

Författarna till denna artikel håller på att skriva på en ny familj av klienter till LysKOM. Vi tänkte berätta lite om hur vi har byggt upp den. Men först, för eventuella nytillkomna läsare av GARB, en kort presentation av vad LysKOM är för något sett ur implementatörens synvinkel.

LysKOM - en kort presentation

När jag[1] började på LiTH 1988 fanns det en dator här som hette Linus. På den fanns ett program som hette KOM, som man kunde använda för att diskutera saker med andra personer. Ganska snabbt blev jag beroende.

1990 skulle IDA köpa in en massa Sparcstationer och Linus skulle avvecklas. Vi var några lysiter som blev kallsvettiga när vi hörde talas om det. Skulle vårt KOM-system försvinna för gott? Vi bestämde oss för att göra något för att förhindra det. Vi skrev ett eget konferenssystem.

Ett drygt halvår efter att vi satte igång fanns det ett system som kunde användas hjälpligt. I en central server lagras alla texter som skrivs och information om alla personer som läser och skriver dem. När man läser KOM använder man en klient som kopplar upp sig mot servern.

Servern är skriven helt i C. Den första klienten som började skrivas var också skriven i C och var gjord för att kunna köras på vilken terminal som helst. Den klient som används mest i dagsläget är skriven i Emacs Lisp. Man måste alltså använda Emacs för att kunna köra LysKOM. Eftersom den lispen inte är gjord med tanke på att skriva LysKOM-klienter blir den rätt långsam.

Objekt i LysKOM-systemet

I grund och botten finns det tre sorters objekt i ett LysKOM-system: texter, personer och möten. Varje objekt har ett nummer. Personer och möten får nummer ur samma nummerserie; texterna har en egen.

1. Detta stycke skrev ceder innan Inge blev medförfattare

En text är uppdelad i en "textstatus" och själva texten. Textstatusen innehåller bland annat följande information:

- Författare (ett personnummer)
- Skapelsetid
- Mottagare (0 eller fler mötesnummer)
- Texter som kommenterar detta inlägg (0 eller fler textnummer)
- Texter som kommenteras av detta inlägg (0 eller fler textnummer)

Ett möte består av bland annat följande information:

- Ett namn (en sträng)
- Medlemmar (0 eller fler personnummer)
- Mötestyp (styr bland annat vilka som får bli medlemmar)
- Texter

Den sista punkten kräver en liten förklaring. Varje text som sänds till ett visst möte får ett lokalt textnummer i det mötet. I mötet finns en tabell där man kan slå upp vilket textnummer ett visst lokalt textnummer motsvarar.

En person innehåller ett lösenord, en lista över de textnummer han har skapat, lite statistikgrejor och information om vilka möten han är med i och vilka lokala textnummer han har läst i de mötena. Det finns alltid ett möte med samma nummer som personen. Det mötets namn är personens namn. Brev till personen hamnar i det mötet.

Protokollet

Mellan servern och klienterna skickas information via ett protokoll som kallas "protokoll A" [1]. Vi hade följande designmål när vi gjorde det:

- För att undvika att servern blir en begränsande faktor skall så mycket som möjligt av jobbet skall utföras av klienterna.
- Det ska gå att göra flera olika klienter.
- Ju enklare dess bättre. Om ett halvår skulle Linus-KOM tas ur drift, och då ville vi gärna ha ett LysKOM igång.

För att underlätta avlusningen av både server och klient är protokollet ett nästan-klartext-protokoll. Inga binära data överförs. Siffror som ska överföras sänds som decimala tal.

Protokollet är assymetriskt. I riktning från klienten till servern skickas frågor. Varje fråga består av ett nummer (som anger vilken fråga det är) och ett referensnummer. De flesta frågor har även mer information. Det finns ett drygt 60-tal olika frågor. Som exempel kan nämnas "logga in", "hämta textstatus", "hämta mötesstatus" och "läsmarkera text".

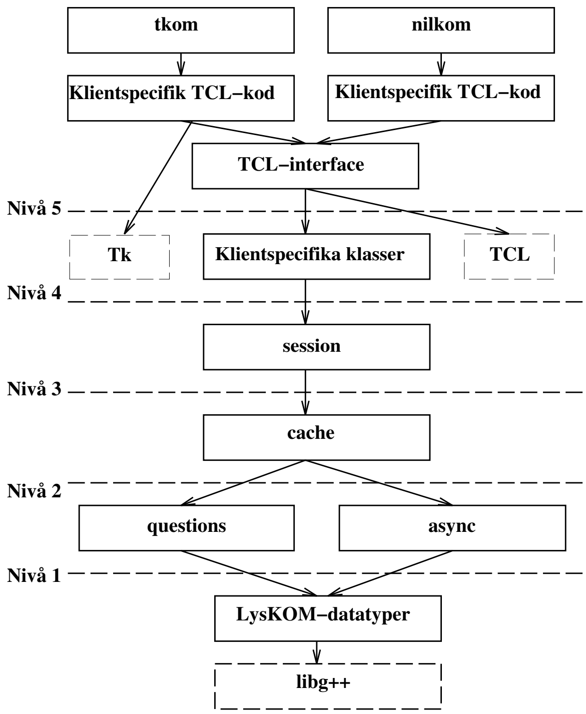
Servern reagerar på en fråga genom att skicka ett svar. Normalt består ett svar av tecknet '=' följt av det referensnummer som klienten gav följt av svaret på frågan. Ibland misslyckas en fråga (till exempel om klienten försöker hämta en text som inte existerar). Då blir svaret tecknet '%' följt av referensnumret och två felkoder (varav bara den ena används idag).

Förutom frågor (från klienten) och svar (från servern) skickar servern ibland ut "asynkrona anrop" för att berätta att saker och ting har inträffat. Det sker till exempel när en ny text har skapats. Klienterna behöver alltså inte polla servern, utan de får sån information automatiskt.

Protokollet är asynkront. Till skillnad från RPC (remote procedure call) där klienten ställer en fråga och väntar på svaret är LysKOM-protokollet designat för att ställa flera frågor på en gång. Särskilt när man har långa fördröjningar i nätverket (till exempel om klienten befinner sig på andra sidan jordklotet) kan man tjäna mycket tid på att göra så.

kom++

Den klientfamilj som vi skriver på nu går under arbetsnamnet kom++. Den är uppbyggd hierarkiskt som ett antal moduler, där varje modul motsvarar en nivå i abstraktionshierarkin. Så här kan det se ut:



Nivå 1: LysKOM-datatyper

I botten ligger ett antal C++-klasser med motsvarighet i LysKOM-systemet. De har namn som `Conf_stat`, `Pers_stat` och `Text_stat`. Dessa objekt kan ibland vara ganska stora, och det har visat sig att man ofta vill skicka dem som argument och/eller returvärden mellan funktioner. För att slippa kopiera stora mängder data har vi gjort dessa klasser automatiskt referensräknande. Själva objektet innehåller endast en pekare till den interna, referensräknade datarepresentationen.

Vi har även implementerat en egen strängklass, `LyStr`, som är anpassad till LysKOMs krav. Den kan till exempel innehålla NUL-tecken, det är effektivt att ta bort tecken i början av strängen och lägga till nya tecken i slutet. En härledd klass, `prot_a_LyStr`, innehåller metoder som hjälper till att avkoda och skapa protokoll-A-meddelanden.

Nivå 2: questions

Som vi förklarade tidigare så består en komsession av att klienten skickar ett antal frågor till servern som svarar på dem. Varje sådan fråga som ställs av klienten representeras av ett C++-objekt, och varje frågetyp (med sitt nummer) har en egen klass. Alla dessa klasser har en gemensam basklass som heter `question`.

Direkt under basklassen `question` finns en klass, `connection`, som sköter uppkoppling mot servern, buffring av data, och liknande uppgifter. Klassen `connection` utnyttjar i sin tur ett bibliotek för kommunikation via BSD-sockets (`ISC`, skrivet av Peter Eriksson i C) och vårt alldeles egna strängpaket.

Förutom ren kommunikation håller `connection` även reda på vilka frågor som servern ännu inte har svarat på. När ett svar kommer skickas det till rätt frågeobjekt, som känner till hur svaret ska se ut och parsar det. Resultatet lagras i frågeobjektet för senare access.

När man skapar ett frågeobjekt konstrueras en sträng innehållande motsvarande fråga enligt protokoll A. Den skickas iväg till servern direkt[1]. Varje frågeobjekt kan vara i ett av följande tillstånd:

- Ivägskickat till servern (`st_pending`)
- Besvarat av servern (`st_ok`)
- Servern gav ett felmeddelande (`st_error`)

Med metoden `status()` kan man se vilket av de tre tillstånden en fråga befinner sig i. Om man verkligen vill ha svaret använder man metoden `receive()` som väntar på att svaret ska komma tillbaka och returnerar dess tillstånd som nu är `st_ok` eller `st_error`. När tillståndet är `st_ok` kan man hämta resultatet med metoden `re-`

1. Nästan direkt. Den mellanlagras ett tag i en buffert. Om två frågor skapas direkt efter varandra kommer de att sändas i samma TCP/IP-paket.

sult().

Ett konkret kodexempel kanske är enklast för att förklara hur man använder frågor. Den här kodsnutten skriver ut namnet på person nummer 5. Vi kommer att lösa detta problem på flera olika sätt längre fram i artikeln. Kodsnuttarna är tagna ur ett fungerande program, som finns listat i sin helhet sist i artikeln.

```
// Hämta namn med hjälp av question.
  get_conf_stat_question c(&conn,
           5);
  if (c.receive() == st_error)
    abort();
  conf_stat = c.result();
  cout <<"Resultat med question:"
       << conf_stat.name()
       << endl;
```

Tillbehör till Nivå 2: async

Som vi beskrev ovan kan servern få för sig att skicka ut asynkrona meddelanden ibland. Dessa måste avkodas och tas om hand på något sätt.

Som spindeln in nätet för hanteringen av asynkrona meddelanden finns klassen `Async_storage`. Den tar emot råa asynkrona meddelanden från `Connection`, ser efter vilken typ de har, och distribuerar sedan ut dem till objekt som ansvarar för en viss typ.

`Async_text_created_administrator` är en klass som administrerar meddelanden om nyskapade texter. Den sköter tre uppgifter:

Avkoda ett meddelande, och skapa ett objekt av typen `Async_text_created` som innehåller all information om objektet.

Hålla reda på en lista av funktioner (handlers) som skall anropas när ett meddelande kommer.

Distribuera ut ett `Async_text_created`-objekt till alla handlers.

Det finns en motsvarande administratör för varje typ av asynkront meddelande.

Vi kommer senare att berätta att det finns en `Text_stat`-cache. Redan nu kan vi avslöja att den är intresserad av att få reda på när nya texter skapas. Den anmäler därför en handler till ett `Async_text_created_administrator`-objekt (som man anger när man skapar cachen).

När ett asynkront meddelande kommer från servern hamnar det som sagt hos `Async_storage`. Den skickar det vidare till en lämplig administratör (till exempel `Async_text_created_administrator`) som returnerar ett informationsobjekt (`Async_text_created`). Detta objekt lagras på en lista.

När klienten är redo att ta hand om de uppköade asynkrona meddelandena ska `Async_storage::handle_async()` anropas. Den distribuerar ut informationsobjekten till respektive handler via rätt administratör.

Många av de högre nivåerna registrerar handlers för att uppdatera sin interna status när något händer i servern.

Nivå 3: cache

Det är naturligtvis onödigt att hämta information som man redan känner till från servern gång på gång. Därför är det lämpligt att införa en cache som lagrar den information man känner till om ett objekt i LysKOM-systemet. Denna cache består av en klass för varje slag av kom-objekt som finns i systemet.

Varje cache lagrar endast en datatyp. Den döljer fullständigt alla `question:s` som hanterar den datatypen för användaren.

Det finns bara ett fåtal operationer man kan göra på en cache. De två viktigaste är `get(Keytype key)` och `prefetch(Keytype key)`. Typen `Keytype` är oftast ett nummer, till exempel ett inläggsnummer eller ett mötesnummer.

`prefetch()` skickar iväg en fråga till servern (om det inte redan är gjort) och returnerar sedan direkt. `get()` väntar tills resultatet har kommit tillbaks och returnerar resultatet. Naturligtvis kan man använda `get()` utan att ha anropat `prefetch()`

först.

Nu frågar sig vän av ordning hur vi har gjort med felhanteringen. I väntan på att gcc ska börja stödja exceptions har vi gjort en temporär lösning. Objektet som returneras av `get()` har en metod som man anropar för att kontrollera om det gick bra. Den returnerar ett felnummer som normalt ska vara 0.

Vårt lilla exempel ser ut så här om man använder cachen:

```
Conf_cache   CC(&conn);
conf_stat = CC.get(5);
if (conf_stat.errno() != 0)
  abort();
cout << "Resultat med cachen: "
     << conf_stat.name()
     << endl;
```

Nivå 4: session

Arbetsnamnet på denna modul är `session`, men egentligen borde den kanske heta `view`. Denna nivå ger nämligen en direkt vy av LysKOM-systemets tillstånd och döljer protokollhantering och cachening helt och hållet. När[1] protokoll B implementeras ska ändringarna på denna nivå bli minimala.

Implementationen av nivå 1 till 3 har använts av oss ett tag, och vi börjar bli ganska övertygade om att klasserna har ett bra utseende. Nivå 4 och uppåt kommer antagligen att genomgå några designiterationer till innan vi är nöjda. Därför blir beskrivningen i fortsättningen mer nedskrivna tankar om hur vi vill ha det, än hur vi faktiskt har implementerat det. Kanske inser vi att vi ska göra annorlunda medan vi implementerar.

På nivå 4 har man ett objekt, som står för "allt i servern". Det objektet har metoder som `create_text(...)`, `goto_conf(Conf_no)`, `tell_server_what_i_am_doing(char *)` och så vidare. Det finns också några stycken arrayer som representerar olika slags data i servern. Varje array innehåller alla objekt av en viss datatyp, till exempel möten, personer och så vidare.

1. Om...

Dessa objekt är *inte* de som beskrevs under rubriken [Nivå 1: LysKOM-datatyper]. Dessa objekt kan man accessa även om de inte är hämtade från servern. De är "handtag" för dataobjekt, och man kan hålla i ett handtag utan att själva objektet finns i klienten. Metoden `prefetch()` hämtar informationen från servern, men returnerar direkt. Andra metoder som hämtar data ur objektet (till exempel `Conference::name()`) skickar själva iväg frågan till servern (om det inte redan är gjort) och blockerar sedan tills svaret kommit tillbaks. Allt detta sker naturligtvis via cachen.

Implementationen har inte kommit riktigt så långt som vi skulle önska, men så här kan man i alla fall göra just nu:

```
Conferences   all_confs(&conn);
if (all_confs[5].errno() != 0)
  abort();
cout <<
  "Resultat med Conferences: "
     << all_confs[5].name()
     << endl;
```

Nivå 5: klient-specifika klasser

Nivå 1-4 kan användas av vem som helst som vill skriva något som kopplar upp sig mot en LysKOM-server. Vilken nivå man vill använda beror på vad man vill göra. För ett program som bara ska se efter om servern är uppe räcker nivå 2. Om man ska skriva en normal klient har man nytta av allt ända upp till nivå 4.

Men nivå 4 räcker inte riktigt till. En människa som läser KOM vill att klienten skall visa de inlägg han inte har läst automatiskt. Men vad är oläst? Och i vilken ordning skall de presenteras? Sådana frågor kan ha många svar, och nivå 1-4 ger inga svar på dem.

Vi har gjort ett försök att svara på dessa frågor. Vi har konstruerat två klasser, `read_order` och `unread_confs`, som implementerar en enkel läsordning.

Klassen `read_order` hanterar olästa inlägg i ett visst möte. Den returnerar olästa inlägg i en förutbestämd ordning och ger också indikationer på vilket default-kommando som kan vara lämpligt att presentera för användaren (till exempel "läsa nästa inlägg" eller "läsa nästa kommentar"). För närvarande kan våran implementation bara ge inläggen i tidsordning, men interfacet till klassen är gjort så att man inte behöver ändra i kod som använder den när man implementerat kommentarsordning.

Klassen `unread_confs` använder `read_order` och dessutom information om mötesprioriteter för att användaren skall få läsa sina möten i rätt ordning. Även `unread_confs` ger tips om vad default-kommandot skulle kunna vara.

nilkom/tkom

Ovanpå alla dessa klasser har vi lagt ännu ett lager. Det är minispråket TCL (se separat artikel av Inge Wallin). Vår förhoppning är att klienterna ska bli mer lättkonfigurerade och enklare att utveckla genom att ha ett interpreterande språk längst upp. Det är antagligen för tidigt att dra några definitiva slutsatser än, men vi tror att det är ett väldigt lyckat val att använda TCL.

TCL-utökningen Tk gör att det är enkelt att skriva X-windowsapplikationer i TCL. En X-windowsklient har funnits på önskelistan länge, och helt plötsligt blev det väldigt enkelt att skriva en. Många program som använder Tk har namn som börjar på `tk` (som exempel kan nämnas `tkinfo` och `tkman`). Vi följer den traditionen och kallar klienten `tkom`, kort och gott.

Alla vill inte köra X-windows jämt. För dessa personer (som har en representant bland artikelförfattarna) skriver vi även en klient som ska fungera på en vanlig textterminal. För alla som förstår p-konventionen är namnet naturligt: `nilkom`.

De bägge klienterna har nästan all C++-kod gemensam. De kan dessutom dela på en del av TCL-koden. Vi får alltså två klienter till priset av en och en halv, ungefär.

I nästa GARB kommer vi att prata mer om nilkom/tkom, och beskriva hur vi har utökat TCL för att kunna skriva en LysKOM-klient i det.

Referenser

[1] Lars Aronsson, "`/ftp@ftp.lysator.liu.se:pub/lyskom/doc/prot-A.txt`" Dokumentation till protokoll A.

// Demoprogram kom++, augusti 1993.

```
//
// Programmet kopplar upp sig mot kom.lysator.liu.se och ser efter vad
// möte nummer 5 har för namn. Det använder sig av ett antal olika
// metoder för att demonstrera dem.
//
// Det är troligt att vissa saker kommer att ändra sig ganska snart
// (kanske redan innan du läser denna GARB). De saker vi räknar med
// att ändra har vi markerat med "Not 1". Det gäller hur man hanterar
// fel som uppstår när man försöker hämta något ur cachen. Just nu ska
// man kontrollera att foo.errno() == 0. Vi inför kanske någon annan
// metod snart.
//
// Detta program är skrivet av Per Cederqvist. Inge Wallin var med och
// bestämde vad det skulle göra.
//
// Detta program är, till skillnad mot resten av kom++, Public Domain.
//
#include <stdlib.h>
#include <iostream.h>
#include "connection.h"
#include "conference.h"
#include "conf-stat.h"
#include "conf-cache.h"
#include "get_conf_stat_q.h"

int main()
{
  // Denna variabel används i alla exempel för att mellanlagra aktuell
  // mötesstatus.
  Conf_stat   conf_stat;

  // Koppla upp mot servern. Objektet ``conn'' står för förbindelsen.
  Connection conn("kom.lysator.liu.se", "4894");

  // Hämta namn med hjälp av question.
  get_conf_stat_question c(&conn, 5);
  if (c.receive() == st_error)
    abort();
  conf_stat = c.result();
  cout << "Resultat med question: " << conf_stat.name() << endl;

  // Gör samma sak med hjälp av cachen. Först måste vi skapa en cache.
  // I det här fallet skapar vi bara en cache för Conf_stat:tar. Vi
  // kallar den f|r CC och hoppas att Pripps ska se detta och sponsra
  // oss. Det har gått åt många liter Coca Cola medan vi skrev
  // LysKOM...
  Conf_cache   CC(&conn);

  conf_stat = CC.get(5);
  if (conf_stat.errno() != 0)// Not 1.
    abort();
  cout << "Resultat med cachen: " << conf_stat.name() << endl;

  // Än en gång gör vi samma sak (det börjar nästan bli tjatigt nu),
  // men denna gång använder vi oss av en ännu högre abstraktionsnivå.
  // Vi börjar med att skapa ett objekt som står för mängden av alla
  // Conf_stat:tar.
```

```
  Conferences   all_confs(&conn);

  if (all_confs[5].errno() != 0) // Not 1.
    abort();

  cout << "Resultat med Conferences: " << all_confs[5].name() << endl;

  return 0;
}
```

Ny LysKOMserver — lyskomd 1.4

av ceder

Nyheter:

- Upp till 450000 texter kan hanteras.
- Bättre felkoder till klienterna:
  + när man ändrar ett namn eller skapar en ny person/ett nytt möte får man felkoden KOM_LONG_STR om strängen är för lång.
  + Man får ett felmeddelande om man försöker avmarkera en text som man inte hade markerat. (Förut accepterade servern detta utan att protestera).
- Nya protokoll-A-anrop:
  + lookup_person, lookup_conf: Använder den vanliga kom-matchningsalgoritmen för att matcha namn, men till skillnad mot lookup_name returnerar de en lista av mötesnummer (och inte två konstiga arrayer). Nya klienter bör på sikt övergå till dessa anrop.
  + set_client_version, get_client_name, get_client_version: Alla klienter bör använda set_client_version för att tala om vad de är för klienter, och vilken version som körs. Skicka ett mail till kom@lysator.liu.se om du vill registrera ett klientnamn. Filen doc/clients.assigned innehåller registrerade namn.

mark_text, unmark_text: Om dessa funktioner används kan man markera texter med markeringstyp 0. Den funktion som förr användes för att markera texter heter nu mark_-text_old (och har kvar samma nummer som förut, så gamla klientter ska inte märka någon skillnad).

