Uptime

Datorföreningen Updates nyhetsblad

Redaktör: Bjarni Juliusson

Nummer 6 2023

Det har pratats ett tag afk om en artikelserie med namnet "Apptajm", där vi kan tipsa om användbara applikationsprogram och introducera läsaren till deras funktion. Nu bestämde jag mig för att mjukstarta serien med en kort artikel om sed, ett verktyg som de flesta medlemmar nog känner till men inte alla kanske har bekantat sig riktigt med. Har du ett favoritprogram att tipsa om? Skicka in ditt bidrag till Uptime!

Vi söker också fler föreläsare till vår föreläsningsserie Updateringar. Kan du tänka dig att tala i en halvtimme eller mer, ämne helt valfritt, hör genast av dig till styrelsen! Föreläsningen kan ske online eller i lokalen, och brukar hållas kvällstid på en helg.

Som vanligt, har du nyheter eller förslag till Uptime, eller har du skrivit något du vill dela med dig av, skicka ett brev till uptime@dfupdate.se!

Vill du komma i kontakt med Update kan du skriva till styrelsen@dfupdate.se eller till vår allmänna diskussionslista update@lists.dfupdate.se. Du är också välkommen att besöka vår IRC-kanal #update på EFnet eller lokalen på Svartbäcksgatan 65 i Uppsala.

Närmast planerade händelser

Onsdagsmöte 7/6 kl 18
Restaurang bestäms på IRC och Updatelistan.

Brädspelskväll fredag 9/6 kl 18 på Svartbäcksgatan 65
Flera spel finns att välja mellan. Lättsam och nybörjarvänlig atmosfär. Pizza är sannolik.

Manual-sweatshop lördag 10/6 kl 15:00 på Svartbäcksgatan 65
Vi gör ett ryck med att skriva användarhandledningar till de körbara datorerna på utställningen.

Updatering lördag 10/6 kl 19:00 på Svartbäcksgatan 65 och via BBB
Hackerspace Design Patterns – What can Update learn from them?, Anke Stüber (Update)
The Hackerspace Design Patterns address common problems and solutions for hackerspaces, like how to make decisions, how to keep the space clean, or how to involve new members. In this workshop I want to present some of the design patterns and discuss how they relate to Update.

Torsdagssweatshop 15/6 kl 18 på Svartbäcksgatan 65
För oss som har för många projekt och för få resultat. Pizza beställs gemensamt.

Datorsweatshop tisdag 20/6 kl 18 på Svartbäcksgatan 65
Vi utforskar, reparerar och leker med månadens dator: Compis.

Plenumsmöte onsdag 28/6 kl 18:30 på Svartbäcksgatan 65
Välkommen att delta i vår löpande planering och utvärdering av verksamheten.

Juni månads dator

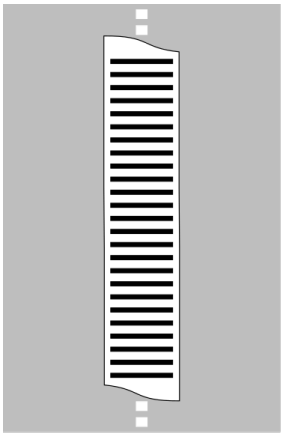
Vi fortsätter på sverigetemat med Compis! Från och med nu har vi också en sweatshop varje månad då vi ägnar oss åt månadens dator.

Nyheter

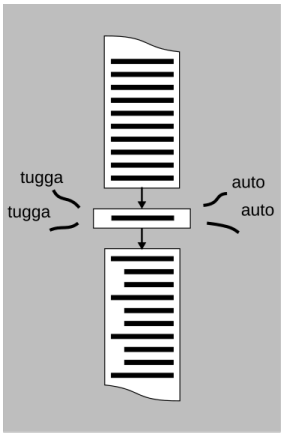
■ Plenumsmöte hölls den 24/5. Protokollet finns här. De viktigaste punkterna var viss omfördelning av ansvar och omprioritering av uppgifter; nyheten att det finns visst förrådsutrymme ledigt hos en av medlemmarna som kan hjälpa oss lätta belamringen av lokalen; och införandet av ett poängsystem med guldstjärnor för medlemmar som uträttar arbetsuppgifter åt föreningen.
■ Nya tag har tagits med utställningen; status finns här. Skyltarna för ABC 80, Amiga 500 och Macintosh Plus får just nu sin sista handpåläggning och har fått sin sista deadline i veckan. Skyltar för PDP-12, PDP-8, och Cray Y-MP EL har plockats fram från tidigare utställningar och ska återanvändas. Lagring har ordnats för Commodore 64, med flera olika nyttoprogram och spel som enkelt kan laddas av besökare.
■ Vi välkomnade en ny medlem. Välkommen, erk!

