SAKU

COLLABORATORS

	TITLE : SAKU	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 13, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SAKU

1.1 Suomen Amiga-käyttäjien unioni ylpeänä esittää: AmigaGuide-SAKU

« AmigaGuide-SAKU »

2/95 #11 - 1. maaliskuuta, 1995

« Kansilehti »

Sisältö

1.2 Pääkirjoitus

Pääkirjoitus

Taas on kaksi kuukautta vierähtänyt, ja on uuden Sakun aika. Määrällisesti jäätiin kyllä artikkeleissa edellisestä numerosta, mutta kymppi olikin erikoisen massiivinen. Pienenä toiveena esittäisin, että kirjoittajat toimittaisivat artikkelinsa vastaisuudessa hieman aikaisemmin. Viimeiset pari päivää ennen julkaisua hurahtivat taas kerran viime hetkellä tulleiden artikkeleiden muokkaamiseen, ja kiireessä voi tulla virheitä...

Disketti-Sakun koodia on taas hieman paranneltu. Suurin uudistus on tuki lehden jakamiseen kahdelle levykkeelle. Monimutkaistunutta asennusta varten mukana levitetään myös erillistä asennusohjelmaa. Muut uudistukset koskevat lähinnä virhetilanteiden hallintaa.

Disketti-Sakun käyttäjien on tärkeätä muistaa, ettei SAKU oikeastaan tiedä levyjen vaihdosta mitään, se vain olettaa löytävänsä tarvitsemansa tiedostot nykyisestä hakemistosta, ja jos se ei niitä löydä, se pyytää vaihtamaan levykettä. Ohjelma on suunniteltu toimimaan näin, jotta toiminta kovalevyllisissä koneissa olisi mahdollisimman läpinäkyvää. Levykkeiltä lukiessa korppua ei pitäisi joutua vaihtamaan muulloin kuin lehteä käynnistettäessä, joten lisälevyasemien omistajia tuskin ylimääräisten asemien tukematta jättäminen suuremmin harmittaa.

Muita uudistuksia Suomen Amiga-käyttäjien saralla ovat Sakunetin avaaminen ja Amphibia-projektin eteneminen. Sakunet on uusi Fidonetin kaltainen, vain paljon paljon pienempi ja vapaampi purkkien välinen viestiverkko, joka tarjoaa mm. laajat SAKU-alueet ja vinon pinon muita viestialueita. Sakunetistä kertoo tarkemmin Esa Heikkinen toisaalla lehdessä. Amphibia on Sakun HTML-lukijalle valittu nimi. Se on latinaa ja tarkoittaa sammakkoeläimiä. Alustavan aikataulun mukaan SAKU siirtyy HTML-hypertekstin aikakauteen uuden lukijansa myötä joskus syksyllä. Siihen saakka lehteä julkaistaan entiseen malliin kolmessa formaatissa: disketti-Sakuna, AmigaGuide-Sakuna ja HTML-Sakuna.

Tiesittekös muuten, että Englannissa Manchesterissä toimii Amiga-lisälaitteita myyvä yritys nimeltään Siren Software. Ei, minulla ei ole mitään tekemistä sen kanssa. :-)

Leppoisia lukuhetkiä ja toukokuussa tapaamme taas!

Janne Siren

Kuinka saada kuvat näkymään?

Kuvien katselemiseksi tulee SAKUVIEW-aliakseen olla asetettuna sopiva IFF-kuvaformaattia tukeva katseluohjelma, joka nappaa kuvan tiedostopolun komentoriviltä. Esimerkiksi Display, MultiView tai Viewtek sopivat mainiosti tähän tarkoitukseen.

```
1> Alias SAKUVIEW C:Viewtek
```

1.3 Toimituksen puheenvuoroja

Toimitus

Päätoimittaja: Janne Siren Yhdistyksen puheenjohtaja: Tomi Jaskari

Materiaalin muokkaus: Janne Siren

Pääasiallinen oikoluku: Anu Seilonen

Kansi: Seppo Sinkkonen

Tavutukseen ja muotoiluun käytettiin Riku Puustisen Tekstinmuotoilijaa.

Toimitus eivätkä kirjoittajat vastaa julkaistuista artikkeleista.
Luet ja käytät oheista materiaalia täysin omalla vastuullasi.

Minäkö avustajaksi?

SAKU kaipaa apuasi, sillä lehteä on vaikea koota ilman artikkeleita. Jos sinulla on tuntemusta jostain asiasta, jonka luulisit kiinnostavan muita, kirjoita siitä. Voitte myös lähetellä toimitukseen Amigaa koskevia kysymyksiä, joihin asiantuntijamme pyrkivät sitten vastailemaan. Julkaistusta materiaalista ei makseta palkkiota.

Lähettämäsi teksti oikoluetaan ja sitä mahdollisesti myös muokataan. Jos mukaan on liitetty copyright kieltäen muokkauksen, artikkelia ei voida julkaista. Emme pysty kehittämään lehteä ilman mahdollisuutta vaikuttaa sen ulkoasuun. Tekstien tulee olla toimituksessa muotoilemattomina (ilman tavutusta) viimeistään viikkoa ennen julkaisua.

Jos et halua omaa nimeäsi julkaistavan toimitukseen lähettämäsi materiaalin ohella, muista mainita siitä erikseen.

Materiaalin saat varmimmin perille ottamalla yhteyttä päätoimittajaan:

Janne Siren
Oravamäentie 2 F 17
02700 Kauniainen

BBS: (90) 505 4201
Fidonet: 2:220/550.0
Internet: jts@krk.fi

1.4 Suomen Amiga-käyttäjien unioni

« Suomen Amiga-käyttäjien unioni »

Suomen Amiga-käyttäjät Ry

Yhdistyksen säännöt

Vyöhykejako

1.5 Suomen Amiga-käyttäjät Ry

Suomen Amiga-käyttäjien (unionin) yhdistyksen perustamiskirja allekirjoitettiin ja toimitettiin oikeusministeriöön syyskuussa 1994. Rekisteröintiä odoteltaessa yhdistys aloittaa toimintansa mm. keräämällä jäsenmaksuja. Yhdistyksen tarkempia toimintalinjoja hahmotteleva ensimmäinen vuosikokous pyritään järjestämään mahdollisimman pian. Jäsenmaksun (100 markkaa vuodeksi 1995) voi maksaa yhdistyksen tilille (muista liittää mukaan nimesi).

Pankkitili: KOP Vantaa-Koivukylä 150630-100355

1.6 Yhdistyksen säännöt

1 § NIMI JA KOTIPAIKKA

Yhdistyksen nimi on Suomen Amiga-käyttäjät ja kotipaikka on Vantaa.

2 § TARKOITUS JA TOIMINTA

Yhdistyksen tarkoituksena on edistää Amiga-tietouden levittämistä ja helpottaa jäsentensä laitteisto- ja ohjelmistohankintoja.

Tarkoituksen toteuttamiseksi yhdistys julkaisee lehteä, jonka ilmestymistiheydestä päättää vuosikokous.

Toimintansa tukemiseksi yhdistys julkaisee mainoksia, kerää jäsenmaksuja, järjestää yleisötilaisuuksia ja ottaa vastaan lahjoituksia.

3 § JÄSENET

Yhdistyksen jäseneksi voi liittyä kuka tahansa tietokoneharrastaja.

Yhdistyksen jäsenet hyväksyy hallitus.

Jäsenellä on oikeus erota yhdistyksestä ilmoittamalla siitä kirjallisesti hallitukselle tai sen puheenjohtajalle taikka ilmoittamalla eroamisesta yhdistyksen kokouksessa.

Jäseniltä perittävän liittymis- ja vuotuisen jäsenmaksun suuruudesta päättää vuosikokous.

Jäsenten yhteystietoja voidaan käyttää suoramarkkinointitarkoituksiin henkilörekisterilain mukaisesti.

4 § HALLITUS

Yhdistyksen asioita hoitaa hallitus, johon kuuluu vuosikokouksessa valitut puheenjohtaja ja 4 muuta varsinaista sekä 5 varajäsentä.

Hallituksen jäsenten toimikausi on vuosikokousten välinen aika.

Hallitus valitsee keskuudestaan varapuheenjohtajan sekä ottaa sihteerin, rahastonhoitajan ja muut tarvittavat toimihenkilöt.

Hallitus kokoontuu puheenjohtajan tai hänen estyneenä ollessaan varapuheenjohtajan kutsusta, kun he katsovat siihen olevan aihetta tai kun vähintään 2 hallituksen jäsentä sitä vaatii.

Hallitus on päätösvaltainen, kun vähintään kolme sen varsinaista jäsentä, puheenjohtaja tai varapuheenjohtaja mukaanluettuna on läsnä. Asiat ratkaistaan yksinkertaisella äänten enemmistöllä. Äänten mennessä tasan ratkaisee puheenjohtajan mielipide, vaaleissa kuitenkin arpa.

5 § YHDISTYKSEN NIMEN KIRJOITTAMINEN

Yhdistyksen nimen kirjoittavat puheenjohtaja, varapuheenjohtaja, sihteeri tai henkilö, jolla on siihen hallituksen erikseen antama henkilökohtainen oikeus.

6 § TILIT

Yhdistyksen tilikausi on 1.7 - 31.6.

Tilinpäätös tarvittavine asiakirjoineen ja hallituksen vuosikertomus on annettava tilintarkastajille viimeistään kaksi viikkoa ennen vuosikokousta. Tilintarkastajien tulee antaa kirjallinen lausuntonsa hallitukselle viimeistään viikkoa ennen vuosikokousta.

7 § YHDISTYKSEN KOKOUSTEN KOOLLEKUTSUMINEN

Yhdistyksen kokoukset kutsuu koolle hallitus. Kokouskutsu on toimitettava viimeistään seitsemän (7) päivää ennen kokousta lähettämällä kutsu sähköpostina tai kirjeenä kullekin jäsenelle. Jäsen saa itse valita kumpaa tapaa hän haluaa käytettäväksi.

Jos mahdollista koollekutsu julkaistaan myös soveliaalla Fidonetin viestialueella kuten sf.amiga.saku sekä Internetissä - esimerkiksi uutisryhmässä sfnet.atk.amiga.

8 § YHDISTYKSEN KOKOUKSET

Yhdistyksen vuosikokous pidetään vuosittain hallituksen määräämänä päivänä heinä-syyskuun aikana.

Ylimääräinen kokous pidetään, kun yhdistyksen kokous niin päättää tai kun hallitus katsoo siihen olevan aihetta tai kun vähintään yksi kymmenesosa (1/10) yhdistyksen äänioikeutetuista jäsenistä sitä hallitukselta erityisesti ilmoitettua asiaa varten kirjallisesti vaatii.

Yhdistyksen kokouksessa on jokaisella jäsenellä äänioikeus ja jo-

kaisella äänioikeutetulla yksi (1) ääni. Myös alle 15-vuotiailla jäsenillä on äänioikeus.

Yhdistyksen päätökseksi tulee, ellei säännöissä ole toisin määrätty se mielipide, jota on kannattanut yli puolet annetuista äänistä. Äänten mennessä tasan ratkaistaan vaalit arvalla. Muutoin päätökseksi tulee kokouksen puheenjohtajan kannattama mielipide.

9 § VUOSIKOKOUS

Yhdistyksen vuosikokouksessa käsitellään seuraavat asiat:

1. kokouksen avaus;

2. valitaan kokouksen puheenjohtaja, sihteeri, kaksi pöytäkirjantarkistajaa ja tarvittaessa kaksi ääntenlaskijaa;

3. todetaan kokouksen laillisuus ja päätösvaltaisuus;

4. hyväksytään kokouksen työjärjestys;

5. esitetään tilinpäätös, vuosikertomus ja tilintarkastajien lausunto;

6. päätetään tilinpäätöksen vahvistamisesta ja vastuuvapauden myöntämisestä;

7. vahvistetaan toimintasuunnitelma, tulo- ja menoarvio sekä liittymis- ja jäsenmaksun suuruus;

8. valitaan hallituksen puheenjohtaja ja muut jäsenet;

9. valitaan tilintarkastaja ja hänelle yksi (1) varamies, tilintarkastajan toimikausi on 1.10 - 30.9;

10. käsitellään muut kokouskutsussa mainitut asiat.

Mikäli yhdistyksen jäsen haluaa saada jonkin asian yhdistyksen vuosikokouksen käsiteltäväksi, on hänen siitä kirjallisesti ilmoitettava niin hyvissä ajoin, että asia voidaan sisällyttää kokouskutsuun.

10 § SÄÄNTÖJEN MUUTTAMINEN JA YHDISTYKSEN PURKAMINEN

Päätös sääntöjen muuttamisesta ja yhdistyksen purkamisesta on tehtävä yhdistyksen kokouksessa vähintään kolmen neljäsosan (3/4) enemmistöllä äänestyksessä annetuista äänistä. Kokouskutsussa on mainittava sääntöjen muuttamisesta tai yhdistyksen purkamisesta.

Yhdistyksen purkautuessa käytetään yhdistyksen varat hyväntekeväisyyteen purkamisesta päättävän kokouksen määräämällä tavalla. Yhdistyksen tullessa lakkautetuksi menetellään samoin.

1.7 Suomen Amiga-käyttäjien unioni: Vyöhykejako

« Vyöhykejako »

Vyöhykkeistä
Kartta

Etelä

Itä

Pohjoinen

Länsi

1.8 Suomen Amiga-käyttäjien unioni: Vyöhykejako

Suomen Amiga-käyttäjien unioni on jaettu neljään vyöhykkeeseen. Kullakin vyöhykkeellä on oma koordinaattorinsa, useita vastuuhenkilöitä sekä yksi tai useampi tukipurkki palvelemassa alueen Amiga-käyttäjiä. Kartasta näette maantieteelliset rajat vyöhykkeille. Lähettäkää SAKU-tilaukset (tilauksiin aina mukaan tiedot siitä mitä haluat, sopiva määrä levykkeitä ja palautuskuori riittävällä postimerkillä varustettuna) ja muu posti koordinaattoreille, ellei lehdessä toisin mainita. Ellet toisin erikseen pyydä, koordinaattoreille tai toimitukseen lähettämäsi kirje voidaan julkaista lehdessä.

1.9 Suomen Amiga-käyttäjien unioni: Vyöhykejako

« Etelä »

Ahvenanmaan maakunta, Uudenmaan ja Kymen läänit

Tukipurkit

Vastuuhenkilöt

1.10 Etelä: Tukipurkit

Epsilon Indi BBS (170:10/103.0@giganet)
Janne Siren
(90) 505 4201
V.32bis, 24h

The Bermuda Triangle (2:221/11.0@fidonet)
Ville Valpasvuo
(914) 434 695

Giga-Box (2:220/222@fidonet, 22:468/620@globalnet, 170:10/100@giganet)

```
Kimmo Mustonen
(90) 875 2828
HST DS V.32bis, 24h

Amiga Night System (2:220/550.0@fidonet)
Janne Saarme
(90) 675 840
V.32bis, 24h

Star Fleet
Sami Klemola
(951) 363 5254
Auki: 16-24

Hangar (2:221/18@fidonet, 40:765/765@lukkiverkko, 112:911/511@cabinet)
Juha Niemi
(951) 375 8236
V.FC, 24h
(951) 311 7053
V.32bis, 24h
```

1.11 Etelä: Vastuuhenkilöt

```
Janne Siren

Tomi Jaskari
```

1.12 Yhteystiedot: Janne Siren

```
Janne Siren
Oravamäentie 2 F 17
02700 Kauniainen

BBS: (90) 505 4201
Fidonet: 2:220/550.0
Internet: jts@krk.fi

SAKU-lehden päätoimittaja
```

1.13 Yhteystiedot: Tomi Jaskari

```
Tomi Jaskari
Rasinkatu 7 C 80
01360 Vantaa

Puhelin: (90) 874 2445 (arkiaamuisin 7-10)
Internet: tomi.jaskari@helsinki.fi

Koordinaattori, yhdistyksen puheenjohtaja
```

1.14 Suomen Amiga-käyttäjien unioni: Vyöhykejako

« Itä »

Keski-Suomen, Kuopion ja Mikkelin läänit

Tukipurkit

Vastuuhenkilöt

1.15 Itä: Tukipurkit

AmigaZone BBS
Esa Heikkinen
(958) 422 757
Auki: 21-04

MegaByte BBS (2:225/20.0@fidonet)
Teppo Peurakumpu
(941) 612 950
Auki: 20-07 (ma-to) 24h (pe-su)

1.16 Itä: Vastuuhenkilöt

Esa Heikkinen

1.17 Yhteystiedot: Esa Heikkinen

Esa Heikkinen
Vilhulantie 6 C 24
76850 Naarajärvi

Internet: oh4kju@kauhajoki.edu.fi

Koordinaattori

1.18 Suomen Amiga-käyttäjien unioni: Vyöhykejako

« Pohjoinen »

Lapin, Oulun ja Pohjois-Karjalan läänit

Tukipurkit

Vastuuhenkilöt

1.19 Pohjoinen: Tukipurkit

```
The Spit (2:228/406.2@fidonet)
Lauri Aalto
(981) 330 434
Auki: 22-06
```

1.20 Pohjoinen: Vastuuhenkilöt

```
Mika Välitalo

Janne Kiiskilä
```

1.21 Yhteystiedot: Mika Välitalo

```
Mika Välitalo
Valtatie 5
99100 Kittilä

Koordinaattori
```

1.22 Yhteystiedot: Janne Kiiskilä

```
Janne Kiiskilä
Kaitoväylä 16 as. 1
90570 Oulu
```

1.23 Suomen Amiga-käyttäjien unioni: Vyöhykejako

```
« Länsi »

Hämeen, Vaasan sekä Turun ja Porin läänit


Tukipurkit

Vastuuhenkilöt
```

1.24 Länsi: Tukipurkit

```
Top 100 Memo (2:222/180.0@fidonet)
Tommi Kangas
(938) 822 4572
V.FC, 24h
(938) 822 4527
V.FC, 24h
(938) 822 4557
V.32bis, 24h

Stafia BBS
(922) 56974
V.32bis, 24h
```

1.25 Länsi: Vastuuhenkilöt

```
                Antti Vähä-Sipilä
```

1.26 Yhteystiedot: Antti Vähä-Sipilä

```
Antti Vähä-Sipilä
Vaajakatu 5 F 125
33720 Tampere

Fidonet: 2:221/360.42
Internet: antti.vaha-sipila@cc.tut.fi
URL: http://proffa.cc.tut.fi/~v150105/

Koordinaattori
```

1.27 Aikaisempien AmigaGuide-Sakujen artikkelit

```
               Aikaisempien AmigaGuide-Sakujen artikkelit
               ------------------------------------------

1/95 #10 - 1. tammikuuta, 1995

Ajankohtaiset

Uutiset
Amiga Report CEI Conference on Internet Relay Chat
World of Commodore '94
World of Amiga '94

Internet

Internet todelliselle aloittelijalle
Internet pähkinänkuoressa
Usenetin uutisryhmät
```

50 markalla Internettiin Kuopiosta
Amigazen tuotehinnasto - tammi-helmikuu 1995

7/94 #9 - 1. marraskuuta, 1994

Ajankohtaiset

Uutiset
Keskustelua Amigan tulevaisuudesta (uusinta)
Amiga Report CEI Conference
Assembly ’94 kilpailujen tulokset

Tietokoneet

Erään kovalevyn tarina
Tietokone auttaa, vai auttaako?
Tietokoneiden ja käyttöjärjestelmien tulevaisuus
Piratismin vaikutukset Amigaan

Tietoliikenne

TechnoBBS, virittelevän sysopin purkkisofta

Testit

DirWork v2.1
Grapevine v1.33
DirectoryOpus v4.0 -tiedostoapuohjelma
PageStream v3.0a -julkaisuohjelma
EGS Spectrum 28/24 -näyttökortti

Ohjelmointi

Resepti ympäristöystävälliselle demolle
HTML ja WWW - hypertekstin suuri mahdollisuus
C-ohjelmointikurssi - Osa 1

Pelit

Air Warrior
Detroit
Ennakkokatsauksessa Pinball Illusions

Kilpailut

Kultarahat

Sekalaiset

Maukasta ja maittavaa, Ison-Wäiskin laittamaa

Ilmoitukset

Obvious Implementations Corporation
Amigazen tuotehinnasto - marraskuu 1994
Use-Computer Ky

6/94 #8 - 1. syyskuuta, 1994

Ajankohtaiset

Uutiset
Messuraportti: Assembly '94
Messuraportti: RoPeCon '94
Ensimmäinen SAKU-kokoontuminen Assemblyillä

Sovellukset

OctaMED v5.0 maallikon hyppysissä
Keinotodellisuus

Ohjelmointi

Joonas Jaskarin ensimmäinen C-ohjelma
Ohjelmoinnin henkilökohtaiset ennätykset: Tomi Jaskari

Pelit

Pikakatsauksessa Sierra BBS
Kruunujen Saari -postipeli
Halpapelejä Anttilasta

Sekalaiset

Huumoria: Programming languages are like women
Elektroniikkakurssi - Osa 4

Ilmoitukset

Use-Computer Ky
Amigazen tuotehinnasto - elokuu 1994

5/94 #7 - 18. heinäkuuta, 1994

Ajankohtaiset

Commodoren neuvottelut jatkuvat
Uutiset

Testit

Toccata-äänikortti
Telmex Mouse
Well AT-2814 V.FAST -faksimodeemi

Sovellukset

FrozenFish CD ja GoldFish CD

Pelit

AmigaGuide-SAKU #6 jätettiin väliin yhtenäistämisoperaation vuoksi.

4/94 #5 - 20. kesäkuuta, 1994

3/94 #4 - 30. toukokuuta, 1994

Uutiset

Commodore ei ole konkurssissa!
Uutiset

Palaute

Korjaus Final Writer III -artikkeliin

Messuraportit

CeBIT'94, Hannover

Tietokoneet

MOD-soittimet testauksessa
PD-kääntäjät
Amigan ohjelmoinnin alkeita
Amiga 3000 vaihtoehtona
Haastattelussa OctaMED:in tekijä, Teijo Kinnunen
Amiga ja Euroopan Unioni
GVP Spectrum ja EMPLANT pikatestissä
Viewtek v2.1
Scala 500
Amiga Voice Mail v1.19
Amiga 500 käyttäjän video
Hypermedia

Pelit

Microcosm (Amiga CD32)
Yo Joe
Castles II - Siege & Conguest! (Amiga CD32)
Combat Air Patrol

Tekniikka

Elektroniikkakurssi - Osa 2

2/94 #3 - 1. maaliskuuta, 1994

Uutuudet

Atari Jaguar, 64-bittinen konsoli

Tietokoneet

PowerPC ja Motorola MC68060
Voiko Kickstart 1.3:n kanssa enää elää?
Haastattelussa Real 3D:n tekijä, Vesa Meskanen
Great Valley Products A530 -turbo
Final Writer III
CanDo v2.51
Minimorph v1.0
Amigan käyttäjät ja kuuluisat käyttötarkoitukset

Uutisia Amiga-maailmasta, koonnut Pasi Kovanen
Mielipide - kömmähdys Mikrobitissä?

Grafiikka

Vista aloittelijan silmin
Piccolo, 24-bittinen näyttökortti
HamLab Plus testipenkissä

Pelit

Cannon Fodder
The Settlers
Hired Guns

Tekniikka

Elektroniikkakurssi - Osa 1

Huumori

Elämää takkatulen ääressä - Tuhertelijan tarina

1/94 #2 - 18. tammikuuta, 1994

Tietokoneet

Amigan tulevaisuus, Lew Eggebrecht haastattelussa
SCSI-levyä asentamassa
SupraTurbo 28
MicroBotics MBX 1200z
Great Valley Products A1230 JAWS
Computer Systems Associates 12-GAUGE

Tietoliikenne

Aloittelevan modemistin ohjeisto
DayDream BBS
TechnoBBS kehittyy

Pelit

Dune II, Battle for Arrakis

Tekniikka

Videonauhurin puhdistus

Huumori

Pekka-sedän näkötesti

1/93 #1 - 14. marraskuuta, 1993

Messuraportit

KT-Data 1993, Helsingin Messukeskuksessa
World of Commodore, 1993

Tietokoneet

Tietokoneiden huominen
Amigalla aivoaaltoja stimuloimassa
Amigan grafiikan tulevaisuudennäkymät
Amiga 4000/030, mitä uutta?
Tietotekniikan sanastoa
AHD, PC:n kovalevy Amigaan
IDE - SCSI historiikki

Pelit ja urheilu

Megazone - The Ultimate Laser Adventure
Budo ja tekniikka
RC-autoilu, mitäs kummaa?

Tietoliikenne

EXCELSIOR!, Amigan purkkisoftien kuningas?
Uusi kotimainen purkkiohjelma, TechnoBBS

Lehdet

Hakkeri-lehti arvostelussa
Tekniikan Maailma tunaroi

Sekalaista

Kuinka luot (lähes) täydellisen julkaisun
Esa Heikkinen: Videokirjasto
Pro Amiga Oy

1.28 Jäsenrekisteri

Jäsenrekisteri

Olen parhaillaan tekemässä yhdistyksen jäsenrekisteriä SuperBase4-ohjelmalla Tomi Jaskarin avustuksella. Jäseniähän ei vielä järin paljon ole, mutta kukapa tietää, vaikka tulevaisuudessa niitä olisikin pilvin pimein - toivottavasti ainakin kaksi-kolme kertaa enemmän kuin nyt. Jokaista jäsentä kohden on tietue, joka muodostuu useasta kentästä. Eri tiedoille on oma kenttä: esimerkiksi sukunimi, osoite, puhelin, sähköposti jne. SuperBasella voi tulostaa esimerkiksi jäsenkirjeisiin tarrat ja erilaisia raportteja ym. yhdistyksen sisäiseen käyttöön helposti, eli jäsenrekisteristä olisi hyötyä monessa rutiiniasiassa. Tällä hetkellä ei kaikista maksaneista jäsenistä ole tarpeellisia tietoja, joten olisi hyödyllistä, että maksaneet jäsenet ilmoittaisivat ainakin nimensä, osoitteensa, puhelinnumeronsa, mahdollisen sähköpostitunnuksensa ja koska jäsenmaksu on suoritettu. Tiedot voi lähettää minulle tai Tomi Jaskarille kirjeellä, postikortilla tai privaattimessulla Epsilon Indiin, Amiga Zoneen tai johonkin tukipurkkiin (mieluus-

ti Sakunetissä olevaan). Tiedot ovat luottamuksellisia, eivätkä niitä ulkopuoliset näe! Ne ovat vain ja ainoastaan yhdistyksen käytössä! Uudet jäsenet: kun maksatte jäsenmaksua pankissa tiskillä tai automaatilla, niin laittakaa ko. tiedot tiedonantona laskuun - onnistuu myös automaatilla! Kiitos!

Hyvää kevättä jäsenille ja kaikille Sakun lukijoille, ja käyttäkää ahkerasti sitä maailman parasta tietokonetta, Amigaa!

Veli-Matti Vuorensyrjä

1.29 Vuosikokous

Vuosikokous

Tässä on joutunut hyppyyttämään Suomen Amiga-käyttäjien yhdistyksen ensimmäisen vuosikokouksen kokouspäivää jo jonkun aikaa. Ei vain tule tarpeeksi valmista ei sitten vaikka pistäisi päänsä pantiksi.