Version 1.4 (se ovan) hade också den skojsiga finessen att följande inlägg (400000) av Per Starbäck numera inte längre existerar, tillsammans med 6000 andra inlägg (hej ceder!)—Red.

```
Jag tycker att detta är rätt tid att komma med en liten berättelse om
just mitt förhållande till dom olika kom-system som finns. Det började
när jag för att bedriva studier hade kommit till Uppsala. Detta var
1983 och faktum är att jag ännu idag kommer ihåg första gången.
Händelsevis råkade strax innan min ankomst databasen gå lite sönder,
så där var förutom det vanliga vaniljkom också ett OLDKOM.EXE med den
gamla databasen. Jag var tvungen att kompensera för att jag börjat
med kom så pass sent med att vara i "koma" med stirrande ögon och ena
fingret på return-tangen i den något komiska pose som sedan blev så
vanlig i långa pass för att hinna oldkom färdigt eftersom det snart
skulle tas bort. Sen var jag förstås komplett fast och var beroende av
kom de närmaste åren. Ungefär som en komet slog jag ner i IÅM för ett
tag sen, så tydligen har jag ännu ej kommit över det stadiet i mitt
liv. Visst har jag i kom tampats med kombatanter som jag hellre klarat
mig utan, men också stött på riktiga kompisar och övat min
uttrycksförmåga, så det har varit en kombination av positivt och
negativt hoppas jag. Snart måste jag komma till slutet av detta inlägg
annars kommer nog detta bara strax bakom någon annans försök, och det
vore ju snopet.
```

Kille-kille-kill

eller

TCL/Tk—

Ett programspråk för programmerbara program

av Inge Wallin[1]

Kraftten ligger i programmerbarheten

Vad är det som gör Emacs så kraftfullt? Varför kan man skriva en vi-emulator i Emacs, men inte tvärtom? Jo, Emacs är programmerbart, dvs man kan utöka Emacs med nya finesser utan att kompilera om det.

Som nästan alla som läser det här känner till så är Emacs programmerbar i en lisp-dialekt som kallas emacs-lisp, eller elisp. Det är en lispdialekt med bland annat massor av inbyggda primitiver för text- och stränghantering och processkontroll. Elisp gör det möjligt att utöka Emacs med så skilda funktioner som:

- språkkänsliga editeringsmoder
- klienter till olika nättjänster som LysKOM, News, irc eller gopher
- stavningskontroll
- ett tetrisspel

Detta att kombinera ett interpreterande språk med domänspecifika primitiver skrivna i något snabbt språk, t ex C eller C++, gör att man får ett flexibelt och utökbart program. Samtidigt så lider det inte av att det interpreterande språket blir för långsamt. Idén är enkel, men egentligen är den ganska sällan utnyttjad. Emacs är ett lysande undantag.

Nästan alla vi som har passerat larvstadiet i vår hackerkarriär har någon gång gett oss på att skriva ett eget språk. De vanligaste språken är FORTH och lisp, dels eftersom det är relativt enkelt att få till en liten kärna av dem, dels därför att de är "rena" och tilltalar oss hackers. Jag har själv hackat på en liten lispkärna som ursprungligen skrevs av Krister Joas[2], mest för att få tillgång till ett enkelt utökbart språk för inbyggnad i mina andra program.

TCL

När ceder och jag började skriva på vår nya LysKOM-klient[3] så ville vi göra något liknande Emacs, dvs lägga ett interpreterande språk ovanpå vårt snabba klientbibliotek skrivet i C++. Jag hade gärna sett att vi använde min lisp som jag nämnde här ovanför. Under tiden hade dock ceder fått upp ögonen för något skumt språk som hette TCL (uttalas "tickle"). Det skulle vara lika bra, eller bättre, än klisp.

Jag ville först inte alls höra på det örat, men efter att ha tittat närmare på TCL så blev jag tvungen att ge med mig. TCL hade nämligen ett antal stora fördelar framför klisp:

- TCL är mycket väldokumenterat. Jag har visserligen skrivit en enkel användarhandledning till klisp, men killen som har skrivit TCL (John Ousterhout) har skrivit en hel bok. Denna bok skall ges ut i vinter, men man kan redan hämta postscripten till den via ftp och skriva ut den på sin laserskrivare. Boken är välskriven och lätt att begripa.
- TCL gör inte en paus med ojämna mellanrum för att utföra garbage collect. Både elisp och klisp har den nackdelen att de måste samla ihop gammalt minnesutrymme ibland, och då stannar alltting för ett kort ögonblick. Detta är egentligen ingen inbyggd nackdel med

1. `inge@lysator.liu.se,ingwa@isy.liu.se`

2. Krister är en gammal lysit som numera bor i Stockholm och jobbar på FFV Testsystems AB.

3. Beskrivs i en annan artikel i detta nummer av GARB

lisp i sig, men för att komma runt problemet så behövs en avancerad garbage collector som jobbar inkrementellt. Sådana finns, men är komplicerade. Varken elisp eller klisp har en sådan och kommer knappast att få heller inom någon överskådlig framtid.

Inte heller en annan känd lisp som är avsedd att byggas in i andra program, xlisp, har en sådan garbage collector.

- Det är mycket lätt att lära sig. Jag kunde visserligen lisp redan, men för den som bara kan Pascal, eller kanske C, så är nog TCL enklare att begripa. Många av de finesser som gör lisp så kraftfullt återfinns i TCL, även om de kommer i något annorlunda skepnad. Man förlorar alltså egentligen ingenting om man väljer TCL i stället för lisp.
- TCL är "standard". Många använder redan TCL och det finns redan ett ganska stort antal utökningar för olika slags domäner. Dessa kan man hitta på ftp-arkiv runt om i världen, och nya kommer hela tiden. Det finns en newsgrupp som heter comp.lang.tcl och där dyker hela tiden upp nya saker. John Ousterhout själv skriver ofta och hjälper nybörjare tillrätta.
- Tk. Den egentligen största fördelen med TCL är en utökning som skrivits av samma person som gjorde TCL. Tk står för TCL Toolkit och det innehåller ett stort antal primitiver för att skriva användargränssnitt ovanpå X windows. Mer om detta senare i den här artikeln.

Bara en datatyp

Hur ser då TCL ut? Jag tänker inte göra några djupdykningar här, utan bara förklara de mest utmärkande dragen. Den som verkligen vill lära sig TCL rekommenderar jag att han hämtar boken via ftp och skriver ut ett exemplar. Jag kan även låna ut min kopia för den som hellre kopierar än skriver ut själv. Kanske boken t o m har kommit ut på riktigt när det här läses.

Det som först slår en är att det bara finns en datatyp, nämligen strängar. Allt lagras som strängar, även heltal och flyttal. Även inuti numeriska uttryck skickas tal fram och tillbaka som strängar. TCL blir alltså inte speciellt snabbt, men det är heller inte meningen. Filosofin är i stället att allt skall vara så lika som möjligt överallt.

Även den exekverbara programkoden lagras som strängar. Det finns dessutom en funktion som heter `eval`, och som i stort sett fungerar på samma sätt som i lisp. Man kan med andra ord konstruera små programsnuttar under exekveringen som sedan kan evalueras med `eval`.

Vi tar ett litet exempel. För att sätta en variabel använder man kommandot `set`. Det kan t ex se ut så här:

```
set a 17
set b 43
set c [expr $a+$b]
```

I sista raden ser vi hur `$`-tecknet används för att hämta fram värden ur variabler. Vi ser också hur `[]` används. Allt inom hakarna evalueras först innan någonting annat görs och resultatet skickas med till `set` som en av parametrarna. Denna substitution kan göras godtyckligt djup genom att ha flera `[]`-inneslutna kommandon inuti varandra. Funktionen `expr` tar, kanske inte helt överraskande, ett numeriskt uttryck och returnerar resultatet.

Om vi vill sätta en variabel till en sträng som innehåller mellanslag måste vi använda krullar, `{}`. Ex:

```
set d {Detta är en lång sträng.}
```

Man kan också använda vanliga citationstecken (`""`), men det finns en viktig skillnad. En sträng inom `{}` blir precis som den står, medan om man använder `""` så kommer delar inom `[]` att evalueras och variabler med `$` framför kommer också att ersättas av sina värden. Ex:

```
puts "a = $a, b = $b och
a + b = [expr $a + $b]"
```

kommer att skriva ut:

```
a = 17, b = 43 och a + b = 60
```

Ett annat sätt att tolka en sträng med flera mellanslagsseparerade[1] tokens i, är att se den som en lista. TCL innehåller flera inbyggda funktioner för att hantera listor av det här slaget. TCL blir alltså lite lispliknande även på det här viset, men eftersom listorna lagras som strängar och inte lagras i något internformat blir det inte riktigt lika effektivt som lisp.

Man kan också använda TCL-variabler som associativa arrayer. Man kan indexera en array med en godtycklig sträng. Ex:

```
set huvudstad(Sverige) Stockholm
set huvudstad(Norge)   Oslo
```

Egna funktioner

Förutom att utnyttja de inbyggda funktionerna, kan man skapa egna. Då använder man funktionen `proc` som tar en definition och lägger in det nya kommandot bland de inbyggda funktionerna. Därefter kan man använda den egna funktionen precis som villket annat kommando som helst. Den välkända funktionen `fak` som implementerar fakultetsfunktionen ser ut så här i TCL:

```
proc fak {n} {
  if {$n < 2} {
    return 1
  } else {
    return [expr $n *
             [fak [expr $n - 1]]
            ]
  }
}
```

Funktionen `proc` tar alltså 3 parametrar: Funktionsnamnet, en lista av parametrar (se ovan om listor) och en lista av exekverbara rader. Hela procedurkroppen är alltså en enda lång sträng, med radbyten och allt! Varje rad tolkas som ett kommando och kommer att evalueras för sig själv. Raden med `if` ovan slutar med ett { och tolkas som en oavslutad sats som fortsätter på nästa rad på samma sätt som hela procedurkroppen gör. Detta kan upprepas hur många gånger som helst inuti varandra.

Förutom `if` finns även alla andra kontrollstrukturer som `while`, `repeat`, `case` och så vidare. Man kan ha lokala variabler inuti en funktion och man kan ange defaultvärden till sina parametrar.

Tk

En riktigt stor fördel med TCL är Tk, som är en uppsättning verktyg för att skapa användardialoger under X windows. Tk innehåller knappar, menyer, ritfönster, textfönster och ett stort antal andra slag av nyttigheter. Gemensamt för dem alla är att man kan komma åt dem från TCL och använda TCL för att ange hur de skall samverka. Det är också extremt enkelt att lära sig och att använda. Hello-world-programmet i Tk kan t ex se ut så här:

```
button .button
  -text {Hello, world}
  -command {destroy .}
  pack .button
```

Den första raden skapar en knapp med texten "Hello, world". Parametern `-command` anger att TCL-kommandot som följer skall exekveras när man trycker på knappen. Den andra raden talar om att knappen `.close` skall packas in i det huvudfönster som öppnas när en Tk-applikation startar.

Fönsterhierarkin hos Tk fungerar ungefär som filsystemet i UNIX, men separatorn är en punkt i stället för ett snedstreck. Fönstret `.` är alltså toppfönstret och `.button` är ett underfönster till `.`. Kommandot `destroy .` förstör toppfönstret och när det sker så avslutas hela applikationen.

På motsvarande sätt fungerar alla widgets i Tk. Man kan registrera små TCL-snuttar som skall exekveras när olika saker händer i

1. Egentligen är det inte bara mellanslag, utan också tab-tecken och newline, men det blir så långt att skriva det varje gång.

en widget, t ex att man klickar med musen i ett fönster, eller om man trycker på en tangent på tangentbordet.

Fördelen med Tk framför t ex Interviews, som är ett klassbibliotek i C++ med ungefär samma funktioner, är att man aldrig behöver kompilera om applikationen bara för att användargränssnittet skall ändras. Man kan ändra sina definitioner i en editor och sedan läsa in dem igen i sitt program samtidigt som det kör.

Alla fönster i Tk har samma utseende som Motif, men det utnyttjar inte någon Motif-kod. Tk är skrivet helt ovanpå Xlib och utnyttjar inga andra toolkits alls. Om man tycker det är en nackdel eller en fördel får väl var och en avgöra själv, men om till och med SUN skall basera sitt Open Windows på Motif i fortsättningen så är det väl inte helt fel att enkelt kunna skapa Motifliknande applikationer.

Hur får man tag på TCL?

John Ousterhout, killen som har skrivit TCL och Tk, jobbar på Purdue University i USA och han lägger alltid upp den senaste versionen för anonym ftp. Både boken jag berättade om här ovanför, TCL- och Tk-distributionerna och diverse utvidgningar skrivna av andra personer finns att hämta på `harbor.ecn.purdue.edu:/pub/tcl`. Den senaste versionen av TCL är 7.0b2 och av Tk 3.3b2.

Där finns också proceedings från en TCL-konferens som hölls i våras upplagda som postscriptfiler. På den konferensen presenterades en massa nyheter som släpps snart, eller som redan har släppts och som ligger på samma ftparkiv.

Bland de intressantare sakerna kan nämnas `Expect`, som är en TCL-debugger. Det finns också en objektorienterad utvidgning av TCL, `[incr tcl]`. Namnet är en ordlek på C++, eftersom motsvarigheten till C:s `++` heter `incr` i TCL.

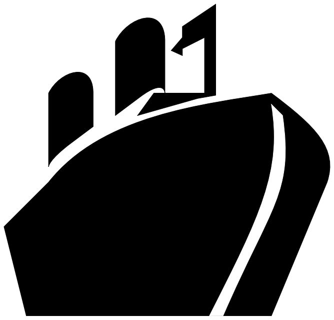
Den som vill titta närmare på TCL/Tk på någon av Lysators maskiner behöver inte göra någonting själv. Såväl bibliotek som körbara program och en massa demonstrationsprogram ligger under färdigkompilerade under `/usr/local/tcl7.0`. Orsaken till att vi inte tog upp det direkt under `/usr/local` är att det redan fanns applikationer som utnyttjar en äldre version av TCL och att ett fåtal inkompatibiliteter har introducerats i version 7.0 av TCL och 3.3 av Tk. När vi har konverterat de gamla applikationerna till den nya TCL-versionen skall den gamla tas bort och den nya läggas upp under `/usr/local` som alla andra bibliotek. Den nya versionen av TCL/Tk är den som beskrivs av boken och antagligen kommer inte nya inkompatibiliteter att introduceras på ett bra tag.

Kontakt!

Svar skickas i slutet e-mail till bug@lysator.liu.se

Generös akad, 179-67, trevl. ing. intr: lysK & allt som gör livet värt att leva.: Söker tjej/qvinna 16-32 för mysiga hemmakvällar framför terminalen. Diskretion utlovas.

— Donald Dick



Resor!

Lysator åker ofta ut på resor för att skaffa datorer eller hälsa på företag och datorföreningar. Här följer några reseberättelser.

Lysiter i Västerled

av Peter Eriksson[1]

och Mattias Olofsson

En historia i 8.5 delar om hur en hämtning av en donerad dator kan utspela sig.

1. Lördag

En vacker vårdag i Mars 1993 begav sig fyra hurtfriska Lysiter ut på en färd som en vecka senare och 2000 km skulle avslutas på samma ställe som den utgick ifrån. De fyra Lysiterna var Peter Eriksson <pen>, Mattias Olofsson <mattias>, Lars Aronsson <aronsson> samt Jens Larsson <jens>.

Alltsammans hade egentligen börjat mer än ett år tidigare då jag (Peter) helt plötsligt fick ett brev från en snubbe i Skottland vid namn Jim Reid där han undrade om vi var intresserade av att ta över en gammal Sequent de hade stående. Visst var vi det, men på grund av diverse byrokratiskt krångel och strul så drog det hela ut på tiden.

När jag sedan precis mer eller mindre hade givit upp tanken på maskinen och bara skulle skicka ett sista brev till Jim för att höra efter så svarade han att han hade fått klartecken för nån månad sedan men hade tappat bort min emailaddress!

Nu vart det snabba ryck med att planera en resa innan sommarsäsongen närmade sig (och båtpriserna skulle stiga) men efter ännu lite mera strul så blev det iallafall av att vi skulle åka, biljetterna var beställda och uthämtade. Enda osäkerhetsmolnet som var kvar var bilen+släpkärra som Jens skulle fixa...

Nåväl, det löste sig iallafall och på lördagsmorgonen så hämtade Jens upp oss andra vartefter vi åkte förbi Q-huset för att hämta lite nödvändiga attiraljer såsom spännband, verktyg och Coca-Cola. Där mötte vi Pär Emanuelsson <pell> och Jan Danielsson <jan> som skulle åka en sväng till Norge för att hämta en del hårddiskar som Lysator skulle få.

Vi passade även på att provlasta släpkärran (som såg oroväckande liten ut) med ett Nanny-skåp, och då det provet utföll till belåtenhet så begav vi oss ut på vägarna mot Göteborg.

Strax efter Jönköping så tyckte vi att det var dax för en liten paus så vi stannade till vid en bensinstation. Vi passade då på att inhandla lite glödlampor och säkringar till bilen och släpkärran då ett antal av dessa var trasiga samt påbörjade en smärre felsökning för att ta reda på varför säkringen till lamporna på släpkärran hela tiden gick så fort man bro-

1. `pen@lysator.liu.se`
   `mattias@lysator.liu.se`

msade. Vi hittade dock inget fel så vi satte i en säkring högre strömtålighet och körde vidare.

Väl framme i Göteborg skulle vi försöka leta oss fram till hamnen vilket visade sig inte var helt utan problem. Speciellt inte utan bromsljus (man undvek då gärna att bromsa kraftigt med bilar bakom). Vi kom fram till hamnen med 10 minuter till godo, varefter vi kör ombord på båten och parkerar bilen.

Mattias, Lars och Jens håller sig flytande

Därefter utbryter den vanliga aktiviteten ombord på båtar, man stuvar in sina prylar i de små hytterna, man går och käkar och besöker de saker som finns att besöka. Efter maten så kör tar vi och bastar och kryper så småningom i säng för att vara utvilade inför morgondagen.

2. Söndag

Vi vaknar vid 11-tiden och avnjuter en frukost ala Aronsson i hytten varefter vi tar och beger oss ut bland de andra människorna. Allt känns lagom och vi räknar med att vara iland i lagom tid, sådär vid 14-tiden. Gott om tid att ta oss till Manchester (en resa på sisådär 7 timmar) så at vi ska komma fram i vettig tid.

Trodde vi ja, medans vi sitter där börjar man ropa ut i högtalarna en massa anrop efter en viss Johan Brown, och efter ett ytterligare tag så får vi reda på att man har vänt båten för att söka efter killen som de tror har hoppat/ fallit överbord.

Då inser vi att det här kommer att ta tid så vi inhandlar en kortlek och börjar fördriva tiden bäst vi kan. Efter många timmars väntan ger man slutligen upp sökandet och båten får åka in till kaj. Vi är iland vid 21.30 och inser att vi inte har en chans att hinna till Manchester den kvällen. Och det blir svårt att hitta någonstans att övernatta när det är så sent. Så vi beslutar oss för att försöka spendera natten med att köra. Aronsson får den roliga uppgiften att vara den första personen som kör bil på vänster sida utav oss.

2 1/2. Natten

Vi åker en liten bit från Harwich och tycker att vi måste nog ringa till Dave Shield i Manchester och meddela att vi inte kommer på kvällen, så vi svänger av in till Colchester där vi hittar en McDonalds och en telefonkiosk. Vi avnjuter några hamburgare och ringer till Dave. Därefter byter vi förare och Mattias börjar köra mot Manchester. När vi börjar närma oss Cambridge så inser vi att det är dax att fylla på med lite bensin i bilen.

Vi tänker att, klart det måste finnas en BP-mack i Cambridge (vi hade nämligen varit så smarta att vi redan hemma i Sverige såg till att skaffa oss en sådant där BP-bensinkort som skulle funka i hela Europa ("för BP måste det ju finnas gott om i England/ Skottland, BP är ju ett brittiskt företag" !) så vi skulle slippa ha en massa kontanter med oss. Efter två och ett halvt varv runt och i

Cambridge klockan 0200 så insåg vi vårat misstag... Shell och Texaco och en massa andra fanns det men inga BP. Nåväl, vi fyllde på en del på en Shellmack och sedan fortsatte jag köra mot Manchester. (Vi hittade faktiskt en del BP-mackar lite senare även om de inte var många).

Därefter kör jag bil, kör jag bil, kör jag bil, vi tankar, jag fortsätter köra bil, kör bil, kör bil, passerar Birmingham, kör bil, kör bil, kör bil (under hela den här tiden regnar det och/eller är dimma), kör bil, kör bil, kör bil tillsammans med en massa lastbilar och sedan framåt morgonen så kommer vi fram till Manchester vid 5.30-tiden. Vi beslutar då att åka förbi Dave och se om de kanske vaknar snart (och göra lite besök på toa å så).