Apptajm: sed

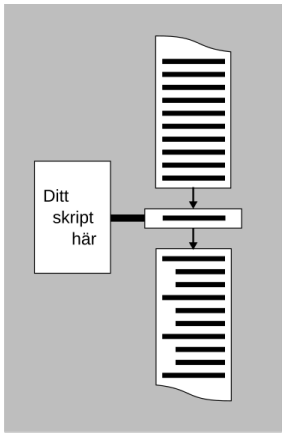
sed är ett klassiskt Unix-verktyg som många av oss använder mer eller mindre dagligen för att på olika sätt formatera om och anpassa text som ett eller annat program spottar ur sig. Med dagens ögon kan det dock te sig en smula kryptiskt, så en liten förklaring av programmet för de mer *casual* datoranvändarna kanske är på sin plats. Det hela är tämligen enkelt:



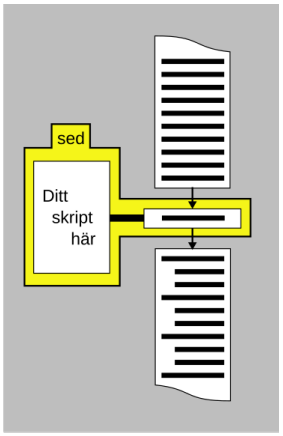
Du har en massa text som du behöver göra en miljon lika-dana rättelser i



Tänk dig att du skickar den rad för rad genom en automatisk editor som gör rättelserna



Tänk dig att editorn styrs av ett skript som du skrivit som säger hur varje rad ska redigeras



Det är sed!

Rent praktiskt kör vi ofta sed som ett led i en pipeline i skalet:

```
$ något kommando | sed 'skript här' | något annat kommando
```

Men vi kan också låta sed läsa igenom och redigera innehållet i en fil:

```
$ sed 'skript här' textfil.txt
```

Vi kan också spara skriptet i en fil om vi vill, istället för att skriva det direkt på kommandoraden:

```
$ sed -f skriptfil textfil.txt
```

Oavsett hur vi startar sed är dess funktion i grund och botten densamma: den läser input-texten rad för rad, och för varje rad genomför den de redigeringar som vi beställt i vårt skript, och så skriver den ut den redigerade raden.

De flesta redigeringar vi behöver göra är väldigt enkla. Ofta är det bara ett enda kommando vi behöver använda i vårt sed-skript, oftast kommandot 's', som substituerar en text i stället för en annan:

```
s/regexp/ersättning/
```

Detta kommando matchar inputraden mot ett reguljärt uttryck, och ersätter matchande text med ersättningstexten. Rader som inte matchar lämnas orörda. Vi kan fånga delar av den matchande texten med parenteser och klistra in de fångade fragmenten i ersättningen med **\1**, **\2**, etc, om vi t.ex. vill byta plats på delar av texten på raden:

```
$ echo "X Y" | sed 's/\(.\) \(.\)/\2 \1/'
```

Output: `Y X`

Punkter matchar vilka tecken som helst, som vanligt i reguljära uttryck. Av default använder sed så kallade "basic regular expressions", som kräver backslashes före parenteser med flera andra specialtecken. Tycker man att det blir för oläsligt kan man ge flaggan `-E` för att få extended regular expressions, som då ser ut såhär:

```
$ echo "X Y" | sed -E 's/(.) (.)/\2 \1/'
```

I ett mer användbart exempel kan vi istället för `'.'` använda `'[a-zA-Z]*'`, där `[a-zA-Z]` matchar alla tecken i de två spannen a-z och A-Z, och `*` är Kleene-stjärnan, som alltså matchar det föregående uttrycket så många gånger som det går.