Suomen Amiga-käyttäjien seuraava online-keskustelu pidetään Internetin IRC-keskustelufoorumissa 11.3.1995 kello 18 alkaen. Oletettavasti kokouksen kanava, #saku, on käytössä ainakin kello 21:een saakka.

Alunperin oli tarkoitus järjestää virallinen ylimääräinen vuosikokous tiedekeskus Heurekan ryhmätyötiloissa tuona päivänä, mutta valitettavasti partiolaiset olivat ehtineet ensin tilojen käyttäjiksi. Seuraavanakin viikonloppuna kyseiset tilat ovat varattuina sähköisen viestinnän tapahtumalle. Harkitsin myös kokouspaikaksi Kumpulassa sijaitsevaa aivan tuliterää Helsingin Yliopiston kemianlaitosta, mutta sijainti on hieman hankala enkä usko että tilojen varaaminen viikonloppuna onnistuu tarpeeksi helposti.

Yhdistyksen jäsenille postitetaan kokouskutsu, kunhan tapahtuman ajankohta varmistuu. Mikäli unohdit antaa yhteystietosi liittyessäsi tai haluaisit vielä harkita jäsenyyttä mutta silti osallistua kokoukseen, ota yhteyttä koordinaattoreihin. Jos ei-jäsenet haluavat kokouskutsun postitse, heidän tulee liittää mukaan palautuskuori ja postimerkki. Koordinaattoreiden yhteystiedot löydät toisaalta lehdestä.

Tomi Jaskari ja Janne Siren

1.30 Uutiset

Uutiset

Janne Siren

Commodore, muistatteko?

1.31 Uutiset: Commodore, muistatteko?

Commodore, muistatteko? Ehkä joskus hamassa menneisyydessä olet kuullut tämän nimen. Niin, sehän oli myynnissä vajaa vuosi sitten? Kyllä, se oli ja on edelleen. Asioihin on tosin tullut taas hieman edistystä. Kaiken sähellyksen keskellä USA:n tuomioistuin, Bahaman tuomioistuin ja Commodoren velkojat pääsivät vihdoinkin sopimukseen tavasta jolla myynti vietäisiin loppuun. Tarkoitus oli siirtyä huutokauppavaiheeseen niin pian kuin mahdollista ja tutkia Commodoren johtoportaan toimet viimeisen 12 kuukauden ajalta.

Vastauksena Commodoren selvitystilaan ajaneen johtoportaan jäsenet, Irving Gould ja Medhi Ali, tekivät valituksen. Commodore International oli kirjoilla Bahamalla, joten heidän mielestään prosessi tulisi hoitaa kokonaan Bahamalaisessa tuomioistuimessa ja Bahaman lain mukaan. Valituksella ei liene mitään tekemistä sen tosiasian kanssa, että Bahamalaisen lain mukaan selvitystilaan ajautuneen yrityksen menneisyyttä voidaan kelata taaksepäin vain kolmen kuukauden ajalta? Ei taida olla miehillä aivan puhtaita jauhoja pussissaan. Bahamalainen tuomioistuin vielä viivytti valituksen käsittelyä erään tuomarin perheessä sattuneen kuolemantapauksen johdosta, joten tässä sitä vain pyörittelemme peukaloitamme.

Kun lakimiehet hiovat pykäliä oikeussalissa, kiinnostus Commodoresta kasvaa. Ennestään mukana ostokilpailussahan olivat Commodoren UK:n johto ja amerikkalainen CEI, mutta nyt lisäksi saksalainen Escom (Escom oli mukana jo aiemmin, mutta tiputtautui pois) ja joku tuntemattomana pysyttelevä amerikkalainen yritys.

Asian tiimoilta mainittakoon, että kun Englannissa tietokonekauppialle esitettyyn kysymys, haluaisivatko he Amigan takaisin - vastasi 82% myöntävästi. Koneelle on siis markkinoita, kunhan lakimiehet pääsisivät asioista yhteisymmärrykseen ja siirtyisivät rahastamaan jonkun muun yrityksen konkurssipesää...

1.32 Uutiset: Kotimainen levykelehti pelaaville amigisteille

Samuli Lehtosen päätoimittama AMIGAmes on reilun puolen vuoden ikäinen diskettilehti pääasiallisena aihealueenaan Amigan pelit.

Lehteä levitetään AmigaGuide-hypertekstinä, joten sen täysipainoiseen lukemiseen tarvitaan joko AmigaOS 3.0:n/3.1:n MultiView tai erikseen hankittu AmigaGuide-ohjelma. Ulkoasultaan lehti muistuttaa aika paljon Amiga Reportia, AmigaGuide-Sakua ja jopa Amiga-purkkilistaa, yksityiskohtia kun on matkittu ahkerasti muista guide-julkaisuista. Ulkoasu on kuitenkin hieman suttuinen: lukuisat kielioppi- ja kirjoitusvirheet häiritsevät, ja suomenkielisen tekstin seassa lymyävät englanninkieliset sanat ja lauseet ihmetyttävät. Kuvia lehdessä ei ole.

Kun oppii jättämään pahat kielioppivirheet omaan arvoonsa ja paneutumaan lehden sisältöön, ei voi olla muistelematta Sakun alkuaikoja. Tyhjästähän mekin aloitimme. AMIGAmesiin on yritetty avata useampia erilaisia palstoja, joista osa on edelleen tyhjillään. Nykyinen anti koostuu mm. peliarvosteluista, uutisista ja AMIGAmesin toimituksen henkilökohtaisista pelikokemuksista. Pelit ovat erittäin tuoreita, joten joko AMIGAmes testaa esittelyversioita tai piraattikopioita. Toivottavasti eivät jälkimmäisiä.

SAKU toivottaa uudelle tulokkaalle onnea matkaan! AMIGAmes-lehdet löydät modeemilla esimerkiksi MegaByte BBS:stä numerosta (941) 612 950 tai Mental Hospitalista numerosta (90) 3742 269.

1.33 Uutiset: Inovatronics tekohengityksessä

Inovatronics, amigisteille tuttu sellaisista laadukkaista ohjelmista kuin CanDo, DirectoryOpus ja GigaMem, on taloudellisesti surkeassa jamassa. Henkilöstön määrä on vähennetty kahteen ja myynti on ollut heikkoa. Nyt olisi aika ostaa ko. firman ohjelmia ja osoittaa tukea, ettei vain taas yksi hyvä Amigan ohjelmistovalmistaja poistuisi keskuudestamme!

1.34 Uutiset: Amiga-klooni MacroSystemiltä

Joskus vuosia sitten maailmalla kiersi huhu Commodoresta riippumattoman tahon valmistamasta kannettavasta Amigasta. Se jopa esiteltiin eräillä messuilla, mutta projekti tyssäsi Commodoren uhattua viedä kloonaajat lakituppaan. Tavallisista pöytäklooneistakin olisi ollut Amigalle eittämättä hyötyä, mutta Commodore luonnollisesti ajatteli omaa etuaan. Onhan Applekin vasta nyt sallimassa koneidensa kloonauksen.

Unelma klooni-Amigasta on kuitenkin tulossa toteen aikana, jona uutinen on todella tervetullut. Commodoren myynti venyy lakipykäliä hierottaessa, eikä edes vanhojen mallien koneita ole kauppoihin vähään aikaan uusina tullut. Saksalainen MacroSystem tietää, että Amigalle on kysyntää ja iskee pöytään oman valttikorttinsa. Se on nimeltään DraCo.

DraCossa ei ole Amigan erikoispiirejä, joten se ei riko tekijänoikeuksia. Samalla voidaan kuitenkin unohtaa useimmat pelit ja muut kovoa suoraan iskemällä tehdyt ohjelmat. Kaikki grafiikkakortteja ja käyttistä siististi hyödyntävät ohjelmat kuitenkin toimivat, joten klooniin sopivia ohjelmia on

runsain mitoin. Käyttiksenä rullaa tuttu AmigaOS tai valinnaisesti joku Amigan UNIX-järjestelmistä. Käyttöjärjestelmä toimitetaan koneen mukana CD-ROM-levyllä.

Prosessori on Motorolan upouusi tehokas 68060 ja näytöstä huolehtii Retina-näyttökortti. Laajennusvaraakin DraCon tornikotelossa on: kolme omaa DraCo-direct-liitäntää (johon on tulossa mm. RISC-prosessorimoduuli) ja viisi Zorro II -korttipaikkaa. Massamuistilaitteista huolehtii SCSI-II-ohjain, ja kirjoittimen voi kytkeä sisäiseen rinnakkaisporttiin. Koneen mukana toimitetaan myös kolminopeus SCSI-romppuasema. RAM-muistia DraCoon mahtuu emolevylle 4-128 Mt.

DraCon toimitukset aloitetaan keväällä 1995, ja hinnaksi on ilmoitettu 5998 DM eli noin 18595 mk.

MacroSystem Computer Trading GmbH
Friedrich-Ebert-Str. 85
58454 Witten
Germany

Puhelin: 990-49-23-02-8-03-91
Faksi: 990-49-23-02-8-08-84

1.35 Uutiset: Commodore UK avaa oman kuuman linjan

Amiga Worldin hot-line numerossa 990-1-603-924-2195 on toiminut luotettavana ja ajan tasalla pysyvänä tiedonlähteenä Commodoren myynnin edetessä. Nyt meille Eurooppalaisille on hot-line lähempänäkin: Commodore UK:n nauhoitteen kuulee soittamalla numeroon 990-44-1628-779-655.

1.36 PCMCIA SCSI-II-ohjain

HiSoft on tuonut markkinoille Amiga 600- ja 1200-mallien PCMCIA-liitäntään sopivan SCSI-II-ohjaimen. Squirrel-nimiseen ohjaimeen saa ketjutettua seitsemän SCSCI-laitetta (kovalevyjä, CD-ROM-asemia, nauha-asemia, jne). Squirrelin hinnaksi tulee £69.95, ja mukana on tarpeellisten ajurien lisäksi mm. ohjelmat musiikki-CD-levyjen soittamiseen ja samplaamiseen.

HiSoft
The Old School
Greenfield
Bedford
MK45 5DE
United Kingdom

Puhelin: 990-44-1525-718-181
Faksi: 990-44-1525-713-716

1.37 PageStream 3.0c korjaa aiempien versioidensa mokia

SAKU kympissä arvosteltu taitto-ohjelma, PageStream 3.0, vakuutti, mutta sen bugisuus harmitti. Soft Logic tietää tämän ja on tasaiseen tahtiin julkaissut ohjelmalleen päivityksiä. PageStream 3.0c on aiempaa vakaampi, ja tulostukseen liittyneet ongelmat on korjattu. 3.0c on myös hieman vanhempaa versiota nopeampi, ja useita keskeneräisiä toimintoja on viimeistelty. Päivitys on ilmainen ja löytyy purkeista.

1.38 Kokoelmaromppuja Englannista

Yhä useammat brittifirmat ovat keksineet kasata myymistään PD-ohjelmista CD-ROM-levyjä. Englantilaisten Amiga-lehtien sivut suorastaan pursuavat mainoksia mitä ihmeellisimmistä kokoelmarompuista. Muutamina esimerkkeinä mainittakoon Speccy Sensation (kasa Spectrum-emulaattoreita ja yli 500 Spectrumin peliä), Network Package (kaikki tarpeelliset ohjelmat CD32:en tai CDTV:n liittämiseksi toiseen Amigaan) ja Assasins CD (yli 650 PD-peliä). Levyjen hinnat pyörivät n. 20 punnan tienoilla ja niitä voi tiedustella esimerkiksi seuraavilta yrityksiltä.

Epic Marketing
First Floor Offices
Victoria Centre
138-139 Victoria Rd
Swindon
Wilts. SN1 3BU
United Kingdom

Puhelin: 990-44-793-490-988
Faksi: -

PDSOFT
1 Bryant Avenue
Southend-on-sea
Essex
SS1 2YD
United Kingdom

990-44-1702-466-933
990-44-1702-617-123

1.39 LZX - pakkausformaatti, joka syrjäyttää LHA:n?

LHA on pitkään ollut johtava ja käytännössä ainoa käytetty pakkausformaatti Amigalla. LHA (LH5) sai alkunsa PC:llä vuoden 1990 tienoilla, ja Amiga sai omat purku- ja pakkausohjelmansa vuotta myöhemmin. LHA syrjäytti nopeastI LZH:n (LH1) huomattavasti tehokkaampana pakkausalgoritmina.

LZX on kokonaan uusi pakkausalgoritmi. Sen takana ovat Jonathan Forbes ja Tomi Poutanen. Algoritmi on kehitetty yhteistyössä. Jonathan on samalla tehnyt Amiga-puolen ohjelmistoja ja Tomi vastaavia PC:lle. MS-DOSin rajoitusten vuoksi PC-versio on kuitenkin laahannut jäljessä, ja toistaiseksi LZX on saatavilla vain Amigalle. LZX on siinä määrin tehokas, että tekijät toivovat sen nousevan PC:llä markkinajohtajien ohitse, kunhan PC-versio saadaan julkaistua. Olisihan se jotain, että alunperin Amigalle julkaistu pakkausalgoritmi kerrankin nousisi PC-maailman kärkeen.

LHA pakkaa 384423 tavua tiedostoja 180981:een tavuun, kun LZX pakkaa samat tiedostot 163397:een tavuun. Erot kasvavat suurilla tiedostoilla: 2201744 tavua pakkautui LHA:lla 887678:een tavuun ja LZH:lla 712764:een tavuun! LZX on yleensä lisäksi Stefan Bobergin LhA-ohjelmaa nopeampi.

Nähtäväksi jää, pystyykö LZX murtamaan johtavan pakkausalgoritmin valta-

aseman. Yksi suurimmista esteistä on algoritmin salaisuus, ja sen seurauksena purkuohjelmia ei ainakaan toistaiseksi ilmesty muille koneille, mikä käytännössä estää LZX:n yleistymisen UNIX-järjestelmissä pyörivissä FTP-palvelimissa. Amiga-purkeissa LZX voi hyvinkin kerätä mainetta ja kunniaa, onhan se huomattavasti aiempaa tehokkaampi.

LZX löytyy hyvinvarustetuista purkeista nimellä lzx100.lha (131 kt).

1.40 Mikä ihmeen Sakunet?

Mikä ihmeen Sakunet?

Esa Heikkinen

Mikä ihmeen Sakunet?

Sakunetin viestejä

1.41 Mikä ihmeen Sakunet?

Ennen kuin uppoudumme Sakunetin syntytarinoihin, täytyy selvittää muutamia peruskäsitteitä lukijoillemme, joille sähköpostimaailma bokseineen, verkkoineen ja nodeineen ei ole ennestään tuttua. Varmasti jo aikaisempia Sakun numeroita lukeneet ovat saaneet jonkinlaisen kuvan siitä, millainen sähköposti (boksi) on. Se on systeemi, jonne voi soittaa modeemilla ja mm. kopioida itselleen ohjelmia ja keskustella muiden käyttäjien kanssa viestejä kirjoittamalla. Tämä on jokaisen modeeminomistajan arkipäivää - vai pitäisikö sanoa -iltaa/-yötä, sillä monestihan yhteydet jatkuvat läpi yön, koska iltaisin soittelu on halvinta. Sysop taas on henkilö, joka ylläpitää systeemiä esimerkiksi omalla tietokoneella kotonaan. Ok, mikä sitten on netti? Tämän selvittämiseksi täytyy ensin miettiä hieman käsitettä viesti.

Viesti on käyttäjän kirjoittama "kirje", joka on yleensä osoitettu toiselle sähköpostin käyttäjälle joko yksityisenä, jolloin vain vastaanottaja voi lukea sen, tai julkisena, jolloin myös muut voivat lukea sen ja kommentoida sitä. Julkinen viesti voi myös olla osoitettu kaikille (All). Yleensä julkisilla viesteillä pyritään aloittamaan keskustelu uudesta aiheesta. Kun käyttäjä kirjoittaa viestin, se tallentuu sähköpostissa kiintolevylle odottamaan, kunnes se henkilö, jolle viesti on osoitettu, soittaa systeemiin. Jos taas viesti on julkinen, se näytetään kaikille viestejä selaaville. Näin siis toimii normaali local-tyyppinen viestinvälitys. Huono puoli on se, että viestin vastaanottajan tulee olla käyttäjänä samassa boksissa, jota viestin lähettäjäkin käyttää. Tätä puutetta korjaamaan on kehitetty verkko (net = netti), joka mahdollistaa sen, että samat viestit ovat luettavissa useassa sähköpostissa.

Käytännössä verkko toimii siten, että bokseissa olevien ns. local-alueiden lisäksi on luotu net-alueet, joiden viestit välittyvät muihin bokseihin. Kun boksi on verkossa kiinni ja välittää viestejä käyttäjiltään verkkoon,

sitä kutsutaan verkon nodeksi. Koska verkkoja on olemassa useita, voi sama boksi olla jopa useassa eri netissä nodena. Viestit siirretään siten, että boksit pakkaavat valmiiksi viestipaketteja aina, kun uusia viestejä kirjoitetaan. Kun toinen boksi ottaa yhteyttä hakeakseen viestit, vaihdetaan valmiiksi pakatut tiedostot molempiin suuntiin, minkä jälkeen yhteys katkaistaan, viestipaketit puretaan ja viestit sijoitetaan oikeille alueille. Soittoa kutsutaan pollaukseksi. Yksi pollaus ei yleensä kestä kovin kauaa, koska paketit ovat valmiiksi olemassa. Pollaus on täysin automaattinen kahden koneen välinen yhteys.

Käyttäjän local-alueelle kirjoittamalla julkisella viestillä on enintään niin monta lukijaa kuin boksissa on käyttäjiä. Verkkoalueilla taas viestillä on paljon enemmän lukijoita, koska viesti leviää kaikkiin samassa verkossa oleviin bokseihin, joiden käyttäjät voivat myös lukea viestin ja vastata siihen. Tämäntyyppisiä julkisia keskustelufoorumeja kutsutaan echoiksi, eli viestit "kaikuvat" kaikkialle verkossa. Kun käyttäjä kirjoittaa esimerkiksi Sakunetiin alueelle AZ.Pelit, on viesti pienen ajan kuluttua luettavissa kaikissa Sakunetin bokseissa alueella AZ.Pelit.

Yksityisviestien välitystä verkossa kutsutaan netmailiksi. Tällöin kirjoitettu viesti välitetään ainoastaan vastaanottajalle, ei muuhun verkkoon. Lähettääkseen yksityispostia täytyy lähettäjän tietää vastaanottajan käyttämän boksin nodeosoite, jotta järjestelmä osaa ohjata (= routata) viestin oikeata reittiä perille. Aivan samoin perinteisessä postissa täytyy tietää vastaanottajan osoite. Yksityisviesti menee ainoastaan niiden boksien läpi, joiden kautta sen on mentävä saavuttaakseen vastaanottajan, ja se voi pahimmassa tapauksessa kiertää useankin systeemin läpi. Sen pystyy kuitenkin lukemaan vain siinä systeemissä, johon viesti on osoitettu.

Ristiriitoja verkoissa

Koska verkko on julkinen järjestelmä, jossa kaikki voivat kirjoitella, on odotettavissa, että ennemmin tai myöhemmin syntyy riitoja. Joku esimerkiksi selvittää jotain teknistä asiaa ja näkee asian eteen paljon vaivaa. Sitten tulee joku toinen ja vastaa viestiin, ettei asia olekaan näin. Alkuperäisen viestin kirjoittaja vetoaa käytännön kokeiluihinsa. Tässä vaiheessa keskusteluun liittyy pari muutakin henkilöä omine käsityksineen, ja sitten onkin käynnissä ilmiriita. Yleensä tällaiset riidat kuitenkin saadaan sovittua, eikä niistä synny sen suurempaa ongelmaa. Poikkeuksiakin kuitenkin on. Suomessa harrastajaverkkojen valta-asemassa oleva verkko, Fidonet, on yksi niistä. Viime aikoina siellä syntyneet ristiriidat ovat saaneet aivan uusia ulottuvuuksia. Jotkut sysopit ovat alkaneet laatia mm. mustia listoja henkilöistä, joista eivät pidä. Listalle pääsyyn riittää, että joku ehdottaa nimeä listan ylläpitäjälle ja kertoo oman käsityksensä ko. henkilöstä. Tämän jälkeen listan ylläpitäjä lisää nimen ja ehdotetun kommentin listaan.

Tällä hetkellä erästä suosituimmista listoista on ilmoittautunut ylläpitämään ja levittämään helsinkiläinen Lassi Kurikka. Hän pyytää julkisessa viestissään muita sysopeja lähettämään hänelle listaehdokkaita. Näitä listoja on sitten levitetty ja levitetään bokseissa tiedostona, julkisina viesteinä tai jopa file-echoina verkon yli siten, että listat leviävät automaattisesti mahdollisimman moneen Fidonet-boksiin. Listoissa esitetään siis perättömiä väitteitä ja haukutaan henkilöitä. Kenenkään mielessä ei ole käynyt, että tämänkaltainen toiminta loukkaa yksilön oikeutta ja saa ihmiset arastelemaan kirjoittamisiaan verkoissa. Edes Fidonetin johtavassa

asemassa olevat henkilöt eivät ymmärrä tätä. Tällainen toiminta on huonontanut Fidonetin mainetta ja saanut monet sysopit näkemään punaista. Koska Amiga- ja SAKU-keskustelut on tähän mennessä käyty juuri pahamaineisen Fidonetin kautta, syntyi idea uuden verkon perustamisesta Sakua varten.

Alkoi tapahtua

Tarina alkoi, kun Fidonetin lievimminkin sanoen diktatuurisen johdon kanssa syntyi riitoja koskien em. mustia listoja. Fidonetistä vastuussa olevat henkilöt, nurmijärveläinen Tuomo Soini ja helsinkiläinen Pertti Heikkinen, osoittivat käytännössä, kuinka Fidossa tehdään mielivaltaisia päätöksiä oletusten ja huhupuheiden perusteella ja kuinka välinpitämättömästi sakulaisia ja muita käsitellään Fidonetissä. Ne henkilöt, jotka esim. vastustavat mustien listojen levittämistä Fidossa, joutuvat hyvin nopeasti vainon ja aiheettomien rankaisutoimenpiteiden kohteeksi, koska Soinin ja Heikkisen mielestä listat ovat hyvin hauska pila, ja niitä tulee jatkossakin ylläpitää ja levittää. Jos uskallat sanoa vastaan näiden henkilöiden mustamaalaus- ja vaino-organisaatiolle, sinut potkitaan ulos Fidonetistä, sinua uhkaillaan ja joudut itsekin listalle. Jos koneenasi on vielä Amiga, se vain pahentaa asiaa. Ihme, ettei kukaan ole vienyt asiaa oikeuteen - aika näyttää...

Näin ollen kirjoittelu Fidonetissä saattaa koitua turmioksi kenelle tahansa aktiiviselle kirjoittajalle. Nimesi saattaa ilmestyä ties mille haukuntalistalle sinun mitään asiasta tietämättäsi. Riittää siis, että kuka tahansa ehdottaa nimesi sinne. Sinulta ei kysytä. Lisäksi Pertti Heikkinen on nyttemmin sanonut, että Fidonet on vain sysopeja varten, tavalliset käyttäjät eivät sinne kuulu. Tekstistä voi lukea rivien välistä, etteivät he ole myöskään tervetulleita. Lisäksi Tuomo Soini toteaa viestissään 19.2.95 alueella SF.Fidonet Sakusta ja sakulaisista seuraavasti:

"Minun tehtäväni ei ole puuttua Sakun toimintaan mitenkään, mutta voin jo nyt sanoa olevani aikalailla kurkkuani myöten sakulaisten toimintaan Fidonetissä."

Vaikka Tuomo ei osaa kunnolla tekstissään itseään ilmaista (kuten em. lainauksesta huomaa), viestin muu ulkoasu kertoo kyllä, mitä Tuomo tarkoittaa. Tästä voi jokainen itse päätellä, ovatko sakulaiset ja SAKU-keskustelu tervetulleita Fidonetiin. Tämän lisäksi Tuomo on jarruttanut Fidonetin kehitystä muutenkin, esimerkiksi estämällä 8-bittisten merkkien käytön viesteissä, mikä helpottaisi viestien kulkua ja niiden kirjoitusta monella tavalla.

Sakunet iskee

Näin siis syntyi sijaa Sakunetille. Fidonetin SF.Amiga- ja SF.Amiga.SAKU-alueilla oli muutamaan otteeseen heitetty ohimennen ajatus uudesta verkosta, mutta kukaan ei viitsinyt/jaksanut/uskaltanut tehdä asialle mitään. Niinpä soitin eräänä lauantaiaamuna Tomi Jaskarille ja kerroin AmigaZone-net työnimellä olevasta kokeiluasteella elävästä verkostamme, josta voitaisiin muokata Sakunet, ennen kuin verkko vielä ehtisi laajemmalle. Fidonetin tilanteen tietäen Tomikin oli asiasta varsin kiinnostunut. Onhan totta, että SAKU on jo sen verran laajentunut käsite, että se ansaitsee kokonaan

oman verkkonsa, jossa sillä on oma päätäntävalta ja jota ei hallita mielivaltaisin Soini-päätöksin. Verkon tulisi olla siinä mielessä uusi, ettei sillä olisi jo valmiiksi huonoa mainetta, joten AmigaZone-netistä tehtäisiin virallisesti Sakunet - ja se tehtäisiin HETI, enää ei jahkailtaisi tippaakaan.

Näin sitten tapahtui. Järjestettiin muutamia tiedotteita uudesta verkosta, ja uusia nodeja sekä kiinnostusta uutta verkkoa kohtaan alkoi nousta, mikä ei Fidon kärjistyneen tilanteen huomioon ottaen ole ihme. Uusien nodejen tullessa myös viestimäärä kasvoi ja ylitti kaikki ennakko-odotukset. Tätä kirjoitettaessa Sakunetissä on kiinni jo 11 boksia:

Node	Boss	Boksi	Sysop
65:958/1	(HOST)	Amiga Zone BBS	Esa Heikkinen
65:90/1	958/1	Epsilon Indi	Janne Siren
65:90/2	90/1	Unbroken Dream	Niko Strandman
65:90/3	958/2	Overscan BBS	Ville-Pertti Keinonen
65:90/4	-	Varattu: Giga-Box	Kimmo Mustonen
65:90/5	917/1	SinCity	Samuli Seppänen
65:90/6	90/1	Graveyard	Jani Tertsonen
65:917/1	958/1	Cool Place	Kai Kasurinen
65:917/2	917/1	Anttila	Antti Halla
65:951/1	958/1	Star Fleet TechnoBBS	Sami Klemola
65:958/2	958/1	System Point	Sami Jäntti
65:971/1	958/1	Bahia Island	Tuomo Kalliokoski

Kuten listasta huomaa, rakentuvat nodenumerot varsin erikoisesti. Verkon zone-tunnus on 65. Numero on A-kirjaimen ASCII-arvo, kirjaimen, jolla Amiga alkaa! Sen jälkeen tuleva verkkotunnus on sama kuin boksin suuntanumero, joten nodeosoite kertoo hieman, missä päin Suomea boksi sijaitsee. Jos samalla alueella on useampia bokseja, saavat ne /2.. /3.. /4 nodenumeron. Node-sarake listassa kertoo boksin nodeosoitteen, ts. jos haluat lähettää postia toiselle käyttäjälle toiseen boksiin, tulee sinun tietää ko. boksin nodeosoite. Boss-sarake kertoo, minne ko. boksi pollaa eli mistä se hakee viestinsä. Myös toisentyyppistä nodenumerointijärjestelmää on harkittu, mutta käytännössä pitäydytään vielä toistaiseksi nykyisessä.