Efter lite snurrande runt en rondell ett par varv så hittar vi till slut hem till Dave's hus. Vi tycker att det kanske är i tidigaste laget att knacka på klockan 0600 så vi tar en liten promenad bland husen. Vid 06.45 tycker Aronsson att nu är det nog läge att storma in hemma hos Dave så då tar vi och ringer på. Efter ett litet tag så öppnar en yrvaken kille och hälsar oss välkomna. Vi får lite frukost vilket inte sitter fel.

3. Måndag

Efter frukosten beger vi oss ut på vägarna, nu med Aronsson bakom ratten. Det är relativt bra väder nu så det är ganska trevligt att sitta i baksätet och titta på landskapet som far förbi. Ett antal timmar senare passerar vi gränsen till Skottland och nästan genast börjar det regna! För att sedan bli solsken! Och regn! och sol! och...

Vid 12-tiden är vi framme i Glasgow i strålande solsken, vi passar på att tanka i en splirrans ny BP-mack vid infarten till Glasgow och besöker en turistinformationsbyrå som ligger bredvid.

Därefter beger vi oss in mot stadens centrum för att leta reda på Strathclyde University och Jim Reid. Knappat har vi parkerat bilen förrän det börjar ösregna som sedan övergår i hagel, sol, regn, hagel... Skottland och Glasgow har väder, det är tydligt!

Vi tar och letar reda på en järnaffär och investerar i hänglås och kätting så vi kan låsa fast släpkärran vid bilen, vartefter vi ringer runt till en del Bed&Breakfast-ställen för att hitta några rum åt oss. Efter några försök får vi napp så vi går dit och bokar rummen åt oss. Damen som tar emot oss undrar om vi tycker om fotboll. Jaa.... det gör vi väl... tycker vi och sedan undrar hon över att vi inte är tyskar? Lustigt tycker vi, men det visar sig snart att anledningen till att det var så svårt att få tag på rum var att Skottland skulle spela fotbollslandskamp mot Tyskland på tisdagen!

Därefter tar vi och irrar runt lite på universitetets område med baktanke att leta upp några hackers ("för det finns nog alltid några hackers sittande någonstans, det brukar det göra"). Det enda vi hittar är några tomma PC-salar med 286-or men annars är det väldigt folktomt. När vi sedan börjar misströsta att hitta några och går ute på universitetsområdet så kommer det en tjej gående mot oss. När hon är framme hos oss så hälsar hon på svenska och undrar vad Linköpings-teknologer gör i Glasgow (vi hade våra teknologmössor på oss, givetsvis)! Det visar sig vara Helena Svensson, före detta kårpamp på LinTek som gör ex-jobb på Strathclyde. Hon tipsar oss om studentpuben Lord Todds.

På kvällen besöker vi studentpuben och insuper atomsfären, trots att vi är ganska trötta vi det här laget. Det här var en viss skillnad mot Herrgår'n en måndagskväll. Fullt med folk och glatt och trevligt. Helena och en väninna till henne kommer dit så vi sitter och pratar hela kvällen. Vid midnatt så tar vi en taxi hem, för den otroliga kostnaden av 2 pund.

4. Tisdag

Vi försöker ringa Jim Reid på morgonen för att bestämma när och var vi ska träffas. Ingen som svarar. Jim är tydligen ganska morgontrött. Vi går till bilen, och upptäcker där att Jens har glömt nycklarna på rummet. Jens får promenera mera (tog 20+20 minuter att gå den biten).

Under tiden njuter vi av det fina vädret som är (inget regn den här dagen) och samtalar med den mycket trevliga parkeringsvakten om Glasgow och Glasgows historia. Vi får även tips om en bättre parkering inför nästa natt som hade bevakning även nattetid.


Jens, Lars och Mattias poserar

När vi har flyttat bilen till denna nya parkering tar vi och går till Jims kontor men han har då inte kommit heller. Mycket morgontrött kille :-) Vi snackar med en kompanjon till honom tills han dyker upp. Därefter flyttar vi ut datorn från datorhallen och bygger om den något för att bättre passa våran ganska minimala släpvagn.

Därefter går vi runt på stan ett par timmar innan vi får för oss att åka ut på tur till sjön/nationalparken strax norr om Glasgow, Loch Lomond. Det visade sig att det nog var i senaste laget att bege sig ut, då det var precis kolsvart när vi kom fram dit. Men man kan ju inte få allting :-)

Vi åker tillbaka och går till Lord Todds där vi träffar en trevlig norrman som läste protesbyggarteknik. Det visade sig att det fanns en stor koloni med Norska studerande vid Strathclyde.

5. Onsdag

Vi käkar ännu en god frukost på B&B, knallar till bilen och kör upp den till huset där Computer Science huserar. Funderar på vart vi ska ställa den medan vi lastar då det fanns gott om skyltar som indikerade att det var förbud mot att parkera bilar på just den gatan. Frågar Jim som bara skrattar och sade att det bara var att strunta i, om polisen såg att man höll på att lasta så struntade de i att sätta dit folk. Klart sympatiskt folk de där skottarna. Även parkeringsvakterna hade en vettig syn på livet (det gör inget om ni kommer efter att tiden ni har betalat för går ut, kom när ni kommer, det är lugnt..)

Vi packar in datorn i släpvagnen, ger våran lilla tack-gåva till Jim, tar bilder, parkerar bilen och går och käkar billighetsmat på 'Student Union'. Men mätta vart vi. Därefter försöker vi reparera våra bromsljusen och blinkers men lyckas bara delvis. Efter diverse ombyggnationer så får vi igång blinkersen men då funkade inte bromsljusen alls längre. Dock brände vi inga säkringar heller. Medans vi håller på med detta kommer Jims medhjälpare ner till parkeringsplatsen med ett extra kretskort som han hade hittat efter att vi hade givit oss iväg.

När vi har slutat fippla med bilens elsystem åker söderut från Glasgow mot Manchester. Kommer fram dit framåt kvällen och spenderar kvällen med att äta mat, prata med Dave och fru, ger dem en liten tackpresent för att vi får övernatta hos dem och sover sedan på golvet i deras vardagsrum.

6. Torsdag

Äter frukost och åker sedan i ottan vidare söderut mot London med Jens bakom ratten. Och åker, och åker. "Vi byter förare efter Birmingham!" - oj, var det visst dåligt med rastplatser på den här nya fina motorvägen den har byggt. "Det kommer nog en snart". Jens kör vidare. "Vi byter innan London!". Oj, vi var visst redan framme i London...

Jens kör och kör och kör...

Efter att vi har läst in oss på vägbeskrivningen hem till Andrew Bennett och kollat på en Londonkarta i god tid innan, plus att vi är fyra som håller utkik efter vägarna i fråga så kör vi lika förbenat fel. Efter 17 svängar hit och dit så kommer vi slutligen rätt och möter Andrew. Vi lastar in datorn i hans hus på grund av den stora stöldrisken i området han bor i.

Därefter drar vi ner på stan för att turista en kort stund (1 timma) men hinner iallafall besöka Foyles bokhandel. Tar tunnelbanan tillbaka till Andrew och följer sedan med honom till en Indisk restaurang med en intressant framtoning (Mycket Mat! Och Fort!). Efter att ha proppat i oss mat så besöker vi en pub innan vi drar oss tillbaka för att försöka sova lite.

7. Fredag

Upp tidigt (en timma tidigare än vi räknade med att det skulle ta att komma ut ur London och ta oss till Harwich), käkade lite frukost och lastade sedan in hela datorn i bilen igen. Jag fick förtroendet att lotsa oss ut ur London. Vi var på rätt väg i hela två kvarter (vi fick en jättebra vägbeskrivning av Andrew, samt en _detaljerad_ Londonkarta i bokformat) innan vi körde fel... Vi fortsatte dock att köra i ungefär rätt riktning och hittade till slut ut. Sedan var det motorvägen i full fart till Harwich, och dit kom vi med 15 minuter tillgodo innan man skulle vara där...

Besvikelse, tullen tittade inte ens på vad vi hade i lasten, och vi som hade förberett diverse papper och dokument... Nåja. Väl ombord på båten så mötte vi samma mogna 9:e-klassare som på vägen till England. Intressant folkslag de där svenska 9:e-klassare. Toppluva på—inomhus. På vägen upp var de två, på vägen hem hade de forkat av sig till kanske 5-6 stycken. Men roligt hade vi åt dem iallafall :-)

8. Lördag

På morgonen (nåja sen förmiddag) vaknade vi sedan och upptäckte att vi nästan var framme i Göteborg. Vi hann precis lagom kliva upp förrän det var dax att köra av båten. Strålande väder och tulltjänstenmännen var på riktigt gott humör. "Vad har ni i släpkärran?" "En gammal dator vi har fått från ett

Maxwell lastas ur utanför Q

universitet i Skottland." "Inget annat?" "Nej" "Då kan ni åka." (De tittade inte ens!)

Vi beslöt att åka förbi CD (Chalmers Datorförening) och hälsa på när vi ändå var där. Åkte dit, hälsade på och käkade sedan pizza tillsammans med en kille från CD vars namn jag inte kommer ihåg... Därefter hoppade vi in i bilen och åkte vidare hemåt. Framåt kvällen var vi sedan hemma i Linköping där vi lastade ur datorn och åkte hem och sov, sov, sov...

Hämtresan till CMArentals.

av Magnus Redin

Den 16:e juli åkte jag, Jens Larsson och kjell-e till göteborg för att hämta en donation på CMA rentals. CMA lever på att hyra ut vaxar och SUN-arbetsstationer, jag tror de stod för de flesta maskinerna på SUN-expo. Nu hade dom fått över lite gamla 11/750 vaxar, reserdelar och gamla diskar och beslutat att vi kunde få grejerna i stället för skroten.

Själv var jag till en början tveksam till att hämta så stora burkar med så lite CPU-kraft, men eftersom kjell-e och Jens var frivilliga till att ha dom hemma (Vi har ingen ström kvar i Q-huset) och använda dom som terminalservrar så var det idé att hämta dom. Nu har Stångåstaden mumlat lite om att några korridorer påminner om datorhallar men det rer sig nog när dom blir färdigreparerade.

Färden ner till götet med en VW pickup var händelselös så när som en guldkommentar. På en mack så undrade jag ljudligt vem som kunde vara så pantad att ett CD ställ behövde skylten "OBS TOMMA FODRAL" varvid mackägaren replikerade: Fotbollsupportrar!

I göteborg hittade vi med hjälp av ett par hjälpsamma bensinmackar fram till CMAs kontor. Där berättade Petter Gräns en hel del anekdoter om gamla vaxar medan vi fick gå igenom deras förråd av skrotningsfärdiga prylar, vi fick två 11/750 maskiner, fyra RA60 diskar, två skrivare, blandade reservdelar och lite smått och gott. (Ibland känner man sig verkligen motarbetad av datorföretagens inbyteserbjudanden—CMA hade skickat tillbaka flera containrar med Sun-2 och Sun-3 till SUN!)

Efter en kopp kaffe tackade vi för gåvan och åkte in till Chalmers för att hälsa på CD, Chalmers Datorförening. Där lånade vi terminaler och fick tipset att sova över på elektros soffor. Dom var obekväma och man väcktes ständigt av folk som gick förbi men billigt var det. På eftermiddagen tittade vi på avenyn och besökte göteborgs bokantikvariat och fyndade. Jens var tvungen att åka tillbaka samma dag och tog tåget till Linköping. Kvar vart jag och kjell-e som besökte Lisebergs alla folkslungemaskiner utom bergbanan som var avstängd.

Efter lite mera LysKOM på CD och några timmars sömn på elektro åkte vi efter att ha konstaterat att ingen hade stulit någon 11/750 från kapellflaket. Om någon hade orkat så hade tjuven varit förtjänt av den... På vägen hem så åkte vi slumpmässigt av väg 40 vid Hökerum mellan Borås och Ulricehamn. Jag chansade på att det skulle finnas ett fik och det gjorde det: en nybyggd liten rosa stuga som finns nedanför Sparbanken.

Den var fullare än lysfika pga att ägarinnan i all hast öppnat lanthandel sedan byns butik lagts ner. Nostalgi att se en handikapptoalett användas som läskförråd ;-) Där fick vi kaffe i porslinskanna med rokoko-koppar. Två-tre koppar kaffe och två varma bullar kostade mig 12:- och så hade dom en söt servitris som studerade på lärarlinjen i Linköping. Glöm inte att besöka Hökerums fik när du har varit i Göteborg och hyrt datorer av CMA!

Efter det så stannade vi och tittade på f.d. A6s försvarsmuseum i Jönköping. Sedan var vi snart hemma och hämtresan och denna sommars semester var slut.

Magnus Redin.

RA60, the choice of a new generation

eller

Göteborg, tur och retur utan otur under 13 timmar[1]

av Peter Johansson

Där sitter man helt oskyldigt en tidig måndag morgon (c:a kl 02) och filar på sitt kom-läsar-script (för att dressera den snälla terminalen (läs: Amiga) hemmavid) när det plötsligt dyker upp ett allmänt meddelande från Redin som undrar om det finns någon frivillig medchaufför som vill åka med till Göteborg för att hämta en donation. Tja, har man semester och inget bättre för sig så, "Varför inte?", tänkte jag. Sagt och gjort, snabbt bestäms avfärd kl 08.55 från Ryd C. Dags att sova. Kl 08 ringer klockan och som genom ett mirakel vaknar jag av det (första gången på flera veckor som jag har försökt gå upp så tidigt, det är skönt med semester) och har gott om tid på mig att äta en ordentlig frukost vilket senare visade sig vara vist.

Vis av erfarenheter genom genomläsning av tidigare reseberättelser ringer jag och kollar med Redin att han är vaken. Det var han. Han hade till och med ringt till Petter på CMA rentals och sagt att vi skulle komma. Kl 08.55 står jag vid Ryd C. Ingen Redin. Men det dröjde bara några minuter så dök han upp där. Men det var inte dags att åka på en gång, först behövdes några gåvoavtal skaffas fram. Det där med några gick inte fram i min morgontrötta hjärna så medans jag skriver ut *ett* ex går Redin iväg för att ta ut pengar på bankomaten så att vi kan tanka innan avfärd. När jag sedan får skriva ut några ex till så hinner Redin äta frukost i Q-huset också. Men bankomaten ville inte ge Redin några pengar så det blev till att försöka tanka på bankkort (nåt konstigt modernt påfund som Redin har skaffat sig, tacka vet jag hålkort, synd att man inte kan tanka med dem). Men då en hutlöst ockrande telefonapparat på Texaco vid Vallarondellen har fått Redin att bli sur så fås ett andra försök göras vid Statoil istället. Andra gången gillt i den här sagan, tur för Statoil det. Det kan löna sig att låna ut telefon gratis till Redin som glad som en boll studsade ut från kassan och tankade bilen full.

Så äntligen var vi på väg, på väg, på väg (eller hur många lager asfalt det nu var). Mycket väg var det vi var på, jag räknade dock inte efter hur mycket väg det var, det lämnas som övning åt läsaren. När vi hade varit på väg så mycket att vi passerat Jönköping och Redin hade kört "fel" så att vi hamnade på flygplatsen var det dags att byta chaufför för att komma vidare så jag satte mig vid ratten. Innan Redin slumrade till lite mumlade han till lite om "1 mil efter Ulricehamn", "Komskälet", "Hökerum", "fik". Jaha, det är väl bäst att se till att Redin är vaken när någon av de uppräknade sakerna inträffar. Det som inträffade först var passerandet av en skylt med ortsnamnet Komskälet på. "Komskälet", sa jag och Redin slog genast upp ögonen och blev klarvaken och sa, "Nej, vi ska inte svänga in där". Det var rätt för det var till Hökerum vi skulle svänga in. Avfarten mot Hökerum från väg 40 ligger som Redin sa ca 1 mil efter Ulricehamn mot Borås. När man åkt någon kilometer från väg 40 dyker en Sparbank upp, där svänger man in till höger och efter några hundra meter syns ett rosa hus på höger sida. Det är Hökerums fik och kiosk/minilivs.

Ont om plats är det ganska gott om inne i lokalen men utbyggnad planerades. Det är ett trevligt litet ställe där man lite försynt ber om en kopp kaffe och den trevliga personalen går och brygger en kanna kaffe som serveras i riktigt porslin ute vid bord med tak över huvudet (option som var påslagen vid tillfället pga risk (våran bedömning) för lokalt mycket hög luftfuktighet, annars finns det bord utan tak ovanför för utnyttjande företrädelsevis vid soligare väderlek). Det är lite annan stil än vid fiken

1. Detta är den andra resan till CMA - Red.

längs med väg 40! Rekommenderas å det varmaste om man inte är en sönderstressad yuppie (men vem är det?) som inte kan vänta några minuter på att kaffet blir klart. Till kaffet blev det en bulle eller (i Redins fall) två. Som om det inte vore nog så var det oförskämt billigt också, hur billigt tänker jag inte tala om utan det får ni upptäcka själva om de inte har lärt sig ta betalt ordentligt tills ni kommer dit! :-)

Efter det uppiggande kaffet fortsatte färden mot resans mål, fortfarande (nej, inte fort farande) med mig vid ratten. Åtminstone tills jag av nöd gjorde en hastig avfart mot Landvetters flygplats för besökande av bekvämlighetsinrättningen i vänthallen. Efter det, byte av chaufför eftersom det var Redin som eventuellt hittade till CMA. Det visade sig att det var just så som var fallet. Utan minsta felåkning (vad jag kunde se) hamnade vi rätt direkt. På CMA blev vi välkomnade av Petter Gräns som utrustade oss med verktyg och visade oss högen med datorprylar som vi fick gå loss på. Resultatet blev 4 st RA60 (varav en med diskpacke, resten utan), ca 8 trevliga smålådor med kragg och D-subbar (nån sorts terminalkopplingsbox, litet kretskort med Z80-kretsar på inuti) lämpliga för små hårdvaruhack, en 11/750 diagnosbandare, lite blandad sladd och ca 8 analoga joystickar.

Efter CMA blev det en kort visit hos CD för att se hur de hade det och dessutom KOMma i kontakt med hemmafolket i Linköping. Nu började klockan bli mycket i förhållande till födointaget så nästa besök blev en pizzeria (vi hittade ingen tidsmaskin) där intagande av lasagna justerade det missförhållandet. Sedan bar det av hemåt med Redin vid ratten i den nedlastade Volvon. På hemvägen lyckades vi undvika alla mellanlandningar på flygplatser och enda förarbytet var i Hökerum där vi naturligtvis fikade igen (öppet till kl 22 tror jag det var, men vi var där mycket tidigare). När vi kom fram till Q-huset lyckades vi lura in några intet ont anande bärare i noppes bil och så bar det iväg till Ryd för avlastning. Det var nog tur att de inte uppfattade att noppe sa att han inte skulle tillbaks till Q-huset efter avlastningen utan skulle hem...

- prj

Sponsorer

Lysator vill tacka alla sponsorer, nya som gamla. Utan sponsorer skulle inte Lysator finnas till i den form vi idag känner. Aktuella sponsorer är framförallt:

Strathclyde University, Glasgow, Skottland:	**Maxwell**
Stråbruken AB	**Hilbert&Hubert**
Svenska Fujitsu AB	**Diskar**
Televerket	**Vitalink**
LysCom AB	**Mikrovågslänk**
Sun Svenska AB	**Datorer**
Signum Support AB	**GNU-manualer**
Digsim Systems AB	**Digsim**
IFM	**Dumle och en 11/780**
ISY, IDA, FOA, LIDAC	**Diverse, ethernet-kablar**
CMA Rentals,	**VAXar, diskar, tillbehör**
Elemtell	**Ethernettranscievrar**
ISY, IDA, IFM, IKP, MAI, KBS	**Ett hus!**

KOMpisklubben är ute och reser.

av kjell-e

Vad är KOMpisklubben (för den som inte redan vet)? KOM är från början ett program från QZ som körde på stora DEC-maskiner, men numera finns även andra liknande system som tex Lysators LysKOM. I KOM och LysKOM diskuterar man allt mellan himmel och jord som datorer, elektronik, politik, mat, relationer, psykologi, pedagogik, fysik mm. För flera år sedan fanns en DEC20 på LiTH vid namn Linus, vilken körde KOM och där skapades KOMpisklubben.

KOMpisklubben var en lös sammanslutning av folk som körde mycket KOM och som ville träffas utanför datorernas värld. Bland de arrangemang som genomfördes fanns bla tårtkalas tex när någon fyllde år, innebandy, badminton och en avskedsfest för våran DEC20 Linus när han skrotades. KOMpisklubben har dock levt vidare i och med LysKOM och träffas alltjämt och firar nyår, valborg, midsommar, födelsedagar, har våffelkvällar, varma mackorkvällar, och åker till fjällen.

Nåväl, i våras tyckte jag att det var hög tid att ge mig ut på en fjällvandring igen då det var ca 10 år sedan sist. Annonsering i LysKOM gav till resultat att ca 5 personer var intresserade förutom jag själv. Tyvärr pga diverse omständigheter blev vi bara 2 till slut, en Lysit och en Luddiot (Systerförening till Lysator vid Luleå tekniska högskola). Först en liten sväng upp till Stockholms skärgård för att fiska och bada några dagar och sedan med tåg från Stockholm central till Luleå.