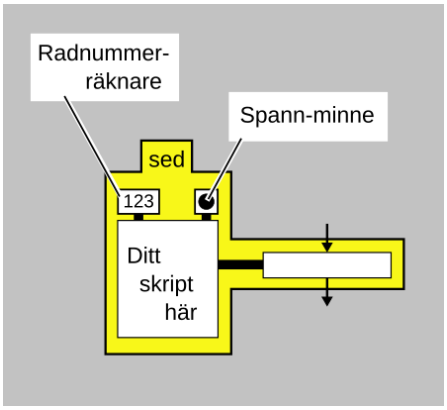
```
$ echo "Haddock, Kapten" | sed -E 's/([a-zA-Z]*), ([a-zA-Z]*)/\2 \1/'
```

Output: `Kapten Haddock`

Ett annat enkelt kommando är 'y', som översätter en uppsättning tecken till en annan:

```
$ echo "Ha ha, ho ho" | sed 'y/ao/ei/'
```

Output: `He he, hi hi`



Utökad sed-maskin med tillstånds-variabler för att hantera adresser

Adresser

Mekanismen i sed är i själva verket lite mer avancerad än att den bara applicerar en och samma redigering på varje rad. Därtill finns villkor man kan lägga till för att säga *vilka* rader som ska redigeras. Dessa kallas *adresser*, och kan utgöras av ett radnummer eller ett reguljärt uttryck mellan snedstreck `/såhär/`. Dessa skrivs före kommandot de gäller. Man kan också ange ett spann, för att välja alla rader mellan två adresser. Dessa skrivs med ett kommatecken emellan: `/så/,/här/`. I sed finns alltså tillståndsvariablar som överlever från en rad till nästa: ett räkneverk som håller reda på vilken rad i texten vi är på, och en flagga för varje spann som säger om vi har sett startraden men inte slutraden, och alltså befinner oss inuti spannet nu. Detta utökar flexibiliteten avsevärt.

För att t.ex. klippa bort alla rader från raden med "begin rant" till raden med "end rant", kan vi använda ett adress-spann tillsammans med delete-kommandot 'd':

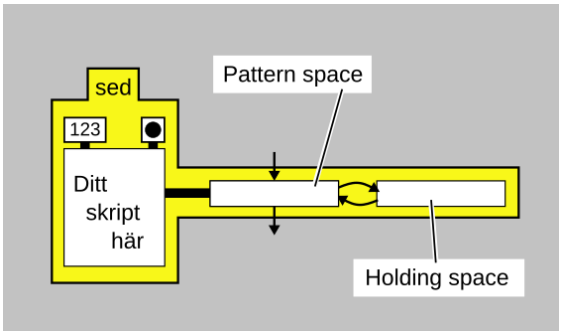
```
$ sed '/begin rant/,/end rant/d'
```

Vi kan också invertera villkoren genom att sätta ett utropstecken efter:

```
$ sed '/begin rant/,/end rant/!d'      # Tar bort allt utom rantandet
```

Och man kan också plocka ut ett visst *antal* rader, t.ex. en matchande rad och raden efter:

```
$ sed '/nästa rad är strunt/,+1d'
```



Holding space

I själva verket är sed-maskinen ännu lite mer avancerad än så, för det finns förutom radbufferten som redigeringen görs på också en till, extra, radbuffert som man kan stoppa undan text i tillfälligt. Den vanliga bufferten, som all text passerar igenom, heter *pattern space*, och den andra bufferten heter *holding space*.

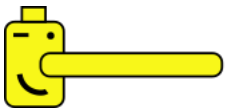
Kommandon finns för att kopiera text från pattern space till holding space och vice versa, samt för att tillfoga text till den text som redan finns, och för att utväxla innehållet mellan pattern space och holding space. Åter igen gör detta stor skillnad för flexibiliteten med mycket enkla medel. Vi kan t.ex. repetera den föregående raden om vi stöter på `-"-` i texten, med hjälp av hold-kommandot 'h' och get-kommandot 'g', som sparar undan och återkallar pattern space till och från holding space:

```
$ sed '/-"-/g;h'
```

Här har vi använt två kommandon, vilka då skrivs med ett semikolon emellan. Observera att adressen endast gäller kommandot den står före, dvs bara 'g' i det här fallet. Vill vi applicera en adress på flera kommandon kan vi gruppera ihop dem med måsvingar. Exemplet här nedan flyttar t.ex. alla rader som matchar "obs" till sist i texten, genom att tillfoga dem till holding space med 'H' och ta bort dem ur pattern space, och sedan med adressen '$' och kommandot 'G' tillfoga all den ackumulerade texten i holding space till den sista raden:

```
$ sed '/obs/{H;d};$G'
```

Lägg för övrigt märke till hur mycket sed liknar en legogubbe med lång näsa och ett snett leende som blinkar åt oss! Det kan inte sägas om många program.