Kuva: Verkon rakenne

Kuvassa "HOST" on keskusboksi jota "HUBit" pollaavat, ja "HUBit" ovat alueellisia keskusbokseja, joita nodet pollaavat. Verkon rakenne on siis tähtimäinen, mutta vapaan rakenteen ansiosta muutama node pollaa suoraan keskuspurkkiakin. Nämä nodet muuttunevat myöhemmin HUBeiksi ottaessaan omia nodeja.

Sakunetin policy

Verkoissa on aina jonkinlaiset säännöt, joita kaikki verkon käyttäjät noudattavat. Sakunetin virallisia sääntöjä ei varsinaisesti vielä ole. On vain sovittu, ettei KENELLÄKÄÄN tässä verkossa ole ehdotonta määräysvaltaa kuten Fidossa, vaan asioista pyritään päättämään yhdessä. Vastuu kirjoitettujen viestien laadusta on sysopeilla, jotka ovat vastuussa siitä, että bokseista lähtevät viestit ovat asiallisia ja noudattavat lakia. Tarvittaessa sysopeilla on täysi valta (ja häiriötapauksissa myös velvollisuus) poistaa

asiattomia viestejä ja myös kirjoitusoikeudet ko. käyttäjältä Sakunetiin.

Vrt. Fidonet: häiriköinnit hoitaa moderaattori, sysopeilla ole mitään oikeuksia kajota viesteihin, jotka heidän systeemiensä läpi kulkevat. Moderaattori voi poistaa kirjoitusoikeudet, mutta kirjoitettu viesti (esimerkiksi lainvastainen huumeiden tai varastetun tavaran myynti-ilmoitus!!) leviää kaikkialle, eikä kenelläkään ole oikeutta sitä estää. Moderaattoreiksi Tuomo Soini valitsee henkilöitä, jotka eivät ole edes paikalla lukemassa viestejä (esimerkiksi armeijassa viikot), ja ehdoton edellytys on, ettei ehdokas vastusta mustia listoja... Tilanne on siis päässyt melko pahaksi, joten toivottavasti Sakunet on yksi niistä vaihtoehdoista, jotka voivat tarjota jotakin parempaa.

Uusia nodeja Sakunetiin otetaan sitä mukaa kuin kiinnostuneita sysopeja löytyy. Mitään äänestystä tai valintaperusteita ei ole, vaan mukaan pääsee kuka vain. Sakunetiä saa pollata mistä tahansa nodelta/hubilta, kunhan vain sopii asiasta ko. paikan sysopin kanssa. Suotavaa olisi silti, että pollaukset keskitettäisiin HUBeille tai nodeille, jotka ovat suoraan HOSTilla, jotta viestien kulku olisi mahdollisimman nopeaa.

Pollausten tiheydestä ei ole määräyksiä. Kukin pollatkoon niin usein kuin tarpeelliseksi katsoo, eli ellei viestiliikennettä omasta purkista tai alanodeja ole, voi aivan hyvin pollata pari kertaa viikossakin.

Sakunetissä käytettävä merkistö on 8-bittinen ISO, joten kaikenlaisten merkkigrafiikasta muodostuvien kaavioiden ja muiden erikoismerkkien käyttö viesteissä on sallittua, ja ne myös kopioituvat eteenpäin oikein:

```
 ___          ___       _____          ___   _____
|   |  | /   |          |    |  |  |   |    |
|   |  |<    |---       |    |---|  |---|    |
|___   _|_   | \   |___   |    |   |  |   |    |
```

Viestien pituudella tai määrällä/vrk ei ole mitään rajaa, vaan saa kirjoittaa niin paljon kuin jaksaa, kunhan keskustelu on asiallista ja pysyy oikeilla alueilla. JA ENNEN KAIKKEA: SAKUNET ON KÄYTTÄJIÄ VARTEN, EI AINOASTAAN SYSOPEJA! Ja toiseksi motto: "Tässä verkossa noudatetaan Suomen lakia."

Keskustelualeet

Sakunet on yleiskäyttöinen viestiverkko. Tämä tarkoittaa, ettei keskusteluaiheita ole rajattu ainoastaan Sakuun, vaan alueita on kaikenlaiseen keskusteluun. Näin pyritään pitämään verkko hengissä, vaikka SAKU-keskustelua ei jostakin syystä sattuisi olemaankaan. Pidemmän päälle tämä on parempi, koska pari SAKU-aluetta sisältävä verkko olisi melko hiljainen, eivätkä sitä lukisi muut kuin Sakusta kiinnostuneet henkilöt. Nyt verkkoa lukevat myös muut, ja siinä ohessa he lukevat SAKU-alueitakin!

Tätä kirjoitettaessa käytössä ovat seuraavat keskustelualueet:

Alue		Tarkoitus
AZ.SysOp	(S)	Sysopien keskustelualue
AZ.Echo	(S)	Sysopien keskustelualue koskien verkkoa
AZ.Amiga		Yleinen Amiga-keskustelu

AZ.Amiga/Uutiset		Amiga-uutisia/uutuuksia/uutta
AZ.BBS		Boksikeskustelu
AZ.Elektroniikka		Elektroniikka-aiheinen keskustelu
AZ.Hifi		Hifi/video/satelliitti-TV-laitteistot
AZ.Huuhaa		Nimensä mukainen, aihe on täysin vapaa
AZ.Info	(R)	Netin tiedotusalue, ainoastaan luettavaksi
AZ.Internet		Internet-aiheinen keskustelu
AZ.Ilmoitukset		Käyttäjien oma tiedostuskanava
AZ.Ilmoitukset/BBS		Alue boksimainoksia varten
AZ.Markkinat		Myynti/osto/vaihtoilmoitukset
AZ.Musiikki		Musiikkiin liittyvä keskustelu
AZ.Muut		Keskustelu, jolle ei löydy muutakaan aluetta
AZ.Netti		Sakunetiin liittyvä keskustelu
AZ.Ohjelmointi		Ohjelmointiasiaa
AZ.PC		Jos jollakin on kakkoskoneena PC... :-)
AZ.Pelit		Peliasiaa
AZ.Point	(S)	Pointtikeskustelua
AZ.Radioamatöörit		Radioamatööriasiaa
AZ.Roolipelit		Roolipeliasiaa tai roolipelejä
AZ.SAKU		SAKU - Yleinen alue
AZ.SAKU/Info		SAKU - Tiedotusalue
AZ.SAKU/Ohjelmointi		Sakuun liittyvä ohjelmointi/kehitystyö
AZ.SAKU/Toimitus		Sakun toimitukseen liittyvät asiat
AZ.SAKU/Yhdistys		Suomen Amiga-Käyttäjät ry -keskustelut
AZ.SciFi		Science-fiction
AZ.Software		Jos etsit jotakin ohjelmaa, niin kysele täällä
AZ.Teeseitse		Tee se itse -asiaa virittelijöille
AZ.Tietokoneet		Muut tietokoneet, oheislaitteet yms.
AZ.Tietoliikenne		Tietoliikenne, modeemit yms.
AZ.TV/Elokuvat		TV-sarjat, TV-elokuvat, videot, elokuvat yms.
AZ.Vitsit		Kerro Sakunetin paras vitsi!

(S) aluenimen perässä tarkoittaa, että alue on ainoastaan sysopien käytössä, eivätkä tavalliset käyttäjät lue/kirjoita sinne.

(R) aluenimen perässä tarkoittaa, että tavalliset käyttäjät voivat lukea aluetta, mutta eivät kirjoittaa. (Käyttäjille on oma AZ.Ilmoitukset.)

"AZ" lyhenne on jäänne AmigaZone-netin ajoilta, mutta käytännössä se on pakollinen ainoastaan echotagissa, jolla viestejä siirretään boksien välillä. Sen tarkoitus on vain toimia Sakunetin tunnisteena, jotta viestit eivät sekoitu muiden verkkojen viesteihin, mikäli boksi on useammassa netissä. Käyttäjille alueet saavat näkyä myös muilla nimillä, joiden ei ole mikään pakko olla AZ.jotakin. Pääasia on, ettei itse alueen nimeä muuteta niin paljon, että tulee epäselvyyksiä mikä alue on kyseessä.

Uusia alueita voidaan perustaa kun tarvetta ilmenee. Uuden alueen tarpeen huomaa ehkä parhaiten siitä, että keskustelu AZ.Muut alueella alkaa laajentua tietystä aiheesta niin paljon, että keskustelulle voisi perustaa jopa oman alueen. Samoin alueita voidaan poistaa, jos vaikuttaa siltä, ettei niillä käydä keskusteluja lainkaan. Tällaisista asioista päätetään yhdessä. Tätä kirjoitettaessa verkossa liikkuu melko mukavia määriä viestejä per vuorokausi, kun ottaa huomioon verkossa olevien nodejen määrän - 11 kpl. Keskustelu on pysynyt myös asiallisena, ja aihepiirejäkin on laidasta laitaan: tavallisesta tietokonekeskustelusta aina todella pitkälle mietittyihin teorioihin valon nopeudesta ja ajan olemuksesta!

Jotta myös modeemittomat lukijamme näkisivät, millaista keskustelua verkoissa käydään, poimin muutamia esimerkkiviestejä Sakunetin julkisilta echoalueilta. Modeemin omistajille tämä onkin jo ehkä tuttua, mutta moni on varmasti kiinnostunut uuden netin tarjonnasta.

Kannattaa siis tutustua. Kokoan tähän lopuksi bokseja, joista voit hakea QWK/WWF:lla (tai vaikka online) Sakunetiä ja osallistua mielenkiintoiseen keskusteluun miltä elämän aihepiiriltä hyvänsä. Tule kehittämään uutta verkkoamme! Tätä kautta tavoitat mm. Sakun tekijöitä, jos sinulla on heille palautetta! Voit myös välittää omia artikkeleitasi Sakuun. Tai jos jokin asia askarruttaa Sakua lukiessa, avaa asiasta keskustelu täällä!

Jos olet sysop, kysy lähimmältä Sakunetin nodelta/hubilta mahdollisuutta liittää boksisi mukaan Amiga-käyttäjien omaan verkkoon!

Sakunetiin voi tutustua näissä bokseissa

Amiga Zone BBS - Netin keskuspurkki, viestikanta alusta lähtien. On varsin ruuhkainen, mutta jos haluat yrittää, niin tee se klo 21-23, joka on varattu viestipakettien hakuun. Boksi on auki klo 21-04 ja numero on (958) 422 757.

Epsilon Indi - Sakun päätomittajan, Janne Sirenin boksi. Palvelee 24h numerossa (90) 505 4201.

Overscan BBS - TechnoBBS:n kehittäjän, Ville-Pertti Keinosen boksi. Auki 24h numerossa (90) 635 296.

1.42 Sakunetin viestejä

Tässä muutamia poimintoja Sakunetin echo-alueilta. Viestit on valittu satunnaisesti eri alueilta, siten että esimerkki kuvaisi hieman ko. alueen käyttöä. Viestejä on muokattu niin etteivät rivit ylitä 75 merkkiä.

```
Area:    AZ.Muut
Date:    15-Feb-95  19:30:24
From:    Antti Halla
To:      Tuomo Mämmelä
Subject: Re: Ennustus...

In a message of 23 Jan 95 Tuomo Mämmelä wrote to Esa Heikkinen:

 Oli ihan pakko replytä tähän hieman vanhaan messuun...

 EH>> SK> toisille ihmisille. Ihmiset kun eivät vielä ymmärrä ajan
käsitettä,

 No niinpä. Se aika kun ei ole mikään niin yksinkertainen käsite (siis jos
se ei ole lineaarista). Tosin minä ajattelen ajan lineaarisena, koska muuta
on vähän hankala "tällä älykkyden tasolla (ihminen)" kuvitella.

 EH>> Tulevaisuuteen pääsisi periaatteessa kulkemalla lähes valon nopeutta,
```

EH>> mutta ongelmana on, ettei sieltä pääse takaisin!

Lähes valon nopeus ei oikein riitä, sitä paitsi jos lähdet yli valonnopeudella maasta poispäin kuljet menneisyyteen. Siis ainoastaan periaatteessa, sillä jos otetaan esimerkiksi joku valovuoden päässä oleva paikka, ja katsotaan sinne täältä, niin mehän olemme periaatteessa vuoden heitä jäljessä (näemme siellä vuosi sitten tapahtuneita).
Katsoessamme sinne aika siellä kulkee "normaalia" vauhtia, ja valon nopeutta sinne matkustaessamme aika siellä näyttäisi kulkevan 2x vauhtia (?) Matkustaisimme siis tulevaisuuteen. Tätä asiaa on nyt messussa hankala selittää, mutta siis valonlähteitä (aurinkoja) on useassa paikassa, joten >VN vauhdilla matkustaminen olisi sekä menneisyyteen, että tulevaisuuteen matkustamista, vertailukohteesta riippuen.
Tuo siis oli "fake"-aikamatkustusta. Yli valon nopeuttahan ei (tämänhetkisten) fysiikan lakien mukaan voi kulkea, ja sen seurauksia on vähän hankala, ellei mahdoton ajatella _lineaarisessa_ ajassa. Takaisin tulevaisuudesta/menneisyydestä pääsisi vain kulkemalla jotain negatiivista valonnopeutta (mitäköhän sekin on, piti jotain keksiä).
"Aitoa" aikamatkustusta voisi ehkä olla "madonreiät", joista joskus oli jossain puhetta, tosin jos aika on lineaarista, niin aikamatkustusta ei ole olemassakaan, ja nuo "oikopolut" vain mahdollistaisivat valovuosien matkan hetkessä. Jos joku kävisi jossain 2VV:n päässä täältä oikomalla, ja palaisi saman tien, voisi hän siis katsoa matkansa uusintana maassa 2 vuoden päästä... Hankalaa.

TM> saavuttaa nykyisin tuntemallamme fysiikalla. Teoriassa valon nopeuteen
TM> voisi päästä mutta käytännössä se ei onnistu.

No onhan se tosiaan käytännössä hankala rakentaa aineetonta kulkuneuvoa..

--- Spot 1.3 Unregistered
* Origin: -*- Anttila -*- (65:917/2.0)

Area: AZ.Muut
Date: 16-Feb-95 14:11:49
From: Sami Klemola
To: Antti Halla
Subject: Ajan todellinen olemus

AH> Oli ihan pakko replytä tähän hieman vanhaan messuun...

Sen kuin. :)

AH> EH>> SK> Ihmiset kun eivät vielä ymmärrä ajan käsitettä
AH> No niinpä. Se aika kun ei ole mikään niin yksinkertainen käsite (siis
AH> jos se ei ole lineaarista). Tosin minä ajattelen ajan lineaarisena,
AH> koska muuta on vähän hankala "tällä älykkyden tasolla (ihminen)"
AH> kuvitella.

Se johtuu vain siitä, että korvaavaa mallia ei ole olemassa. Kun sellainen on, asian ymmärtäminenkin paranisi huomattavasti.

AH> Lähes valon nopeus ei oikein riitä, sitä paitsi jos lähdet yli
AH> valonnopeudella maasta poispäin kuljet menneisyyteen.

Minä olisin valmis heittämään koko teorian romukoppaan. Yhtä hyvin

voitaisiin ottaa äänen nopeus ja esimerkiksi ukkosen jyrinän kuuluessa
sanoa, että se tuli menneisyydestä. Toki ääni on syntynyt aikaisemmin
kuin se kuullaan, mutta ei se tarkoita, että se paikka olisi
absoluuttisesti jäljessä ajan suhteen.

AH> Siis ainoastaan
AH> periaatteessa, sillä jos otetaan esimerkiksi joku valovuoden päässä
AH> oleva paikka, ja katsotaan sinne täältä, niin mehän olemme
AH> periaatteessa vuoden heitä jäljessä (näemme siellä vuosi sitten
AH> tapahtuneita).

Niin, me näemme. Jos me katsomme täältä Englantiin (jos näkisimme sinne),
vaikuttaisi, että olisimme hieman heistä menneisyydessä, koska valo kulkee
hetken sitäkin matkaa, mutta elämme kuitenkin aivan sitä samaa aikaa kuin
englantilaisetkin - sama pätee etäisempiinkin kohteiisin, muihin
galakseihin.

AH> Katsoessamme sinne aika siellä kulkee "normaalia" vauhtia, ja valon
AH> nopeutta sinne matkustaessamme aika siellä näyttäisi kulkevan 2x
AH> vauhtia (?) Matkustaisimme siis tulevaisuuteen.

Höpön. Kuten jo sanoin, jos matkustaisimme kaksinkertaisella äänen
nopeudella kohti edellä mainittua salamaniskua, kuuluisi ääni
tuplanopeudella, mutta ei se tarkoita että menisimme tulevaisuuteen. Minä
en usko, että valoon liittyy mitään magiaa. Valo vain sattuu kulkemaan
nopeudella c, mutta mitään tekemistän ajan kanssa sillä ei ole.

AH> matkan hetkessä. Jos joku kävisi jossain 2VV:n päässä täältä oikomalla,
AH> ja palaisi saman tien, voisi hän siis katsoa matkansa uusintana maassa
AH> 2 vuoden päästä... Hankalaa.

Tuo oli ihan hyvä esimerkki. Toinen voisi olla, että kun käyt kilometrin
päässä huutamassa oikein kovaa ja tulet heti takaisin, kuulet huutosi
kolmen sekunnin kuluttua. :)

AH> No onhan se tosiaan käytännössä hankala rakentaa aineetonta
AH> kulkuneuvoa..

Tiedätkö, miten se on todennettu että valon nopeudella kulkevalla kohteella
ei voi olla massaa? Mihin tuo "tieto" perustuu ja kuka sen on käynyt
kokeilemassa?

--- TechnoBBS 0.93
 * Origin: SFL (65:951/1)

Area: AZ.Muut
Date: 19-Feb-95 10:16:39
From: Esa Heikkinen
To: Sami Klemola
Subject: Re: Ajan todellinen olemus

SK> Höpön. Kuten jo sanoin, jos matkustaisimme kaksinkertaisella äänen
SK> nopeudella kohti edellä mainittua salamaniskua, kuuluisi ääni
SK> tuplanopeudella, mutta ei se tarkoita että menisimme tulevaisuuteen.

No ei äänen nopeudella tulevaisuuteen pääsekään. Valon nopeudella
matkustamisesta taas ei voi pahemmin puhua kun sitä on niin vaikea
kokeilla. :-)

SK> Tuo oli ihan hyvä esimerkki. Toinen voisi olla, että kun käyt
kilometrin
SK> päässä huutamassa oikein kovaa ja tulet heti takaisin, kuulet huutosi
SK> kolmen sekunnin kuluttua. :)

No tuon voi toteuttaa paikallaan seisoenkin sopivassa ympäristössä jossa
kaikuu.

SK> Tiedätkö, miten se on todennettu että valon nopeudella kulkevalla
SK> kohteella ei voi olla massaa?

Ei kait tuota ole mitenkään todennettu. Toisaalta menee hulluksi kun alkaa
miettiä, miltä NÄYTTÄISI jos kappale kulkisi lähes valon nopeutta, tasan
valon nopeutta tai sen yli. Maasta katsoenhan yli valon nopeudella taikka
tasan valon nopeutta pois päin kulkeva kappale katoaisi näkyvistä kokonaan.
Entä miltä näyttäisi ko.kappaleen pinnalta katsottuna? Äkkiä ajatellen en
keksi muuta kuin että tulee täysin pimeää, taikka sitten vastaan tuleva
valo menee tuplanopeudella, siirtyy jollekin UV-alueelle tai paljon ylikin.
Doppler-ilmiöhän muuttaisi jo pienemmilläkin nopeudella esim. värit aivan
toisiksi. Kun kuljetaan lähes valon nopeutta, muuttuu ilmeisesti valon
taajuus niin paljon että värähtely siirtyy radioalueelle. Tästä saamme taas
sellaisen mietintämysteerin, että miltä mahtaa näyttää radiolähettimen
antenni kun lähetin on päällä ja antennia katsotaan lähestyen sitä melkein
valon nopeutta, jolloin radioaallot muuttuvat valoksi? Valo ja
radioaallothan ovat ainakin minun käsityksen mukaan samaa sähkömagneettista
säteilyä, ovat vain eri taajuuksilla. Ainakin valo leviää samaan tyyliin
kuin radioaallot, joten kyllä sen täytyy olla niin.

--- TechnoBBS 0.93
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)

Area: AZ.Muut
Date: 18-Feb-95 17:46:00
From: Tuomo Mämmelä
To: Antti Halla
Subject: Re: Ennustus...

AH> No niinpä. Se aika kun ei ole mikään niin yksinkertainen käsite
AH> (siisjos se ei ole lineaarista). Tosin minä ajattelen ajan
AH> lineaarisena, koska muuta on vähän hankala "tällä älykkyden tasolla
AH> (ihminen)" kuvitella.

 Minä en ainakaan ole nähnyt mitään "järkeenkäyviä" todisteita siitä
että aika ei olisi lineaarista. Parempi se on ajatella asiat samalla
tavalla kuin muutkin maan päällä asuvat ihmisrotuun kuuluvat. :)

AH> periaatteessa, sillä jos otetaan esimerkiksi joku valovuoden päässä
AH> oleva paikka, ja katsotaan sinne täältä, niin mehän olemme
AH> periaatteessa vuoden heitä jäljessä (näemme siellä vuosi sitten
AH> tapahtuneita).

 Niin ja he näkevät vuosi sitten täällä tapahtuneita, eli ovat
periaatteessa vuoden meitä jäljessä.

AH> Katsoessamme sinne aika siellä kulkee "normaalia" vauhtia, ja valon
AH> nopeutta sinne matkustaessamme aika siellä näyttäisi kulkevan 2x
AH> vauhtia
(?)

 Tähän en sano mitään...

AH> Matkustaisimme siis tulevaisuuteen. Tätä asiaa on nyt messussa hankala
AH> selittää, mutta siis valonlähteitä (aurinkoja) on useassa paikassa,
AH> joten >VN vauhdilla matkustaminen olisi sekä menneisyyteen, että
AH> tulevaisuuteen matkustamista, vertailukohteesta riippuen.

 Eli kun vertaamme nopeuttamme yhden valonlähteen valoon, menemme
menneisyyteen. Kun vertaamme samaa nopeutta toiseen valonlähteeseen,
menemme tulevaisuuteen. Kokonaisuutena emme siis liiku ajassa yhtään
mihinkään (?)

AH> vähän hankala, ellei mahdoton ajatella _lineaarisessa_ ajassa. Takaisin
AH> tulevaisuudesta/menneisyydestä pääsisi vain kulkemalla jotain
AH> negatiivista valonnopeutta (mitäköhän sekin on, piti jotain keksiä).

 Nopeus ei voi koskaan olla negatiivista muuten kuin villeissä
ajatuksissa. :)

AH> hetkessä. Jos joku kävisi jossain 2VV:n päässä täältä oikomalla, ja
AH> palaisi saman tien, voisi hän siis katsoa matkansa uusintana maassa 2
AH> vuoden päästä... Hankalaa.

 Eikö tuo toteutunut Paluu Tulevaisuuteen -leffassa? Siinähän se jätkä
lopuksi katsoi vierestä "matkalle" lähtönsä.

* WWF-Reader v0.60 *

--- TechnoBBS 0.93
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)

Area: AZ.Amiga
Date: 16-Feb-95 00:13:54
From: Mikko Sipiläinen
To: Kimmo Suorsa
Subject: Re: Amigalle kuuluu??

KS> Eikai tämä laite tunne myös nimeä Amiga 2200?? Vai mikäs laite tämä
KS> Amiga 2200 sitten on?? Näin siitä mainoksia tammikuun CU-Amigassa.

Vanha viesti, mutta....
A2200 on itseasiassa tämän päivän A4000, joka oli sellainen prototyyppi
uudenlaisesta amigamallista. Loppujenlopuksi A2200 syrjäytettiin uusien
AGA-grafiikkapiirisarjojen myötä ja pohjaksi valittiin A3000, johon tämä
grafiikkapiiristö upotettiin. A2200 on todellakin ollut olemassa mutta sitä

on valmistettu ainoastaan 200 kappaletta, jotka olivat developerkäytössä. Kone oli muuten A3000:n kaltainen, mutta siinä oli kuutamia muutoksia grafiikkaväylissä ja ECS deniseä oli jonkun verran paranneltu (mm. denice pyöri kaksinkertaisella kellotaajuudella vanhaan nähden). Kone oli myöskin 100%:sti 32bittinen myöskin grafiikkaominaisuuksiltaan, lukuunottamatta deniceä, joka toimi äänissä edelleen 8 bittisenä.

--- TechnoBBS 0.93
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)

Area: AZ.Amiga/Uutiset
Date: 14-Feb-95 12:30:08
From: Janne Siren
To: Esa Heikkinen
Subject: Re: DraCo

EH> On kuva. Tarviitko digitoituna? (järjestyy, kun kerrot missä
EH> formaatissa tarviit). Lehti on muuten sellainen kuin "Amiga Magazin",
EH> ilmeisesti

Jonkinlainen pieni kuva koneesta olisi uutisen ohelle mukava saada lehteen. Muutta IFF:ksi ja 16 väriseksi (jos saat harmaasävyskannauksen) ja muokkaa jos haluat, minä sitten muutan sen koon Sakun sivulle sopivaksi.

EH> No, kirjoitan tämän nyt siinä toivossa, että jos joku ymmärtää, niin
EH> suomentaisi tämän ja lähettäisi viestinä tänne alueelle. Koko jutun
EH> siis.

Jep. Internetistäkin tuli jo tietoa, joten sain tuosta uutispalstalle jutun aikaiseksi. Millonkohan saamme Sakuun testin? Oisko vähän upeeta... :-)

Jts

--- TechnoBBS 0.93
 * Origin: Epsilon Indi BBS, (90) 505 4201, 24h, V.32bis (65:90/1)

Area: AZ.Elektroniikka
Date: 16-Feb-95 08:11:26
From: Esa Heikkinen
To: Sami Klemola
Subject: Re: Onnistuneekohan?

SK> En koskaan puhu paskaa! Tarkoitin ainakin kyllä sanoa, että useimmissa
SK> koneissa en usko sen onnistuvan, koska se ainakin vaatii kovasti
SK> elektroniikkaa.

No se vaatii pari TTL-piiriä, toisella synkronoidaan kytkimeltä tuleva vaihtosignaali siten, että vaihto tapahtuu tarkalleen oikealla hetkellä tiettyyn kelloon tahdistettuna. Toisella piirillä sitten suoritetaan itse vaihto, eli kytketään kellotaajuus lähteestä toiseen.