Resan tog ca 14 timmar, men det fanns ju bio ombord och så en man som jobbade med att serva SJ's RC-lok i Luleå att prata med. Väl framme blev jag mött av Jörgen och vi åkte bort till kårhuset för att dumpa mina prylar i ragges korridorrum, vilket han fortfarande antagligen inte känner till att jag lånade. En tur till Ludd för att köra KOM följt av middag där det bjöds på Palt, vilket var första gången för mig men gott var det. På lördagen försökte medeltidaföreningen att resa sitt tält som de hade sytt ihop vilket faktiskt lyckades. Så var det dags att handla mat och de prylar som behövdes, plasta in kartan och packa och så sova.

Söndag 25:e Juli begav vi oss iväg mot Kiruna, och en sak är då säker, många och långa vägarbeten fanns det. Det längsta var en grusväg på 1,7 mil. Dessutom passerade vi förbi en liten by där det gällde att se upp så man inte råkade missa den när man passerade, men ett trafikljus hade de i alla fall. Kiruna var just då inte direkt den varmaste staden med sina 11 plusgrader men tack och lov så blåste det inte så mycket så det gick att gå omkring i shorts och t-shirt, vilket vi var ganska ensamma om förutom en typisk fransman som gick omkring i shorts och täckjacka. Tanka bilen köpa mat, äta ett superskrovmål och så vidare mot Nikkaluokta via ännu fler vägarbeten.

Lagom framme efter 40 mils biltur så börjar det regna, men har man bestämt sig för att ge sig av så kan inte lite regn stoppa en. Bort till fjällstationen vid Kebnekaise är det ca 1,9 mils vandring på en bred och väl upptrampad led vilken närmast kan liknas vid en mindre motorväg så mycket folk som går där. Blöt och hungrig kommer man fram till fjällstationen och då känns det jätteskönt att kunna byta om till torra kläder, hänga in de blöta i torkrummet, och så äta. Dessutom lyckades jag med konsten att bli utkastad från fjällstationen, men det berodde på att de skulle stänga och ingenting annat.

Var hittar man då en lämplig tältplats? En kort vandring upp på ett litet berg bredvid gjorde att vi fann en alldeles utmärkt plats på en liten platå en ca 50 meter ovanför fjällstationen med en alledeles ypperlig utsikt om det bara hade varit lite bättre väder. Somna var inte så svårt att göra, och då vädret nästa dag var ungefär detsamma som dagen innan beslöt vi oss för att ta en kortare tur runt den närmsta trakten i avvaktan på en toppbestigning. Den här dagen hanns det dessutom med att bada lite bastu och ringa hem. En dam som skulle ringa hem till Småland lyckades ringa fel och hamna på Esrange, bara några mil fel.

Tisdag 27:e, nu skulle vi upp på toppen. En guidad tur avgick 8.30 och sällskapet var ca 45 personer som begav sig av. Första biten är ganska enkel och inte så jobbig att ta sig fram på, men det är sedan när man kommer till första snön som det börjar. Tack och lov så tar guiderna paus efter ca 20 minuters vandring vilket behövs då det lutar ganska så mycket uppför och är nog så jobbigt att gå. Värsta biten var nog innan andra snön, för efter den snöfläcken så lutade det inte fullt lika mycket längre och det var här som plötsligt molnen sprack upp ett tag så att det gick att se långt ner i dalgången. Tyvärr höll inte det i sig utan det mulnade på igen och blåste ganska så bra så efter ett par timmars vandring då vi var på ca 1580 meters höjd tycke guiderna att det inte var någon mening med att fortsätta längre. Nåja expedition 2 sommaren 1994 ska lyckas bestämde vi då.

Nerför var det betydligt lättare och mycket roligare. Först går man en bit över glaciären för att sedan ta på sig regnstället och åka kana ett par hundra meter utför på snön. Alla blev som barn på nytt och åkte för glatta livet. Nu vet vi så här i efterhand att ska man på en sådan tur så är det bra att ha ordentliga stövlar med bra mönster, regnställ, handskar, en varm tröja, en mössa eller huva på jackan, termos med varm dryck, mackor, och så en åklapp att sitta på under nerfärden för bästa hastighet. Det är också efter det här som den nya 6-kampen har kommit till vilken lyder, 1,9 mils fotvandring från Kebnekaise till Nikkaluokta, 40 mils bilkörning, fotografera dig själv eller någon annan vid polcirkeln, äta ett skrovmål, och åka raggarrundan i Överkalix.

Den sistnämnda var den mest komiska delen. En kort gatstump där de åkte fram och tillbaka hela tiden i sina volvobilar, några nersänkta (patetiskt). Fast det gick också att åka runt om man fortsatte gatan fram en bit till och sedan svängde in till höger och så höger igen lite längre fram. 6-kampen avklarades på tiden 9 timmar och 45 minuter.

En tur till Luleå innebär oftast också en tur till Haparanda eller Haaparanta som det heter på finska och över bron till finska sidan. Där i Torneå finns trevliga butiker som heter Oy ALKO Ab på vilka det går att köpa Minttu, Gin Longdrink mm. Priserna är ungefär som de svenska men det finns sorter som inte Systembolaget har. Förutom att åka till Finland passade vi på att hälsa på Bettan, ännu en Luddiot, som guidade folk vid kyrkan i Gammelstad, och så naturligtvis järnvägsmuseet, Malmbanans vänner.

Helgen närmade sig med stormsteg och Storsjöyran likaså. Packa bilen igen, hämta Bettan, fortsätta vidare förbi Piteå, vidare förbi alla vägarbeten till Skellefteå där nästa Luddiot skulle hämtas. Mil på mil, solstrålarna stod som spön i backen ända fram till strax innan Östersund, där kom regnet. In till stan och bort till bussstationen, men var håller Robert från Lysator hus ? Han skulle ju vara här. Nåja bort till järnvägsstationen och möta de andra, men hur hittar man dit. Leta, leta, hitta kårhuset, hitta Storsjön, men ingen station. In på turistinformationen och hämta karta. Ah, inte så långt bort men säkrast att ta bilen. Där minsann fanns Robert och de andra, nåja nu skulle vi till nästa Luddiot och inkvartera oss hemma hos hans föräldrar.

Maken till gästfrihet har jag inte varit med om på länge. Förfest i en stor jättemysig träkåk inne i stan. Antalet festdeltagare steg med ca 100 % då vi anlände. Vissa lokalinvånare fick prova på våra specialdrycker men blev bara konstigare och konstigare, vilket var märkligt då vi inte påverkades alls på samma sätt. En guldkommentar var då en person försökte avgöra kryddan i Vodkan och kom fram till att det var sån där Turklakris om den turkiska pepparen som det egentligen var. Även här lyckades man bli utkastad men även denna gång pga att de skulle stänga.

Trapphusfest för att tömma flaskorna eftersom dessa inte fick föras in på yranområdet. Bästa kommentaren där var från han som drack ren Lime och tyckte att det var gott och att det inte smakade speciellt mycket alkohol för att genast ta en klunk till. Bort till yran i regnet, där Carola sjöng och inne på en klubb spelade sextor sex, ett punkband. Själv smet jag iväg och tittade på DiLeva. Senare framåt kvällen spelade bla Wilmer X.

På lördagen tog vi en liten tur runt stan i det strålande vädret. På kvällen var det så dags för förfest igen. Någonting vi fick lära oss då var att Svarta döden inte är speciellt lämplig att blanda med. Många konstiga häxblandningar kom till världen den kvällen. Flaskor fick man inte ta med sig in på yranområdet vilket Jörgen fick erfara då de tog hand om hans öl. Lagom tills kvällsaktiviteterna börjar så börjar även regnet, typiskt.

Fler bra band spelade däribland Euskefeurat, Inner Circle, Lisa Nilsson, Peps Persson, Quireboys, Atomic Swing, Ulf Lundell mfl. Vid halv tolv höll presidenten Ewert Ljusberg tal för alla Jamtar i republiken Jamtland. Inne på puben försökte Jörgen och Charlotte att stjäla skyltar men blev utkastade. Dock var det så att puben hade två ingångar så de gick bara in andra vägen.

Söndag, sova ut. De som skulle norrut åkte iväg, Ragge och Charlotte åkte iväg till Sundsvall, kvar var bara jag och Robert som beslöt oss för att ta natt-tåget till Stockholm och därifrån vidare hem till Linköping.

Ludd—

Luleå Högskolas Datorförening

av Bettan

Lysator är inte den enda universitetsdatorföreningen i Sverige!! Även här uppe bland isbjörnarna har det skapats en som av outgrundliga skäl givits namnet Ludd. Bakom den står i första hand ett gäng outtröttliga D:are men vi har medlemmar från alla Luleå Högskolas linjer och gör allt för att skapa möjligheter och aktiviteter för alla. Vi har en lokal i ett av skolans hus där vi i första hand kör Sun-stationer. Vi har en stor ftp-bas i samarbete med skolan och vi kan oxå inbjuda till LuddKOM, Ludds egen version av Lysators KOM...

Kontakterna med Lysator har hittills varit mycket goda och även om de mestadels gått över dator har vi oxå varit och hälsat på varandra, bytt datorer och datordelar mm.

Lokal: D348-352, Högskolan i Luleå, Porsön.
Maskot: Vicke, ett stridslystet litet datavirus.
Antal medlemmar: Nu innan hösten ca 250.
Medlemsaktiviteter: Förutom allt som hör hackandet till årsfest, sommarfest och ett otal improviserade bastufester.
Kontaktperson: Vem som helst med `.ludd.luth.se` i sin e-mailadress. Vill du ha säker respons kan du maila `ludd@ludd.luth.se`.

Projekt!

Ett flertal projekt drivs av Lysators medlemmar. Några har fått egna rubriker (MUD, LysKOM) och här är några till spridda rapporter.

Lite Rydnetläge och Rydnethistoria.

av Magnus Redin

Rydnät[1] är sveriges första studentnät av den typ som är vanlig på universitet i USA. Målet med rydnät är att alla studenter som bor i Ryd skall kunna ansluta sina datorer till universitetsnätet. Det målet är nu efter två års arbete uppnått för ca 1700 studentlägenheter. Genom att dra en kabel från trapphuset till sitt rum kan alla studenter koppla in sig med tunnt ethernet eller rs232 om utrustningen finns. Nätet är det första stora studentnät som vi känner till som har byggts av studenter och inte tillkommit genom beslut av skolledningarna som näten i USA.

Tanken på ett nät i Ryd har funnits i minst 6 år och det gjordes en liten ansats för 4 år sedan. Det blev dock inget seriöst projekt förrän hösten 91 då jag hade slutat lumpen och den rätta studielusten inte infann sig. Jag ville ha en snabb förbindelse till Lysator från mitt rum och eftersom jag visste att fler ville ha det tänkte jag att vi kunde bygga ett litet nät och dela på ett höghastighetsmodem. Så jag kollade med bostadsbolaget Stångåstaden och fick ett entusiastiskt ja till att använda rören i backen mellan husen och skrubbarna i trapphusen till nätutrustning så med ritningarna under armen samlade jag glatt lite entusiaster och skulle bygga nät. Klart till julen var det tänkt.

Jag bestämde mig för att bygga med ethernet och något billigare till att börja med. Appletalk/rs232 på serielina verkade billigt och det fanns ett par friviliga som entusiastiskt började skissa på hårdvara. Efter ett år insåg jag att de aldrig skulle löda en krets, må dom glömmas. Finansieringen ordnade vi genom att lägga 1500:- var. Så drogs det kabel och draglinor mellan husen så nu är jag riktigt bra på det. Men nästan ingen hade dator med ethernet och appletalk/rs232-hacket blev aldrig av. Pengarna började ta slut och vi stod där med ett fint kabelnät i väggarna. Jag hade sedan början av projektet letat hårdvarusponsorer och första halvåret hittade jag en, DotCom som gav oss en 10Base-T hubb. 10Base-T var uppenbarligen den ideala tekniska lösningen för ett "spretigt" nät men visade sig vara för dyrt. 10Base-T räddade ändå projektet genom att göra tunt och tjockt ethernet omodernt under året som följde!

För att göra en lång historia kort så tiggde jag sedan envist i nästan två år och lånade ur egen ficka till frakt och tusen småprylar som inte gick att tigga. Jag fick hälften av datornätsföretagen att lova att skicka inbytta grejer hit men bara mycket få fick fram något. Genombrottet kom i julas då Waves, nuvarande LysCom (i Lysekil) gav oss en 2Mbit mikrovågslänk—först då paserade Rydnät krönet.

1. Redin kan inte komma överens med sig själv om det heter Rydnet eller Rydnät och jag är inte rätt person att avgöra—Red.

Jag kan inte göra en komplett lista över alla sponsorer men jag kan antyda lite genom att visa min förbindelse från mitt rum till Lysators Q-hus:

Tunn koax såld med rabatt av Ericsson Cables.

Transcievrar från Ellemtel.

Repeater skänkt av DotCom.

Beg PC med pcroute och ethernetkort med rabatt från PM data.

SUN-3/280 som huvudrouter skänkt av Sun Sverige.

Vitalink fjärrbrygga från Telverket data Göteborg.

2Mbit/s mikrovågslänk ifrån Lyscom.

Antennfästen skänkta av Kävelsbo smide.

Vitalink fjärrbrygga från Televerket data Stockholm.

Gul koax från Lidac, ISY och IDA[1].

Det var ca en fjärdedel av det nu installerade nätet. Jag har inga anlag för huvudvärk men jag misstänker vad som kommer att drabba den som sköter nätet om några år.

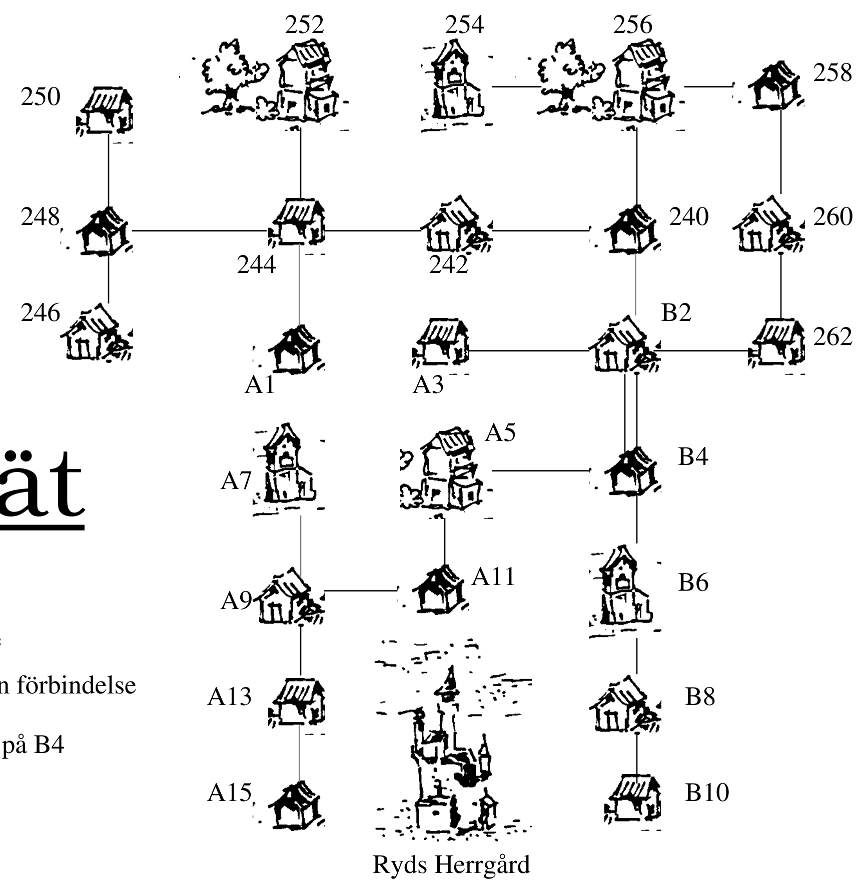
Nu när jag skriver det här så är de flesta av de ursprungliga entusiasterna anslutna, totalt ca 11 stycken. Med dom intresseanmälningar vi har så räknar jag med 50 anslutna till julen och ca 100 till nästa sommar. Vi är nu nära ekonomisk jämvikt, dvs anslutningsavgifterna räcker till att köpa gamla 286or och ethernetkort så att vi kan bygga ut nätet med PC/BRIDGE-bryggor.

Detta är också en del av mitt mål med Rydnät: att Lysator återigen skall finnas i datoranvändningens frontlinje på samma sätt som när mikrodatorn LYS-16 byggdes på 70-talet. Nu har studenterna på LiTH möjlighet att få erfarenhet av framtidens kommunikationssystem. Detta kommer troligen att förbli unikt för LiTH då ingen annan högskola i Sverige har ett lika koncentrerat studentboende. Och vem vet—om ett par år kanske det finns samma krav på studenter som börjar på LiTH som på en del skolor i USA: att de skaffar en dator med nätanslutning för att kunna studera effektivt.

— Magnus Redin, nöjd projektledare.

1. Institutioner på LiTH—Red.

Rydnät



— Dragen förbindelse

— Planerad, lättdragen förbindelse

Mikrovågslänken sitter på B4

Annons

Nutshell—

Ett nytt skal för Linux!

av Jonas Melin

Nu finns det ett skal som gör att du kan använda funktionstangenterna (F1-F5) när du kör Linux. Skalet heter Nutshell eller bara nutsh och kan köras "ovanpå" Linux-skalet. I kommandofilen (nut.cmd) kan man definiera de kommandon som ska bindas till respektive funktionstangent. Det här skalet blir speciellt smidigt att använda i RydNET när man snabbt ska koppla sej vidare efter att ha loggat in. Genom att skriva in några rader i din `.profile` kan du dessutom ordna så att du kör nutsh som logginskal. Du loggar då bara in och trycker på en knapp och 'vips' kommer du fram till nästa dator (via telnet, ftp, rsh, rlogin, ...). Eftersom skalet är en form av "kommunikationsskal" avsett att snabbt koppla vidare med till en annan dator med fullgott skal är dess funktion begränsad i vissa avseenden. För närvarande är t ex inte 'pipes', bakgrundsprocesser och 'history' implementerade. I övrigt fungerar det i princip som ett vanlig skal. Det är bara att ge kommandon med argument efter `RydNET>` prompten.

Detta är defaultinnehållet i nut.cmd:

```
ls -l
ps
more .nutsh_history
ping -c 10 -s 1024 rydvax.rydnet.-
         lysator.liu.se
telnet rydvax.rydnet.lysator.li-
         u.se
```

Ändra det precis som du vill ha det när du kör nutsh!

Detta ska du lägga till din .profile för att köra nutsh som logginskal på <Alt>-F1 och <Alt>-F2:

```
TTY=`tty`
case "$TTY" in
         /dev/tty1)
         exec nutsh ;;
         /dev/tty2)
         exec nutsh ;;
esac
```

Skalet generar en loggfil `.nutsh_history`. I denna står alla kommandon som givits i de skal du kör.

För RydNET-användare finns nu filerna `<README,nutsh*,rydNET.cmd>` att hämta från `/users/jonas/linux/nutsh/rydnet` på Lysators solsystem. En annan variant kommer så småningom att finnas tillgänglig. Jag tar gärna emot kommentarer och svarar på frågor i LysKOM. Jag nås också per elbrev till `<jonas@lysator.liu.se>` eller min kombrevlåda `<p274@lyskom.lysator.liu.se>`.

Mycket nöje med det nya skalet!

En kort introduktion till ka9q

av kjell-e

Vad är ka9q?

ka9q är ett nätverksprogram som går att köra på PC, Amiga, Macintosh. Ursprungligen är det gjort för datorkommunikation över amatörradio men finns numera även för fasta nät. Namnet ka9q kommer från radioamatörbeteckningen som Phil Karn, mannen bakom programmet, har. Det man kan göra med ka9q är att köra terminalsessioner mot andra datorer, överföra filer till och från andra maskiner och skicka elektroniska brev. Den hårdvara som krävs är antingen ett ethernetkort som ger hög överföringshastighet eller en serieport vilken är betydligt långsammare.