Hopp och labels

Till sist vill jag visa en typ av kommandon som får sed att nästan likna ett riktigt programmeringsspråk, och som åter igen förbättrar flexibiliteten för oss. Det rör sig om möjligheten att hoppa omkring i skriptet, villkorligt eller ovillkorligt, inklusive att repetera en del av skriptet i en loop om man vill.

Kommandona som används för detta är ':' för att placera en label, 'b' för ovillkorligt hopp, 't' för villkorligt hopp om ett s-kommando har matchat, och 'T' om inget s-kommando har matchat, sedan senaste t/T-kommandot. Hoppkommandon följs av namnet på den label de ska hoppa till, eller så kan de användas utan label för att hoppa förbi resten av skriptet och därmed avsluta redigeringen av den nuvarande textraden.

Här är ett exempel som slår samman långa rader som wrappats med '\':

```
$ sed ':l; /\\$/{N;s/\\\n//;bl}'
```

Ser det krångligt ut? Vi kan skriva ut det på flera rader för att göra det mera läsligt:

```
:l
/\\$/{
  N
  s/\\\n//
  bl
}
```

Det börjar med att definiera en label 'l' i början av skriptet. Därefter följer en adress /\\$/ som matchar rader som slutar med ett '\'. Resten av skriptet står inom {måsvingar}, och alltihop utförs alltså endast på matchande rader, dvs rader som har wrappats och ska slås samman med raden efter. Kommandot 'N' läser in nästa textrad och tillfogar denna till slutet av texten i pattern space. Kommandot s/\\\n// ersätter backslash+newline med ingenting. Slutligen loopar kommandot bl tillbaka till labeln 'l'. Loopen avbryts när adressen inte matchar, dvs när vi läst in en rad som inte slutar med ett backslash, varpå all texten i pattern space skrivs ut som vanligt och sed går vidare till nästa rad med tom pattern space.

Vi tar ett exempel med ett villkorligt hopp också. Här är ett skript som skriver ut namnet på alla genier ur texten, och inga andra rader:

```
$ sed 's/är ett geni//;t;d'
```

Här provar vi att redigera bort texten "är ett geni" med ett s-kommando. Sedan använder vi ett t-kommando för att hoppa till slutet av skriptet om s-kommandot lyckades, dvs om det stod "är ett geni" på raden (vilket i så fall har redigerats bort). Om hoppet tas hoppar det över d-kommandot, som annars slänger bort raden. Resultatet är att endast rader med "är ett geni" skrivs ut, men utan den texten, så "Bobo är ett geni" skrivs ut som bara "Bobo", medans "Laban är dum" inte skrivs ut alls.

Ett alternativ är att använda sed:s flagga -n, som gör att den inte automatiskt skriver ut pattern space efter redigeringen av varje rad. Då kan vi skriva om kommandot ovan såhär:

```
$ sed -n 's/är ett geni//;T;p'
```

Här använder vi kommandot 'p', som skriver ut pattern space, och istället det omvända villkorliga hoppet 'T' för att hoppa över p-kommandot om s-kommandot *inte* matchade. Effekten är densamma.

Till sist vill jag bara säga att det givetvis går att spara sed-skript i filer och stoppa en Unix-shebang i dem och få ett körbart program. Vill vi t.ex. göra ett skript som tusengruppifierar långa nummer kan vi göra det såhär:

```
#!/bin/sed -f

:l
  s/\([0-9]\)\([0-9]\{3\}\([^0-9]\|$\)\)/\1,\2/
  tl
```

Sparar vi det skriptet i en fil med namnet "tusengrupper", säg, och gör den filen körbar med chmod, så kan vi sen köra det med ./tusengrupper precis som vilket program som helst. Så kan man alltid göra, inte bara med sed, men det är kanske något som är värt att påpeka för medlemmar av den här artikelns målgrupp.