--- TechnoBBS 0.93

```
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)


Area:    AZ.Hifi
Date:    30-Jan-95  01:04:01
From:    Esa Heikkinen
To:      Mikko Sipiläinen
Subject: Re: hmm.

MS> Tarvitsisin kaavat kaiuttimen litratilavuuden, refleksiputkin pituuden
MS> ja optimikotelon suunnittelukaavat.

No imaiseppa Amiga Speaker Design 1.00 originista. Siinä on valmis ohjelma.
Syötät vain elementin tiedot sinne ja halutun litratilavuuden yms. Softa
löytyy kätevimmin hakusanalla "speaker".


--- TechnoBBS 0.93
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)


Area:    AZ.Huuhaa
Date:    09-Feb-95  19:11:00
From:    Tuomo Mämmelä
To:      Mikko Sipiläinen
Subject: Re: Pääskyjen lentonopeus

MS> Olen kuitenkin sitä mieltä, että afrikkalaisten pääskysten lentonopeus
MS> on suurempi kuin eurooppalaisten kun otamme huomioon lämpötilavaihtelut
MS> ja saastuneen ilman aiheuttamat tahmaukset...

 Tosin pääskynen on kooltaan pienempi Euroopan kylmässä ilmassa (lämpö-
laajeneminen) kuin kuumassa Afrikassa. Tästä johtuen ilmanvastus on
pienempi Euroopassa huolimatta ilmansaasteista. Ja tähän kun vielä
lisätään se, että Afrikassa pääskyjen pitää kuljettaa omat eväät
repussa kun siellä on ruokapaikkoja niin harvassa. Tietysti reppu lisää
painoa ja hidastaa täten taloudellista lentonopeutta n. 20 prosenttia.
:-)



* WWF-Reader v0.60 *


--- TechnoBBS 0.93
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)


Area:    AZ.Huuhaa
Date:    09-Feb-95  23:37:25
From:    Mikko Sipiläinen
To:      Tuomo Mämmelä
Subject: Re: Pääskyjen lentonopeus

TM>  Tosin pääskynen on kooltaan pienempi Euroopan kylmässä ilmassa (lämpö-
TM> laajeneminen) kuin kuumassa Afrikassa. Tästä johtuen ilmanvastus on
TM> pienempi Euroopassa huolimatta ilmansaasteista. Ja tähän kun vielä
```

TM> lisätään se, että Afrikassa pääskyjen pitää kuljettaa omat eväät
TM> repussa kun siellä on ruokapaikkoja niin harvassa. Tietysti reppu lisää
TM> painoa ja hidastaa täten taloudellista lentonopeutta n. 20 prosenttia.
TM> :-)

Aivan, mutta toisaalta, kun otetaan huomioon pääskysen selkään aerodynaamisesti muotoiltu reppu ja paino, voidaan todeta, että sen massa ja ilman vastus ovat hyvin ratkaiseva tekijä. Väittäisin, että jos asetamme repullisen afrikkalaisen ja reputtoman eurooppalaisen pääskysen lähtölinjalle, odotamme 10 sekunttia ja ammumme kummatkin alas, niin eroa ei juurikaan tule. Tällöinhän pääskystemme erilaiset lentotottumukset eivät aiheuta ongelmia liitonopeuden määrittämisessä, mutta toisaalta. Kovin hienojyväisellä patruunalla ei varmasti kannata tussauttaa, sillä olisihan sääli, jos jompikumpi linnuistamme katoaisi salaperäisesti olemattomiin kesken testi. Tästä pääsemmekin taas seuraavaan asiaan, eli lentotottumuksiin, jotka tietysti kummalakin linnulla ovat erilaiset. Tuli vain tässä mieleeni, että afrikkalaisella pääskysellä täytyy sittenkin olla surkastuneemmat lihakset, kuin eurooppalaisella. ;*)

--- TechnoBBS 0.93
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)

Area: AZ.SAKU
Date: 15-Feb-95 21:51:21
From: Tomi Jaskari
To: Esa Heikkinen
Subject: Re: Alueen käyttö

Esa, sinä täydensit ajatusta oikein osuvasti...

EH> Niin ja tietty itse kukin voisi hieman kertoilla itsestään, koska/miten
EH> on aloittanut tietokoneharrastuksen, miten tutustui Amigaan ja miten
EH> aikoo tulevaisuudessa. Näinhän moiseen artikkelisarjaan saisi ahdettua
EH> yllättävän paljonkin asiaa, joka saattaisi jotakuta kiinnostaakin.
EH> Joskus on tullut kirjeitä, jossa pyydetään "SAKU-tyyppejä"
EH> kirjoittamaan hieman itsestään ja laitteistostaan.

Voisitko/ehtisitkö tehdä artikkelin itsestäsi ja Amiga-urastasi numeroon 11? Vai pitääkö minun uhrautua? Vai odotetaanko vielä yksi numero (ei hyvä)?

Tai jos saisi Anu Seilosen haastattelemaan...

--- TechnoBBS 0.93
 * Origin: Epsilon Indi BBS, (90) 505 4201, 24h, V.32bis (65:90/1)

Area: AZ.Saku/Yhdistys
Date: 11-Feb-95 10:52:42
From: Janne Siren
To: Sami Klemola
Subject: Re: Ehdotus

SK> mielestäni voitaisiin kirjoitaakin sanana, tietysti erisnimenä, eli

SK> Saku, ainakin epävirallisissa yhteyksissä. Virallisestihan se on
SK> kirjoitettava niinkuin yhdistyksen perustamiskirjassa on määritelty.

Otammeko käytännön tästä lähtien että lehden nimi ja SAKU kaikissa
yhteyksissä kirjoitetaan Saku, ei enää SAKU. Itseasiassa yhdistyksenkään
nimi ei ole Saku vaan Suomen Amiga-käyttäjät Ry (Ry:tä tietty ei ole vielä
hyväksytty). Sakun oikeinkirjoitusta ei ole siten perustamiskirjassa
mainittu.

Selventäisi ainakin tilannetta jossain määrin.

Tietysti lehdelle voisi keksiä jonkun vähän vetävämmänkin nimen. Eihän Saku
sanana kerro meille lehdestä yhtään mitään ja eihän lehden nimen tarvitse
olla sama kuin yhdistyksen nimen epävirallinen muoto?

Jts

--- TechnoBBS 0.93
 * Origin: Epsilon Indi BBS, (90) 505 4201, 24h, V.32bis (65:90/1)

Area: AZ.SciFi
Date: 16-Feb-95 09:58:44
From: Sami Klemola
To: Janne Siren
Subject: Re: Trek

JS> Harvasen kertahan Picard heittää: 'Mr. Data, you have the bridge'.
JS> Tietääkseni Data on arvojärjestyksessä seuraava Rikerin jälkeen.

No kai nyt vähän... Eihän LaForgen asemapaikka edes ole sillalla.
Meinasin vain semmoisessa erikoistilanteessa, tai siis, äh en minä
enää muista mitä oikein meinasin. Jaksossa Disaster Troi muuten
sai komennon. Ei hän kyllä tiedä laivan toiminnasta mitään. Ei tiennyt
edes sitä, että laiva räjähtää, jos antimateriaa koossa pitävä kenttä
pettää. Kyllähän tuollainen pitäisi tietää, vrt. auto pysähtyy, kun
bensiini (dieselöljy) loppuu...

--- TechnoBBS 0.93
 * Origin: SFL (65:951/1)

Area: AZ.Teeseitse
Date: 17-Feb-95 23:52:00
From: Sami Jantti
To: Sami Ylonen
Subject: Re: SCSI-2

SY> EH> Tätäkö etsit?

SY> Jee, taas yksi jonka pitäisi toimia.. Kiitti nyt kuitenkin, olen
SY> melkolailla luopunut toivosta...

 Sattumoisin olen rakentanut ton ohjeen perusteella itselleni
 kaapelin, ja hyvin toimii..

```
--- TechnoBBS 0.93
 * Origin: Miten saat koneesi rikki? Hajoita se! (65:958/2)


Area:    AZ.Tietokoneet
Date:    15-Feb-95  00:33:31
From:    Esa Heikkinen
To:      Kai Niemela
Subject: Re: DD <-> HD

KN> EH> puuttuminen tai vääränlainen formaatti.
KN>                     ^                   ^
KN> Mitäs tuo viimeksi mainittu tarkoittaa?

No ainakin eräästä asemasta löytyi diskchange 3:lla eri tavalla, kun
siltauksia muutti.

KN> Pitää harkita. Mutta antanevatko kokeiltavaksi kun kuulevat, että yritän
KN> sovittaa sen Amigaan. Toisaalta, jos en mainitse mitään, niin saattavat
KN> tuumata, että ei sitä tarvi testailla kun se on standardi koneisto :)

Niinpä.. Se on aina virittämistä, kun noissa HD-asemissa ei välttämättä ole
sitä Diskchangea. Sen verran vihjaan, että vanhemmat TEACin koneistot
[sellainen melko jämäkkä koneisto jossa on alumiinipeltiä oleva,
mattapintainen kuori (uudemmissa on kiiltävä)] toimii ainakin suoraan
ilman ongelmia. Minulla on CDTV:n asemana sellainen.

KN> Uhm.. onko tuo nyt sitten tulkittavissa sitten, että DD ja HD aseman
KN> luku/kirjoituspäät on saman levyiset?

Juu... HD:ssahan on vain enemmän sektoreita/ura.


--- TechnoBBS 0.93
 * Origin: Amiga Zone BBS - (958) 422757 - Open: 21-04   (65:958/1)


Area:    AZ.TV/Elokuvat
Date:    06-Feb-95  00:29:51
From:    Ville-Pertti Keinonen
To:      Sami Klemola
Subject: Re: Grannin TV-sarja

SK> nauraa itsensä vähintään kuoliaaksi. Siinä ohjelmassa kun muuten
SK> sanottiin, että alienit olisivat halunneet tulla julkisuuteen, mutta
SK> USA:n hallitus esti, HAH! Jos olennot osaavat matkata toiselle
SK> planeetalle, osaavat he kyllä iskeä itsensä vaikka kaikille
SK> satelliittikanaville. Tuo on yksi esimerkki sellaisesta pupusta, mikä
SK> saa

Siinä ohjelmassa myös sanottiin, että USAn hallitus on levittänyt paljon
epäuskottavaa ja ristiriitaista keksittyä tietoa UFO-aiheesta, jotta
ihmiset nimenomaan eivät uskoisi.

Sitäpaitsi luultavammin tuolla tarkoitetaan sitä, että avaruudesta tulleet
```

ensin tarkistivat USAn hallituksen kanssa, voisivatko he virallisesti tulla julkisuuteen ja vakuuttuivat hallituksen perustelujen pohjalta siitä, että se ei ole kannattavaa.

Toisaalta ei pidä uskoa kaikkea, kyseisessä ohjelmassa lienee aika paljon virheitä, ja Juhan Af Grannin erikoisala on kaiken saattaminen naurunalaiseksi aika heikosti uskottavalla esitystavalla. Kuitenkaan kaikkea ei pidä myöskään leimata suoraan "huuhaaksi"kaan (ihan hyvä, ettei tämäkään keskustelu ole sillä alueella).

SK> minut epäilemään koko hommaa. Tulisivat tänne, että näkisin omin
SK> silmin...

Kerro, jos käyvät..

--- TechnoBBS 0.93
 * Origin: Overscan BBS - (90) 635296 - 24H - V.FC (65:90/3)

Area: AZ.Vitsit
Date: 08-Tam-95 00:09:44
From: Mika Hamalainen
To: All
Subject: Mitä eroja?

Mitä eroa on helmitaululla ja Pentiumilla?

Helmitaululla saa laskettua oikein.

--- TechnoBBS 0.93
 * Origin: Sami's System Point, only for YOU! (65:958/2)

1.43 Mitä merkittävää taivaankannen alla?

Mitä merkittävää taivaankannen alla?

Heimo Laukkanen

Sitten viime Sakun ilmestymisen ovat asiat seuranneet toisiaan. Joskus on tullut vielä kolmaskin ja muodostanut ehkä neljännen kanssa viimeisen parin, joka on tullut uunista ulos. Tässä minusta tärkeitä ja ei niin tärkeitä poimintoja ja pohdintoja. Olkaa hyvät.

Internet on todellakin pulpahtanut pinnalle. Jos ette sitä nyt usko, niin milloin sitten? Maamme lähes yksinvaltiaan asemassa oleva ainakin pääkaupunkiseudun valtalehti, Helsingin Sanomat, on käsitellyt Internetiä muutaman viikon sisällä niin monessa eri artikkelissa, että kohta saa ottaa jo toisen käden laskuavuksi. Se ei välttämättä tunnu paljolta, mutta monen vuoden hiljaisuuden jälkeen tällainen ylitsepursuavuus on jo enemmän kuin huomiota herättävää.

Internet levittäytyy myös uusille aloille. Helsingin kaupunki liittyy 4000 - 5000 työasemalla eli koko omalla verkollaan Internetiin. Imagon kohottajaiset, joilla Helsinkiä nähtävästi puffataan tulevissa esitteissä edelleen kulttuurikaupungiksi vuodeksi 2000, maksavat kertakustannuksina Helsingin kaupungille noin 60 000 markkaa. Virastot päättävät itsenäisesti verkon käyttämisestä, mikä vaikuttaa juoksevien kulujen suuruuteen.

Kohta myös kirjastoista päästään Netiin. Tähän mennessä varsin menestyksekkäästi Kaapelitehtaan Kaapelikirjastossa toiminut Internet-yhteys on poikinut seuraajiaan jo Kemin ja Maarianhaminan kaupunginkirjastoihin. Jatkona tulevat myöhemmin Kaarinan, Jyväskylän ja Tampereen kaupunginkirjastot sekä yhteensä viisi helsinkiläistä kirjastoa. Yhteydet löytyvät jo Pasilan pääkirjastosta ja Töölön kirjastosta.

Seitsemäs helmikuuta Internet nousi jälleen otsikoihin lapsipornon avulla. Helsingin Sanomat siteerasi ruotsalaisen tietokonetutkijan Mats Wiklundin tutkimuksia ja Sole Lahtisen kirjoittamassa artikkelissa lähes ristiinnaulitsi Johan Helsingiuksen ylläpitämän anonymous-palvelimen pornokuvien ja lapsipornon pesänä. Samalla sivulla Antti Penttisen artikkelissa tietosuojavaltuutettu Jorma Kuopus kertoi paneutuvansa asiaan ja tutkivansa sitä mm. ihmisten henkilötietojen tallettamisen ja laittoman rekisterin keräämisen kannalta.

Henkilökohtaisesti minusta on aivan sama, onko jo muutenkin älyttömiä tietosuojalakeja rikottu rekisterin pitämisen suhteen, sillä ainakin minun käsitykseni mukaan Netin anonymous-palvelimet saavat enemmän hyvää kuin pahaa aikaan. Ymmärrän hyvin, että on ihmisiä, jotka eivät halua kysellä täysin avoimilla alueilla hyvin intiimeihin kysymyksiin vastauksia tai apua. Oikean henkilöllisyyden selvittäminen ilman anonymous-palvelimia on liian helppoa. Sen sijaan palvelimien ylläpitäjien pitäisi varata itselleen oikeus luovuttaa tietoja eteenpäin, mikäli on syytä epäillä, että palvelinta käytetään rikolliseen toimintaan. Poissuljettakoon toki sellaiset tapaukset kuten poliittiset rikokset.

En toisaalta tiedä, miten tietopyynnöt voitaisiin sitten tarpeeksi hyvin eritellä, mikäli jokin taho käsittelisi esimerkiksi poliittista vastarintaa edustavaa henkilöä huumausainerikollisena ja yrittäisi siten hankkia anonymous-palvelimelta tietoja. Olisiko tutkiminen tietojen luovuttamista tai kieltoa varten annettava tehtäväksi jollekin kansainväliselle poliisielimelle, kuten Interpolille?

Oli miten oli, vastine Helsingin Sanomien mielipidesivuille on lähtenyt ja toivottavasti jo julkaistukin. En tiedä, reagoinko tietokoneharrastajana liian voimakkaasti tähän asiaan, mutta sanottakoon, että kerrankin puolustan sanan vapautta sen oikeassa muodossa.

Mainittakoon toki, että Helsingin Sanomat on kertonut Internetistä positiivisiakin asioita. Suoralta kädeltä esimerkiksi muistuu mieleen juttu, jossa kerrottiin, että Bill Clinton kyllä vastaa Netin kautta lähetettyihin kirjeisiin, Martti Ahtisaari ei. Mitä me voimmekaan tästä päätellä? Joko amerikkalaiset ovat julkisuuden kipeämpiä, tai sitten Ahtisaari ei vain ole vielä oppinut käyttämään konettaan.

Varsinainen virstanpylväs oli kuitenkin City-lehden siirtyminen myös sähköiseen aikaan. Pienellä palstalla ilmoitettu uutinen oli sellainen sähköinen shokki, että Sakun lukijoidenkin sietää huomioida se. Cityn kotisivu on osoitteessa http://www.clinet.fi/city. Osoitteesta voi urkkia tule-

via aiheita ja vastaavasti myös antaa palautetta lehdestä. Sähköposti puolestaan kulkee osoitteeseen city@clinet.fi. Portit on avattu, käyttäkää niitä!

Jääkäämme muutenkin odottamaan vesi kielellä Helsingin Sanomien Talousosassa mainittua tietoverkkoprojektia. Artikkelin mukaan mm. Nokia, puhelinyhtiöt, Tele, muutamat lehtitalot ja Tekes eli Tekniikan edistämiskeskus puuhailevat laajakaistatietoverkkojen parissa tarkoituksenaan saattaa verkot myös tavallisten kansalaisten käyttöön kotipäätteiden, puhelimien ja television avulla. Satojen miljoonien markkojen laajuinen projekti kuulostaa herkulliselta ja voi poikia mielenkiintoisiakin tuloksia. Odottakaamme. Merkittävintä allekirjoittaneelle kuitenkin oli se, että Ismo Alangon uusi CD, Taiteilijaelämää, ilmestyi. Helsingin Sanomissa olleesta Matti-Esko Hytösen murska-artikkelista huolimatta kehotan kuuntelemaan Alangon levyä ja nauttimaan.

Tämä ei ollut maksettu mainos.

1.44 Yksityisyys ja vapaa tiedonlevitys Internetissä

Yksityisyys ja vapaa tiedonlevitys Internetissä

Antti Vähä-Sipilä

Suomessa sijaitsee eräs erittäin suosittu elektronisen postin anonyymipalvelin, anon.penet.fi. Internetissä kulkevaan postiin kiinnitetään aina tieto postin lähettäjästä, eikä tietyissä tilanteissa ole lainkaan suotavaa, että viestin lähettäjä voidaan tunnistaa postin perusteella. Erityisesti anonyymipalvelimesta hyötyvät henkilöt, jotka kirjoittelevat Usenetin uutisryhmiin (newsgrouppeihin, ks. SAKU #10) alueille, joilla heidät voidaan leimata esimerkiksi vakaumuksensa, kantansa tai seksuaalisen suuntautumisensa vuoksi.

Anonyymipalvelimeen lähetetään viesti, joka sisältää tiedon vastaanottajasta. Palvelin poistaa viestistä kaikki alkuperäiseen lähettäjään liittyvät tunnistetiedot ja lisää omansa niiden tilalle. Sen jälkeen palvelin lähettää viestin edelleen vastaanottajalle tai vaikkapa johonkin uutisryhmään. Tässä nimenomaisessa tapauksessa anonyymipalvelin tallentaa lähettäjän todellisen henkilöllisyyden tietokantaan ja luo lähettäjälle anonyymitunnuksen. Mikäli joku haluaa vastata viestin alkuperäiselle lähettäjälle, vastaa hän tälle anonyymitunnukselle. Viestin saapuessa palvelimeen lähettäjän tunnistetieto poistetaan ja vastaanottajaksi asetetaan tietokannasta löytyvä henkilö. Näin henkilöt voivat keskustella kummankaan todellisen henkilöllisyyden paljastumatta.

Parina viime viikkona on useissa eri yhteyksissä kohuttu anon.penet.fi:n ympärille kehittyneistä "skandaaleista". Ensimmäinen kohu syntyi, kun ruotsalainen tutkija oli tutkinut eroottisia kuvia sisältäviä uutisryhmiä (tutkimisesta voidaan olla monta mieltä, subj. huom.) ja todennut, että niissä levitetään lapsipornoa. Tästä paikalliset iltapäivälehdet väänsivät otsikoita tyyliin "Internet tulvii lapsipornoa". Kuvat tulivat lehden tietojen mukaan Helsingissä sijaitsevasta anonyymipalvelimesta, anon.penet.fi:stä. Palvelimen ylläpitäjä ja EUnet Finland Oy:n toimitusjohtaja Johan "Julf"

Helsingius onkin sittemmin yrittänyt korjata saamaansa julkisuutta useaan otteeseen.

Parin päivän päästä otsikoitiin, tosin huomattavasti pienemmällä pistekoolla, että kuvat tulivatkin Englannista. Lapsipornotulvakin oli kutistunut muutamaan hassuun kuvaan, joista pahimpia lienee televisiossa nähty kuva nokkahuilua soittavasta alastomasta lapsukaisesta. En yhtään epäile, etteikö verkossa liikkuisi myös "kovaa" lapsipornoa, mutta sensaatiolehdille tyypilliseen tapaan oli asiaa taas kerran paisuteltu yli äyräiden.

Tuskin oli lapsipornokohu ehtinyt laskeutua, kun kuvaan jo astuivat skientologit. Skientologikirkko väitti, että eräs amerikkalainen ex-skientologipappi olisi julkaissut Internetin skientologiaa käsittelevillä alueilla "salaista" tai "tekijänoikeuslain alaista" tietoa. Kirkko lähestyi Helsingiusta vaatien häntä luovuttamaan kyseisen henkilön tunnistetiedot, mutta hän kieltäytyi. Loppujen lopuksi kirkko sai tahtonsa läpi tehtyään ilmoituksen Interpolille, joka sai hankittua tiedon Helsingiukselta Suomen poliisin avulla. Asioiden todellinen tapahtumajärjestys löytyy esimerkiksi Helsingiuksen lehdistötiedotteesta, joka julkaistiin useilla Usenetin alueilla.

Kun henkilöllisyys oli saatu selville, hankki skientologikirkko oikeuden tehdä kotietsinnän viestin lähettäjän kotiin. Electronic Frontier Foundationin (EFF) julkaisemien lausuntojen mukaan "miehet, joiden henkilöllisyyttä ei saatu selville, tunkeutuivat asuntoon, tutkivat jokaisen paikan, veivät kovalevyn tiedot mennessään nauhavarmistimilla ja tuhosivat kovalevyn tiedot". Kirkko on nostanut syytteet viestin lähettäjää, hänen käyttämänsä purkin sysopia ja purkin palveluntarjoajaa Netcomia vastaan. Tätä kirjoitettaessa tilanne on kaikkea muuta kuin selvä.

Internetin paisuttua näihin mittoihin se on tiedon levittämisessä useassakin suhteessa ainutlaatuinen. Sitä ei omista mikään yksityinen instanssi, jolla olisi mitään oikeuksia valvoa siinä kulkevaa tietoa. Sillä ei myöskään ole yhtä systeemin ylläpitäjää, joka voisi poistaa häiriköiviä käyttäjiä. Sillä on paljon käyttäjiä, eikä yksityiselle käyttäjälle yhden viestin postittaminen maksa käytännössä mitään; kun tätä vertaa esimerkiksi lentolehtisten painamis- ja jakelukustannuksiin, on Internet ylivoimainen - puhumattakaan nopeudesta ja maantieteellisestä kattavuudesta.

Tämänkaltainen vapaa tiedonvälitys nähdään useiden ryhmien kannalta uhkana - vain yhtenä esimerkkinä yllämainittu skientologikirkko. Myös esimerkiksi USA:n valtion turvallisuudesta vastaava NSA (National Security Agency) on huolestunut siitä, että verkossa voidaan välittää huomattavia määriä aineistoa, johon se ei pääse käsiksi, ja pahinta on se, että aineisto leviää helposti ja kontrolloimatta yli rajojen. Internetissä ei ole tulliasemia.

Valitettavasti näillä vapaata tiedonvälitystä vastustavilla tai sen rajoittamista haluavilla ryhmillä on rahaa sekä voimaa vaientaa arvostelijansa ja vastustajansa. Tietenkin todelliset rikokset esimerkiksi tekijänoikeuksia vastaan ovat tuomittavia, mutta asiallinen kritiikki ja tiedonlevitys ovat ihmisen perusarvoja, jotka on kirjattu useimpien valtioiden perustuslakeihin. Jotta Internet säilyisi vapaana foorumina, tarvitaan sen käyttäjien apua.

Jotta "isoveli" voisi paremmin valvoa verkon liikennettä, haluaa esimerkiksi NSA rajoittaa kryptattuna eli salattuna liikkuvan tiedon määrää. Netissä de facto -standardiksi on noussut julkisen avaimen salakirjoitusjärjestelmä

PGP (Pretty Good Privacy), jonka uusin versio 2.6.2 on käännetty myös Amigalle. (Sen vienti USA:sta on laitonta, mutta sen käyttö USA:n ulkopuolella on sallittua.) PGP:llä luodaan kaksi avainta, joista toinen on julkinen ja toinen salainen. Voit antaa julkisen avaimen kenelle tahansa, ja tämä henkilö voi koodata sinulle tulevat viestit tällä julkisella avaimella. Viestit voi tämän jälkeen avata vain omalla salaisella avaimellasi. PGP:n salaus on erittäin voimakas. Se käyttää RSA- ja IDEA-kryptausalgoritmeja. Nykyiset avaimet ovat tyypillisesti yhdestä kahteen kilobittiä pitkiä, ja niiden "kräkkääminen" raakaa voimaa käyttäen on nykykoneilla lähes mahdotonta.

Koska anon.penet.fi on joutunut viranomaisten silmätikuksi, on varteenotettavaksi vaihtoehdoksi anonyymimailaukseen tullut Cypherpunk-mailereiden käyttö (cypher = cipher = salaus). Ne ovat yleensä yksityisillä mail-tunnuksilla pyöritettäviä uudelleenlähettäviä anonyymipalvelimia, jotka eivät pidä lokia siitä, mistä viesti saapuu ja minne se menee. Cypherpunk-mailereita voi ketjuttaa, ja niistä useimmilla on oma PGP-avain. Voit esimerkiksi salata tiedoston vastaanottajan PGP-avaimella, sen jälkeen cypherpunk-mailerin A avaimella, vielä mailereiden B ja C avaimilla ja lähettää sen mailerille C, joka avaa ensimmäisen kryptaustason, lähettää sen anonyyminä mailerille B, joka taas purkaa yhden tason ja anonymisoi postin vielä kerran. Kierrettyään vielä mailerin A läpi posti jatkaa vastaanottajalle, joka purkaa viimeisen koodaustason ja pystyy lukemaan postisi. Cypherpunk-mailerien käyttö on hieman haastavampaa kuin anon.penet.fi-tyylisen, mutta ei mahdotonta.