Installation

Ka9q är väldigt enkel att installera och det finns ett färdigt installationsprogram, men om man nu vill installera allt för hand så är inte heller det särskilt svårt. På en PC gör du bara så här: skapa ett nätverksbibliotek tex `net`. Under `net` behövs sedan följande bibliotek:

```
              net
pub   bin   dist  spool
 |
doc   help  mail  rqueue  mqueue
```

Alla program som `net.exe`, `ne1000.com`, `pktaddr.com`, `pktall.com` mfl läggs lämpligtvis i `bin`. I `net` ska filer som `autoexec.net`, `domain.txt`, `ftpusers` ligga under förutsättning att net startas med:

```
net -d c:\net c:\net\autoexec.net
```

där i det här fallet allt ligger på c-driven. `ftpusers` innehåller alla användare som får komma åt maskinen via telnet eller ftp. Den filen innehåller loginnamn lösenord bibliotek och behörighet, där användaren endast kommer åt den drive på vilken ka9q körs och där bara det bibliotek som anges i `ftpusers` och dess underbibliotek. En stjärna i lösenordsfältet innebär att kontot är lösenordsfritt. `domain.txt` är en tabell med maskiner och deras ip-adresser för att underlätta så att man kan ange maskinnamn istället för ett nummer. `autoexec.net` är den initieringsfil som sätter upp allting så att det fungerar när man startar programmet. I `autoexec.net` skall stå den IP-adress man har blivit tilldelad, namnet på maskinen man kör ka9q på, mjukvaruinterruptet där ka9q kan hitta packetdrivrutinen, routingen, och så de program som ska startas som tex telnet, smtp, ftp, finger mfl. `autoexec.net` på Tekla ser ut så här:

```
# Vem vi är   IP-nummer och
# maskinens namn
ip add 192.165.122.42
host Tekla
# vi skall lyssna på
# (interrupt 0x7e)
att pack 0x7e ne0 10 65535
ifconfig ne0 netmask 0x00
# IP-adress och eternetkortets adress
arp pub 192.165.122.42 ether
            00:40:95:00:25:2d
# routingen så att man kommer åt de
# andra maskinerna på nätet
# 192.165.122.1 är Ryder som sitter
# som spindeln i nätet
route add 192.165.122/24 ne0
route add 192.165.122.64/26 ne0
            192.165.122.1
route add default        ne0
            192.165.122.1
# starta servicear
start telnet
start ftp
start smtp
start echo
start finger
```

För den som har ett ethernetkort så behövs en packetdrivrutin till detta kort, vilket finns till de flesta korten, som startas innan ka9q. För att initiera ett ne1000-kort skriver man:

```
c:\net\bin\ne1000 -w 0x7c 3 0x300
```

där 0x7c är det mjukvaruinterrupt som används mellan drivrutinen och ka9q, 3 är det hårdvaruinterrupt på vilket kortet finns, och 0x300 är den adress där packetdrivrutinen ligger. Det går även att köra under windows på PC och för det krävs en speciell drivrutin som ligger mellan den hårdvarurelaterade och windows som heter winpkt. winpkt tar två argument, nya mjukvaruinterruptet och det gamla mjukvaruinterruptet, vilket med ovanstående exempel blir winpkt 0x7e 0x7c. Ka9q emulerar ingen terminal så för det behövs en terminalemulator. Till PC finns bla `ansi.sys` och `tansi.sys` vilka duger men det skulle behövas fler. Lägg till

```
device=c:\dos\ansis.sys
```

i din `config.sys`. Det bör även finnas terminalemulatorer till Mac och Amiga. En fördel med ka9q är att källkoder till allt finns tillgängligt. Titta på `/usr/local/src/ka9q` på Lysator. En del omhackningar av ka9q har skett nu under sommaren, en bugg i telnet har fixats, lösenord krypteras numera, och de svenska PC-tecknen omvandlas till 7-bitars ascii-tecken. Dessa ändringar finns att hämta på `Rydvax.rydnet` och på `Tekla.rydnet`.

För den som vill veta mer om telnet, ftp, tcp/ip så finns rfc959, rfc845/5, rfc791, rfc793 mfl.

Denna text är skriven via RydNet från en PC med ka9q körandes under windows.

Ping, ping (Rydnätsången)

text av Lars Aronsson[1]

Melodi: ABBAs Ring, ring

Ping, ping, bara mitt nät gav nåt svar...
Ping, ping, Rydnät är Liv-löst idag.
Ping, ping, Tekla och Rydvax, hallå!
Ping, ping, C-nät ett hundra tjutvå.

Jag är satt på en ö, och helt perplex,
för vårt C-nät når inte till två fem sex!

Å, å,
ping, ping, bara mitt Rydnät gav svar!

Förklaringar: Rydnät är Lysators datanät i studentbostadsområdet Ryd i Linköping. Liv är namnet på min maskin, som har problem att kommunicera med resten av Rydnät. Tekla och Rydvax är två andra maskiner i Rydnät. Rydnät är ett C-nät med nummer 192.165.122. Ett C-nät har plats för 256 adresser, i Rydnäts fall från 192.165.122.0 till 192.165.122.255. Jag bor på Rydsvägen 256, vilket visserligen inte har nåt med de 256 adresserna i C-nätet att göra, men det gav visan knorr.

1. `aronsson@lysator.liu.se`

Projekt Runeberg

av Lars Aronsson

Du känner kanske till att bibeln finns i maskinläsbar form på Lysators datorer? Klassiska texter från Shakespeares samlade verk till Det kommunistiska manifestet finns numera fritt tillgängliga "på nätet". Det kostar ingenting att plocka hem en kopia med anonym FTP. Det enda felet är att alltsammans är på engelska, och den enkla anledningen är att det var amerikaner som skapade dessa elektroniska utgåvor.

Elektronisk text är kanske en ny företeelse i Sverige, men Michael Hart i Lisle, Illinois är redan professor i ämnet. Hans vision är att på elektronisk väg revolutionera utgivningstakten lika mycket som en gång tryckkonstens fader Gutenberg gjorde. Michael Hart har därför startat Project Gutenberg, som ger ut elektroniska texter och gör dem fritt tillgängliga för alla som vill ha. Målet är att sprida 10.000 titlar till 100 miljoner läsare före sekelskiftet. Project Gutenberg spottar för närvarande ur sig fyra elektroniska böcker i månaden, och utgivningstakten fördubblas varje år.

Några av den svenska historiens största kulturgärningar har bestått i att ta utländska idéer och klä dem i svensk språkdräkt. Gustav Vasas bibel, den första på svenska, tillkom när Olaus Petri såg hur Martin Luther hade översatt bibeln till tyska, och sedan själv fortsatte att översätta från tyska till svenska. Svensk-MUD uppstod när Linus Tolke tröttnade på att spela engelska äventyr, och i stället översatte Nanny-MUD till svenska. Likadant var det när jag tröttnade på dessa engelska elektroniska texter. Därför startade jag Projekt Runeberg.

Namnet är medvetet valt för att skapa associationer hos dem som redan känner till Projekt Gutenberg. Men Finlands nationalskald Johan Ludvig Runeberg är också författare till Fänrik Ståls sägner, som var den första kompletta elektroniska utgåvan från mitt projekt. Denna förelåg i början av mars i år, och då tillkännagav jag också projektets existens i USENET News.

Redan i december 1992 hade jag emellertid satt upp mål för projektet och börjat samla texter. Jag arbetade från starten med menytjänsten Gopher, som jag berättat om i tidigare GARB. I mars när projektet kom igång mera på allvar skapades också en datorpostlista med tillhörande möte i LysKOM. Alla utgivna texter är dessutom tillgängliga med anonym FTP.

I början byggde Projekt Runeberg mer på vilja än på vetande. Jag knappade sålunda in Fänrik Ståls sägner för hand, vilket förvisso medförde att jag lärde mig skriva med korrekt fingersättning, men också tog en väldig tid. Pär "Pell" Emanuelsson visade mig senare ett utmärkt och gratis OCR-program och på Institutionen för systemteknik fick jag låna en flatbäddscanner under lediga stunder. Detta tillsammans med att jag fått texter från utomstående medhjälpare har gjort att utgivningen idag närmar sig två megabyte ren text.

Projekt Runeberg definieras genom tre begränsningar: Utgivningen skall för det första vara i maskinläsbar form, alltså på datafiler. För det andra ska utgivna texter vara skrivna på ett skandinaviskt (nordgermanskt) språk eller dialekt. Vid utgivning av notskrift, illustrationer och liknande som saknar språk, ska det finnas en rimlig anknytning till de nordiska länderna. För det tredje ska utgivningen vara fri, och får alltså inte skyddas eller hindras av upphovsrätt.

Den andra av begränsningarna utesluter finska texter, vilket vissa finnar funnit illa vara. Jag har medvetet gillrat denna fälla för att provocera fram ett självständigt finskt projekt (lämpligen tillsammans med estniska och lappska). Något sådant har dock ännu inte visat sig.

För att uppfylla den tredje begränsningen utnyttjas upphovsrättslagens tidsbegränsning. När en författare varit död i mer än femtio år upphör nämligen all upphovsrätt till hennes verk. Det kan verka makabert, men plötsligt blir det lika intressant att veta när en författare dog som vilka böcker hon skrivit. Många stora svenska författare som August Strindberg, Selma Lagerlöf, Hjalmar Bergman och Karin Boye dog alla före 1943, vilket gör att deras verk nu kan spridas fritt.

Projektets utgivning är uppdelad i filer, som kan hämtas med FTP, där varje fil är tänkt att motsvara en bok. Filerna är sedan indelade i kapitel eller verser, men bara i en nivå. Under Gopher har varje bok en egen meny och varje kapitel blir en Gopher-text. Denna struktur är lätt att administrera, men passar inte alltid de aktuella texterna, så den kan komma att ändras. I synnerhet Bibeln borde nog hanteras som kapitel inom böcker inom samlingen, alltså med menyer i två nivåer.

De största filerna från projektet är svenska Bibelöversättningen av år 1917 (64.000 ord, vilket ändå bara är två av evangelierna), romanen Kallocain av Karin Boye (56.000 ord), Fänrik Ståls sägner av Johan Ludvig Runeberg (28.000 ord), Nils Holgerssons underbara resa genom Sverige av Selma Lagerlöf (24.000 ord, vilket bara är de fyra första kapitlen), Herr Arnes penningar också av Selma Lagerlöf (23.000 ord), Svenska bilder av Carl Snoilsky (22.000 ord) samt den poetiska Eddan (18.000 ord, ej fullständig).

Därtill kommer flera mindre samlingar av sånger och dikter, alla ännu ofullständiga (av Fredmans epistlar av Carl Michael Bellman finns 6.800 ord), samt mindre stycken ur den svenska lagboken (8.800 ord). En mindre samling norska folkvisor utgör den första utgåvan på annat språk än svenska.

Vad ska man då ha texterna till? Det lämnar jag till läsarna. Helt klart verkar det finnas intresse. Under de senaste månaderna har Projekt Runebergs Gopher-menyer haft cirka 1.000 besök i veckan. Lagtexterna tillhör de populäraste, trots att de ännu är få, och kanske är det uppslags- och referensverk som kommer att bli de mest använda elektroniska texterna.

Per Hedbor, som startade Lysators World-Wide Web-tjänst, använder sig av Projekt Runebergs texter för att slumpa fram "dagens dikt". Antagligen är det inte lika populärt att sträckläsa en roman från bildskärmen; det gör man nog fortfarande bäst från papper. Men när man vill gå tillbaka och citera ett visst stycke, så erbjuder en elektronisk text stora fördelar jämfört med skrift på papper.

Lingvister vid Göteborgs Universitet har länge arbetat med datorer. Avdelningen Språkdata, där Sture Allén på 1960-talet skapade Svensk frekvensordbok, ingår i Institutionen för svenska språket. I Språkdatas arkiv finns en stor samling texter av August Strindberg, men ett avtal med förlaget hindrar fri kopiering. Språkdatas samlingar av medeltida svenska texter är emellertid fritt tillgängliga, och finns i kopia hos Lysator.

Nästa steg i utvecklingen av Projekt Runeberg blir att nå ut till läsare som inte har tillgång till datanäten. Jag tänker mig att göra en diskett för MS-DOS, Apple Macintosh och/eller Commodore Amiga där några av texterna ryms tillsammans med ett program för att bläddra och söka. Jag har redan varit i kontakt med några blinda som har dator med blindskriftsterminal hemma, och de verkade intresserade.

Samtidigt med detta kommer naturligtvis fler texter att läggas till samlingarna. Jag vill passa på att tacka alla som hjälpt mig med texter. Lars Willför hade en liten samling dikter (Karin Boye med flera) som han hade knappat in tidigare. Anders Andersson i Uppsala har gett många intressanta kommentarer och även bidragit med en del texter. Joakim Ragnvaldsson har tagit på sig den stora uppgiften att knappa in hela Nils Holgersson, och han är på god väg.

Till avslutning har jag lagt in den blänkare som jag numera brukar skylta med lite varstans. Den sammanfattar hur du kommer åt Projekt Runebergs texter. Ta en titt och låt mig få veta vad du tycker.

```
PROJECT RUNEBERG--free electronic texts in Scandinavian languages:
Find our Gopher menus  at  host = gopher.lysator.liu.se; port = 70
World-Wide Web server  at  http://www.lysator.liu.se/runeberg.html
Anonymous FTP archive  at  ftp.lysator.liu.se /pub/runeberg/README
Subscription requests  to  <runeberg-list-request@lysator.liu.se>.
```

Lys-32—

Planer och Funderingar

av Leif Stensson

1 Bakgrund

Det har inom Lysator länge talats om att bygga en 32-bitars flerprocessordator, som gått under arbetsnamnet "Lys-32". Från början var tanken att man skulle använda INMOS-transputrar; T9000 som huvudprocessor(er) och eventuellt några mindre (typ T425) som I/O- och systemprocessorer. När T9000 dröjde, visade sig sakna[1] stöd för virtuellt minne [1], och dessutom såg ut att bli ganska dyr, övergavs tanken på transputern som huvudprocessor.

I och med att transputeridén dog uppstod flera problem, varav det förmodligen största var hur processorerna skulle kommunicera med varandra; i en flerprocessormaskin är snabb och effektiv kommunikation mellan processorerna mycket viktig. Med transputrarna hade det bara varit att koppla ihop deras "transputerlänkar" med varandra (vilket var en av anledningarna till att man helst ville ha använt transputrar), men med andra processorer blev man tvungen att göra på något annat sätt. De huvudsakliga alternativ som diskuterades var

1) att låta varje processor ha ett "nästan privat" minne som den, helt eller delvis, delar med en hjälpprocessor som sörjer för kommunikation med andra processorer.

2) helt delat minne, dvs alla processorer delar allt minne, och

3) delvis delat minne, dvs processorerna delar minne, men kan även ha privata minnesområden,

De olika alternativen har, naturligtvis, olika för- och nackdelar:

1) är det mest skalbara alternativet, vilket naturligtvis är en stor fördel, men samtidigt är det ett ganska dyrt alternativ då varje processor måste ha en hjälpprocessor. En annan nackdel är att man riskerar lång latens (dvs långa väntetider) även för enkla saker som låsning av en semafor. Specialskriven mjukvara till hjälpprocessorn, så att semaforering och annan synkronisering hanteras av den, kan lindra problemet, men lär å andra sidan ta tid att utveckla och debugga.

2) är antagligen det som är lättast att skriva program (både system- och applikations-), men är samtidigt det alternativ som har sämst skalbarhet, eftersom alla processorer blir tvungna att hela tiden konkurrera med varandra om att komma åt minnet. Den flaskhalsen kan vidgas dels genom att ge varje processor en relativt stor minnescache (minst i storleksordningen 64KB), och dels genom att tillhandahålla speciell hårdvara för att hantera syn-

1. Det stöd som fanns delade upp minnet i ett fåtal segment, vilket innebar att "paging" blev i det närmaste omöjligt, och att man dessutom riskerade att få stora problem med minnesfragmentering. Se referens 1 för detaljer.

kronisering (semaforer, lås, monitorer...), som annars skulle riskera att sakta ner hela maskinen. Man har gjort på dessa sätt i bl a Sequent Computer Systems' Sequent Balance-serie (ref 2, 3 och 4) och DECs Firefly-arbetsstation (ref 5), men tyvärr lär vi inte kunna göra på detta sätt i Lys-32, eftersom Lysator knappast har de resurser som krävs.

3) är en variant på 2), men lättare att implementera. Det privata minnet hos processorerna kan, helt eller delvis, användas som en mjukvarukontrollerad minnescache.

Som de som läst Lys-32-mötet i LysKOM antagligen redan vet så tycker jag att det är alternativ 3 som är det som bäst lämpar sig för Lys-32 (om vi inte kan använda transputrar som huvudprocessorer, vill säga). Jag ska nedan redogöra för mina planer och funderingar kring att bygga Lys-32 på detta sätt.

2 Målsättningar

Primära målsättningar för Lys-32:

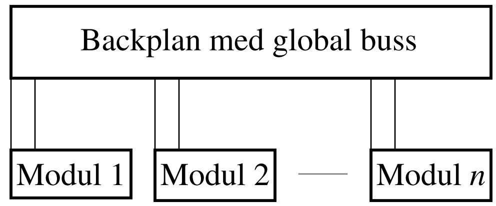
- Goda prestanda.
- Ska kunna byggas till rimlig kostnad.
- Modulär arkitektur (för att förenkla expansion och reparation).
- Ska kunna köra ett fleranvändaroperativsystem utan att vara ineffektiv.
- Någotsånär skalbar arkitektur; ska teoretiskt kunna användas till allt ifrån hemdatorer till arbetsstationer till "liten" superminidator[1]. (OBS: detta gäller alltså _arkitekturen_.)

3 Minnesorganisation

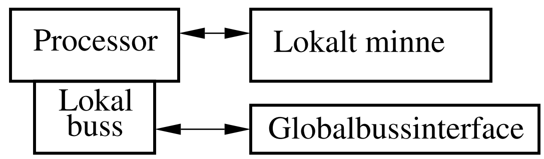
Minnet delas upp i två huvudklasser: det globala minnet, som delas av alla processorer, och lokalt minne, som är privat för varje enskild processor. Om minnet är lokalt eller globalt markeras med lämpligt antal (förslagsvis 2) av de mest signifikanta bitarna i minnets fysiska adress.

1. "Bättre lyss till den sträng som brast..."

Den globala bussen sitter på ett backplan som moduler med minne och processorer pluggas in i (ungefär som en VME-buss):



Processormodulerna byggs enligt följande skiss:



Interfacet till den globala bussen byggs (naturligtvis) så att det bara kräver bandbredd på den globala bussen när det verkligen tänker använda den.

3.1 Den globala bussen

I och med att all åtkomst till den globala bussen går genom ett speciellt interface är det teoretiskt möjligt att inom rimliga gränser byta implementation av den globala bussen utan att framtvinga några större ändringar av arkitekturen processormodulerna. Vi kan därför, av kostnads- och resursskäl, välja en enkel implementation av global buss till att börja med, och, om behov skulle uppstå i framtiden, modifiera den för att förbättra prestanda till nästa eventuella generation av Lys-32.

Jag föreslår att första generationens globalbuss implementeras ungefär enligt följande specifikationer:

- 32 ledare som används till att överföra både data och adress- information. Ledarna heter "D0, D1 ... D31" när de överför data, och "SEL0, SEL1, SEL2, SEL3, A2, A3, ... A29" när de överför adressinformation. (SEL0..3 markerar vilka "bytes" som berörs. SEL0 markerar den minst signifikanta byten D0..D7, etc.)

- 4 ledare för handskakning mellan det minne som adresseras och den processormodul som adresserar det: "Read/ Write", "Data Strobe", "Address Strobe" och "ACKnowledge".
- 1 ledare vardera för jord "GND" = "SGND", matningsspänning "Vcc", skyddsjord "PGND", och "RESET".
- 2 ledare reserveras för framtida utökningar. Den ena kallas "Command Strobe", den andra lämnas utan namn.
- 2 ledare, "REQuest" och "GRant", för arbitrering (se nedan).

Totalt 44 ledare. Eventuellt reserverar man några till, beroende på hur många poler en lämplig kontakt har...

Alla ledare utom de två sista ("REQuest" och "GRant") är samma fysiska ledare för alla moduler; rent elektriskt ska de hållas vid Vcc med pull-up-motstånd, och "dras ner" till GND av de moduler som vill signalera med dem. Idealt ska de ligga precis vid Vcc respektive GND för att signalera 1 respektive 0; praktiskt ska de åtminstone ligga inom 5-volt-TTLs toleranser.

3.2 Minnesreferenser via den globala bussen

Följande protokoll används för en åtkomstcykel på den globala bussen:

1) Sätt "REQuest" låg. Vänta på att "GRant" går låg.
2) Lägg ut adress, R/W och lämpliga SELectsignaler på bussen och sätt "Address Strobe" låg. Vänta på att "ACKnowledge" går låg.
3) Sätt "Address Strobe" hög.
4) Sätt "Data Strobe" låg. Om data ska *skrivas* till minnet så lägg ut det på databussen. Vänta på att "ACKnowledge" går hög.
5) Om data ska *läsas* från minnet så finns det nu på databussen, färdigt att läsas in.
6) Sätt "Data Strobe" hög. Sätt "REQuest" hög.
7) Klart.

Teoretiskt kan ovanstående punkter implementeras som separata steg i en "state machine", men vissa punkter kan kombineras i en "state" om man så vill.

Observera att de olika enheterna på bussen skulle kunna arbeta asynkront, på olika klockfaser och -frekvenser, om man är noggrann med tidsmarginalerna.

Det kan noteras att klisterlogiken kring själva minneskapslarna inte behöver buffra upp hela adressen, trots att den "försvinner" från bussen innan datat ska överföras; för ett dynamiskt minne kan man starta RAS-cykeln (Row Address Strobe) medan adressen ligger på bussen, och bara buffra kolumnadressen, signallera "ACKnowledge", sedan i nästa cykel ge CAS (Column Address Strobe) från buffern och samtidigt koppla in bussen på minnets "datapinnar", och sedan vänta tillräckligt länge för att datat ska hinna skrivas/ läsas.