Det finns betydligt mer funktionalitet än så här i sed, och jag rekommenderar att läsa man-sidan för att se allt som finns. Av alla standardverktygen i Unix är sed ett av de allra mest användbara, och en genomläsning av mansidan är väl spenderad tid!

Månadens dator: ABC 80

Denna månads dator är en svensk klassiker från 1978. ABC 80 skapades på initiativ av Scandia Metric, som under 1970-talet sålde förra månadens dator Alpha LSI till bl.a. skolor i Sverige, men började känna att det fanns behov av en mikrodator. Man slog sig i slang med Dataindustrier AB (senare DIAB), som hade erfarenhet av att utveckla datorer som bl.a. DataBoard 4680, en familj av standardiserade kort och bakplan som industrikunder själva kunde komponera sin datorutrustning med. Man bestämde sig för att utgå ifrån DataBoard 4680 med en Z80-CPU som arkitektur för den nya mikrodatorn. Dataindustrier skötte designen, och så anlitade man en tredje part, radio- och TV-tillverkaren Luxor AB, för att sköta tillverkning och service. Luxor hade också butiker runtom i landet där de kunde sälja datorn, medans Scandia Metric skötte försäljning till skolor och företag. Projektet startade i februari 1978, och i augusti var datorn färdig att presenteras för allmänheten. Priset var drygt 7000kr, omkring 25.000kr i dagens pengar, vilket var en bråkdel av vad t.ex. en Apple II kostade.

Det blev succé. Beställningarna kom in fortare än Luxor kunde tillverka datorerna, och i slutet av 1979 var ABC 80 den största persondatorn i Sverige med 80% av marknaden. Siffrorna höll i sig i några år till, medans 8-bitars datorer var gångbara, men tappade i mitten av 80-talet marknadsandelar till IBM PC. Produktionen upphörde i slutet av 1985, och man hade då sålt 33.000 exemplar.

Arkitekturen hos ABC 80 är ganska typisk för en mikrodator från slutet av 1970-talet: en 3MHz 8 bitars Zilog Z80-CPU, 16Kbyte RAM, och 16Kbyte ROM med en inbyggd BASIC-tolk som startar direkt när man slår på datorn. Skärmen visar 24 rader svartvit text om 40 kolumner vardera.

Även viss grafik är möjlig. Ett särskilt tecken används för att växla visningen av en rad på skärmen till grafikläge, och i detta läge ersätts delar av teckenuppsättningen med grafiktecken om 2x3 pixlar, där teckenkoden avgör vilken kombination av pixlar som är tända. På detta sätt kan man kombinera ihop en skärmbild med upp till 78x72 pixlar fri grafik blandat med mer högupplöst text.

För ljud har datorn ett SN76477, ett mycket enkelt ljudeffektschip som mest användes i leksaker och spel (bland annat Space Invaders!), och som inte är fritt programmerbart utan delvis analogstyrt och kräver externa kretsar för att t.ex. sätta tonhöjd. I ABC 80 har kretsen konfigurerats för en djupt inskränkt repertoar bestående av pling i två olika tonhöjder, ett fräsande ljud, och en sireneffekt.

Data lagrades från början på en av användaren själv tillhandahållen kassettbandspelare via en utgång på datorns baksida. Snart kom en särskild bandspelare ut med namnet ABC 820, som sedan såldes ihop med datorn. Data lagrades i 700 bps, vilket ger ett par hundra kilobyte kapacitet på en 60-minuterskassett.

Förutom kassettporten finns på baksidan av datorn också en separat kontakt för motorstyrning till kassettbandspelaren (även användbar för valfri annan funktion), en serieport, en reset-knapp, en kontakt för skärm, samt en långsmal kontakt för bussexpansion, den så kallade ABC-bussen. Denna är baserad på den tidigare nämnda 4680-bussen för vilken ett brett utbud expansionskort fanns med funktioner för styr- och mättillämpningar, anslutning till olika instrument etc, vilket gjorde ABC 80 populär i industrin. Skärmen är en modifierad version av en svartvit TV-modell som Luxor tillverkade, särskilt avsedd för ABC 80, och innehåller också datorns nätdel. Skärmen är alltså nödvändig för att förse datorn med ström.