Niille, joilla on Internet-tunnus, on seuraavassa listattu muutamia paikkoja, joissa omasta turvallisuudestaan ja tiedonvälityksen vapaudesta kiinnostuneiden kannattanee käydä:

- Electronic Frontier Foundation. Pitää ajan tasalla esimerkiksi skientologikirkko vs. Internet -tapauksessa, vastustaa näkyvästi isoveljen valvontaa ja omaa laajat kokoelmat linkkejä alan palvelimiin:

<URL:http://www.eff.org/>

- Eurooppalaisille PGP 2.6.2:n saanti on ongelmallista, koska sitä ei saa viedä ulos USA:sta. Ståle Schumacher kuitenkin ylläpitää linkkejä sopiviin paikkoihin:

<URL:http://www.ifi.uio.no/~staalesc/PGP/>

- Amigalle PGP:n löytää seuraavasta paikasta. Koska PGP 2.6.2:n levitys on joidenkin FTP-admineiden mielestä epäilyttävää USA:n ulkopuolella, ei täältä ainakaan vielä löytynyt kuin 2.6ui, joka on 2.3a:n modifioitu versio mutta turvallisuudeltaan samaa luokkaa.

<URL:ftp://ftp.uni-paderborn.de/ftp/aminet/pub/util/crypt/>

- Uutisryhmästä alt.security.pgp löytyy kryptaamiseen ja yleiseen Internetin turvallisuuteen liittyviä artikkeleita. Anonyymipalvelimien kohtalosta saa tietoja uutisryhmästä alt.privacy.anon-server.

- Suomalaisen anon-palvelimen tueksi on perustettu postituslistoja. Oman, anonyymipalveluja puoltavan kommenttinsa voi lähettää listalle osoitteeseen anon-support@lists.otol.fi.

- Cypherpunk-remailereita voi käyttää myös WWW:n kautta osoitteesta:

<URL:http://www.c2.org/remail/by-www.html>

1.45 Ensikokemuksia World Wide Webistä

Ensikokemuksia World Wide Webistä

Heimo Laukkanen

Keskiviikkona 22. helmikuuta pääsin ensimmäistä kertaa tutustumaan Internetin parempaan puoleen, World Wide Webiin, eli WWW:hen. Tomi Jaskarin saattelemana pääsin vierailemaan Helsingin Yliopiston tietojenkäsittelytieteen laitoksen tiloissa niin tavallisten VT-320-päätteiden kuin myös itse varsinaisten X-päätteiden luona.

"Internet tulee, Internet tappaa, Internet"

Internetistä puhutaan paljon. Joskus jopa liian paljon. Kuten aikaisemminkin olen todennut, mediat ovat huomanneet Internetin arvon, tosin sen käyttö on ollut parhaimmillaankin vajavaista - muutamia poikkeuksia lukuunottamatta.

Puoluepelit, lehdet, yritykset ja yhteisöt ovat löytäneet oman sijansa netin maailmasta ja tarjoavat palveluitaan aivan uudella tavalla. City-lehteen voi lähettää palautetta netin kautta, Kokoomuksen vaaliehdokkaista ja uutisista saa tietoa netistä, Amnesty Internationalilla on oma sivu. Kohta koko maailma on netissä.

"If you think virtual reality is interesting, try reality" - Amnesty Interactive

Suomen oma Internet-lööppikin paljastui uutisankaksi. Lapsipornokuvat, jotka olivat levinneet Internetissä, eivät olleetkaan tulleet Johan Helsingiuksen anonymous-palvelimesta, vaan Englannista. Se siitä huippuhienosta tutkimuksesta ja journalismista.

"World Wide Web, W3, WWW"

Internetin taika on kuitenkin täysin ymmärrettävää. World Wide Web on hämmentävä mediatyökalu, jonka käyttöönottaminen vaatii oman käsittelytaitonsa, mutta luo ympärille kuitenkin särkymättömän taian. Tai särkymättömän ainakin siihen asti, kunnes käyttöaika loppuu tai yhteyttä ei saadakaan aikaan.

Kun minä pääsin ensimmäistä kertaa Webin maailmaan, olin myyty mies. On hämmentävä kokemus seurata nettisurffailua parilla eri mantereella. Hetkeä aikaisemmin ollaan Helsingissä, Jaskarin omalla kotisivulla, sitten Tampereella katsomassa hienoa ilmakuvaa Tampereen keskustasta. Yllättäen linkkaudutaankin Lulajan yliopistoon Ruotsiin, mistä löydetäänkin tie Lontooseen ja yhä Gdanskin Teknilliseen korkeakouluun. Kuitenkin kaiken aikaa ollaan edelleen Helsingin Vallilassa.

"Taidetta, kulttuuria, pornoa, propagandaa"

Netissä maailma joutuu kuitenkin kohtaamaan taas vanhat tiedon oikeellisuusrealiteetit. Lehdistä olemme tottuneet lukemaan todellisuuden. Reuterin, Itar-Tassin ja STT:n uutisia on pidetty lähes täydellisesti luotettavina, sillä uutisten todenperäisyyttä ei ole tarvinnut epäillä.

Sinisilmäisinä nettiin siirryttäessä pitää kuitenkin omata itsenäisen kyynninen ja epäilevä katsanto sieltä saatavaan tietoon. Vanhat kunnon tiedustelun periaatteet, eli varmistaminen muualta ja tietoketjun mahdollinen jäljittäminen taaksepäin, ovat edelleen hyviä neuvoja. Pieni viaton huhu, joka siirtyy nettiin ja elää netin eri osissa niin, että se tavataan useasti ja monissa eri yhteyksissä, muuttuu nopeasti totuudeksi.

Netti on siis vaarallinen propagandatyökalu, jonka käyttöönottaminen ei entisten medioiden, paperisten lehtien ja TV:n tavoin tarvitse sen suurempia uhrauksia. Tietokone, puhelinyhteys ja käyttöoikeus nettiin riittävät tiedon syöttämiseen ja mielikuvien muokkaamiseen.

"Tää on tietoyhteiskunta, varokaa vaa - maailma muuttuu, sen nähdä saa"

G7-maat ovat pitäneet oman kokoontumisensa tietoyhteiskunnasta. Paikalla oli meille myös Mikrobitistä tutuksi tullut Kari A. Hintikka, jonka mielipiteitä esiteltiin lähes jokaisessa uutislähetyksessä. Ehkäpä se on parasta viimeinkin uskoa, että cyberpunk-unelmista, ei niinkään painajaisista, voi vielä meidänkin elinaikanamme tulla totta. Kun ministeritason virkamiehet saa mukaan tietotekniikan puffaustapahtumaan niin suurella innolla kuin uutiskuvista näkyi, niin mikä tahansa voi lähivuosina olla mahdollista.

"No money, no honey"

Datalasit, Internet ja kaikki muu, joista virtuaalisurffarit orgasmimaisesti huohottaen puhuvat, ovat siis totisinta totta. Meillä on ihmiset, jotka osaavat sekä haluavat tehdä ja käyttää niitä, mutta meillä ei ole rahaa siihen kaikkeen.

Huipputekniikka maksaa. Vaikka Internet on oikeastaan nykyajan perusväline tiedonhankinnassa, sen käyttökustannukset ovat suurilta osin tavallisen opiskelijan budjetin ulkopuolella. Mikäli ei satu olemaan kirjoilla yliopistolla, teknillisessä korkeakoulussa tai muussa laitoksessa, joka tarjoaa opiskelijoilleen mahdollisuuden netissä seikkailemiseen, on tyydyttävä vain lukemaan Usenet-uutisia ja odottamaan sitä aikaa, kun pääsee itse vapauttamaan sielunsa elektroniseen viidakkoon - nettiin.

1.46 Tee se itse: Amiga 1200T

Tee se itse: Amiga 1200T

Kari Melkko

Pikku-Amigoiden huonoimmat puolet (ainakin tehokäyttäjälle, toim. huom.) ovat olleet niiden kiinteä näppäimistö ja kaikki-yhdessä-laatikossa-paketti, joka muutaman lisälaitteen kytkemisen jälkeen muuttuu todella sekavaksi

ja tilaa vieväksi johtorykelmäksi. Kovin ammattimaista kuvaakaan moinen paketti ei luo, ja Amigan pelikoneen mainekin lienee osittain tämän syytä. Mutta rahavarojen puutteen ja A4000/030-mallin heikon saatavuuden vuoksi minunkin oli tyydyttävä vain A1200:een. Vuoden verran jaksoinkin kärsiä kiinteän näppiksen kiroista, kunnes sietokyky ylittyi. Erillinen näppis ja tornikotelo oli saatava.

Huhuja tornitetuista A1200:ista (
Ramiga Tower
) on kuulunut jo pitkään,
ja ftp.funet.fi:stä löytyneet ohjeet Amigan tai PC:n irtonäppäimistön liittämisestä A1200:een saattoivat projektin aluilleen. Epätoivoisen irtonäppiksen etsimisen jälkeen löysinkin sattumalta rikkinäisen A2000-näppäimistön, josta tuli virittämällä uuden veroinen. Ohjeita seuraamalla ja vakaalla kolvikädellä A1200:ni siirtyi käyttämään irtonäppistä puolessa tunnissa. Pieni varoituksen sana lienee paikallaan: näppiksen liittäminen vaatii todella pienikärkisen ja tehoiltaan elektroniikkakäyttöön suunnitellun juottimen käyttämistä. Millään 60W tuuballa on turha yrittääkään, sillä kolmen johdon juottaminen suoraan pintaliitospiirien jalkoihin on suhteellisen tarkkaa puuhaa. Liitettävät piuhat ovat RESET (CTRL-Amiga-Amiga ei toimi ulkoisen näppiksen kanssa A1200:ssa), jonka maihin tökkäämällä saa koneen resetoitua, KBDAT ja KBCLK itse näppistä varten. Näppiksen DIN-liittimeen pitää vielä törkätä +5V, GND ja suojamaa.

Seuraavaksi etsin mahdollisimman halvan (virhe näin jälkikäteen ajatellen) minitornikotelon, 200W virtalähteellä ja 'hienolla' MHz-näytöllä varustettu kotelo (sellainen kun on muodikkain vaihtoehto) maksoi 380 markkaa. Myöhemmin sitten kokoa mittaillessani päädyin ratkaisemattomaan yhtälöön: A1200-emolevyn pituus muistikortin kanssa on 40.5 cm, ja kotelon ulkomitta oli tuo samaiset 40.5 cm. Jonkin aikaa emolevyä kotelon sisässä pyöriteltyäni sain sen sinne mahtumaan, pystysuuntaan ja pohjaan nähden vinottain. Emolevyn kiinnittäminen koteloon kaikkien taiteiden sääntöjen mukaan onkin sitten tarina erikseen. Itse päädyin epoksilla koteloon liimattuihin kiinnityskorviin, joihin emolevy ruuvataan koneruuveilla kiinni. Kooltaan suurempi laatikko (esimerkiksi IPS:n 42.5 cm) olisi helpottanut tehtävää huomattavasti.

Kotelon mataluuden vuoksi A1200-virtaliittimen käyttö on mahdotonta, joten ainoaksi ratkaisuksi jäi lyhyiden johdonpätkien juottaminen suoraan emolevyyn. PC:n virtalähteestä tulevat virtajohdot liitin sokeripaloilla näihin pätkiin. Ensimmäisellä käynnistysyrityksellä ei tapahtunut mitään: kone ei suostunut heräämään henkiin. Hetken pohtimisen ja emolevyn tutkailemisen jälkeen tajusin, että A1200 tarvitsee maan kahteen paikkaan: sekä virtaliittimen nastaan että virtaliittimen runkoon. Tämän operaation jälkeen kone jo toimikin kuten pitää. Varoitus: liittämällä virtajohdot väärin on mahdollista tuhota koneensa varsin helposti. Huolellisuus on tarpeen, ellei ole tekemässä kalansavustinta itselleen.

Seuraava vaihe operaatiossa onkin luultavasti työläin, joskaan ei vaativin. Liittimien vetäminen A1200:n emolevystä PC-kotelon taakse kun onnistuu vain tekemällä lyhyet välijohdot. Tinankäryä saa haistella useamman tovin, ellei sitten osta kaupasta lyhyitä ja edullisia välijohtoja. Välijohdoissa kannattaa käyttää lattakaapelia, sillä se on tässä tapauksessa ehkä nopein ja inhimillisin juotettava. Ainoa ongelmallinen tapaus liittimistä on RGB-liitin. PC-koteloissa on valmiina 2 kpl 9 pinnin D-liitinaukkoa ja toiset 2 kpl 25 pinnin D-aukkoa. Nämä reiät tukitaan hiiriporteilla, serial- ja parallel-liittimillä. RGB-liitin on 23-nastainen ja ilman paikkaa. Ratkaisin

päätä askarruttavan ongelman repimällä vanhasta XT:stä irti laajennuskortin, jossa oli reikä 25-napaiselle D-liittimelle. Koska 23-nastainen liitin ei aukossa suostunut pysymään kiinni, tein 25-napaisen liittimen rungosta 23-napaisen ja länttäsin kiinni. Ainoa lievä haitta tästä virityksestä on se, että monitoripiuhaa ei voi ruuvata kiinni kuin yhdellä ruuvilla, mutta pysyypä tuo mukana niinkin. Amigan RCA-liittimien kytkemistä varten (audio, video, RF) revin samaisesta XT:stä tyhjän korttipaikan tukkona olleen peitelevyn, johon sitten porasin RCA-liitinten reiät. Levyaseman asentaminen paikoilleen on helppoa, kunhan ostaa sille pidemmän datakaapelin. 3.5 tuuman kovalevyn asentaminen vaati myös datajohdon pidentämistä, sentti kun puuttui alkuperäisen pituudesta. Vielä virtajohdot levyasemaan ja kovalevyyn, reset-piuha reset-nappulaan ja ledien piuhat (power, HD, DF0) kiinni, ja johtosotku on melkoinen.

Kotelo kiinni - poissa silmistä, poissa mielestä. Ulkoisesti paketti on siistin näköinen, joskin etulevytön levyasema hieman pistää silmään. Jos omistaa lisälevyaseman, tätä ongelmaa ei tule... Suunnitelmissa on jostain kaivaa viallinen PC:n korppuasema, jossa olisi ehjä etulevy.

Summa summarum, tornittaminen kotikonstein on hieman vaivalloista ja aikaavievää. Iltapuhdetta riittää useallekin illalle, ja kolvia saa kärytellä ahkerasti. Jos innostusta riittää, kolvikäsi on vakaa eikä pelästy pieniä vastoinkäymisiä, niin tornittaminen kannattaa tehdä oitis. Irrallisen näppiksen ja minitornin lumoihin kerran päästyään ei halua enää missään tapauksessa palata takaisin alkuperäisen kotelon käyttöön.

Jos jo koneen avaaminen pelottaa, kannattaa harkita valmiin tornituspaketin ostoa. Maahantuoja: Data Service, PL 50, 02771 Espoo. Puhelin: (90) 859 7459.

Muutama vinkki: varaa projektille runsaasti aikaa ja kärsivällisyyttä. Hanki hyvä pienikärkinen kolvi, iso kasa liittimiä, johtoa, ruuveja, työkaluja ja jos mahdollista, kaiva kaapista vanha XT, sen osasille löytyy käyttöä.

Näppisohjeet: ftp.funet.fi (/pub/amiga/hacks/misc/A2000Keybd_2_A1200.lha)

1.47 Ramiga Tower

Valmiita tornitusratkaisuja Amiga 1200:lle on tulossa markkinoille useita. Ramiga International esitteli omansa World of Amiga -messuilla joulukuussa.

Ramiga Tower sisältää muutakin kuin paikan emolevylle ja massamuistilaitteille: tornin sisältä löytyy sisarkortti, jolla on tilaa neljälle Zorro II -kortille. Laajennusmurheistakin päästään kertaheitolla. Lisäksi tornissa on liitäntä PC-näppäimistölle, joten irtonäppiskään ei ole enää ongelma.

Janne Siren

1.48 Tietokone nimeltään Amiga - eli Amigan historia

Tietokone nimeltään Amiga - eli Amigan historia

Janne Siren

Amigan kaltaista konetta ei kehitetä yhdessä yössä. Ei edes kahdessa. Alkuperäisen Amigan kehittämiseen kului 40 miljoonaa dollaria ja satoja miestyövuosia, mutta lopputulos olikin sitten näkemisen arvoinen.

Tämä on Amigan tarina.

Amiga Computer Inc. sai alkunsa vuonna 1982 Amerikan Yhdysvalloissa RJ Mical -nimisen henkilön tavattua kolme floridalaista lääkäriä, joilla oli seitsemän miljoonaa dollaria sijoitettavanaan. Lääkärien ajatuksena oli ollut avata tavarataloketju, mutta he halusivat kokeilla jotain hieman jännittävämpää. Tietokoneet olivat tuolloin kovasti muodissa, joten he päättivät perustaa tietokoneyhtiön.

Uutta konetta kehittämään palkattiin Jay Miner Atarilta ja Dave Morse Tonka Toys -leluyhtiöltä. Ajatus oli alun perin tehdä niin hyvä pelikone kuin vain suinkin mahdollista. Pelikone, ei muuta. Jay Minerillä ja teknikoilla oli kuitenkin muita suunnitelmia, mutta ne he toistaiseksi pitivät visusti ylemmältä johdolta salassa.

Yhtiön nimi oli alkujaan Hi Toro, mutta se ei miellyttänyt sijoittajia. Uusi nimi napattiin sanakirjasta: Amiga. Nimen haluttiin ilmaisevan ystävällisyyttä, ja sana amiga tuli ensimmäisenä vastaan. Amiga tarkoittaa espanjaksi naispuolista ystävää, ja olihan se lisäksi ennen Applea aakkosissa. Itse kone sai nimensä markkinoinnista vastanneen Dave Morsen vaimon mukaan: Lorraine.

Lorrainesta alettiin siis kehittää pelikonetta. Ennen kuin konetta kuitenkaan voitiin julkaista, päätettiin hankkia hieman nimeä ja yhteyskanavia markkinoimalla muilta lisenssoituja pelejä ja lisälaitteita. Yksi tärkeimmistä oli nk. joyboard: eräänlainen joystick, jonka päällä seisten ja lanteita heiluttaen ohjattiin peliä. Lorrainen kehittäjät naputtelivat joyboardia varten pelin nimeltä Zen Meditation, jossa piti pyrkiä istumaan joyboardilla mahdollisimman paikallaan, kuin itämainen guru. Se olikin ainoa tapa rentoutua ja säilyttää järkensä, kun Lorraine tuppasi jatkuvasti kaatuilemaan. Joyboard ei valloittanut maailmaa, mutta Lorrainen virheenkäsittelyrutiini sai nimensä: Guru Meditation.

Lorrainessa ei alunperin ollut edes näppäimistöä, mutta suunnittelijoiden kaukonäköisyys palkittiin. Loppuvuodesta 1983 rahoittajat huomasivat, että pelikonemarkkinoilta alkoi pudota pohja pois ja tarvittaisiin jotain muuta kuin pelkkä pelikone. Lorrainessa oli jo valmiiksi tekniikkaa muuhunkin kuin pelikäyttöön, joten näppäimistön, levyaseman ja muiden liitäntöjen lisääminen ei tuottanut ongelmia. Jay Miner ja kumppanit saivat vihdoinkin toteuttaa suunnitelmansa: Lorrainesta tulisi oikea tietokone.

Lorraine messuilee

Tammikuun neljäntenä 1984, CES-messuilla Las Vegasissa, oli tullut aika esitellä Lorraine maailmalle. Ennen sitä kone oli pidetty visusti salassa, ja teollisuusvakoojat oli pidetty loitolla valmistamalla mm. peliohjaimia peitetoimena. Ikkunasta sisään kurkistava kilpailija näki vain joystickeja

ja jonkin hipin keikkumassa joyboardilla. Piilossa kuitenkin kehitettiin laitetta, jonka Business Week tulisi nimeämään henkilökohtaisten tietokoneiden Porscheksi.

Ohjelmistot olivat valmiina kymmenen päivää ennen messuja ja pyörivät moitteetta simulaattoreilla. Valitettavasti kovopuoli ei aivan vastannut simulaatioita, ja koneen kimpussa saatiin työskennellä kuumeisesti aina messujen ovien avaamiseen saakka. Lorraine muistutti tuolloin ulkoisesti lähinnä Cray-supertietokonetta. Erikoispiirejä (Agnus, Daphne ja Portia - jälkimmäisistä tulisivat myöhemmin Denise ja Paula) ei ollut vielä valettu piilastuiksi, vaan ne muodostivat piirilevyn päälle kolme johtomeren peittämää massiivista pyöreää tornia. Jos sivulliselle olisi kerrottu koneen joskus mahtuvan niinkin pieneen tilaan kuin Amiga 600, hän tuskin olisi uskonut sitä.

Lorraine saatiin kuin saatiinkin toimimaan ajoissa, ja kuukausien puurtaminen palkittiin. Lorrainea esiteltiin yksityisesti lähinnä suurten yritysten johtajille, ja sen mahdollisuudet herättivät suurta mielenkiintoa lähes kaikissa sitä ihastelemaan päässeissä.

Ensimmäisen onnistuneen messupäivän jälkeen markkinointiryhmä vei koko kehittelyporukan juhlimaan. Aamuyön tunteina italialaista ruokaa suupielissään humalainen ryhmä hoiperteli takaisin messuhalliin parantelemaan demoja ja korjaamaan bugeja. Messuilla kaikki työskentelivät 20 tuntia päivässä. Mical ja Dale Luck joutuivat soittamaan musiikkia täysillä ja tanssimaan vain pysyäkseen hereillä ohjelmien käännösten aikana. He saivat lisänimen "The Dancing Fools". Tanssivat tollot saivat juovuspäissään valmiiksi myös yhden Amigan tunnetuimmista ohjelmista, Boing!-demon.

CES-messujen jälkeen Amiga Computer Inc. oli lähes rahaton ja isoissa veloissa. Lorraine oli maksanut paljon enemmän kuin alkuperäiset seitsemän miljoonaa, ja tarvittaisiin vielä paljon aikaa Lorrainen kehittämiseksi markkinointikuntoon. Valitettavasti aika oli kuitenkin rahaa, eivätkä rahoittajalääkärit halunneet uhrata enempää, vaan päättivät irrottautua projektista. Ulkopuolista rahoitusta tarvittiin ja pian.

Rahaa saatiin haalittua kasaan tarpeeksi kesän CES-messuille matkustamista varten, ja Lorraine sai vihdoinkin ansaitsemansa huomion lisäksi myös rahoitusta. Näihin aikoihin Lorrainen nimi vaihtui Amigaksi, jona kone yrityksen nimestä johtuen jo paremmin tunnettiin. Rahaa oli kuitenkin juuri ja juuri tarpeeksi yhtiön pitämiseksi poissa konkurssituomioistuimista, sillä kaikki raha meni suoraan kehitystyöhön. Pian oltiin siinä pisteessä, että palkkoja oli leikattava. Markkinointipäällikkö Dave Morse joutui kiinnittämään talonsa pelastaakseen Amigan, mutta sekään ei ollut tarpeeksi.

Commodore ostaa Amigan

Amiga Computer Inc. oli kovaa vauhtia menossa nurin, ellei lisää rahoitusta saataisi jostain ja äkkiä. Neuvotteluja käytiin mm. Sonyn, Applen ja Silicon Graphicsin kanssa. Tulokseen päästiin lopulta vasta Atarin kanssa.

Atarin oli juuri ostanut Commodorelta suutuspäissään lähtenyt Jack Tramiel, joka kaavaili kostavansa menneet vihamielisyydet Commodorelle ostamalla Amigan ja päihittämällä Commmodoren tuotteellaan. Tramiel näki, ettei Amiga Computer Inc. ollut kovin hyvässä neuvotteluasemassa, ja asteittain hän

tarjosi vähemmän ja vähemmän tietäen, että Amiga Computerin olisi pakko lopulta hyväksyä naurettavan alhainen hinta ja myydä Lorraine Atarille pikkurahalla.

Commodoren onnistui kuitenkin sotkea Tramielin kuviot ostamalla Amigan Atarin nenän edestä. Commodore tarjosi kehittelyryhmälle 27 miljoonaa dollaria. Samana päivänä jokaiselle ohjelmistokehittelijälle kannettiin eteen upuusi SUN-työasema, ja kehitystyö pääsi taas vauhtiin.

Commodore teki paljon hyvää Amigalle. Se laski koneen hintatasoa kadottamatta paljoakaan sen ominaisuuksista, ja sillä oli aiemmasta poikkeava näkemys Amigasta työkoneena. Olisi Commodore silti asiansa paremminkin voinut hoitaa. Mical ei saanut haluamaansa kahdeksaatoista kuukautta lisää kehitysaikaa Commodoren omien taloudellisten vaikeuksien vuoksi, ja ensimmäinen Amiga, Amiga 1000, julkaistiin jouluna 1985 hieman raakileena. Commodore tuntui myös tyystin unohtaneen markkinoinnin, ja johtoportaan kiintymys vanhoihin malleihin hidastaisi Amigan kehitystä tuntuvasti seuraavan kymmenen vuoden aikana.

Se oli virhe, joka tulisi maksamaan Commodorelle kaiken.

1.49 Tee se itse: RGB-portti Amiga CD32:een

Tee se itse: RGB-portti Amiga CD32:een

Harri Laitinen

Kaikissa CD32:issa ei ole Amigan RGB-liitintä, vaikka niissä on TV-modulaattori, S-VHS ja Composite Video. Tämä omituinen puute käy oleelliseksi, jos haluaa käyttää CD32:a vaikkapa Commodoren 1942-monitorin kanssa. Syksyllä 1994 pulpahti kysymys CD32:n RGB-liittimestä tuon tuostakin esille Internetin Amiga-uutisryhmissä, joten sama puute näytti vaivaavan muitakin kuin itseäni. Vasta kun ryhmässä comp.sys.amiga.cd32 julkaistiin lokakuussa 1994 tieto CD32:n sisäisestä RGB-liitännästä ja sen johtojen järjestyksestä, asia alkoi hitaasti edistyä. Vuoden alusta minulla sitten on ollut myös monitoriini sopiva liitäntä, jollaisen rakentelusta kerron tässä.

1. Mitä tässä syntyy?

CD32:n oikeaan kylkeen tulee yksi 9-piikkinen D-liitin ja siihen liitettävä videovahvistimen laatikko. Vahvistamattomat RGB-signaalit tulevat ensin CD32:n piirilevyltä liitäntäkohdasta TP9 uuteen pieneen piirilevyyn konsolin kuorissa. Siinä puskuroidaan kuvan tahdistussignaalit. RGB-signaalit vahvistetaan ulkopuolella vahvistinlaatikossa, mistä kaikki johdetaan Amigamallisen monitoriliittimen kautta eteenpäin. Vahvistin saa käyttöjännitteensä CD32:sta. Tätä laajennusta voi käyttää, jos monitori kykenee näyttämään Amigan näyttötiloja videonäyttötiloista (PAL 50 Hz vertical, 15,6 kHz horizontal) alkaen.

Esimerkeissäni C.Sync-linjaa ei ole täydennetty loppuun, mutta tarvittaessa sen saa käyttöön täydentämällä kaaviot samanlaisella kytkennällä kuin H.Sync ja V.Sync.