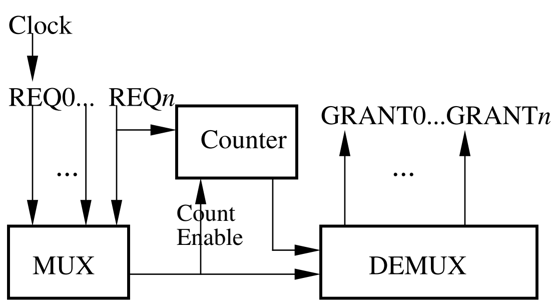
3.3 Arbitrering

"REQuest" och "GRant" är individuella för varje modul, och förbinder modulen med "bussdomaren" ("bus arbiter"). En modul sänker "REQuest" till 0 (GND) för att markera att den vill komma åt den globala bussen, och väntar sedan till "bussdomaren" sänker "GRant" till 0. Då (och endast då) får modulen använda bussen. Efter att ha använt bussen ska modulen "släppa upp" "REQuest" till 1 (Vcc) igen.

Kraven på själva "arbitreringen" är att den ska se till att endast en modul får använda bussen samtidigt, samt att ingen modul får vänta alltför länge. (Exakt vad som är "alltför" länge beror bl a på hur många processorer som finns i systemet, och vilken klockfrekvens som används, men det lär som mest bli frågan om något i storleksordningen 5 mikrosekunder.)

En lämplig implementation är ren round-robin-arbitrering, dvs att bussdomaren ger bussen till modul 1, väntar en cykel, väntar tills "$\overline{\mathrm{REQuest}}$" blir 1 (Vcc), ger bussen till modul 2, väntar en cykel, väntar tills "$\overline{\mathrm{REQuest}}$" blir 1, etc... Om modulen inte vill ha bussen så kommer "$\overline{\mathrm{REQuest}}$" från modulen redan att vara 1, och bussen går vidare till nästa modul; om modulen ville ha bussen är "$\overline{\mathrm{REQuest}}$" 0 tills den är klar. (Här får man naturligtvis ställa krav på hur länge en modul får använda bussen...)

Implementationsskiss:



(Vill man använda asynkron klockning kan det vara lämpligt att lägga in en D-vippa precis före enable-ingången (den nedre i skissen) till DEMUXen, och låta DEMUXens ingång drivas av en OR-grind vars ingångar drivs av D-vippans ingång och utgång.)

3.4 Cachning av globalt minne

Eftersom det lokala minnet är tänkt att användas som en mjukvarustyrd cache så ska inte det globala minnet behöva någon annan cachning. Trots detta kan naturligtvis vissa prestandaökningar uppnås med en cache, men i samband med sådan cachning kan även vissa problem uppstå, beroende på hur cachen implementeras, och framförallt var den placeras.

Det enklaste, men samtidigt mindre effektiva, är att placera cachen mellan den globala bussen och *minneskretsarna*, för att t ex minska åtkomsttiderna till långsamt dynamiskt minne. Eftersom minnet, precis som processorerna, sitter på löstagbara moduler kan man lätt ersätta en minnesmodul utan cache med en som har en cache utan att det påverkar systemets funktion (förutom, förhoppningsvis, effektivitetsmässigt).

Det krångligare sättet är att placera det i anslutning till varje processors globalbussinterface, men då kan man råka ut för att två olika processormoduler läser in samma minnescells värde i sina lokala cachar, att de sedan skriver olika värden för minnescellen i sina cachar, och att det sedan finns två olika värden i "samma" minnescell. Det finns olika varianter av lösningar på detta s k "cache inconsistency"-problem, men de kräver extra hårdvara, och lär inte bli aktuellt för Lys-32 om det inte dyker upp färdiga cachekretsar till rimligt pris...

4 Processormodulen

Eftersom det enda som krävs för att man ska kunna koppla in en processormodul på Lys-32s globala buss är att den har ett interface som passar ihop med bussen har man avsevärd frihet i att välja hur man vill bygga de enskilda modulerna; alla moduler behöver inte se ut på samma sätt, och behöver i extremfallet inte ens ha samma sorts processor.

I praktiken vill man nog ställa några krav till för att underlätta att skriva program, framför allt operativsystem och systemprogram, som fungerar på mer än just den maskin de skrevs på, men dessa krav är inte absoluta, utan relativa till vad man vill använda för operativsystem, så de behöver inte påverka arkitekturen som sådan.

5 Val av processor

Vad man väljer för processor för Lys-32 är av sekundär betydelse, eftersom själva arkitekturen är relativt processoroberoende; pris, prestanda och bekvämlighet får bestämma. Man behöver heller inte binda sig till någon speciell processor.

Första generationen av Lys-32 lär dock använda processorer ur AMDs 29000-serie, eftersom AMD varit vänliga nog att donera något dussin vardera av 29030 och 29200 till Lysator. :-)

Referenser och rekommenderad läsning för intresserade:

1. /* T9000-boken från INMOS som finns i Q-husets bokhylla. */

2. /* Sequent Balance, hårdvarumanual. Q-huset. */

3. Fielland, Gifford, Thakkar; "Balance: A Shared Memory Multiprocessor" ("Proceedings, 2nd International Conference on Supercomputing", Santa Clara, May 1987).

4. Beck, Kasten, Thakkar; "VLSI Assist For A Multiprocessor" ("Proceedings, 2nd International Conference on Architectural Support for Programming Languages and Operating Systems", Palo Alto, October 1987). (ACM 0-89791-238-1/87/1000-0010).

5. Stewart, Thacker; "Firefly: a Multiprocessor Workstation" ("Proceedings, 2nd International Conference on Architectural Support for Programming Languages and Operating Systems", Palo Alto, October 1987). (ACM 0-89791-238-1/87/1000-0164).

Utvecklingscykler

Cykeln **editera-kompilera-provkör** är välkänd. Numera är det **editera-starta kompileringen-spela flipper-provkör** som gäller i Q-huset.

—ceder

29000—

AMDs RISC-processor-familj

av Leif Stensson

Inledning

29000 är beteckningen på Advanced Micro Device's familj av högpresterande 32-bits RISC-processorer.

Översikt

De olika varianterna av processorn skiljer sig på en del punkter, bland annat vad gäller klockfrekvens, cachning, flyttalshantering och busstruktur, men på det hela taget fungerar alla varianterna av processorn (med undantag för microcontrollermedlemmen i familjen, Am29200) på samma sätt. Den äldsta medlemmen i familjen, Am29000, är en god representant för hela seriens arkitektur:

- Harvardarkitektur[1].
- 32-bits adressområde (dvs 4GB adresserbart minne).
- 192 allmänna ("general purpose") 32-bits register.
- Stöd för virtuellt minne via en 64-sidors TLB[2].
- Flyttalsinstruktioner. (Dock har Am29000 ingen inbyggd flyttalsprocessor - däremot finns en flyttalscoprocessor Am29027 som kan kopplas ihop med den.)
- Både instruktions- och databussen kan

1. Harvardarkitektur innebär att data och instruktioner har logiskt (och ofta även fysiskt) skilda minnesområden, som referas med separata bussar; man kan alltså t ex läsa nästa instruktion från instruktionsminnet *samtidigt* som man läser data åt den "nuvarande" instruktionen.

överföra data i "Burst Mode".

- Inbyggd timer.
- Kan mjukvarumässigt välja om "big Endian" eller "little Endian" ska användas. (Dvs om oktetterna i ett ord ska ordnast efter ökande eller minskande signifikans.)

Instruktionsuppsättning

Allt som allt har 29000-serien drygt hundra olika instruktioner, men alla instruktioner är exakt ett ord långa, och består av 4 oktetter, varav den mest signifikanta är själva instruktionskoden och de övriga tre är antingen registernummer eller heltalskonstanter.

29000-serien är en så kallad Load/Store-arkitektur, vilket innebär att det endast är Load/Store-instruktioner som läser och skriver till minnet; alla andra instruktioner arbetar enbart med register. För att göra det enklare att arbeta med ett så pass stort antal register som 29000-seriens processorer har så finns det, förutom vanliga Load/Store-instruktioner även Load Multiple- och Store Multiple-instruktioner som överför data mellan ett antal register och en följd av minnesadresser.

Hoppinstruktionerna är alla av typen "delayed branch", vilket innebär att instruktionen precis *efter* hoppinstruktionen alltid utförs medan hoppet pågår. Detta vid första anblicken konstiga beteende kommer sig av att processorn är "pipelined", dvs bearbetar instruktioner i flera steg:

→ Hämtning → Tolkning → Exekvering

så att flera instruktioner bearbetas samtidigt, en i varje steg, vilket leder till att instruktionen efter hoppinstruktionen hinner hämtas och tolkas innan hoppinstruktionen utförs. När hoppet väl görs blir man tvungen att vänta minst en cykel innan instruktionen man hoppar till hinner hämtas och tolkas; medan detta sker kan man "gratis" utföra den instruktion som redan hämtats. Det är tänkt att en optimerande kompilator ska kunna flytta om instruktionerna så att en instruktion som "logiskt" sett ligger före hoppet placeras efter den i minnet och på så sätt spara en cykel. Detta fungerar oftast mycket bra för ovillkorliga hopp och funktionsanrop, men sämre för villkorliga hopp, bland annat för att man bara kan flytta fram instruktioner som inte påverkar villkoret. För ovillkorliga hopp kan man även ta till knepet att flytta en kopia av destinationsinstruktionen till direkt efter hoppinstruktionen, och hoppa till instruktionen direkt efter den instruktion man ursprungligen hoppade till.)

Ett antal instruktioner är reserverade för hantering och räkning med flyttal, men alla processorer i serien kan inte själva exekvera de instruktionerna, utan försöker först skicka dem till en flyttals-coprocessor, och om det misslyckas inträffar en "trap" som åtminstone möjliggör emulering av den felande instruktionen. Några av de nyaste processorerna, bl a Am29050, har inbyggd flyttalsprocessor.

Register

De 192 generella register som ett program normalt förfogar över är uppdelade i två grupper, globala och lokala. De globala registren är 64 till antalet och fungerar som register brukar; de lokala är däremot lite speciella: när man använder dem så adresserar man dem relativt stackpekaren, så att det ser ut som om innehållet i de lokala registren "glid-

---

2. TLB = Translation Lookaside Buffer; en tabell som innehåller ett antal översättningar från virtuella till fysiska adresser. När ett program försöker komma åt en virtuell adress slås adressen upp i TLB. Om adressen finns där så är översättningen klar, annars gör processorn en "trap" som hoppar till en rutin som översätter adressen och stoppar in den nya översättningen i TLB.

er runt" när stackpekaren ändras. Tanken bakom detta är att man ska kunna använda de 128 lokala registren som en mjukvarukontrollerad stackcache, vilket gör att man kan emulera s k "sliding/overlapping register windows", men på ett sätt som är betydligt flexiblare än det som används i processorer som RISC och SPARC.

Utöver de 192 generella registren finns 1 stackpekare och tre "indirektregister" som gör att man kan referera till register indirekt genom att ladda in deras nummer i indirektregistren.

Internt i processorn finns även ett antal specialregister som på olika sätt styr processorns funktion, men dessa register kommer bara systemprogram (eller, rättare sagt, de program som kör i Supervisor Mode) åt. Bland dessa register finns bland annat två register för kontroll av den inbyggda timern, inte mindre än tre programräknare (en för varje steg i processorns pipeline), ett "register bank protect" register, som man kan använda för att blockvis reservera grupper om 16 av de 192 generella registrena för speciella ändamål, med mera.

Prestanda

Am29000 i grundutförande kör på en klockfrekvens av 25MHz, och klarar då att kontinuerligt exekvera 17 miljoner instruktioner per sekund, förutsatt att den hinner hämta dem från minnet tillräckligt fort. (Detta görs praktiskt möjligt i "Burst Mode" där ett instruktions- och ett dataord kan överföras samtidigt, på en cykel, förutsatt att den externa minneslogiken stöder "Burst Mode".)

En av de nyaste processorerna i serien, Am29050, kan köras på 50MHz, och har dessutom en inbyggd flyttalsprocessor.

Har Apollo flugit till månen?

eller

Stod det inte ett soffbord här nyss?

av Peter Johansson

För de observanta Q-husbesökare som har noterat att det inte bara tillkommer maskiner till Q-huset utan även ibland försvinner maskiner därifrån tänkte jag berätta en liten saga som börjar såhär:

Va e're om?

Lysator fick en donation av HP för ett par år sedan som bestod av 4 maskiner, DN560, DN660, DN550 och DN300 (med separat 8 tums HD och floppy). Dessa maskiner är alla grafikarbetsstationer som antagligen har använts till att köra någon form av CAD-program på. De kan presenteras lite närmare (främst hårdvarumässigt) som:

```
DN560, CPU-RAM: 3.0 Mbyte,
   CPU 68020@12MHz + 68881,
   Namn: //anna,
Disk: ca  64 Mbyte form.,
   Micropolis 1325 (ST-506),
Grafik: 1024x800 bildpunkter,
   8 bitplan (256 färger).
Övrigt: 60 Mbyte 1/4"-
   cartridgetapestation (Archive)
Fysisk storlek: Resväska

DN660, CPU-RAM: 4.5 Mbyte,
   CPU Apollo bitslice (ung
   68020),
   Namn: //amalia,
Disk: ca 143 Mbyte form.,
   okänt fabrikat (har ej tittat
   efter)
Grafik: 1024x800 bildpunkter,
   4 bitplan (16 färger)
Övrigt: 8 tums floppydrive,
   CPU:n har 4 kbyte instr.-
   och 16 kbyte datacache
Fysisk storlek: Soffbord eller
   mindre kassaskåp

DN550, CPU-RAM: 3.0 Mbyte,
```

```
   CPU 68010@10MHz,
   Namn: //alma,
Disk: ca  64 Mbyte form.,
   Micropolis 1325 (ST-506),
Grafik: 1024x800 bildpunkter,
   4 bitplan (16 färger)
Fysisk storlek: Resväska

DN300, CPU-RAM: 3.0 Mbyte,
   CPU 68010@8MHz,
   Namn: //alla,
Disk: ca  64 Mbyte form.,
   Separat 8 tums, hög,
   inklusive floppy-drive
Grafik: 1024x800 bildpunkter,
   1 bitplan (svart-vit)
Fysisk storlek: Integrerad med
   bildskärmen (förutom disk)
```

Alla maskinerna kör DomainOS 10.3 i BSD4.3-universum. Dvs, DomainOS är egentligen Aegis (kommandon i `/com` och en shell `/com/sh` som inte liknar unix mer än möjligtvis i princip) men det är tillräckligt kraftfullt och likt unix för att man ska kunna få det att nästan bli antingen BSD4.3 eller SVR3 (valfritt, dessutom konfigurerbart per användare). Det verkar (åtminstone vid första anblicken) inte så värst mycket mer inkompatibelt är diverse unix:ar är sinsemellan. Aegis-kommandona verkar i allmänhet ha lite fler och flexiblare flaggor än unix-kommandona. T.ex. `pst` som är en motsvarighet till ps har flaggan `-n` för att ange på vilken nod (fysisk maskin) som man vill göra ps och det skapas ingen extra process eller inloggning på den andra maskinen för det ändamålet. Dessutom finns det just i fallet `pst` en grafisk variant, `dspst`, som visar processernas cpu-utnyttjande och nätverks/disk IO m.m. som staplar, det finns en Null Process som motsvarar idle-tid.

Va e're som hender?

Dessa trevliga maskiner hände det inte så mycket med (jo, de drogs igång och kopplades in till lysator via SLIP och det hackades lite sporadiskt) i föreningen då de flesta hackers inbillade sig att SunOS var roligare än DomainOS och att Sparc:ar går fortare än Apollo-maskinerna vi fick (hrm, en gnutta sanning här iofs, men med Solaris 2 verkar det börja ordna till sig åt rätt håll :-)). Nåja, det började bli ont om plats i Q-huset efter att det drällde in VAX:ar, Sequent Balance, Pyramid, ett par HP9000/3xx maskiner så det började höras röster som förespråkade skrotning av Apollo-maskinerna medelst utkastning i sopcontainer. Detta kunde undertecknad vän av ordning inte acceptera och därför har jag nu bastu i ett rum därhemma (nåja, 30-32 grader är inte så farligt, snart är det vinter), dvs jag har förbarmat mig över maskinerna och lånat hem dem för att rädda dem från ett oförtjänt grymt öde. Eller så är det så att jag har lånat ut ett rum hemma hos mig för uppställande av Lysators Apollo-maskiner, beroende på hur man vill se saken.

Status på maskinerna för tillfället är att jag efter att ha släpat hem dem och lyckats boota dem efter lite strul med att få igång `rgyd` och `llbd`:ar (DomainOS variant på YP/NIS/NIS+, lösenorden, person, grupp, organisation och sånt hanteras i något som kallas registry, förkortas rgy) har lagat bandstationen (det var bara läs/skrivhuvudet som hade lossnat, fixat enkelt med lite lim) och installerat cc (C), ftn (F0RTRAN), pas (PASCAL), gmr3d (3D-bibliotek) och tittat lite på demoprogrammen. Dessutom har jag installerat en Micropolis 1325:a i DN550:n (c:a 70 Mbyte). Den separata 8 tums HD:n har slutat fungera för tillfället, den varvar inte upp men fläkten går. Det bzztar lite konstigt om den. Kanske har en konding i nätdelen gett upp eller så får man försöka meka lite med motorn. Musen funkar inte som den ska på färgmaskinerna (560, 660, 550) när man kör Display Managern (DM) men den verkar OK när man kör musdiagnos som standaloneutility (sau). På DN300:an funkar musen som den ska i DM också.

Jag har hämtat hem ett musdiagnosprogram från nätet också (bzone V1.0 i F0RTRAN, :-)) och det ger samma resultat, musknapparna fungerar, musflyttning fungerar inte på färgmaskin, men musflyttning funkar på DN300:n. Det tråkiga är att piltangenterna inte funkar som substitut för mus i bzone (ännu, blir jag tillräckligt frustrerad hackar jag om det :-)).

Det är problem att få igång någon vettig Xserver på maskinerna. Jag har testat 3 olika som jag hittade på nätet och 1 som fanns på maskinerna från början men det funkar inte bra med musen och samarbetet mellan fönsterhanterarna tycker jag. Jag har sett att det finns en borrow-mode server (Xdomain, X11R4) men den lyckas jag inte få igång, undefined global står det bara när man försöker starta den och i release-notes står det att bara sau7 och uppåt är supportat :-(DN300 är sau2. DN660 är sau4. DN550 och 560 är sau5. Jag funderade lite smått på att försöka bygga en X11R5-server från MIT-distributionen, men har för lite ledig disk för att lägga in hela X11R5 och ska väl försöka lägga in bara det som behövs för en server någon dag när jag orkar. Det är dock inte så givande att köra X när man bara har 3 Mbyte RAM...

För övrigt har jag kompilerat ihop en bildvisare för X (`xli`, bygger på `xloadimage`) och den verkar funka på gråskale-jpeg-bilder, vilket är det enda jag har testat att titta på med den än så länge. Och `stwar` (Star Wars) som inte går att köra då det krävs färgskärm och mus (jo, den startar och verkar väl fungera). Och `dviapollo` (previewer för TeX/LaTeX:s utmatningsfiler) bara för att upptäcka att den läser det obsoleta 128-teckens pxl-formatet (nja, aaf-format men det fanns konverterare från pxl till aaf) som jag inte har några fonter till (får väl hacka om den till att konvertera från pk-fonter om jag inte hittar någon nyare version, det är nog ingen ide' att ta hem en gammal MetaFont bara för att konvertera pk-fonter till pxl-fonter, de tar bara en massa mera plats och har inte svenska tecken).

Vadå, asså? Va mena'ru?

Vad är då dessa maskiner för ena konstiga djur? När maskinen har bootat färdigt slutar det med en nästan helt blank skärm sånär som på nedersta raden där det står `login:` i ett fält till vänster. Detta är en rad som hör till DM. Normalt sett (dvs efter inloggning) är det DM:s kommandorad och dit kommer man genom att trycka på CMD-knappen. Här kan man ge kommandon som `LO` (logout), `SHUT` (Shut down), definiera tangentbordsmacron, visa/editera en fil i ett nytt fönster, m.m.m. (med mycket mera). Till en tangent kan man alltså knyta en följd av DM-kommandon. T.ex. kan man programmera högerknappen på musen till att läsa det ord som curson står på som ett filnamn och öppna ett nytt fönster med den filen i. Kraftfullt och användbart. Den funktionen finns normalt även på SHIFT-Read. Trycker man bara på Read får man ange filnamnet via tangenbordet (eller Mark, Copy, Paste "för hand") på DM:s kommandorad. En fil som man har gjort Read på (på ett eller annat sätt) kan man inte ändra i av misstag genom att bara sätta igång och knappa, då får man pip och meddelande om att det inte är en editeringsfil. Hur ändrar man detta? Jo, man trycker på Save-knappen. Det kanske verkar lite ologiskt vid första anblicken, men det finns även en Edit-knapp som gör att man t.ex. kan skapa nya filer som man får knappa i direkt från början (och öppnar man en befintlig fil med Edit så får man ändra i den genom att knappa direkt). Då man har ändrat färdigt trycker man logiskt nog på Save varvid filen sparas på disk och sätts till Read-läge så man inte ändrar i den igen av misstag (t.ex. under kompilering). Efter ett tag vänjer man sig vid Save-Save för att spara tillfälligt och t.ex. Shift-Read Save för att editera en fil vars namn står under cursorn för tillfället.