Diskettstationer, som tillverkades i flera olika modeller, anslöts via ABC-bussen. Dessa tog formen av en expansionslåda, med plats för kontrollerkortet för diskettenheten samt ytterligare kortplatser för vidare expansion. I expansionslådan satt också ett ROM med diskettoperativsystemet, som i likhet med många andra operativsystem vid tiden hette "DOS".

T.v. syns ABC 80 med bandspelare ABC 820, diskettenhet FD2 UD, samt bildskärm; t.h. en ABC 800 med bildskärm

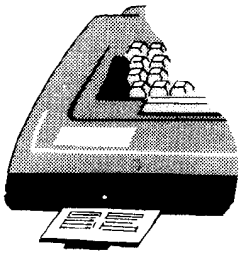
Update har två diskettexpansioner till ABC 80: modell FD2 UD respektive FD4 UD. Bokstaven U anger att modellen är en bred låda med plats för att ställa skärmen ovanpå, och med plats för sex expansionskort inuti. Disketterna är 5,25 tum, double density. FD2 UD är enkelsidig, 160Kbyte per diskett, och FD4 UD är dubbelsidig och lagrar 320Kbyte. Det fanns också varianter utan "U", som var en mindre låda med bara en kortplats, samt en FD2 utan "D", som använde single density-disketter om 80Kbyte. Alla modellerna innehåller två stycken diskettstationer. I U-modellerna är dessa placerade på varsin sida om skärmen.

Expansioner kom också ut för RAM upp till 32K och 80 kolumners text. Exempel på kringutrustning som fanns att köpa är modem, skrivare, plotter, och digitaliseringsbord, samt alltså en mängd olika gränssnitt för givare, reläer, osv.

Datorn har en inbyggd tolk för en egen dialekt av programmeringsspråket BASIC. Den är någorlunda kompetent och ganska snabb, och har stöd för bl.a. omnumrering, modifikation av programrader, infogning av programrader från fil, och plottning av grafik. Dialekten stöder flyttal, 16-bitars heltal, samt strängar, och även arrayer av alla typer. Funktioner finns också för BCD-aritmetik med strängar. Variabelnamn är dock bara en bokstav eller en bokstav och en siffra, följt av typkoden '%' för heltal och soltecken '¤' för strängar (dvs inte det vanligare '$'). Kommandon finns också för att läsa och skriva datafiler på band. Varken labels eller funktioner finns (utöver DEF FN), utan all flödeskontroll använder radnummer, som typiskt är för tiden.

En till ABC 800, expansionskort, ett "ABC-lab", kassetter, böcker. I bubbelplast: en till ABC 80, FD4 UD, och skärm.

Det man saknar mest är dock felmeddelanden. När man gör nåt fel får man istället ett nummer, och får slå upp själv vad det betyder. För att underlätta finns dock ett kort med alla felnummer och förklaringar fastsatt på undersidan av tangentbordet, och kan dras fram med ett litet handtag för snabb referens!



Felmeddelanden under datorn

I januari 1980 bildades ABC-klubben för att samla användare av ABC 80. Klubben gav ut ABC-bladet (arkiv här), ett magasin med artiklar, tips, tester, och BASIC-program man själv kunde skriva in på sin dator. Man skickade också ut kassetter med mjukvara till sina medlemmar, och sände även programvara i radio som datoranvändare kunde spela in på kassettband och ladda i sina datorer. Klubben finns fortfarande kvar, se deras website, men har ändrat sitt fokus till mer moderna datorplattformar.

1981 följdes ABC 80 av ABC 800, en utökad modell med bland annat högre skärmupplösning och mer minne. Update har två exemplar av vardera modellen.

För mer dokumentation av ABC 80, se t.ex. dessa två arkiv.

Vad vi gjorde under maj månad var huvudsakligen att plocka fram alla ABC 80-relaterade saker ur förrådet, inklusive ABC 800, och gå igenom vad som fanns. En ABC 80 och en skärm var redan uppackade efter flytten, så dem kopplade vi ihop för att testa att köra datorn, men det skulle visa sig besvärligare än väntat.