2. Mitä jos joku tampio tunkee joystickin RGB-liittimeen?

Ikävä kyllä 9-piikkiseen D-liittimeen saa kiinni monenlaisia laitteita. Liittimessähän ongelmia aiheuttaa +12 V napa. Täysin idioottivarman liitännästä saa vain ottamalla vahvistimen jännitteen erillisestä jännitelähteestä, niin ettei sitä voi oikosulkea minkään kuvasignaalin tai CD32:n +5 V käyttöjännitteen kanssa. Mallissani napojen järjestys ja suojaus on mietitty niin, että RGB-liittimeen voi tunkea myös hiiren tai joystickin, eikä se polta CD32:n sisuskaluja. +12 V:n johdossa on sulake siksi, ettei oikosulkemalla +12 V ja GND voisi rikkoa muuntajaa. Diodi +5 V johdossa taas estää polttamasta CD32:n piirilevyä +12 V jännitteellä. Vahvistinlaatikon voi myös vaaratta kiinnittää hiiriporttiin.

Kaikesta huolimatta +12 V jää ongelmaksi, sillä mikä tahansa RGB-liittimen navan 9 (+12 V) johonkin navoista 1-6 yhdistävä lisälaite voi aiheuttaa tuhoa. Ongelman voisi ratkaista käyttämällä epästandardeja liittimiä, joihin ei saa kiinni mitään muuta lisälaitetta tai ottamalla vahvistimen käyttöjännitteen muualta.

Kuva: Kytkentöjen periaatekaaviot

3. Varotoimenpiteitä

Konsolin sisuskaluissa puuhastelu saa aivan varmasti takuun raukeamaan, jos se vielä sattuu olemaan voimassa. Tätä ohjetta ei muutenkaan kannata ryhtyä soveltamaan ainakaan ensimmäisenä elektroniikan rakenteluprojektinaan!

CD32 on Amiga 1200:n tavoin staattiselle sähkölle arka 32-bittinen tietokone, joten työskentelyyn pitäisi varata mahdollisimman vähän hankaussähköä sisältävä ympäristö. TP9:n liitosjohtojen juottamisen ajaksi olisi myös aihetta maadoittaa juotoskolvi ja CD32 samaan maahan. Tosin ainakaan laitteen oman muuntajan kautta tämä ei aivan helposti onnistu, sillä sen verkkojohtoa ei ole maadoitettu.

Erittäin hyödyllinen varotoimenpide on katkaista CD32:sta virta aina, kun siihen kiinnitettyä elektroniikkaa käpälöidään. Itse en rakentelun loppuvaiheissa malttanut pitää tästä kiinni, ja niinpä tulinkin rikkoneeksi Alicen. Varomattomuus maksoi sellaiset 700 markkaa ja viikkokausia hukattua vapaa-aikaa, ennen kuin kone taas suostui toimimaan.

4. Synkronointipulssien puskurointi

Alice-piiristä tulevat synkronointipulssit on syytä ottaa loogisen portin tai vastaavan suojauselektroniikan lävitse, sillä Alice on kallis ja hankalasti vaihdettava lastu. Väliin on viriteltävä elektroniikkaa senkin vuoksi, että koko Amiga kieltäytyy käynnistymästä kunnolla, jos jännite synkronointijohdoissa laskee liian alas. Väliin soveltuvat monenlaiset loogiset portit, esimerkiksi AND-portit, joiden sisäänmenot kytketään yhteen, kaksi NOT-porttia peräkkäin tai OR-portit. Minä käytin piiriä 74HCTLS08N, jonka kaltaista on käytetty myös Commodoren VGA-adapterissa C=390682-01.

Kuva: Pienempi piirilevy

5. RGB-signaalien vahvistaminen

RGB-signaalit tulevat TP9:ään suoraan D/A-muuntimesta vahvistamattomina, joten niitä on vahvistettava ennen monitorille johtamista. Vahvistimen elektroniikka suojaa myös D/A-muunninta. RGB-signaalien vahvistamiseen olen käyttänyt yksinkertaista jännitevahvistinta, jonka perusteet noukin Pentti O. A. Haikosen kirjasta Videotekniikka (1992). Erityisiä videovahvistimiksi rakennettuja integroituja piirejäkin on olemassa, mutta ikävä kyllä ainakaan Tampereen Bebekillä ei ollut niitä myynnissä. Ilman IC:tä edessä on hiukan vaivalloisempi urakka.

Kuva: Videovahvistin

6. Piirilevyn suunnittelu

Jos haluaa siistin lopputuloksen, on parasta valmistaa painopiirilevyt videovahvistinta varten. Levylle pitää mahduttaa ainakin kolme videovahvistinpiiriä ja mahdollisesti synkronointipulssien puskurointi. Itse asensin puskuroinnin pienemmälle piirilevylle, jonka sijoitin CD32:n kuorien sisäpuolelle. Isommalla piirilevyllä ovat sitten videovahvistimien seurana transistorit, joiden vahvistusta säätämällä saa synkronointipulssien jännitteet monitorille sopiviksi. Näitä jäljessä seuraavia transistoreja ei välttämättä edes tarvita, jos monitorin kaapeli on järjellisen pituinen.

Piirilevyihini tuli niin paljon hyppylankoja, että niiden juottamisen vaivalla olisi voinut suunnitella koko kortin siistimmäksi. Maajohdin kannattaa sijoittaa silmukaksi ison piirikortin ulkolaidalle, jolloin häiriöt siinä tasaantuvat mahdollisimman tehokkaasti. Omassa versiossani silmukka ei tule umpeen, mutta häiriöitä ei silti juuri huomaa. CD-ROM-aseman koneisto aiheuttaa pieniä jännitevaihteluita, jotka näkyvät joskus monitorissa heikkona sykkivänä ilmiönä, mutta sitä ei täysikään silmukka poistaisi.

Kuva: Piirilevy Kuva: Piirilevy päältä

7. Ruumiinavaus CD32:lle

Jotta CD32:n muovikotelon saisi auki, on kaikki laitteen pohjassa olevat ruuvit kierrettävä irti ja lisäksi irrotettava laitteen takaa yhdellä isolla ruuvilla kiinni oleva laajennusportin luukku. Tämän jälkeen kotelon voi avata kohottamalla kuulokepistokkeen puoleisesta päästä kansiosan irti pohjasta. Avaa varovasti, sillä CD-ROM-asema on edelleen kiinni kansiosassa, ja sen lattakaapeli on lyhyt. Kun kotelo on saatu auki, pitäisi saada avattua vielä peltikotelo, jonka sisällä itse kone on. Sitä varten on ruuvattava irti kaikki peltikotelosta ylöspäin näkyvät ristipääruuvit ja käännettävä pystyyn kotelon laidoille taitetut metalliliuskat. Lisäksi on irrotettava koneen piirilevyn etulaidasta ulos pistävä kaapelinippu, joka johtaa ohjauspaneeliin. Liittimen saa vedettyä irti työntämällä kapeateräisen veitsen liittimen kuoren ja pistokeosan väliin ja taivuttamalla niitä varovasti irti toisistaan niin, etteivät lukitustapit enää estä pistokeosan irrottamista. Nyt peltikotelon yläpuoliskon pitäisi olla irti ja nousta paikoiltaan vastustelematta.

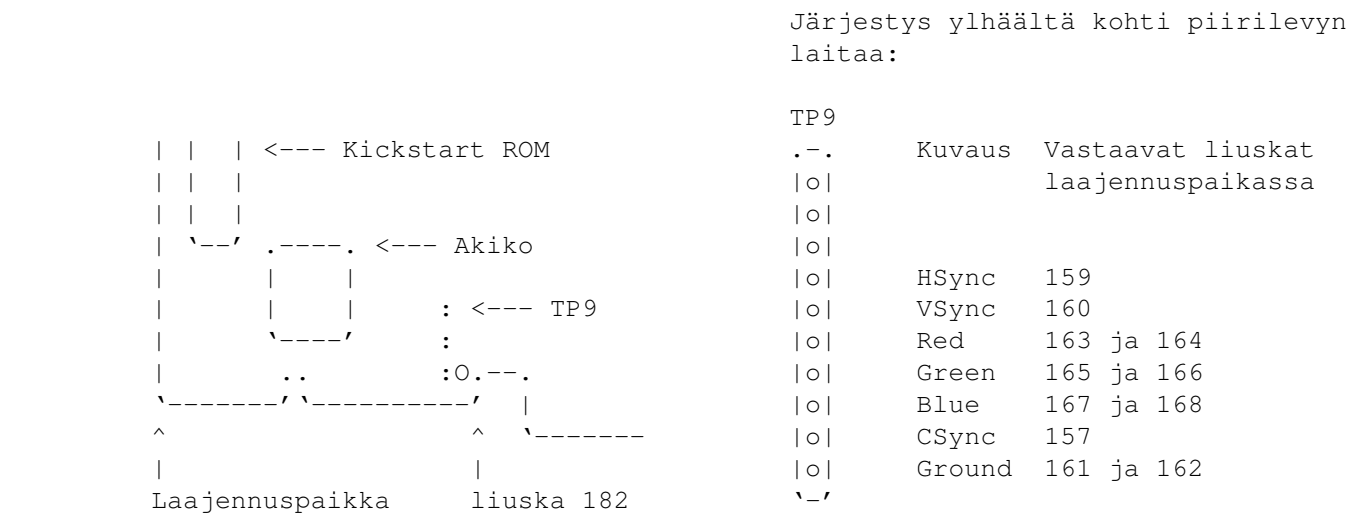
Seuraavaksi on saatava irti CD-ROM-aseman kaapeli. Se irtoaa helposti kohottamalla sitä piirilevyyn kiinnittävän muovisen liittimen kehystä noin

puoli senttiä. Kaapelin voi noukkia irti myös ennen peltikuoren avaamista, mutta kuoren alla olevan liittimen saavuttamiseen tarvitaan jokin pieni koukkupäinen työkalu, ja on vaikea nähdä mitä oikein tekee. Myös kokoamisvaiheessa romppuaseman kaapeli on hankala tapaus.

Peltikotelon sisältä paljastuu siis Amigan piirilevy. Portti TP9 on rivi tinapisteitä lähellä ulkoisen laajennuksen paikkaa. Pisterivin ympärille on maalattu kapea valkoinen suorakaide ja teksti TP9.

8. Sisäinen RGB-liityntä TP9

Seuraavaa kaaviota tutkimalla selviää, mikä tinapiste on mikäkin. RGB-signaalit löytyvät testiportista TP9.



Järjestys ylhäältä kohti piirilevyn laitaa:

TP9

	Kuvaus	Vastaavat liuskat laajennuspaikassa
	HSync	159
	VSync	160
	Red	163 ja 164
	Green	165 ja 166
	Blue	167 ja 168
	CSync	157
	Ground	161 ja 162

RGB-signaalit tulevat vahvistamattomina suoraan D/A-muuntimesta TP9:ään ja laajennuspaikkaan, ja S-VHS-liitäntä vie vielä osan niiden muutenkin heikosta tehosta. Tahdistussignaalit ovat puskuroimattomia, ja niistä HSync:iä ja VSync:iä voisi käyttää paitsi monitorin kuvan tahdistamiseen, myös tahdistamaan CD32 ulkoisen kellon mukaan. CSync taas toimii vain ulospäin.

Juotin johdot suoraan tinapisteisiin, mutta siistimpää olisi varmasti juottaa TP9:n päälle jonkinlainen liitinkampa ja käyttää sitä. Tosin silloin peltikuoresta joutuu myös leikkaamaan palan pois.

Koska videovahvistimet tarvitsevat toimiakseen sähkövirtaa, täytyy piirilevyltä vetää vielä ainakin kaksi johtoa: Sync-puskuroinnille +5 V ja videovahvistinpiireille +12 V. Molempiin sopivat liitäntäpisteet löytyvät läheltä CD32:n virtakytkintä. Omassa laitteessani (piirilevyssä lukee Spellbound rev 3) oli kolmen sentin päässä virtakytkimen edessä vierekkäin kolme tinapistettä, joista yksi oli +12 V, toinen +5 V ja kolmas maata. Jännitteet on parasta mitata ennen johtojen kiinnittämistä.

9. Kotelo

Kuva: CD32:n kuoret

Koko lisälaite kuuluu sulkea metallikoteloon, ja välikaapelien pitää olla suojattuja, muuten tämä viritelmä ei ainakaan paranna kuvan laatua. Metal-

likuorista pitäisi muodostua katkeamaton ketju CD32:n rungosta aina monitoriliittimen kuoreen saakka.

Kuva: Laitekotelo

10. Viritystä

Kun kaikki on valmista, kytkennät ovat oikein eikä missään ole oikosulkua tai epäonnistuneita juotoksia, koittaa odotettu hetki. Lopultakin saadaan kytkettyä CD32 kiinni monitoriin. Pitäisi vielä uskaltaa kytkeä laitteisiin virta päälle.

Jos kaikki menee hyvin, pitäisi monitoriin ilmestyä jonkinlainen kuva. Elleivät synkronointisignaalien jännitteet ole sopivat, kuva vipeltää hallitsemattomasti vaaka- ja pystytasossa. Tähän auttaa kuitenkin säätö kahdesta vahvistinpiirilevyn (-.Sync) trimmeristä. Kierrä ne aluksi asteikkonsa puoleenväliin ja säädä sitten yhtä kerrallaan.

Kun kuva pysyy ruudulla, videovahvistimen kolmea trimmeriä kääntelemällä pitäisi monitorin kuvan kirkkaus saada sopivaksi ja värisävyt oikein. Luultavasti kaikki trimmerit tulevat lähes samaan asentoon lähemmäksi vahvistuksen pienintä kuin suurinta arvoa. Lähelle värien oikeita sävyjä pääset ottamalla testikuvaksi valkoihoisen kasvot ja koettamalla saada ne näyttämään terveiltä. Säädöt ovat kohdallaan, kun kasvot eivät ole sairaan vihreät tai kuumeisen punaiset eivätkä sinerrä.

11. Kannattaako?

Lyhyesti voi todeta, ettei kannata, ellei rakentelu ole mukavaa. Jos hyvin käy, rakenteluun tuhrautuu muutama päivä ja tarvikkeiden hinta. Jos käy huonosti, menee pidempään ja voi tulla vaikka kuinka kalliiksi. Mutta saa tässä ainakin jännitystä elämään, ja rakentelu on vähintään yhtä mielenkiintoista kuin keskiverto seikkailupeli.

Arvio tarvikekuluista:

Transistorit	30 mk
Kondensaattorit	10 mk
Vastukset	5-10 mk
Metallikotelo	20 mk
Eristintapit	4 mk
Liittimet ja kuoret	20 mk
Johdot	5 mk
Painopiirilevyt	20 mk
IC 74HCTLS08N	6,50 mk
Holkin-rimat	5 mk
Yhteensä noin	130 mk

Kytkennöissä voi käyttää muitakin kuin esimerkissäni mainittuja transistoreja, kunhan taajuusalueet riittävät. (S2C1417-transistorit irrotin kiinalaisesta kelloradiosta.) Myös Haikosen kirjan esimerkeissä on eri komponentit.

1.50 TechnoBBS - Osa 3: Menukieli

TechnoBBS - Osa 3: Menukieli

Sami Klemola

Artikkelin tässä osassa käsittelen TechMenun ja sen käyttämän kielen. Kieli on C:n tyylinen, mutta paljon yksinkertaisempi. Ensiksi kuvailen kielen rakenteen, sitten siirrytään ohjelmointiin ja tehdään muutama komento. Sen jälkeen esittelen lisää TechnoBBS:n funktioita ja komentoja. Lopuksi tulee valmiita uusia komentoja menukoodilistauksina. Tämä on näillä näkymin viimeinen TechnoBBS:ään liittyvä artikkeli. Jatkossa käsiteltäviä aiheita kuitenkin voisivat olla TechMail ja TechnoBBS:n liittäminen verkkoon.

1.51 Ohjelmat ja niiden ajaminen

TechMenu on menukooditulkki, joka ajaa tokenisoitua menukoodia. ASCII-muotoinen lähdekoodi on ensin käännettävä ajettavaksi koodiksi. Siihen tarkoitukseen kuuluu ohjelmistoon TechMenuComp:

TechMenuComp <source> <destination>

Lähdekoodin oikeellisuuden kanssa kannattaa olla tarkkana, koska kun TechMenuComp sekoaa, se valittaa aivan jokaisesta merkistä, joka tiedostossa on. Virheilmoituksia tulee satoja... Lisäksi virhetilanteessa ohjelma aiheuttaa laittomia muistiosoituksia, Enforcer-hittejä ja jumiutuu joillakin tietyillä virheillä ja kaatuileekin! Virheettömän koodin kääntämisessä ei ole onglemia.

ASCII-tiedoston pääte on dat ja käännetyn koodin menu. Esimerkiksi Doors-valikko kääntyy näin:

TechMenuComp Doors.dat Doors.menu

TechMenuComp kääntää Doors.dat:n ja tuottaa sen pohjalta Doors.menun, jonka

TechMenu lataa ja suorittaa. TechMenulle annetaan myös noden numero:

```
TechMenu <node> <menu>
```

Silloin käynnistetään päävalikko tai jokin muu, se valikko, johon käyttäjän halutaan tulevan sisään:

```
TechMenu 1 Main.menu
```

Valmis menukoodi on nimeltään MainMenu.menu. Minä käytän kuitenkin vain valikon nimeä, koska "Menu" olisi nimessä aivan turha ja epälooginen lisä. Login-koodissa suoritettava käsky tuli jo edellisessä osassa, mutta laitan sen tähän vielä uudestaan. Heti login-ohjelmien ajamisen jälkeen suoritetaan:

```
address command "TechMenu "||ln||" BBS:Menu/Main.menu"
```

TechMenun ajo keskeytetään esimerkiksi, kun käyttäjä antaa Goodbye-komennon. Silloin suoritus palaa Rexx-ohjelmaan, ja siinä tulisi seuraavana olla tarvittava Goodbye-koodi. Täydellinen login-ohjelma julkaistiin edellisessä osassa.

1.52 Lähdetiedosto

TechMenuCompille annettava lähdekoodi on tavallista ASCII-tekstiä. Siihen voidaan sisällyttää kommentteja tavalliseen C-tyyliin. Kommentti alkaa merkeillä "/*" ja päättyy merkkeihin "*/". Kieli on muutenkin hyvin lähellä C:tä. Kielen operaattorit ovat samantapaisia, lauseet päätetään ";"-merkkiin ja ryhmät kootaan kaarisulkeisiin. TechMenu tuntee kahdeksan toimintoa:

Command	Määrittelee komennon
Execute	Suorittaa toimintoja
MenuName	Asettaa valikon nimen
MenuText	Määrittelee valikkotekstitiedoston nimen
Prompt	Määrittelee kehoitteen
HotOn	Asettaa ohjelman hotkey-tilaan tai sitten ei
LoadMenu	Lataa toisen valikon
Unknown	Määrittelee virhemerkkijonon

Näistä merkittävin on Command. Se määrittelee purkkiin komennon. Useita näistä ei tarvita kuin kerran tai ei ollenkaan yhdessä menukooditiedostossa. Käyn nyt nämä kaikki läpi yksitellen.

```
Formaatti   MenuName <string>
Esimerkki   MenuName "Extras"
```

MenuName asettaa valikon nimen. Nykyisellään sitä ei käytetä muuhun kuin valikon nimen näyttämiseen TechCon-ikkunassa, joten sillä ei ole juuri käyttöä.

```
Formaatti   MenuText <filename>
Esimerkki   MenuText "Extras"
```

MenuText määrittelee sen tiedoston nimen, jonka TechnoBBS lähettää komen-

nolla SendMenu. Päätettä nimeen ei tarvitse laittaa, vaan TechnoBBS kokeilee eri vaihtoehtoja ja katsoo käyttäjän Extensions-määrittelyn. TechnoBBS:n mukana tulevissa valikoissa tätä toimintoa ei hyödynnetä.

```
Formaatti   Prompt <expression>
Esimerkki   Prompt "\(27)[35m<\(GetTimeLeft())> \(27)[32mExtras: \(27)[0m"
```

Prompt määrittelee kehotteen, jonka TechnoBBS lähettää käyttäjälle aina, kun se on valmis vastaanottamaan komennon. Se voi olla yksinkertainen merkkijono, tai kuten yllä, sisältää funktiokutsuja. Yllä määritelty kehote näyttäisi kutakuinkin tältä:

```
<30> Extras:
```

Siinä 30 kuvaa jäljellä olevaa aikaa minuutteina. Nuo kaksi osaa tulostuisivat vielä eri väreillä. Määrittelyn ymmärtäminen ei ole tärkeää tässä vaiheessa. Myöhemmin tutustutaan tarkemmin kielen koukeroihin. Erikoistapaus on viestivalikko, jossa promptissa kannattaa kertoa lisätietoa. Minulla näytetään viestivalikon promptissa nykyisen viestialueen nimi ja olemassa olevien viestien rajat eli alueen ensimmäisen ja viimeisen viestin numerot.

```
Formaatti   Unknown <expression>
Esimerkki   Unknown "\(27)[31mKomentosi \"%s\" on minulle vieras.\(27)[0m"
```

Unknown määrittelee merkkijonon, joka lähetetään käyttäjälle, kun hän on antanut tuntemattoman komennon. Komento on tuntematon, kun sitä ei ole määritelty nykyisessä menukoodissa tai missä tahansa ladatussa menukoodissa globaalina komentona. Unknown hyväksyy samanlaisen määrittelyn kuin Prompt. Merkkijonossa oleva %s korvataan komennon nimellä, ja tässä se tulostuu lainausmerkkien sisään, jotka on escapoitava slashilla, jotta tulkki ei luulisi quoten olevan merkkijonon terminaattori.

```
Formaatti   HotOn <misc>
Esimerkki   HotOn "HOTKEYS"
```

HotOn määrittelee UserMisc-muuttujan, jonka arvo määrää, mennäänkö Hotkey-moodiin vai ei. Jos määritellyn muuttujan arvo on nolla, toimitaan, kuin HotOn-toimintoa ei olisi määriteltykään. Muussa tapauksessa siirrytään Hotkey-moodiin, jossa komentojen nimien kirjoittamisen sijaan ne valitaan yksittäisillä näppäimenpainalluksilla. Tällöin luonnollisesti kaikkien käytettäväksi tarkoitettujen komentojen on erottava ensimmäisestä merkistään, joka on Hotkey-moodissa se koodi, joka laukaisee komennon.

```
Formaatti   LoadMenu <menu>
Esimerkki   LoadMenu "Extras.menu"
```

LoadMenu määrittelee ladattavan menukoodin. Normaalisti TechMenu ajetaan vain kerran, jolloin sen käsketään ladata päävalikko. Päävalikossa on erinäinen määrä LoadMenu-toimintoja, jotka lataavat kaikki muut valikot. Tällöin voidaan helposti siirtyä valikosta toiseen menu()-funktiolla. TechMenun sisäiset funktiot käsitellään yksityiskohtaisesti myöhemmin. Kussakin valikossa voi olla myös globaaleja komentoja, joiden ajaminen onnistuu mistä tahansa valikosta, kun kaikki valikot on ladattu.

```
Formaatti   Execute { <actions> }
```

Execute suorittaa kaarisulkuihin sisällytetyt toimenpiteet. Tämä toiminto ei ole ollenkaan käyttökelpoinen, koska sen ollessa käytössä kaikki muut toiminnot jätetään huomiotta. Executea käytettäessä tuloksena ei olekaan valikko vaan skripti, joten toiminnossa ei ole mitään järkeä. Parempi vaihtoehto olisi ollut, että nämä toimenpiteet suoritetaan valikkoon siirryttäessä ja sen jälkeen aloitetaan normaali toiminta, mutta sen sijaan koodin suoritus päättyy. Järjetön toiminto!

Formaatti Command "<id>" [<options>] { <actions> } ["<name>"];

Command on TechMenun tärkein toiminto. Sillä määritellään purkkiin komento. Komennon nimi on ensimmäisessä merkkijonossa, ja se on merkitty yllä id:ksi. Tämän jälkeen tulevat optiot, jotka määräävät, kenellä on oikeus käyttää komentoa. Seuraavaksi tulevat komennossa suoritettavat toimenpiteet kaarisuluissa eli varsinainen komennon koodi. Toimenpiteet käydään läpi seuraavassa luvussa. Viimeisenä voidaan vielä määritellä komennolle nimi, joka lähetetään käyttäjälle Hotkey-moodissa, kun se on tunnistettu.

Command-toiminnon optiot voivat olla standalone-optioita tai niihin voi liittyä parametri. Optioilla rajoitetaan komennon käyttöä:

A:<value>	Asettaa minimioikeustason komennon käyttäjälle
G:	Määrittelee komennon globaaliksi
M:<value>	Asettaa maskin komennon käyttäjälle
N:<value>	Määrittelee tarpeellisten merkkien määrän
O:<expression>	Tarjoaa mahdollisuuden monimutkaiseen kontrolliin

Optiot A ja M asettavat käyttäjältä vaadittavat oikeusmäärittelyt. Voidakseen käyttää komentoa käyttäjän oikeustason tulee olla vähintään yhtä suuri kuin A-optiossa annettu luku. Lisäksi M-optiossa annetun luvun päällä olevien bittien tulee olla päällä myös käyttäjän maskissa. Kolmas käyttörajoituksiin liittyvä optio on O. Sitä seuraa kokonainen lause, jonka pitää olla tosi (arvo muu kuin nolla), jotta komentoa voi käyttää.

Optio G määrittelee komennon globaaliksi. Globaali komento voidaan suorittaa missä tahansa valikossa. Optio N määrittelee, kuinka monta merkkiä komennon nimestä tarvitsee täsmätä, että katsotaan komento annetuksi. Tämä luku kannattaa laittaa aika pieneksi, jotta komennot voidaan lyhentää reilusti. Lyhyillä komennoilla optiota ei kannata käyttää ollenkaan, vaan komennot kannattaa edellyttää kokonaisina.

1.53 Toimenpiteet

Komennossa suoritettavat toimenpiteet ovat lähinnä muuttujan asettaminen, suorituksen ohjaus tai funktiokutsu. Muuttujat menukielessä alkavat aina dollarimerkillä. Suorituksen ohjaukseen ovat olemassa funktiontyyliset if() ja while() sekä quit ja break. Funktiokutsu voi olla TechMenun sisäinen tai ulkoinen TechCon:n tai TechnoBBS:n funktio. TechnoBBS:n funktioita menukoodista kutsuttaessa tulee huomata, ettei noden numeroa tarvita. TechMenu hoitaa sen automaattisesti.