Varje fönster är på det här sättet automatiskt en editor och ett normalt kommandofönster består av två delar, en inmatningsdel och en utmatningsdel. Man skriver kommandona i inmatningsdelen och om man anger flera rader med "typeahead" på det här sättet så ser man vad man har skrivit och kan gå upp och editera kommandona om man hinner... Allt eftersom burken hinner så betar den dock av rader från inmatningsdelen (som ligger ovanför cursorn) och flyttar dem till utmatningsdelen där sedan också utmatningen från kommandot kommer. Utmatningsdelen kan man gå upp till och scrolla tillbaka, inte 17 rader, inte 4711 rader, utan ända tillbaks till första raden då fönstret skapades (t.ex. när man loggade in). Man kan inte gå längre tillbaks om man inte har sparat utmatningsdelen på fil innan man tar bort fönstret (vilket görs automatiskt då man loggar ut). Skojiga tangentbordsmacron är t.ex. sådana som går över till utmatningsfönstret, söker rätt på en prompt och kopierar raden (efter prompten) till inmatningsfönstret. Dvs, scrolla tillbaks bland gamla kommandon man har gett. Det enda ovanliga är att utmatningsfönstret scrollar med allt eftersom man scrollar sig bakåt bland kommandona. Nu finns det iofs också en tcsh (dock ej som standard i OS:et) som tillfälligtvis under interaktiv inmatning slår ihop in- och utmatningsdelen av fönstret så att man får det som man är van vid att ha det om man nu ska envisas med det.

Andra "konstigheter" (läs: icke-unix) i OS:et är att man har en toppnivå ovanför `/` i bibliotekshierarkin som heter `//` och där kan man se maskinerna på sin tokenring (eller vilka maskiner som man har katalogiserat där, och har man många maskiner går det att centralisera katalogiseringen) och komma åt deras lokala disk via `//node`, t.ex. `//anna/local2/games/bzone`. Dessutom används symboliska länkar väldigt mycket och ` betyder `/sys/` på den maskin som processen kör så man kan ha länkar till `node_data för att komma åt rätt maskinspecifika filer. Detta är inbyggt i kärnan och man behöver inte bekymra sig om att få igång nfs och automounters och annat krångel (vad vet jag?). Man behöver alltså inte ta upp maskinen i multiuser eller på något sätt starta annat än själva kärnan för att komma åt filer och på det sättet kunna dra igång en maskin via nätet (om man nu är root och har konfigurerat det så med rättigheterna till filer) och fippla med filer. Praktiskt.

Lider man av säkerhetsmani så är DomainOS trevligt också, jag har inte undersökt det så mycket men det är enkelt att t.ex. sätta skydd på en fil så att den får läsas av alla utom användare som heter nisse utan att behöva skapa en grupp där bara han är med. Det hela implementeras med något som i DomainOS kallas ACL (Access Control List, gissar jag[1]). Dessutom kan man ha ACL:er för bibliotek som anger vilket skydd filer/bibliotek som skapas i det ska få, lite mer användbart än unix:s umask... Man har både grupper och organisationer och kan sätta skydd på filerna beroende på urval av endera eller båda (eller alla tre om man räknar med användare som det tredje).

Ja, det får vara nog för den här gången. Keep hackin'.

/Peter

1. Jupp—se MULTICS - Red.

CRACK

av Pell Pell Pell

Förspel

När jag som vanligt loggade in en måndagkväll vid 8-tiden möttes jag av följande meddelande i min brevlåda:

```
Date: Mon, 14 Jun 1993
           03:11:49 -0700
From: "Erik E. Fair"
       <fair@apple.com>
To: jeff@nmsu.edu,
    domainmaster@liu.se,
    cert@cert.org
Cc: staff@apple.com
Subject: suspicious account
          activity at Apple
```

Det innehöll en redogörelse över hur Erik Fair hade hittat en misstänkt login mitt i natten (som körde IRC) och spårat den bakåt, via en cisco-växel som tillät anonym login, till Lysator. Han hade också "fingrat" Lysator och kommit fram till att den skyldige var "zap"—alla andra var idle.

Efter att ha pratat med zap ansåg jag att han kunde avskrivas som misstänkt. Frågan var då "vem annars"? En komplikation var att vi körde GNU-finger, som ju bara ger första inloggningen för varje person och maskin. Teoretiskt kunde alltså kanske någon av dem som var idle på Lysator ha andra inloggningar som var aktiva. Dessutom är det inte så svårt att gömma sig så man inte syns med "finger". En titt i last-loggen visade emellertid att det inte fanns några andra aktiva inloggningar än just de personerna som syntes. Däremot fanns det några inloggningar som syntes i "finger", men inte i last-loggen. Hur gick det till? En möjlighet kunde vara att någon hade klippt bort en del av last-loggen, t.ex. för att gömma sig själv, och därigenom även tagit bort information om inloggningar av andra personer. Det kändes dock ganska långsökt så jag lät det bero. Jag trodde fortfarande på att det var någon som i oförstånd hade fått reda på lösenordet till något apple-konto och loggat in "bara för att testa". Trist nog hade det tagit 8 timmar för Eriks brev att rassla igenom diverse personer innan det nådde Lysator. Hade han mailat root@lysator direkt hade det funnits två inloggade rötter som direkt hade kunnat kolla upp vem som körde.

Jag försökte kolla vår nätverkloggningsmaskin, men det visade sig då att den inte hade fungerat hela juni pga värmeproblem.

Jag skrev brev till Erik Fair och bad om en last-log för det aktuella kontot. Med hjälp av det skulle man kunna samköra tidpunkter med inloggningar på Lysator och kanske se något samband. Det kändes inte som om det fanns så mycket mer att göra.

Någon dag senare kom information från administratören av ciscon. Förutom Lysator-uppkopplingen fanns en annan uppkoppling från en maskin i Chile till en av Ciscos (företagets) egna datorer. Verkade underligt. Jag fick tag på en checklista från CERT[1] om vad som bör kollas om man misstänker att någon obehörig har lyckats bli root och skickade den till de andra rötterna med en begäran om hjälp. Jag hade nämligen mycket att göra på jobbet och skulle resa bort.

Inte mycket verkade hända—en eller två av rötterna började titta igenom listan, leta setuid-prorgam, rensa bort alla `.rhosts`, etc. Ingen annan verkade intresserad och Erik Fair var tyst. Jag åkte bort och återkom lördag

1. Computer Emergency Response Team. Bildades efter incidenten med Morris-masken och samlar information om knäckta system, säkerhetshål, skydd och lösningar.

natt.

Upptrappning

När jag loggade in mötte mig brev från PJN@IDA, där det framgick att Björn Eriksen på SUNET stötte på och ville stänga av Lysators nätanslutning. Björn hade tydligen fått information från CERT om att Lysator hade gjort nya elaka saker, men inte vad eller när. Efter samtal med BoK@LIDAC och PJN kom vi överens om att inte koppla bort Lysator, utan att åtminstone vänta på mer information från CERT (hittills hade ingen av oss hört mer än rykten från dem) och kanske Erik Fair. Om vi kopplade bort oss skulle ju problemet "försvinna", men då skulle vi antagligen inte få reda på vem som var boven och skulle Lysator någonsin kunna återfå nätanslutning så länge situationen var sådan?

Jag skrev ett långt brev till CERT och klagade på dem. Det gav resultat, för på tisdag fick jag slutligen svar från dem. (Tyvärr var jag återigen bortrest så jag läste det inte förrän torsdag förmiddag)

Det visade sig att det var flera maskiner i Chile som blivit knäckta. Förövarna hade bl.a. använt dem som "trampolin" för att koppla sig vidare mot andra system. Efter att administratörerna där uppdagat vad som hände hade de installerat en "fälla" som loggade vilka som försökte logga in och varifrån. Där fångade de bl.a. ett inloggninsförsök från root@lysator! Det skedde onsdagen den 16:e juni, dvs två dagar efter vi först fick reda på att nåt var skumt.

Jag fick också information om vad Chilenarna upptäckt: bl.a. hade "telnetd" blivit utbytt mot en som gav en root-login om man gav ett speciellt login-namn (bl.a. "foko" och "SMI" hade använts). "telnet" hade också bytts ut mot en version som loggade alla tangenttryckningar (dvs bl.a. namn och lösenord när man kopplar sig ut mot nätet). Loggningen gjordes till filer i biblioteket "`/usr/lib/xglfonts/stroke/. `". Notera mellanslaget efter punkten.

Crackade!

Jag tittade snabbt på Lysator och—mycket riktigt! Det fanns ett sånt bibliotek, men det var tomt. Senaste modifieringsdatum var onsdag i föregående vecka, när också inloggningsförsöket mot Chile gjordes. Det var nu tydligt att någon fixat root-access på Lysator[1].

Jag skulle åka bort igen och skickade brev till pen. Han beslutade sig därefter för det enda raka, nämligen att koppla bort nätet och "undersöka skadans omfattning".

Systemprogrammen (telnet, telnetd, m.fl.) verkade orörda. ceder hittade en två veckor gammal backup där det fanns en fil "stroke6" (tror jag) i det skumma biblioteket. Filen visade sig dock vara tom. Biblioteket självt hade funnits sedan två veckor innan Apple-rapporten, så man kan misstänka att bovarna kunnat logga flera användares lösenord på externa system. De hade förstås också haft alla möjligheter att lägga in bakdörrar på möjliga och omöjliga ställen i filsystemet.

Upprensning

Pen bestämde sig för att installera om operativsystemen på alla Lysators maskiner. Det var nödvändigt för att få "rena" systemfilsystem. Han spärrade också alla användare, vilket var nödvändigt eftersom det kunde antas att en stor andel av kontona var knäckta, så efter en systemrensning skulle förövarna direkt ha kunnat komma in igen.

Som en liten extra krydda (:-) bestämde han sig dessutom för att installera Solaris-2 på alla maskiner som kunde köra det. Det var

1. Det var egentligen allmänt känt att det inte var så fruktansvärt svårt, om man bara lyckats ta sig in som en vanlig användare. Det är inte heller så svårt—starta ett MUD och annonsera. 90% av dem som kopplar upp sig kommer att använda samma lösenord som de har på sitt konto. Vi litade på allas goda vilja och att ingen skulle bry sig tillräckligt mycket. Vi körde `crack` då och då och implementerade de säkerhetspatchar vi hörde talas om—inte mycket mer.

nog ett bra drag, trots att det har framkallat åtskilliga gråa hår vid plågsamma porteringsförsök. En del saker går därför numera på Sun-3-maskiner, t.ex. mail.

Epilog

Lysator har åter nätkontakt. De flesta Lysiter har varit förstående, även om mängden brev av typen "jag MÅSTE FTP:a, fixa mitt konto NU!" har varit jobbigt hög ibland. Åtskilliga manveckor har lagts ner, men vi är ännu inte tillbaka till den "funktionalitet" vi hade innan attacken.

Jag har haft lite samtal med CERT under tiden som har gått, både via mail och telefon. En tid efter händelsen fick jag ett brev som fick mig att svettas lite:

```
% grep law certmail
investigation via law
enforcement.
name/contact information along to
the law enforcement folks.
If you agree to release this
information to law enforcement,
we have
when this might occur. Should you
be contacted by law enforcement
and
to confirm the law enforcement
contact if you so wish.
information may have been passed
along to law enforcement from the
%
```

Kort gick det ut på att "en av de drabbade parterna" gått till nån "law enforcement agency" och ville driva det hela vidare och CERT sade att de kunde skydda mitt namn om jag ville, men att de inte kunde garantera något eftersom andra inblandade kände till min adress. Tonen i brevet gjorde att det verkade som om det var på gång en internationell lagsökning mot "Lysators illegala aktiviteter" och jag såg framför mig en skylt på Q-husets dörr: "Stängt för alltid".

Ett telefonsamtal till CERT redde dock upp frågan—det är inget speciellt som pekar ut Lysator som skurk utan "law enforcement" kanske vänder sig till mig om de vill ha ytterligare information som ett led i undersökningen. Än har dock inget hänt, så vi får väl se... Allt tyder ju på att det var folk utifrån som använde Lysator som språngbräda precis som maskinerna i Chile.

Hela den här händelsen får mig dock att känna en sorts vanmäktigt raseri över vilka idioter som finns därute.

Min uppmaning till alla Lysiter är att vara försiktiga i framtiden:

- Om ni har externa konton som ni gjort telnet till från Lysator så byt lösenord på dem och kolla att ingen annan loggat in där!
- Kolla regelbundet att ni inte har fått någon `.rhosts`-fil med underlig information.
- Använd aldrig samma lösenord på system med känt dålig säkerhet, som t.ex. MUD. Flera fall är kända där MUD-administratörer har loggat användares lösenord och sedan brutit sig in på användarnas konton. Byt lösenord då och då.
- Läs ceders artikel om Xauth på annat ställe i denna tidskrift!

Jag vill avsluta med att tacka alla som har ställt upp och jobbat med att fixa till saker när det har känts tungt. Tack till BoK och PJN för fint stöd! Ättika till Erik Fair för att han aldrig svarade på mitt brev.

```
390752 1993-07-31  18:15  /1 rad/ Den gamla hunden (Ceci.143)
Mottagare: Pl|tsliga aggressioner <4032>
Ärende: Rinn, kula, rinn.
------------------------------------------------------------
Frustration, ditt namn är flipperspel./dgh
(390752) ------------------------------------
```

Sett i LysKOM av ceder

Solaris 2.2—SunOS 5.2

av Peter Eriksson[1]

1. Introduktion

Som ni säkert har märkt så har vi nu bytt operativsystem på våra Sun4 (SPARC) datorer från det gamla hederliga SunOS 4.1.3 till det sprillans nya Solaris 2.2.

Vad är då Solaris 2.2 för ett djur till operativsystem? Ja, det är egentligen inte ett operativsystem utan det är samlingsnamnet på kombinationen av SunOS 5.2 (operativsystemet), OpenWindows 3.1 (fönstersystemet), DeskSet (Open Look-applikationerna man kan köra under OpenWindows, med mera.

Solaris 2.2 går att köra på Sun4 (SPARC)-baserade datorer och även Intel 386 och 486-baserade datorer. Helst ska man ha minst 16MB RAM och 300MB disk för att det ska gå med vettiga prestanda.

2. SunOS 5 / Unix System V Release 4

Version 5 av SunOS är i princip en total omskrivning av hela operativsystemet jämfört med gamla version 4 (som vi fortfarande kör en version av på våra Sun3 och Sun2 datorer). I grund och botten bygger SunOS 5 på Unix System Laboratories (USL)'s version av Unix som kallas System V Release 4 (SVR4), Denna version är framtagen i samarbete med Sun Microsystems och är en 'merge' av 'features' från gamla SunOS 4 och System V Release 3 samt en massa nya finesser.

3. Nyheter för användarna i Solaris 2.2

Största förändringen för användare under Solaris 2.2 jämfört med SunOS 4 är att argument/optioner till program/kommando numera följer System V istället för BSD-standarden. Det gör att det kan kännas lite ovant till en början innan man har vant sig. (Det *går* att komma åt BSD-varianter av de flesta kommandon om man verkligen vill, men det finns en hel del nackdelar med detta. Bland annat att dessa program går via ett BSD-emuleringspaket med effektivitetsproblem och intressanta buggar till följd). Om man är van vid en del andra Unix:ar, tex Hewlett-Packards HP-UX, eller Silicon Graphics IRIX, så känner man snarare igen sig.

Några exempel på kommandon som har ändrat sig en del (se manualsidorna för mera detaljerad information):

```
SunOS 4 -> SunOS 5
---------------------------
ls -Flag    ls -Fla
ps -aux     ps -elf
pstat -T    sar -rv 5
stty pass8  stty cs8 -istrip
yppasswd    nispasswd (Se 4)
```

Det finns även en ganska bra liten Solaris 2 bok som tar upp skillnaderna mellan SunOS 4 och SunOS 5, den skall/bör finnas bland de andra manualerna i bokhyllan.

Då man i SunOS 5 har tagit bort den C-kompilator som brukade skickas med systemen tidigare så kan det vara bra att veta att SPARC-assembler, länkare med mera finns i filkatalogen `/usr/ccs/bin/` samt att det finns en del bibliotek med C-rutiner i `/usr/ccs/lib/`. Man bör se upp så att man inte försöker länka sina program med den länkare som ligger i `/usr/ucb/` då den länkar programmen med BSD- emuleringsbiblioteken vilket med 99.99% sannolikhet kommer att medföra problem. Man bör undvika att bygga programmen på detta sätt om möjligt.

4. YP/NIS -> NIS+

Ännu mera kryptiska förkortningar?!?! YP står för Yellow Pages och var namnet på Suns 'nätverksinformationssystem' tills British Telecom tyckte illa vara (de hade visst 'YP' registrerat som varumärke). Då döptes Suns YP raskt om till NIS i alla manualer å så. Programmen/C-rutinerna heter dock fortfarande saker med 'yp'...

Med tiden märktes det mer och mera att

1. pen@lysator.liu.se

YP/NIS inte var helt perfekt (snällt sagt), bland annat så saknas nästan allt vad det gäller säkerhet och mer än en cracker har utnyttjat säkerhetshål i YP/NIS-servrar runt om i världen för hämta hem lösenordsfiler med mera. Ett annat problem är att effektiviteten hos YP/NIS inte är speciellt bra vilket medför problem på stora installationer. Ett tredje problem är att det i YP/NIS inte finns något bra sätt att bygga upp hierarikier av NIS-"domäner" (en domän är en grupp av maskiner/servrar som har gemensam information av något slag).

För att råda bot på detta påbörjades ett projekt på Sun att ta fram en ersättare för YP/NIS som skulle klara av ovanstående problem på ett bättre sätt. Resultatet av detta är något de kallar NIS+ (även känt som NIS version 3 (YP/NIS är version 2)).

Ett problem med NIS+ (för oss med gamla maskiner) är att det inte är kompatibelt med gamla YP/NIS. Så vi får använda oss av ett paket som emulerar vissa delar av YP/NIS för att de gamla maskinerna ska kunna få den information de behöver.

En sak som dock inte fungerar även med detta emuleringspaket är att kunna ändra lösenord från en maskin utan NIS+, vilket medför att man måste logga in på en av Lysators Sun4 (SPARC)-datorer för att utföra detta. Kommandot för att byta lösenord heter "`nispasswd`".

En sak som kan förvirra i sammanhanget är att man införde en extra säkerhetsnivå (för att täppa till en del hål i gamla YP/NIS) som kallas för 'network credentials' och är ett sätt att unikt identifiera vilken person som försöker få saker och ting utförda över ett nätverk (till skillnad från gamla YP/NIS som blint litade på att man var den man sade att man var när det kom en förfrågan över nätverket).

Detta implementeras via DES-cryptering och öppna nycklar. Dessa nycklar baseras på ett 'nätverkslösenord' som man i normalfallet sätter till samma som det vanliga lösenordet man använder för att logga in på en viss maskin (kom ihåg att det är viss skillnad på att logga in på en viss maskin och köra program, och att logga in på NIS+-servern för att kunna få den att utföra uppgifter över nätverket).

När man sedan loggar in på en Sun4 (SPARC) så loggar login-programmet automatiskt in användare både på maskinen och i nätverket om man har samma lösenord i båda fallen. Även kommandot 'nispasswd' uppdaterar båda lösenorden. Ibland händer det dock att något går fel och då kan man bli tvingad att göra en manuell inloggning med hjälp av kommandot '`keylogin`'.

Se även manualsidorna för "`nis`", "`nispasswd`", "`keylogin`" för (mycket) mera information.

5. Finesser i SunOS 5.2-kärnan

Bland nyheterna i SunOS 5 kan bland annat nämnas att kärnan är uppbyggd av dynamiskt laddningsbara moduler som automatiskt laddas in efter behov. Dvs, man kan hålla Unix-kärnan precis så liten som behövs. Dessutom försvinner behovet av att manuellt bygga en helt ny kärna när man har ändrat något, bara att byta ut den modul som behövs.

Andra nyheter är att den virtuella minneshanteringen har förändrats radikalt. Bland annat så behövs numera inte längre någon 'backing-store' för RAM på disk. Dvs, man kan spara in många MB diskutrymme om man har många maskiner. Totala virtuella minnesstorleken är alltså numera RAM + Swap-utrymme på disk, och inte som tidigare bara Swap-utrymme på disk.

Vidare har hela nätverkshanteringen skrivits om till att vara uppbyggt av STREAMS-moduler (som givetsvis också är dynamiskt inladdningsbara efter behov). Så nu kan man teoretiskt byta ut t.ex. IP-modulen mot en egenhackad dito utan att behöva hacka om TCP och UDP-modulerna.

Man har också infört ett interface så att program kan fråga dessa moduler i kärnan om information på ett vettigt sätt, så man slipper gå in och gräva i /dev/kmem och hoppas på det bästa. T.ex. så använder programmet '`netstat`' denna finess.

6. Tillägg[1]

Virtuellt minne kollas gärna med programmet "swap". Tex:

```
swap -l
```

Visar swapdevices, storlek och hur mycket som är fritt på dem.