Sagan om den eländiga smittsamma trasigheten

När vi slog på strömmen visade datorn bara skräptecken på skärmen. Reset hade ingen effekt. Då packade vi upp den andra datorn och kopplade in den, men den visade också bara skräp. Det här var ju märkligt, eftersom vi demonstrerade en av dem på VCF Berlin häromåret, och då fungerade den bra.

Då tänkte vi att det kanske kunde vara något fel med nätaggregatet, som sitter i skärmen, så vi packade upp och kopplade in den andra skärmen istället. Då fungerade det. Problemet var då att T-tangenten var defekt. I övrigt fungerade tangentbordet fint. Så vi testade att koppla in den andra datorn i den fungerande skärmen. Den fungerade inte, så där har vi en trasig ABC 80. Då bytte vi tillbaka till den fungerande datorn, men till vår häpnad fungerade då inte den heller längre, utan gav åter bara skräp på skärmen.

Vi misstänkte fortfarande nätdelen, så vi letade upp pinout för skärmkontakten och mätte spänningarna. +9V fanns där och -17V fanns där, men +17V saknades. Detsamma gällde den andra skärmen. Vi plockade fram kopplingsschemat, och såg att det finns separata säkringar inuti skärmen för var och en av de tre spänningarna. Då misstänkte vi att det var den första datorn vi hade testat som hade förstört säkringen för +17V-matningen, först på den första skärmen, och sedan på den andra. Vi öppnade skärmarna, och fann att de båda hade trasiga säkringar för +17V. Tyvärr hade vi ingen reserv, men eftersom säkringarna för +17V och -17V är likadana hade vi ändå två stycken fungerande: en i vardera skärmen. Så vi bytte dem mellan skärmarna så att vi fick en skärm med alla säkringarna hela.

När vi sedan kopplade in vad vi *noggrannt påminnt oss* var rätt dator, fungerade allting som det skulle igen. Förutom T, alltså.

Sagan om tangentbordets beklagansvärda jolmighet

Så vi plockade isär datorn också, och undersökte tangentbordet. Tangentborden i ABC 80 är av en, för att använda lite modernt språk, *cursed* typ som kallas "foam-and-foil". Dessa har en liten bit skumgummi på undersidan av varje tangent, med en bit metallfolie i sin tur fastlimmad på undersidan av denna. Denna foliebit trycks mot kretskortet, och skumgummit fungerar som fjädring.

Tyvärr förvandlas oundvikligen, som alla datorhobbyister vet, skumgummi efter ett antal år till en bajsliknande substans som inte alls har samma mekaniska egenskaper som skumgummi. Detta är skumgummits sanna form, som en gåva av guld och ädelstenar man givits av djävulen som sedan förvandlas till sågspån till morgonen. Under vår presentation på VCF Berlin ledde detta till att tangentbordet på demo-datorn helt slutade fungera, och vi genomförde då en komplett restauration med nytt färskt skumgummi (ve oss i framtiden), vilket vi också demonstrerade live som en del av vår utställning.

I det här fallet kunde vi dock inte se något fel med T-tangenten. Skumgummit och folien satt båda där de skulle och allt såg riktigt ut. Vi plockade därför isär den andra datorn för att se om vi kunde låna kretskortet till tangentbordet för att se om det var det som det var fel på. Då upptäckte vi att det inte var så förvånande att den datorn hade haft sönder säkringar för oss; insidan var full av rost, och skruvarna i tangentbordet hade börjat krypa ur sina hål och plonka ner på moderkortet i datorn, till allmän kortslutelse. Den hade också givetvis helt och hållet pulvriserat skumgummi, som skräpade ner och tappade foliebitar överallt när vi tog isär tangentbordet. I vilket fall som helst lånade vi tangentbordskortet, testade detta i den fungerande ABC 80:n, och konstaterade att det inte gjorde någon skillnad — T var fortfarande trasigt.

Som ett sista test tog vi tangentsatsen, med alla de fungerande tangenterna i, och tryckte in en av dessa och använde den för att liksom dutta på kretskortet vid positionen för T-tangenten. Detta gav T på skärmen, och vi konstaterade alltså att det är något mystiskt fel med själva foliedelen på T-tangenten, som inte går att se men som gör att just den foliebiten inte registreras som en tangenttryckning. Vi gav upp för den här gången och skruvade ihop datorerna igen.

T började fungera igen lite halvt, helt av sig självt.