Muuttuja asetetaan tutusti:

$<id> = <expression>

Tässä id on muuttujan nimi ja expression lause, joka tulkitaan ja jonka evaluaation tulos asetetaan muuttujaan. Lauseiden rakenne käydään läpi yksityiskohtaisesti myöhemmin. Koodin suoritusta ohjaavista toimenpiteistä if() on useimmin käytetty ja hyödyllisin. Sen avulla voidaan koodin suoritusta ehdollistaa:

```
if(<expression>) <action>
```

tai

```
if(<expression>) {
    <actions>
}
```

Ylempää tapaa voi käyttää silloin, kun ehdollistettavia toimenpiteitä on vain yksi. Kun niitä on useampia, ne erotetaan ryhmäksi kaarisuluilla. Myös yksittäinen toimenpide voidaan laittaa kaarisulkeiden sisään. Ehdollistettu koodi, if():n perässä olevat toimenpiteet, suoritetaan silloin, kun if():lle annettu lause on tosi. Jos on tarpeen suorittaa tietyt toimenpiteet, kun se ei ole tosi, lauseen voi invertoida. Lisäksi on mahdollista käyttää else-rakennetta:

```
else <action>
```

tai

```
else {
    <actions>
}
```

Tällöin else-rakenne tulee heti if():n perään, ja sen sisältämät toimenpiteet suoritetaan silloin, kun lause ei ole tosi. If() ja else ovat erilliset rakenteelliset osat, eikä kaarisulkeita tarvitse käyttää kummassakin, vaikka toisessa ne olisivatkin. Toinen koodin suoritusta ohjaava toimenpide on while():

```
while(<expression>) <action>
```

tai

```
while(<expression>) {
    <actions>
}
```

While() suorittaa perässä tulevia toimenpiteitä niin kauan kuin sille annettu lause on tosi. Jos se ei ole tosi ensimmäisellä while():n suoituskerralla, ei perässä tulevia toimenpiteitä suoriteta ollenkaan. While()-looppi voidaan keskeyttää break:lla. Mikäli break ei ole while()-toimenpidesarjassa, se keskeyttää koko komennon suorituksen. On mahdollista tehdä looppi, josta ei poistuta ollenkaan, paitsi break:n avulla:

```
while(1) {
    /* suoritettavat toimenpiteet */
}
```

Neljäs koodin suoritusta ohjaava toimenpide on quit. Se keskeyttää suori-

tuksen, ja quit:n kohdatessaan TechMenu palaa sen ajaneeseen ohjelmaan.

1.54 Sisäiset funktio

TechMenu tarjoaa neljä funktiota, jotka eivät palauta arvoa, vaan ne toimivat enemmänkin komentoina ja suoritetaan toimintoina:

```
menu()          Vaihtaa valikkoa
dos()           Suorittaa DOS-komennon
rexx()          Suorittaa Rexx-komennon
rxport()        Vaihtaa Rexx-porttia
```

Jo aikaisemmin tuli ilmi mahdollisuus ladata useita valikoita yhtaikaa. Kun valikko on LoadMenu-toiminnolla ladattu, voidaan siihen vaihtaa funktiolla menu(). Parametriksi funktiolle annetaan käännetyn menukoodin tiedostonimi ilman hakemistopolkua.

Esimerkki: menu("Extras.menu");

Funktio dos() suorittaa DOS-komennon. Yleensä sille annetaan suoraan merkkijono, mutta sille voidaan antaa myös funktiokutsu tai niitä voi olla osana merkkijonoa. Parametrinä annetun lauseen evaluaation tulos ajetaan DOS-komentona.

Esimerkki: dos("MakeFList");

Rexx-ohjelmalle annetaan yleensä parametrinä, kuten tunnettua, noden numero, jonka avulla ne kommunikoivat suoraan TechnoBBS-prosessin kanssa. Ulkoinen ohjelma ei kuitenkaan välttämättä osaa käyttää TechnoBBS:n Rexx-liittymää, vaan kommunikoi standardien I/O-väylien kautta. Avuksi tulee TechIO, jonka avulla saadaan ohjelman stdin- ja stdout-kanavat ohjattua modeemille. Esimerkiksi tiedostolistaus lähetettäisiin näin:

```
dos("List BBS:Menu <>TECHIO:\(node())");
```

TechIO:n täytyy tietää, minkä noden kanssa ohjelman tulee kommunikoida. Tieto saadaan sisäisellä funktiolla node(). Näin mitä tahansa ohjelmaa voidaan helposti käyttää linjalta, kunhan se ei avaa omia ikkunoita tai pyytimiä.

Funktio rexx() suorittaa Rexx-komennon. Se toimii samaan tapaan kuin dos(). Komento lähetetään aktiiviseen Rexx-porttiin, jonka voi vaihtaa funktiolla rxport(). Se vastaa toiminnaltaan melko lailla Rexx-kielen Address-komentoa.

```
Esimerkki: rexx("SendModem Hello, world!");
Esimerkki: rxport("TECHCON");
```

Yleensä Rexx-portti, johon menukoodista lähetetyt komennot menevät, on kyseisen noden TechnoBBS-prosessin portti. Näin esimerkiksi yllä oleva SendModem-komento menisi oikeaan osoitteeseen ja toimisi samoin kuin Rexx-koodista. Kaikille funktioille annettavat parametrit ovat lauseita, ja funktiolle varsinaisesti annettavat argumentit ovat niiden evaluaation tuloksia.

TechMenun varsinaiset sisäiset funktiot ovat:

```
arg()           Palauttaa komennon argumentteja
node()          Palauttaa noden numeron
str()           Ottaa osan merkkijonosta
len()           Palauttaa merkkijonon pituuden
upper()         Muuttaa merkkijonon isokirjaimiseksi
lower()         Muuttaa merkkijonon pienikirjaimiseksi
split()         Etsii merkkijonosta toista merkkijonoa
```

Komennon nimen saa selville kutsumalla arg(0):aa. Muut arvot palauttavat varsinaisia argumentteja, yksi ensimmäisen, kaksi toisen jne. TechnoBBS ei osaa käsitellä lainausmerkkejä komentorivillä, joten monisanaiset argumentit tulevat erillisinä. Esimerkiksi komennolle parametrinä annettava käyttäjän nimi tulee aina vähintään kahtena argumenttina. Näiden kanssa tulee olla tarkkana. Kuten jo tulikin ilmi, node():lla saa selville, millä nodella koodia ajetaan.

Funktio str() vastaa BASICin MID$()-funktiota. Esimerkiksi str("tapuli", 2,3) antaisi ulos merkkijonon "apu". Toinen argumentti siis kertoo, mistä kohtaa aletaan ottaa merkkijonoa. Arvo 1 tarkoittaa merkkijonon ensimmäistä merkkiä. Viimeinen parametri kertoo, kuinka monta merkkiä merkkijonosta otetaan. Funktio len() palauttaa merkkijonon pituuden. Upper() ja lower() muuttavat kaikki kirjaimet isoiksi ja pieniksi. Funktio split() palauttaa kohdan, jossa merkkijono alkaa toisessa:

```
split("hirsikasa","sika")
```

Tämä lause saisi arvon 4. Merkkijono "sika" löytyy ensimmäisestä merkkijonosta alkaen merkistä 4. Jälleen arvo 1 tarkoittaa merkkijonon ensimmäistä kirjainta. Itse asiassa näiden lauseiden ei tarvitse olla merkkijonoja. Periaatteessa TechMenu käsittelee kaikkia lauseita merkkijonoina, ja siksi sen toiminta on joissakin tilanteissa vähintään omituista.

1.55 Lauseet

Lauseet koostuvat kahdentyyppisistä elementeistä, operaattoreista ja arvoista. Operaattorit ovat samantyylisiä kuin C:ssä ja niitä on kasapäin. Ne käsitellään yksitellen myöhemmin. Lauseessa voi olla yksi tai useampi elementti. Arvoja on oltava vähintään yksi, mutta niitä voi olla useita, jolloin ne erotetaan toisistaan operaattoreilla, jotka suorittavat niiden välillä jonkin operaation. Lopulta koko lauseen arvo saadaan selville, kun kaikki yksittäiset arvot on selvitetty ja operaatiot ratkaistu. Arvot voivat olla seuraavia:

```
$<id>             muuttuja
"<string>"        merkkijono
<number>          numero
<name>([<args>])  funktiokutsu
```

Muuttujan nimi, esimerkiksi $muuttuja, tuo sen kohdalle lauseessa muuttujan arvon. Merkkijono ja numero ovat kiinteitä arvoja, jotka ovat samat joka kerta kun lausetta ratkaistaan. Funktiokutsu tuo sen kohdalle lauseeseen funktion palauttaman arvon. Lisäksi lauseessa voi olla kolmella tavalla lisää lauseita:

```
~<expression>     käänteistys
!<expression>     looginen käänteistys
(<expression>)    sulkeistus
```

Normaalisti suoraan, muuttujasta tai funktiolta saatu arvo tulee lauseeseen sellaisenaan, mutta laittamalla "~"- tai "!"-merkki lauseen eteen sen arvo voidaan invertoida. Sulkeiden avulla lauseiden suoritusjärjestystä voidaan muuttaa. Normaalisti lauseet tulkitaan vasemmalta oikealle, eikä laskusäännöistä ole tietoakaan. Lause ei siis oikeastaan olekaan vain yksi lause, vaan siinä voi olla vaikka kuinka monta lausetta, jotka on kaikki ratkaistava, ennen kuin sen arvo on selvillä.

Menukoodissa hyväksyttävät operaattorit ovat:

`==` yhtäsuuruus: arvo on 1, jos operaattorin kahta puolta olevat arvot ovat samat

`!=` erisuuruus: arvo on 0, jos ne ovat samat, muuten 1

`>` suurempi kuin: arvo on 1, jos arvo vasemmalla puolella on suurempi

`<` pienempi kuin: arvo on 1, jos arvo vasemmalla puolella on pienempi

`>=` suurempi tai yhtä suuri kuin: arvo on 1, jos arvo vasemmalla puolella on suurempi tai sama kuin oikealla puolella

`<=` pienempi tai yhtä suuri kuin: arvo on 1, jos arvo vasemmalla puolella on pienempi tai sama kuin oikealla puolella

`+` yhteenlasku: arvo on operaattorin kahta puolta olevien arvojen summa

`-` vähennyslasku: arvo on vasemmalla puolella oleva arvo vähennettynä oikealla puolella olevalla arvolla

`*` kertolasku: arvo on operaattorin kahta puolta olevien arvojen tulo

`/` jakolasku: arvo on vasemmalla puolella oleva arvo jaettuna oikealla puolella olevalla luvulla

`++` merkkijonoyhteenlasku: yhdistää operaattorin kahta puolta olevat merkkijonot

`|` tai-toiminto operaattorin kahta puolta olevilla arvoilla

`||` looginen tai-toiminto

`&` ja-toiminto

`&&` looginen ja-toiminto

`^` ehdoton tai -toiminto

Merkkijonoa on syytä vielä käsitellä tarkemmin. Siinä käytetään kenoviivaa escape-merkkinä. Esimerkiksi lainausmerkki saadaan merkkijonoon, kun sitä

edeltää kenoviiva. Muuten se tulkitaan merkkijonon päätösmerkiksi. Näin myös kenoviiva itse on escapoitava. Kenoviivaa voivat seurata sulkeet, jolloin merkkijonoon tulee sulkeissa olevaa arvoa vastaava ASCII-merkki. Se voi olla myös merkkijono - jotkin funktiot palauttavat merkkijonon.

Rivinvaihtoon on käytettävä CRLF:ää eli tässä tapauksessa "\(13)\(10)", mikä on hieman hankalaa. Värejä voi tulostuksessa käyttää ESCAPE-koodin avulla, kuten normaalistikin ANSI:n mukaan. ESCAPE-koodin saa merkkijonoon laittamalla sinne "\(27)".

Sulkeiden sisällä voi olla myös kokonainen lause, jonka arvo sisällytetään merkkijonoon. Lause voidaan suorittaa käynnistysvaiheessa tai joka kerta uudelleen. Mikäli se halutaan suoritettavaksi vain kerran, sitä tulee edeltää avainsana "PARSE". Lauseen arvo voi olla myös merkkijono, jolloin se liittyy varsinaiseen merkkijonoon siihen kohtaan, jossa lause on.

1.56 Ohjelmointi menukielellä

Menukielellä voi tehdä miltei samat asiat kuin Rexx-ohjelmassa. Jotkin asiat ovat helpommin tehtävissä, jotkin vaikeammin ja jotkut ovat jopa mahdottomia. Menukieli sopii lyhyiden koodien tekemiseen ja yksinkertaisten komentojen implementointiin. Vähänkään monimutkaisemmat toiminnot kannattaa tehdä Rexx-kielellä. Rexx-ohjelman voi helposti ajaa menukoodista dos()-funktiolla. Rexx() ei ole tähän tehtävään oikea funktio, koska se lähettää Rexx-komennon ohjelmalle, eikä aja mitään ohjelmaa.

Komentojen nimeämiselle ja lyhennyksille kannattaa omaksua jokin tyyli, joka toistuu läpi valikoiden. Komentojen nimet voivat esimerkiksi olla yksittäisiä verbejä tai sitten aina tekemistä ja sen kohdetta kuvaava predikaatin ja objektin yhdistelmä, jonka lyhenne tulee aina kummastakin sanasta.

Useimmiten komennot kannattaa tehdä tiettyyn valikkoon. Hirveää määrää globaaleja komentoja tulee välttää. Usein komennot liittyvät esimerkiksi viesteihin tai tiedostoihin, joten niiden sijoitusvalikko on selvä. Itselläni on joukko yleisiä komentoja, jotka eivät liity mihinkään erityisesti, joten tein niille oman Extras-valikon. Minulla on komentoja esimerkiksi koneen muistin ja prosessilistan tutkailuun sekä ajan kyselyyn ja ohjelmiston versionumeron tiedusteluun. Lähdekoodeja näihin on viimeisessä luvussa.

Voi myös olla, että teet jonkin erityisen palvelun purkkiisi. Tällöin on kannattavaa tehdä sille oma valikko. Minulla on Saku-valikko, jossa voi lukea linjalla aina uusimman Sakun artikkeleita sekä imeä niitä yksittäin. Tällaisia toimintoja varten on yleensä syytä tehdä oma valikko. Ihan muutamaa komentoa varten ei niin kuitenkaan kannata tehdä.

Tässä vaiheessa on jo syytä antaa vähän muutakin kuin irrallisia esimerkkejä. Tässä on esimerkki oman valikon käynnistämiseen:

```
Command "extras" (N:1) {
    rexx("LogEntry Extras");
    menu("Extras.menu");
};
```

Tämä on yksinkertainen komento, jossa vaihdetaan valikkoa. Komennon ajami-

seen riitää ensimmäinen merkki, E. Tässä on ensimmäinen kunnollinen komennon lähdekoodi tuosta Extras-valikosta:

```
Command "lc" {
    rexx("LogEntry Viewing last callers");
    rexx("SendASCII Text/LastCallers.txt");
    rexx("SendModem \(13)\(10)");
};
```

Janne Sirenin LastCallers-ohjelma, jonka käyttämisen kuvasin artikkelin edellisessä osassa, ylläpitää listaa viimeisistä soittajista tiedostossa Text/LastCallers.txt. Komennolla lc käyttäjä voi nyt katsoa edelliset soittajat myös purkissa erittäin yksinkertaisella koodilla. Kaikkien komentojen on syytä lopuksi lähettää rivinvaihto eli CRLF, koodit 13 ja 10.

Käyttäjästä voisi olla mukavaa saada jotenkin apua hankalampien komentojen käyttämisessä. Yksi mahdollisuus on tehdä Help-komento, jolle annetaan komennon nimi, josta halutaan apua. Tällainen tosin kannattaisi tehdä Rexx-kielellä. Ohjelma voisi käyttää kullekin valikolle vaikkapa helppitiedostoja, joista se tulostaisi halutun komennon osuuden rivipohjaisesti. Toinen tapa on antaa lyhyesti tietoa, kun komennolle annetaan parametriksi kysymysmerkki, vastaavasti kuin shellissä. Itselläni jokainen komento antaa kysymysmerkillä formaattinsa, mutta apu voi olla muutakin:

```
Command "jokainenkomento" {
    if(arg(1) == "?") {
        rexx("SendModem <Jonkinlaista apua>\(13)\(10)");
        break;
    }
    ...
};
```

Järjestelmän toiminnasta kertovia komentoja saa helposti aikaiseksi useita. Tässä on yksi:

```
Command "sys" {
    rexx("LogEntry Showing system information");
    dos("CPU >TECHIO:\(node())");
};
```

Käyttäjälle ajetaan komento CPU, jonka tuloste ohjataan TechIO:n avulla modeemille. Tuloste voi olla esimerkiksi:

```
System: 68030 68882 FastROM (INST: Cache Burst) (DATA: Cache NoBurst)
```

Moni käyttäjä ei ehkä ole tällaisesta tiedosta kiinnostunut, mutta heitä kuitenkin on. Eri asia on, haluaako tästä tehtävän lokimerkintää. Minulla tehdään varsin monipuoliset lokimerkinnät, mutta joku muu voi haluta vähemmän merkintöjä. Minulla ei käyttäjän tarvitse kuin tulla sisään ja katsella ympärilleen kymmenisen minuuttia, niin loki kasvaa kilokaupalla... Komento VER kertoo ohjelmiston versionumeroita:

```
Command "version" (N:3 G:) {
    rexx("LogEntry Inquiring software version");
    dos("Version >TECHIO:\(node())");
    dos("Version BBS:Bin/TechCon >TECHIO:\(node())");
    dos("Version BBS:Bin/StarTech >TECHIO:\(node())");
```

```
};
```

Tuota viimeistä ei tietenkään kannata ajaa, jos käynnistää nodensa jollakin muulla ohjelmalla. Tällöin komennon voi korvata asianmukaisella komennolla, esimerkiksi "Version Bin:TrapDoor", jolloin komennon tuloste voisi olla:

```
Kickstart 40.9, Workbench 40.6
TechCon 0.93
TrapDoor 4.279
```

Ehkei tuo olekaan hyvä idea. Tietääkseni TrapDoorini versio on 1.83 - jostakin syystä se kuitenkin kertoo Version-komennolle ihan muuta. Valikkoon on muistettava laittaa mahdollisuus palata päävalikkoon:

```
Command "quit" {
    Menu("Main.menu");
};
```

1.57 Lisää TechnoBBS-komentoja ja -funktioita

Tässä luvussa esittelen tukun tehokkaita TechnoBBS:n komentoja ja funktioita. Yksityiskohtaisessa käsittelyssä ovat komennot ListFiles ja SelectFiles. Seuraavassa luvussa on valmiita menukoodeja uusien komentojen aikaansaamiseksi. Niissä hyödynnetään tässä luvussa esiteltäviä komentoja ja funktioita.

ListFiles on erittäin tehokas tiedostojenlistauskomento. Sillä voi listata kaikki, tietyn ajankohdan jälkeen upitut tai kuvaukseen osuvat tiedostot kaikilta tai tietyltä alueelta tai yhdeltä alueelta ja kaikilta sen ala-alueilta joko erikseen tai yhdessä vanhimmasta uusimpaan, uusimmasta vanhimpaan, aakkosjärjestyksessä tai käänteisessä aakkosjärjestyksessä.

Tässä ovat komennon optiot:

```
        A       Kaikki tiedostot kaikilta alueilta
        E       Kaikki tiedostot tällä alueella ja sen ala-alueilla
        C       Jatkuva listaus, ei promptia
        B       Ei ala-alueita
        I       Ei hakemistoja, kaikkien alueiden tiedostot samassa listassa
        N       Ei tyhjiä hakemistoja
        M       Näytä tiedostojen numerot
        L       Näytä pitkät kuvaukset
        G       Kokonaiset aluenimet (todelliset hakemistopolut)
```

Näiden optioiden lisäksi on viisi optiota, joille annetaan myös parametri:

```
        O <formatointimerkkijono>
        P <pattern>
        F <string>
        D <date>
        S <method>
```

Komennon tulosteen voi formatoida haluamallaan tavalla. Optioon O liitetyssä formatointimerkkijonossa voi käyttää seuraavia määrittelyjä:

```
%n      tiedoston nimi
%s      tiedoston pituus
%c      kommentti
%d      päiväys
%t      aika
%a      päivä
%p      protection-bitit
```

Tiedostoja voi myös etsiä komennon avulla. Tiedostoja voi listata määrittelemällä patternin (optio P), johon niitä verrataan. Voi myös antaa merkkijonon, jota etsitään tiedoston nimestä ja kuvauksesta (ei kuitenkaan pitkästä kuvauksesta) ja listataan ne, joista se löytyi (optio F). Optiolla D voit antaa päiväyksen, josta lähtien tiedostot listataan. Päiväys voi olla joko standardissa dd-mmm-yy -muodossa tai päivien määrä tammikuun ensimmäisestä päivästä vuonna 1978 (TechnoBBS:n yleinen päivämääräformaatti).

Optiolla S (sort) voit valita, miten tiedostot listataan:

```
N       Uusimmasta vanhimpaan
O       Vanhimmasta uusimpaan
A       Aakkosjärjestyksessä (suositeltavin tapa)
Z       Käänteisessä aakkosjärjestyksessä
```

Seuraavassa luvussa on esimerkki ListFiles-komennon implementoinnista purkkiin. Annan suoraan koko lähdekoodin omassa purkissani olevaan List-komentoon. Sillä voi myös käynnistää kokoruututiedostolistaimen, joka aktivoituu komennolla SelectFiles. Myös se on hyvin tehokas tiedostojenlistauskomento ja pystyy samaan kuin ListFiles, mutta se on interaktiivinen, ja sen avulla voi seikkailla alueelta toiselle ja myös imeä tiedostoja ja merkitä niitä.

SelectFilesin optiot poikkeavat hieman ListFilesin optioista:

```
A       Kaikki tiedostot, kaikki ala-alueet
E       Kaikki tiedostot tällä alueella ja sen ala-alueilla
H       Näytä nykyisen tiedoston nimi vahvennetulla (katso alla)
C       Näytä ala-alueet ja salli alueen vaihto (korvaa E:n)
```

Kokoruututiedostonvalitsin toimii niin, että se näyttää ruudullisen tiedostoja, ja listauksessa voi liikkua nuolinäppäimillä. Normaalisti kursorin paikka ilmaistaan ">"-merkillä vasemmassa reunassa, mutta sen sijaan voidaan käyttää nimen vahventamista (optio H), mikä ei kuitenkaan ole suotavaa. Optiot A ja E näyttävät tiedostot samassa listauksessa. Paras vaihtoehto on optio C, joka näyttää tiedostot alueittain ja antaa mahdollisuuden vaihtaa aluetta.

Normaalien Download- ja Upload-komentojen lisäksi TechnoBBS:ssä on mahdollisuus siirtää tiedostoja yhtä aikaa kumpaankin suuntaan. Sitä varten on olemassa komento DlAndReceive. Sillä voi imeä ja uppia yhtä aikaa. Ensin imettävät tiedostot tulee merkitä ja sen jälkeen vaihtaa alueelle, jolle uppii. Komennon käyttäminen edellyttää kaksisuuntaista protokollaa. Sellainen on ainakin DModem, joka on myös yksisuuntaisena nopeampi kuin ZModem. Jostakin syystä vain harvat purkit tukevat sitä. Esimerkki DModemin alustusmerkkijonoksi on "XY,LY,IY,B2048".

Lisäksi tiedostoja voi siirtää funktioilla SendFiles() ja ReceiveFiles(). ReceiveFiles():iä käytetään esimerkiksi vastaanotettaessa replypakettia. SendFiles() taas on käytössä tekstikirjastosovelluksessani, jonka lähdekoo-

di on seuraavassa luvussa. Nodella on lähetettävistä tiedostoista lista, jonka voi tyhjentää komennolla ClearFiles. Tämän jälkeen lähetettävät tiedostot lisätään listaan yksitellen komennolla AddFile. Lopuksi ne lähetetään kutsumalla SendFiles():iä, joka palauttaa nollan, jos siirto epäonnistui.

Viestipuolella on monia kiinnostavia komentoja. Yksi niistä on ReWriteMsg. Sen avulla voidaan toteuttaa BBBS:n dup-komentoa vastaava komento. ReWriteMsg <msg> lataa määritellyn viestin editoriin, jolloin sitä voidaan muutella ja tallentaa siitä uusi versio. Tämä uusi versio jää voimaan ja vanha tuhotaan. Viesti voidaan tuhota komennolla KillMsg. Viestejä ei kuitenkaan koskaan varsinaisesti poisteta viestikannasta, koska se olisi liian hidasta, vaan ne ainoastaan merkitään kuolleiksi.

Ihmettelen kyllä, miksei sitten ole olemassa komentoa UnkillMsg, koska viestin herättäminen kuolleista olisi erittäin yksinkertaista. Olen kuullut, että viestejä tapetaan helposti vahingossa, joten tällainen olisi tarpeen. ReWriteMsg edellyttää käyttäjältä viestineditointioikeuksia, jos viesti ei ole hänen itsensä kirjoittama. KillMsg ei tee tarkistuksia, joten on syytä itse varoa antamasta kaikkien käyttää sitä.

ConvertMsg <msg> <set> konvertoi viestikannassa olevan viestin annetusta merkistöstä ISO:ksi. Tämä on kätevä toiminto, jos esimerkiksi käyttäjä on uppinut replynsä väärällä merkistöllä. Tällöin tosin inbound-konversiossa merkit ovat saattaneet kääntyä väärin. TechnoBBS tarjoaa erittäin hyvät merkistökonversiotoiminnot. Hankalinta onkin saada käyttäjät valitsemaan oikea merkistö!

Komennolla EditMsg <msg> pääsee editoimaan viestin statusbittejä. Se vastaa E:n painamista viestejä lukiessa. Bittieditorissa voi esimerkiksi tehdä viestistä yksityisen. Suositeltava tapa viestien lukemiseen on ReadNext. Se aloittaa viestien lukemisen ensimmäisestä lukemattomasta viestistä ja lukee ne loppuun kyseisellä alueella. Toinen vaihtoehto on ReadAllNew, joka toimii muuten samoin, mutta lukee uudet viestit kaikilta alueilta.

Vielä yksi komento on syytä ottaa puheeksi. ShowMessage <text> näyttää käyttäjälle lyhyen tiedonannon. Sen perään tulostetaan prompti ja odotetaan näppäimenpainallusta. Tämä komento on tarkoitettu käytettäväksi noden laukaisemissa asynkronisissa ohjelmissa. Esimerkiksi TechQWK ja TechWWF käyttävät ShowMessagea kertoessaan, että viestipaketti on valmis.

Funktioiden puolella tulee ensiksi kiinnostavista vastaan GetDateVal(). Se palauttaa systeemikellon ajan päivien määränä tammikuun ensimmäisestä päivästä vuonna 1978. Tämä on TechnoBBS:n käyttämä päivämääräformaatti. Tiedostoalueista saa tietoa funktioilla GetLowMsg(), GetHighMsg() ja GetHighRead(). Funktiot palauttavat alueen ensimmäisen ja viimeisen viestin numeron sekä viimeisen luetun viestin numeron. Näitä käytetään useissa seuraavassa luvussa esiteltävissä komennoissa.

Viestialueen ja alueryhmän numeron saa selville funktioilla GetMsgArea() ja CurrentSIG(), alueen nimen funktiolla GetMsgAreaName(). SIG:n nimen pitäisi selvitä funktiolla SIGName(), mutta se ei toimi. Ainakin minulla se palauttaa aina ensimmäisen SIG:n nimen, joten siitä ei ole hyötyä. Käyttäjän oikeudet jollekin alueelle voi tarkistaa funktiolla HasMsgAcc(). Se palauttaa arvon yksi, jos käyttäjällä on pääsy alueelle.