Ett annat program som är skoj att köra är "`sar`" (System Activity Report). Tex:

```
sar -A 5 | more
```

Ger information om nästan allting, samplat under ett 5-sekundersintervall...

De gamla textfilerna för systemadministration, t.ex. `/etc/services` är numera föråldrade och stämmer inte säkert med vad systemet anser. För att uppdatera en sån tabell från kommandoraden kan man t.ex. göra:

```
nistbladm -a cname=telnet
name=telnet port=23
services.org_dir.lysator.liu.se.
```

för att lägga in information om telnettjänsten. Allt skrivet på en rad givetvis!

1. Resten i denna artikel lades till av Red, samlat från Peters spridda insikter..

Säkrare X

av Per Cederqvist

X-windows torde vara ett välvälbekant [sic] fönstersystem för de flesta. Det är ett fönstersystem som är allt populärare, trots att det har sina brister. Den största bristen är kanske att det är så komplext att det är svårt att få en ordentlig överblick.

Jag har just börjat utforska det lite djupare, och gjorde då en chockerande upptäckt: sen jag började köra X har vem som helst har kunna läsa av allt jag skrivit på mitt tangentbord! Det krävs visserligen att man kan en del om X-windows (mer än jag kan just nu), men det är möjligt.

Problemet är att jag använde `xhost +` för att kunna öppna fönster från andra datorer än den jag satt vid. Men `xhost +` ger inte bara mig den möjligheten, utan hela världen! Den som kan öppna ett fönster på din X-windowsskärm kan också läsa av ditt tangentbord. En snabb koll på ett visst datorsystem visar att inte mindre än 75 procent av alla `.xinitrc`-filer innehåller `xhost +`.

LÖSNINGEN

Lösningen på problemet är enkel: använd inte `xhost`! Det finns säkrare sätt att öppna fönster. Filen `.Xauthority` och programmet `xauth` spelar en central roll i den lösning som jag har upptäckt, och som jag tänkte beskriva här.

Med `xhost` talar man om att en viss maskin är betrodd att öppna fönster. Man kan få X-servern att i stället kräva att klienten känner till en hemlig nyckel för att den ska få öppna fönstret. Nyckeln ligger lagrad i binärform i filen `.Xauthority`, som inte skall vara läsbar av någon annan än dig (`chmod 600 .Xauthority`). Programmet `xauth` används för att editera filen.

För att öka på säkerheten (eller i alla fall krångla till det hela lite grann) så har man inte bara en nyckel, utan hela två nycklar per display som man använder! Den ena nyckeln används för uppkopplingar som görs lokalt på maskinen (den heter till exempel `robin/unix:0`). Den andra nyckeln används vid nätuppkopplingar och heter `robin:0` eller något liknande. Nycklarna består av en lång sträng av hexadecimala siffror. Hur lång strängen ska vara kan du välja själv, men det måste vara ett jämnt tal.

PRAKTIK

Hur gör man nu för att använda `.Xauthority`?

Ta bort `xhost +` ur din `.xinitrc`.

Se till att din X-server använder sig av `.Xauthority`. Det gör du enklast genom att skapa en fil vid namn `.xserverrc` i ditt hemdirectory och lägga följande två rader i den:

```
#!/bin/sh
X :0 -auth $HOME/.Xauthority
```

Se till att det finns en nyckel för den display du tänker använda. Om du har loggat in på robin kan du till exempel göra följande för att starta X.

```
$ xauth
Using authority file
       /users/ceder/.Xauthority
xauth> add robin:0 . 1234
xauth> add robin/unix:0 . 5678
xauth> exit
Writing authority file
       /users/ceder/.Xauthority
$ xinit
```

Det kan vara lämpligt att använda fler än fyra hexadecimala siffror. Nycklarna ligger kvar i `.Xauthority` tills du tar bort dem. Du behöver alltså bara använda `xauth` om den display du ska använda saknas i filen. Du kan skriva `xauth list` för att se vilka displayer du har skapat nycklar för.

TEORI

Följande manualsidor läste jag i innan jag insåg hur man använde `xauth`: xauth, Xsun, Xserver, xinit, startx, xdm, xhost, xterm. Det är främst Xserver, xinit och xauth som innehåller relevanta fakta.

EMACS-19

av Peter Eriksson[1]

Nu gör Emacs såhär i `default.el`:

Variabeln '`8bit-terminal`' sätts till `T` om `LC_CTYPE` är satt till `iso_8859_1` eller swedish. '`iso-insert`'-paketet laddas alltid. (Så man kan använda `Ctrl-X 8` som Compose tangent). Även '`iso-syntax`' laddas, plus att ÅÄÖ med mera fixas till.

Om variabeln '`8bit-terminal`' är `T` så:

Sätts Emacs om att visa 8-bitstecken direkt.

Annars (om den är `NIL`) så:

Om variabeln '`8bit-charset-string`' inte är satt så:

Kollas om `TERM` är satt till:

`vt102` -> då antas B = US ASCII, 1 = Svensk 7-bitars

`avi100` -> då antas A = US ASCII, B = Svensk 7-bitars

`vt220`, `vt320`, `finxterm` -> då antas B = US ASCII, H = Svensk 7-bitars

och så byggs strängen 8bit-charset-string upp enligt ovan.

Om variabeln '`8bit-charset-string`' nu är satt så:

Sätts variabeln '`8bit-terminal`' till `t`.
'`iso-swed`' laddas (den kod som sköter smartswitchandet mellan teckenuppsättningar på VT102an).
Strängen '`8bit-charset-string`' skickas till terminalen.

Om variabeln '`8bit-terminal`' nu är satt till `T` så sätts:

`kom-emacs-knows-iso-8859-1` till `T`

(Hmm.. Här kan det blir lite strul med inmatning av kommandon och text till Lyskom.el - med `kom-emacs-knows-iso-8859-1` satt till `T` så vill den ju även ha ISO-ÅÄÖ (Tex 'Återse' ...). Personligen har jag löst det genom att när jag kör igång `lyskom.el` globalt binda om klamrar till ÅÄÖ, men det är inget snyggt... Vore snyggt med en `lyskom-keymap`, finns det kanske?

För mig funkar det så att när jag har den ommappningen så blir klamrar ÅÄÖ, men om jag skriver Ctrl-Q först så blir det riktiga klamrar. Fast det funkar dåligt i minibufferten...

Om man vill att Emacs ska köra 8bitarsrent under en vanlig TTY så får man sätta `LC_CTYPE` till `swedish` eller `iso_8859_1` innan man startar den.

1. Handlar om svenska tecken i Lysators emacs-19 och är snott från LysKOM av Red.

Hur man i smyg utvecklar en databashanterare

eller

sommarjobb på vintern

eller

jag har också gjort ett hjul

eller

källarföretag högre upp

av Jonas S Karlsson

Det var en gång en firma, de hade två sommarjobbare anställda. Den ena tog det en dag för att löda en 25 polig kontakt den andra ritade något kretskort som inte skulle användas. Det vill säga vi har en helt vanligt normal-produktivt helsvenskt källarföretag. Alla vet vi ju hur små källarföretag är så låt oss ta fakta i besiktning och faktiskt lyfta fram att det inte alls ligger i källarvåningen utan på bottenvåningen - helt ok, fast då i en gammal lekskolas lokaler hrm...

Nåja för att komma tillbaka till ämnet, efter det att man börjar på högskolan anser man ju att man bör få trevliga sommarjobb, och till dessa räknar jag inte att rensa ogräs på kommunen, inte då i vart fall.

Ja databas-nånting var det, det börja som en mycket blygsam historia och är väl det fortfarande i någon mån. I USA hade det på senare tid dykt upp viss programvara som ännu inte hade hunnit till Sverige, och man ska väl inte vara sämre än Japsarna så man tar väl ideerna och gör nåt liknande. Det finns många exempel på mer närliggande fenomen, LYSATOR började ju på samma sätt, "LYS-16 - den helsvenska datorn från Flen" (om ni inte visste det byggdes den kommersiellt och marknadsfördes av ett företag i Flen, marknadsföraren kunde förexten träffas på en av i dagarna nyligen Unix-mässa där han marknadsförde en "helsvensk Unix från XXXX". Hur han lyckats få marknadsföringsjobbet är mig en helt okänt, då marknadsföringen av LYS-16 var urschell...).

Tillbaka till Japanerna som aldrig var med i bilden, men väl senare faktiskt... Vi gjorde som dem, "vad kan vi göra bättre, billigare och snabbare" var frågan vi ställde oss. Svaret var ju naturligtvis ett "helsvenskt program" (till försvar kan nämnas att Flen ej figurerar i sammanhanget). Företaget har genom nära(nde) utveckling i sammarbete med ett visst större (hel?)svenskt biltillverkarbolag lyckats prångla ut annan produkt, och varför skulle det inte gå med en ytterligare?

Jag skrev ett program som hanterade en applikations-databas med ett "1:n" förhållande, där "1" är överordnat och "n" underordnat, mycket finurligt. Det underordnade kan ses som en "logg" över förändringar som berör det överordnade, exempelvis ett utlåningsregister med grejor och lån. Databasen av fix-post format med fasta fält, byggd på nåt ISAM-liknande paket. Språket var naturligtvis Turbo Pascal, dBase var i åtanke men föga billigt då det hade måst levereras till varje kund, därmed också minskat vinst, jag hoppar över resonemanget, det är trivialt för folk som har (haft/fått) pengar.

Nätverk kom till PC, och blev vår nästvärk. Kunder vill köra i nätverk, trevliga nätverk har datorer med multitasking, PC är en dator, mer behöver väl inte sägas? Alltså, vi har inte ett trevligt nätverk, client-server är inte att tänka på, det har också en viss nackdel då det inte fungerar så bra i icke-nätverk. Då hittar vi väl en ISAM för nätverk då, med fildelning och -låsning och allt trevligt man kan önska. Sagt och gjort - "våfflan är eder" om än smetig.

Bara att sätta igång och byta ut den gamla modulen mot den nya, helt vanligt generationsskifte. Problematiskt? Nej, inte att byta ut! Men på nåt vis vid övergångar blir det lätt så att båda systemen samexisterar i en "övergångsperiod". Sämre än politiker kan man ju inte vara tänkte jag så Chefen(ägaren) bestämde att Medicinföretaget-som-vi-inte-nämner- vid-namn som ville ha fler fält skulle få det, det skulle "prioriteras upp", som allt annat, tyvärr. Därför infördes "Extrafälten", de hade unika egenskaper då de av kunden själv kunde defineras, ändras och därigenom få databasen att tappa data.

Har vi nu nått en ohållbar situation? Nejdå, onej (vi vill ju fortfarande jobba där på sommaren och vid sidan om, ogräs ***ÄR INTE KUL*** att rensa, ogräs ***ÄR INTE KUL*** att rensa...), Två databaslådor i samma program, blir inte det onödigt mycket kod, och dubbelt så sakta? Oneeej - de får ju lika stor cache var 256KB eller så... Ånej, nu var det dags för nätverksversionen med bara nätverkslådan, och alla fält mer eller mindre definerbara. Användare kan dessutom designa egna inmatningsformulär, utskriftsformulär (rapportgenerator), makron för sökningar. Sökningarna som kan vara otroligt avancerade utan att man märker det, jag menar "Gubben på Golvet" (TM) vet inte vad Snitt och Union är, men kräver dock att få använda det.

Fortfarande har vi "1:n" i databasen med det hindrar inte att vi nu öppnar flera databaser, i ren och fin objektorienteringsanda motsvaras gamla programmet av ett Objekt som kan instansieras många gånger. Jag arbetar på det. Det vore föresten roligt om man kunde på något finurligt sätt koppla samman dessa med varandra i sökningar och så, ja ni vet som om det vore tabeller...

Vi har också funderat på att kunna ha formler i fälten, men det får nog vänta till nästa version.

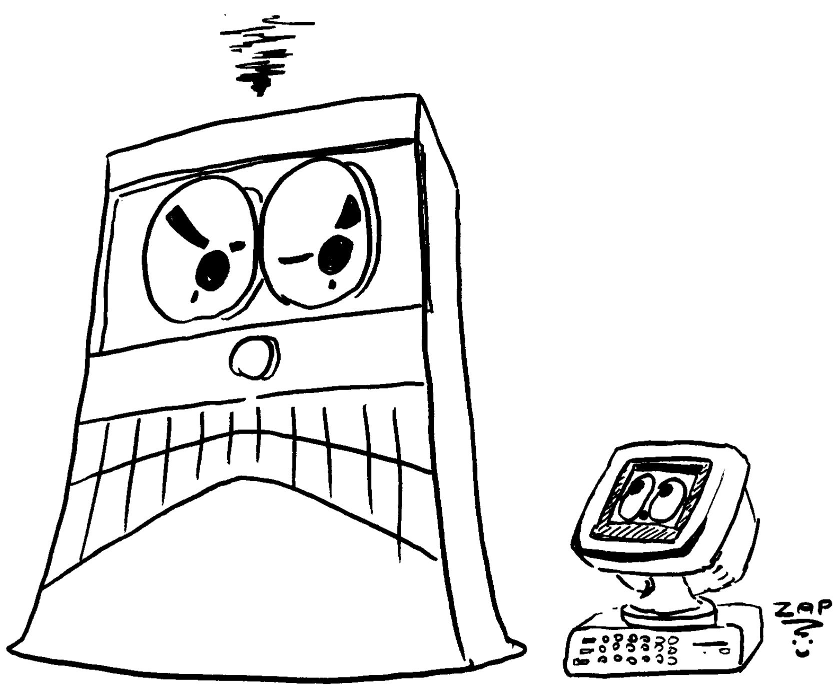
Men man får nog nöja sig med att faktiskt ha gjort en programvara som står **ÖVER** fönstersystem, vi startar nämligen upp WINDOWS och ett scan-program för bilder. Detta kan man göra i ett av inmatningsfälten med en enkel tangenttryckning. Chefen kallar funktionen för "Programväxel".

...

Jag kan för övrigt nämnas att kontakten, den 25-poliga, fungerade mycket väl. Men tyvärr så tillkom arbetet att i efterhand tillfoga hylsan som kontakten skulle suttit i, som tur var var jag den andra.

Jonas S Karlsson,

direkt från Verkligheten (TM).



BNC-kontakternas sång

Melodi: Båtlåt

Nyare text: Linus Tolke

En BNC han sa' till en hona va
du var stilig
vi borde koppla varann
jorde me' varann
och leda lite grann
som vi koaxer kan

Honan hon sa "ja
klart att jag vill va
med och leda
känna ditt stiliga krom
inte långt ifrån
nej bara några Ohm
du skickar elektron.

När vi kopplat in bodil och robin med varan-
dra
kan vi ta ledigt och vila en stund
Inget kan oss fresta sitta som den bästa
hane och hona

Aldrig ens tänka på att komma loss
men (förståss)

Om någon av oss ser
andra BNCer
hänga lösa
kan vi på BNCers sätt
öppna bajonett
och sedan med ett snäpp
förena oss så lätt

Vi har ett jättehack.

Melodi: Roy och Rogers mack.

Nyare text: Linus Tolke

Elispen parseas, elispen körs
i min jätteemacs
Garbage collecting, väntar en stund
men strax så är det dags
för Garbage collecting...done
allting går igång på samma gång

Jag och Stallman har ett hack ihop
Jag och Stallman har ett liv ihop
Jag och hela fria världen har ett jättehack till-
sammans
Skall jag nånsin ändra i en fil
exec emacs är min .profile
Jag och hela fria världen har ett jättehack till-
sammans

Nya finesser fixas just nu
i min jätteemacs
Versionsnumren sprutar ut ifrån GNU
till min jätteemacs
plötsligt en bugg dyker opp
det säger stopp
jag skickar rapport.

42

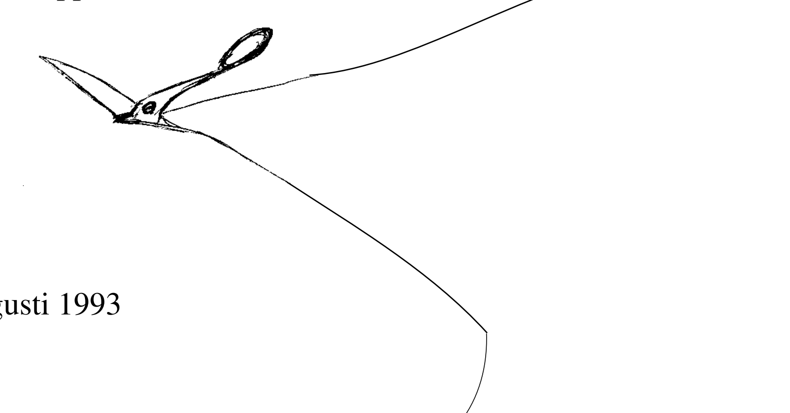
av noppe

Svaret. Alla känner väl till det vid det här laget. Men frågan då? Vad man alltsom oftast inte tänker på är att 42 bas 10 är 54 oktalt. 54 är med romerska siffror liv, för det har aronsson sagt. Och liv är just vad den här artikeln ska handla om, för det har redaktören sagt. Därför sitter jag nu och skriver om livet. Till att börja med ska vi dissekera livet i Q-huset. Det verkar för många bestå av att tillbringa dagar och nätter i ända framför en skärm spelandes mud. För den som inte vet är mud en virtuell verklighet där folk utlever sitt virtuella liv genom att springa runt efter bästa förmåga och befria andra från detsamma. Virtuell verklighet är ett (egentligen två) modeord, vilket innebär att man kan tjäna pengar på det. Att man kan tjäna pengar på nåt implicerar inte att det fungerar. Se bara på Mic...... äsch, förresten, glöm det. De lysiter som inte spelar mud spelar nåt annat, exempelvis flipper. Flipper är ett högst reellt spel där man med ett par små plastbitar slår på en stålkula som därav far ned i ett hål i spelplanen och så får man börja om. Så kan man hålla på i timmar. Undrar just varför ... Nåja, en stålkula är väl inte direkt bekant för sin låga smärttröskel, så man bör kanske roa sig med att slå flipperspelare med små plastbitar istället. Hittar jag några (plastbitar alltså, flipperspelare behöver man inte leta efter) ska jag prova.

Om man i sitt sökande efter liv beger sig utanför Q-huset, vad hittar man då? I Ryd hittar man folk som spelar rydnet, som är en virtuell utsträckning av Q-huset, fast utan flipper och lysfika. I Ryd finns dessutom icke-rydnetansluta, men de räknas inte. Och även om man räknade dem skulle deras antal ändå inte vara 42, så det så. Resten av Linköping innehåller inget liv alls. Som tur är är detta inte typiskt för större delen av jorden. Det finns gott om livsformer som kan indelas i fyra olika arter: Jag, saker som växer i mitt kylskåp, och saker som är i vägen. Dessa arter kan sedan delas upp i flera underarter, undantaget mig, som om någon försöker dela kommer att slå vererbörande med något från andra arten. Av saker som är i vägen märks speciellt folk, som inte är en underart utan avart. Man gör sitt bästa för att ignorera folk, men de märks i alla fall. Folk gör nämligen i stort sett vad som helst för att bli uppmärksammade. Ju mer man ignorerar dem desto mer börjar de likna T. DiLeva. Helt apropå ingenting har vi en depression i Sverige numera. Resten av världen spelar mud, fast utan datorer.

De är också deprimerade.

En stor fråga är: Finns det liv i andra delar av universum? Svaret på den frågan är inte 42, för den frågan finns längre ner i spalten, liksom ett bevis för Fermats sats och ett recept på guld, eftersom jag för skojs skull funderat på dessa frågor ett tag. Men hur var det nu med liv i universum? Svaret är med stor sannolikhet Ja. Än sen då? Precis, vad spelar det för roll att det finns små bakterier och abborrar i ett träsk på en planet i lilla Magellanska molnet? Ingen alls, om man inte är sportfiskare förstås, men även då är det mindre intressant på grund av den akuta bristen på metmask där, vilket högst väsentligt skulle påverka fångstresultatet negativt om man kunde ta sig dit, men det kan man inte. Eftersom man inte kan ta sig dit kan man alltså inte ta med mask från jorden, och på det viset undgå ovannämnda brist. Nej, vad man vill veta är om det finns intelligent liv i andra delar av universum. Intelligent liv på jorden finns, i exakt ett exemplar, jag, och .. (redaktören vill också räknas som intelligent, men det går jag inte med på, fast han hotar med att klippa resten av art



Styrelsen[1] i Lysator, 1993-94

Ordförande:	Ture Pålsson	013 - 26 13 10 ture@Lysator.LiU.SE
Kassör:	Jens Larsson	013 - 17 81 74 jens@Lysator.LiU.SE
Sekreterare:	Björn Olofsson	013 - 17 28 35 bear@Lysator.LiU.SE
Ledamöter:	Tom Carlsson	013 - 17 69 80 tomca@Lysator.LiU.SE
	Jones Desougi	013 - 17 50 92 axel_f@Lysator.LiU.SE
	Per Hedbor	cardeci@Lysator.LiU.SE
	Henrik Rindlöw	013 - 17 82 12 rindlov@Lysator.LiU.SE

1. Har tyvärr inte fått information om adresser, etc—Red.