Viimeinen tarkastelun kohteeksi pääsevä funktio on SIGtoReal(). Se muuttaa

annetun viestialueen numeron todelliseksi aluenumeroksi. Sille annetaan alueen numero SIG:n sisällä, ja se palauttaa numeron, joka vastaa msga<n>.dat- sekä muiden tiedostojen numerointia. Jotkin komennot haluavat parametrinään alueen todellisen numeron. Tällainen on esimerkiksi MoveMsg. Tästä komennosta on lisää tietoa seuraavassa luvussa, jossa käsittelen lähemmin myös muutamia muita komentoja.

1.58 Uusia komentoja

Tässä viimeisessä luvussa julkaisen useita valmiita komentoja ↩ vapaasti portattavaksi mihin tahansa purkkiin. Jos käytät tässä julkaistua koodia, kerro kaikille siitä!

Johdanto

MsgMenu: Move

MsgMenu: Scan

MsgMenu: SIG

MsgMenu: Convert

MsgMenu: Duplicate

Extras: PR, MEM, TIM, DS

Bulletins-valikko

Paranneltu etälukutuki

1.59 Johdanto

Aloitamme omalla versiollani tiedostojenlistauskomennosta. Komento korvaa alkuperäisen list-komennon.

```
Command "list" (N:1) {
    $sel = 0;
    $selmode = "-C";
    $sort = "-SA";
    $an = 1;
    $opt = "";
    if(arg(1) != "") while(arg($an) != "") {
        $curr = upper(arg($an));
        $temp = "";
        if($curr == "ALL") {
            $temp = "-E";
            $selmode = "-E";
        }else if($curr == "SELECTOR") $sel = 1;
```

```
        else if($curr == "FILES") $temp = "-I";
        else if($curr == "DAYS") {
            $an = $an + 1;
            $temp = "-D\(GetDateVal() - arg($an))"
        } else if($curr == "SINCE") {
            $an = $an + 1;
            $temp = "-D\(arg($an))"
        } else if($curr == "NEWFILES") {
            $temp = "-D\(GetUserMisc("LASTFSCAN"))";
            SetUserMisc("LASTFSCAN",GetDateVal());
        } else if($curr == "ALPHA") $sort = "-SA";
        else if($curr == "REVERSE") $sort = "-SZ";
        else if($curr == "CRON") $sort = "-SO";
        else if($curr == "REVCRON") $sort = "-SN";
        else if($curr == "FIND") {
            $an = $an + 1;
            $temp = "-F'\(arg($an))'"
        } else $temp = "-P\($curr)";
        if($temp != "") $opt = "\($opt) \($temp)";
        $an = $an + 1;
    }
    if($sel) {
        rexx("LogEntry Selecting");
        rexx("SelectFiles \($selmode) -H \($opt) \($sort)");
    } else {
        rexx("LogEntry Listing");
        rexx("ListFiles -G -M -L \($opt) \($sort)");
    };
};
```

Komennon formaatti on tällainen:

```
LIST [ALL] [FILES] [NEWFILES] [DAYS <n>] [SINCE <date>] [SELECTOR]
     [ALPHA|REVERSE|CRON|REVCRON] [FIND <string>] [<Pattern>]
```

Tämä on epäilemättä eräs tehokkaimmista ja monimutkaisimmista list-komennoista kautta purkkimaailman. Komennolla voi tehdä melkein mitä vain - listata tiedostoja, etsiä niitä, imeä niitä, ottaa newfiles-listauksen jne. Tässä on selvitys komennon optioista:

ALL	Kaikki tiedostot kaikilta alueilta
FILES	Vain tiedostot (ALL-option kanssa myös ala-alueiden)
NEWFILES	Uudet tiedostot (viime filescanin jälkeen upitut)
DAYS <n>	N päivää sitten ja sen jälkeen upitut tiedostot
SINCE <date>	Sama juttu, toimii aivan kuin DOS-Listissä
SELECTOR	Käytä tiedostojen näyttämiseen FSE-valitsinta
ALPHA	Näytä tiedostot aakkosjärjestyksessä (oletus)
REVERSE	Käänteinen aakkosjärjestys
CRON	Aikajärjestys
REVCRON	Käänteinen aikajärjestys
FIND <string>	Vain tiedostot, joissa on annettu merkkijono
<Pattern>	Pattern, johon täsmäävät tiedostot listataan

FIND-option yhteydessä annettua merkkijonoa siis etsitään tiedoston nimestä ja lyhyestä kuvauksesta - ei tiedostosta itsestään. Merkkijonossa ei saa olla välilyöntejä. Optiolla ALL listataan kaikkien alueiden tiedostot aluekohtaisesti. FILES-optiolla listataan vain tiedostot, eikä ala-alueita

näytetä ollenkaan. Yhdessä käytettynä ALL- ja FILES-optiot saavat aikaan sen, että kaikkien alueiden tiedostot näytetään samassa listassa! Kokoruutuvalitsinta käytettäessä tähän riittää pelkkä ALL-optio - itse asiassa sen kanssa FILES-optiolla ei ole merkitystä, koska SelectFiles ei tue I-optiota.

Virheellisistä tai tunnistamattomista optioista ei huomauteta, vaan ne ohitetaan olankohautuksella. Parametrillisten optioiden parametrien olemassaoloa ei tarkisteta, mutta en usko TechnoBBS:n siihen kaatuvan, vaikka jonkin option parametri puuttuisikin, mutta tietysti käyttäjän tulee varmistua siitä, että kaikki parametrit annetaan. BBS:ni on "high-end", joten myös käyttäjien tulee olla asiantuntevia!

Muutama esimerkki kertoo vähintään yhtä paljon kuin 1024 sanaa:

```
list all files newfiles
```

Näyttää uudet tiedostot samassa listassa (files-optio) kaikilta alueilta (all). Itse asiassa kaikki alueet ei ole absoluuttinen käsite. Kaikilla alueilla tarkoitetaan sitä aluetta, jolla käyttäjä on sekä sen ala-alueita. ListFiles- tai SelectFiles-komennolle (kumpaa käytetään, ratkeaa siitä, onko komentorivillä optio SELECTOR) annetaan all-option ollessa määritelty optio E. Optiota A ei tämä komento koskaan käytä.

```
list days 30 cron
```

Listaa viimeisen kuukauden aikana upitut tiedostot aikajärjestyksessä.

```
list all find Tech
```

Listaa kaikki tiedostot, joiden nimessä tai lyhyessä kuvauksessa esiintyy sananpätkä "Tech". Tiedostoja etsitään kaikilta alueilta ja ne näytetään aluekohtaisesti aakkosjärjestyksessä.

```
list all files since 01-Jan-95
```

Voiko enää inhimillisemmäksi BBS:n käyttäminen mennä? Kyllä voi, mutta sen käsittely jääköön toistaiseksi. Tämä komento vastaa itse asiassa täysin DOSin List-komennon toimintaa. Komento luettelee kaikki tällä alueella ja sen ala-alueilla olevat tiedostot, jotka on upittu tänä vuonna.

Olen kirjoittanut useimmat TechnoBBS:n komennot suurilta osin tai täysin uusiksi. Vakiona TechnoBBS:ään kuuluvat viestikomennot ovat aika sekavia. Jotkin käyttävät optioita, mutta ne ovat kryptisiä Unix-tyylisiä optioita. Tässä on esimerkiksi vielä viestivalikon list-komentoni:

```
Command "list" (N:1) {
  $area = GetMsgArea();
  if(GetHighRead($area) == GetHighMsg($area)) $defstart = GetLowMsg($area);
      else $defstart = GetHighRead($area) + 1;
  if(arg(1) == "") $start = AskInput("\(27)[32mFrom message? \(27)[0m",
      $defstart, 10, "NUMERIC");
      else $start = arg(1);
  if(arg(2) == "") $end = AskInput("\(27)[32mTo message? \(27)[0m",
      GetHighMsg(GetMsgArea()), 10, "NUMERIC");
      else $end = arg(2);
  rexx("SendModem \(13)\(10)";
```

```
  rexx("ListMessages \($start) \($end)");
};
```

Komento ottaa parametreikseen listauksen alun ja lopun. Mikäli niitä ei määritellä, ne kysytään. Oletuksena tarjotaan promptissa aloitusviestiksi ensimmäistä lukematonta viestiä tai, jos alueen kaikki viestit on luettu, ensimmäistä olemassa olevaa viestiä sekä lopetusviestiksi viimeistä viestiä. Samoja oletuksia tarjoaa myös esimerkiksi Find-komentoni. Lisäksi minulla on BBBS-tyylinen Mark Group -toiminto, joka käyttää samantapaista logiikkaa. En listaa enää muita vakiokomentoja, mutta niiden lähdekoodia saa minulta pyytämällä.

Siirtyminen viestivalikkoon ei ole niin helppoa kuin voisi luulla. Tässä on paranneltu versio tehtävän tekevästä komennosta:

```
Command "messages" (N:1) {
    rexx("LogEntry Messages");
    rexx("SelectSIG \(arg(1))");
    if(!CurrentSIG()) rexx("SelectSIG 1");
    rexx("SendModem \(12)");
    rexx("SelectSIGArea \(arg(2))");
    menu("Msg.menu");
};
```

Tällä komennolla varmistutaan siitä, että päästään viestivalikkoon, vaikka typerä käyttäjä painaisikin vain enteriä tajuamatta valita SIG:iä tai viestialuetta. Alkuperäisellä koodillahan silloin silloin takaisin päävalikkoon, mutta tämä koodi vie silloin ensimmäisen SIG:n ensimmäiselle viestialueelle, joka yleensä on postialue, tai sinne, missä viimeksi on oltu.

Viestit luetaan kätevästi painamalla enteriä, mutta miten olisi viestivalikkoon meneminen enterin painalluksella, kuten BBBS:ssä? Kas tässä:

```
Command "" {
    $sig = GetUserMisc("DEFAULTSIG");
    if($sig == "") $sig = 1;
    if(($sig < 1) | ($sig > 6) {
        rexx("SendModem\(27)[31mYour default SIG setting is invalid.
        \(13)\(10)\(13)\(10)\(27)[33m
        Using default default setting.\(13)\(10)");
        $sig = 1;
    }
    rexx("LogEntry Messages");
    rexx("SelectSIG \($sig)");
    rexx("SelectSIGArea 1");
    menu("Msg.menu");
};
```

Tämä tulee siis päävalikkoon, ja se vie käyttäjän viestivalikkoon painettaessa enteriä tyhjälle riville. Näin viestejä päästään lukemaan loginin jälkeen kahdella enterin painalluksella! Laita ennen kääntämistä ensimmäinen rexx()-funktiokutsu yhdelle riville.

Komento vaatinee hieman selvitystä. Aikaisemmin minulla oli siinä vain "SelectSIG 1", mutta nyt SIG:n numero otetaan käyttäjämuuttujasta DEFAULTSIG. Tälle tuli tarvetta, koska haluan yleensä päästä suoraan SakuNet-alueille, mikä näin onnistuu helposti laittamatta sitä kiinteäksi oletusarvoksi.

Riittää, että DEFAULTSIG:n arvo on 3, joka on SakuNetin SIG:n numero. Tämän asettamiseksi tarvitaan tietysti sopiva koodi esimerkiksi terminaaliasetus-valikkoon:

```
    des = GetUserMisc(ln, "DEFAULTSIG")
    select
        when des = 1 then des = "Yleiset alueet"
        when des = 3 then des = "SakuNet"
        when des = 6 then des = "WebNet"
        otherwise des = "<NONE>"
        end
    SendModem "12     "||CYAN||"Default SIG: "||WHITE||des||CRLF
```

Tämä koodi tulee sinne alkuun, missä tulostetaan olemassaolevat valinnat. Seuraava koodi tulee loppuun. Itselläni valinnan numero on 12, mutta siihen voi laittaa muutakin, mikä sitten sattuu olemaan ensimmäinen vapaa numero. SIG-koodeistani on tarkoituksella poistettu SIG:t 2, 4 ja 5.

```
        when cmdid = "12" then do
            dez = GetUserMisc(ln, "DEFAULTSIG")
            if dez = "" then dez = 1
            des = "<NONE>"
            do while des = "<NONE>"
                SendModem CRLF
                SendASCII "Text/DefaultSIG.txt"
                des = AskInput(ln, "Select: ", "", 2, "NUMERIC");
                select
                    when des = "1" then nop
                    when des = "2" then nop
                    when des = "3" then nop
                    when des = "4" then nop
                    when des = "5" then nop
                    when des = "6" then nop
                    when des = "" then des = dez
                    otherwise des = "<NONE>"
                    end
                end
```

Koodi toimii vastaavasti kuin esimerkiksi offlinerin valinta (tulee myöhemmin). Ovelalla manipuloinnilla saadaan käyttäjä valitsemaan oletusarvo hänen tietämättään. Tosin myös koodi, joka arvoa käyttää, osaa valita oletusoletuksen, jos arvo on väärä tai puuttuu. Ohjeeksi lähetetään tiedosto DefaultSIG.txt:

```
    The following SIGs are available in this system:

       1 Yleiset alueet
       3 SakuNet
       6 WebNet

Select one or press enter for no change or default selection.
```

1.60 MsgMenu: Move

Ensimmäinen varsinainen uusi komento tulee tässä. Komennolla voi siirtää viestin toiselle alueelle:

```
Command "move" (N:2) {
    if(GetUserAccess() < 1000) {
        rexx("SendModem \(27)[31mNo access\(13)\(10)");
        rexx("LogEntry Attempted to move a message");
        break;
    }
    $msg = arg(1);
    $area = arg(2);
    if($msg == "") {
        $msg = AskInput("Message number: ", "", 10, "NUMERIC");
    }
    if($area == "") {
        $area = AskInput("Destination area: ", "", 10, "NUMERIC");
        rexx("SendModem \(13)\(10)\(13)\(10)");
    }
    if($msg != "") if($area != "") {
        rexx("SendModem Moving message.\(13)\(10)");
        rexx("MoveMsg \($msg) \($area)");
        rexx("KillMsg \($msg)");
        rexx("LogEntry Moved message \($msg) from \(GetMsgAreaName(
            GetMsgArea())) to \(GetMsgAreaName($area)) as #\(
            GetHighMsg($area))\(13)\(10)");
    }
};
```

En tiedä, miten TechMenuComp suhtautuu tuolla tavalla jaettuihin riveihin, mutta uskoisin, että viimeinen rexx()-funktiokutsu kannattaa palauttaa yhdelle riville. Jaoin sen kahdelle riville taittoteknisistä syistä... Komento varmistaa, että käyttäjällä on SysOp-oikeudet (oikeustaso => 1000). Komennolle annetaan siirrettävän viestin numero ja alue, jolle viesti siirretään. Jos niitä ei anneta komentorivillä, ne kysytään erikseen.

Kohdealueen numero on tässä nyt alueen todellinen numero sellaisena kuin se msg<n>.dat-tiedoston nimessä on. Tätä voisi helpottaa käyttämällä SIGtoReal()-funktiota, mutta ainakin minulla on tarve siirtää joskus viestejä SIG:stä toiseen, joten tuollaista ei voi käyttää, koska siirto estyisi. Viestin siirtämisen jälkeen alkuperäinen tuhotaan, koska MoveMsg ei tee sitä itse. Itse asiassa se onkin toiminnaltaan pikemminkin CopyMsg...

Jälleen tehdään perusteellinen lokimerkintä... Alkuperäisen viestin numerolla ei tosin enää ole merkitystä, eikä uudenkaan viestin numero välttämättä ole aina oikein, mutta mahdollisuus sen väärin menemiseen on hyvin pieni (jos jokin toinen node ehtii välissä laittaa kohdealueelle toisen viestin).

1.61 MsgMenu: Scan

TechnoBBS:ssä ei ole minkäänlaista viestialueiden läpikäyntikomentoa. Esimerkiksi BBBS:ssä on hyvä show (conference status) -komento. Olen kirjoittanut vastaavan TechnoBBS:lle. Se ei ole yhtä monipuolinen, koska olisi käytännössä äärettömän hidasta ja miltei mahdotonta skannata lennossa, kuinka monta viestiä nimenomaiselle käyttäjälle on. Senhän saa selville

mc-komennolla, joten alueilla olevien viestien kokonäismäärä ja uusien viestien määrä riittänevät:

```
Command "scan" (N:2) {
    $curr = GetMsgArea();
    $signum = arg(1);
    $mode = arg(2);
    if($signum == "") $signum = CurrentSIG();
    if(upper(arg(1)) == "NEW") {
        $signum = CurrentSIG();
        $mode = "NEW";
    }
    if($signum == 6) {
        $sig = "WebNet";
        $area = 101;
        $max = 130;
    } else if($signum == 3) {
        $sig = "SakuNet";
        $area = 31;
        if(GetUserAccess() > 999) $max = 64; else $max = 61;
    } else {
        $sig = "Yleiset alueet";
        $area = 3;
        $max = 12;
    }
    rexx("SendModem \(27)[32mScanning message areas: \(
        $sig)\(13)\(10)\(13)\(10)");
    rexx("LogEntry Scanning SIG \($sig)");
    $hightotal = 0;
    $msgtotal = 0;
    while($area <= $max) {
        $highread = GetHighRead($area);
        $highmsg = GetHighMsg($area);
        $areaname = "\(GetMsgAreaName($area)):";
        if($highmsg) {
            $doarea = 1;
            if($mode != "") if($highmsg == $highread) $doarea = 0;
            if(!(GetAreaMode($area))) $doarea = 0;
            if($doarea) {
                rexx("SendModem \(27)[33m\(str($areaname,1,24))\(27)[0m\(
                    $highmsg - $highread) unread messages (\(
                    $highmsg) total, last read \($highread))\(13)\(10)");
                $hightotal = $hightotal + $highread;
                $msgtotal = $msgtotal + $highmsg;
            }
        }
        $area = $area + 1;
    }
    rexx("SendModem \(13)\(10)\(27)[33m\(str("Total:",1,24))\(27)[0m\(
        $msgtotal - $hightotal) unread messages (\(
        $msgtotal) total, read \($hightotal))\(13)\(10)");
};
```

Komento käy läpi halutun SIG:n alueet ja kertoo niistä tietoa. Tässä on esimerkkituloste:

```
AZ.Muut:                4 unread messages (61 total, last read 57)
```

```
AZ.Net:                 43 unread messages (282 total, last read 239)
AZ.SciFi & Trek:        2 unread messages (22 total, last read 20)
AZ.Tee-se-itse:         0 unread messages (26 total, last read 26)
AZ.TV & Elokuvat:       0 unread messages (50 total, last read 50)
AZ.SAKU/Yleinen:        5 unread messages (28 total, last read 23)
AZ.SAKU/Info:           0 unread messages (21 total, last read 21)
AZ.SAKU/Ohjelmointi:    0 unread messages (6 total, last read 6)
AZ.SAKU/Toimitus:       0 unread messages (3 total, last read 3)
AZ.SAKU/Yhdistys:       0 unread messages (6 total, last read 6)

Total:                  54 unread messages (505 total, read 451)
```

Komento ottaa parametrinään skannattavan SIG:n numeron. Jos sitä ei anneta, skannataan se SIG, jossa ollaan. Lisäksi komento tunnistaa option NEW, joka määrittelemällä näytetään vain alueet, joilla on uusia viestejä. Option NEW voi antaa myös ilman SIG-numeroa. Koodissa on hieman logiikkaa. Vain alueet, joille käyttäjä on liittynyt ja joilla on viestejä, näytetään.

SIG:n nimet on asetettava käsin, koska SIGName ei toimi. Lisäksi SIG-numeron vertailussa asetetaan asianmukaiset aluerajat. Huomaa SakuNetin tapauksessa vaihtoehtoinen loppumisalue. Lopussa on kolme aluetta, jotka eivät näy loppukäyttäjille, joten niitä ei myöskään näytetä skannatessa kuin SysOpille.

Janne Siren on tehnyt myös viestialuestatistiikkaa kertovan ohjelman. Olen virittänyt myös siihen SIG-pohjaisen toiminnan:

```
Command "mstats" (N:2 G:) {
    $signum = arg(1);
    if($signum == "") $signum = CurrentSIG();
    if($signum == 6) {
        $sig = "WebNet";
        $area = 101;
        $max = 130;
    } else if($signum == 3) {
        $sig = "SakuNet";
        $area = 31;
        if(GetMask()&1024) $max = 64; else $max = 61;
    } else {
        $sig = "Yleiset alueet";
        $area = 3;
        $max = 12;
    }
    rexx("SendModem \(27)[32mViewing message area statistics for SIG \(
        $sig)\(13)\(10)\(13)\(10)");
    dos("RX >NIL: BBS:Rexx/MessageStats.rexx \(node()) \($area) \($max)");
    rexx("LogEntry Examining message area statistics for SIG \($sig)");
};
```

Tämä vaatii muutoksia myös itse ohjelmaan. Poista MessageStats.rexxistä FirstArea- ja LastArea-määrittelyt ja korvaa ln-määrittely komennolla:

```
Parse ARG ln FirstArea LastArea .
```

Älä unohda lopussa olevaa pistettä. Näillä muutoksilla myös MessageStats käy läpi vain yhden SIG:n, mikä on järkevin toimintatapa.

1.62 MsgMenu: SIG

TechnoBBS:n mukana ei tule myöskään tarpeellista SIG:n vaihtokomentoa. Seuraavalla koodinpätkällä homma hoituu:

```
Command "sig" (N:1) {
    $sig = arg(1);
    $area = arg(2);
    if(($sig != "") && ($area == "")) $area = 1;
    rexx("SelectSIG \($sig)");
    if($area == "") {
        rexx("SendModem \(12)");
        rexx("SelectSIGArea");
    } else rexx("SelectSIGArea \($area)");
    rexx("LogEntry Message area: \(GetMsgAreaName(GetMsgArea()))");
};
```

Komennolle annetaan halutun SIG:n numero sekä viestialueen numero siellä. Jos näitä ei anneta, ne kysytään. Poikkeustapaus on kuitenkin se, että SIG:n numero on määritelty, mutta alueen ei. Silloin siirrytään suoraan alueelle yksi. Tämä on helppo tapa vaihtaa nopeasti alueryhmää.

1.63 MsgMenu: Convert

Joskus verkosta tulee viestejä, joiden merkistö on väärä. Näitä voi tulla myös käyttäjien replypaketeissa. Viestin voi kuitenkin merkistökonvertoida sen ollessa jo viestikannassa:

```
Command "convert" (N:4) {
    if(GetUserAccess() < 1000) {
        rexx("SendModem \(27)[31mNo access\(13)\(10)");
        rexx("LogEntry Attempted to convert a message");
        break;
    }
    $msg = arg(1);
    $set = arg(2);
    if($msg == "") $msg = AskInput(
        "\(27)[32mConvert which message: \(27)[0m", "", 10, "NUMERIC");
    if($set == "") $set = AskInput(
        "\(27)[32mFrom which charset? \(27)[0m", "", 10);
    if($msg != "") if($set != "") {
        rexx("SendModem \(27)[32m\(13)\(10)Converting message \(
            $msg) from \($set) to ISO\(13)\(10)");
        rexx("LogEntry Converting message \($msg) from \($set) to ISO");
        rexx("ConvertMsg \($msg) \($set)");
    } else rexx("SendModem \(27)[31m\(13)\(10)Required argument missing");
}
```

Komento ottaa parametrikseen konvertoitavan viestin numeron sekä merkistön, josta konvertoidaan. Yleensä lähdemerkistö on SF7 tai IBM. SF7:ssä olevassa viestissä näkyvät esimerkiksi pienet ä-kirjaimet avautuvina kaarisulkuina ja pienet ö-kirjaimet pystyviivoina. IBM-merkistöllä esimerkiksi ä-kirjaimesta tulee kursorin siirto alaspäin - TechnoBBS:llä IBM-merkistön havainnee niin, että skandimerkit puuttuvat kokonaan.

1.64 MsgMenu: Duplicate

BBBS:ssä on myös Duplicate-komento, jolla voidaan jo kirjoitettu viesti ottaa uudelleen editoitavaksi. TechnoBBS tarjoaa komennon ReWriteMsg, jota käyttämällä vastaavan komennon toteuttaminen siihen on erittäin helppoa:

```
Command "duplicate" (N:3) {
    $msg = arg(1);
    if($msg == "") $msg = AskInput(
        "\(27)[32mDuplicate which message: \(27)[0m", "", 10, "NUMERIC");
    if($msg != "") {
        rexx("LogEntry Duplicating message \($msg) as \(GetHighMsg(
            GetMsgArea()) + 1)");
        rexx("ReWriteMsg \($msg)");
    } else rexx(
        "SendModem \(27)[31m\(13)\(10)Operation cancelled\(13)\(10)");
}
```

Komennolle annetaan parametrinä dupattavan viestin numero. Viestin voi dupata vain sen kirjoittaja tai käyttäjä, jolla on viestineditointi- tai SysOp-oikeudet.

1.65 Extras: PR, MEM, TIM, DS

Laitan tähän nyt niitä lupaamiani Extras-valikon ylimääräisiä komentoja. Komennot ajavat jonkin järjestelmäkomennon ja kertovat käyttäjälle näin tietoa systeemistä.

```
Command "processes" (N:2) {
    rexx("LogEntry Listing processes");
    if(upper(arg(1)) == "ALL") dos("PS >T:pr_\(node())");
        else dos("Status >T:pr_\(node())");
    rexx("SendASCII T:pr_\(node())");
    dos("Delete T:pr_\(node())");
};
```

Tämä ohjelma lähettää käyttäjälle prosessilistan. Vaihtoehtoisesti käytetään Status-komentoa tai perusteellisempaa PS-komentoa, joka listaa myös Exec-tehtävät. Tulostus voi olla pitkä, joten se ohjataan temppitiedostoon, joka lähetetään SendASCII-komennolla. Näin listaukseen saadaan tarpeen vaatiessa More-prompti. Käytän temppitiedoston nimessä nodenumeroa, koska on olemassa mahdollisuus, että komento ajettaisiin yhtä aikaa useammalla kuin yhdellä nodella.

```
Command "memory" (N:3) {
    rexx("LogEntry Checking available memory");
    dos("Avail >TECHIO:\(node())");
};
```

Yksinkertainen komento MEM kertoo koneen muistitilanteen.

```
Command "time" (N:3 G:) {
    rexx("LogEntry Acquiring current time");
    dos("Date >TECHIO:\(node())");
```

```
};
```

Yhtä yksinkertainen TIM-komento kertoo oikean ajan.

```
Command "ds" {
    rexx("LogEntry Checking disk space");
    if(upper(arg(1)) == "FULL") dos("Inf >T:ds_\(node())");
        else dos("Info >T:ds_\(node())");
    rexx("SendASCII T:ds_\(node())");
    dos("Delete T:ds_\(node())");
};
```

Komento DS toimii samalla tavalla kuin PR. Se kertoo massamuistien tilan. Sitä tarkoitusta varten ajetaan Info-komento. Vaihtoehtoisesti ajetaan Inf-komento, joka kertoo enemmän tietoa kuin vakiokomento Info. Tiedot tulostetaan samaan tapaan kuin prosessilistakin - Inf tulostaa myös muistitilanteen ja Volume-luettelon, joten senkin tuloste voi olla yli sivun mittainen.

1.66 Bulletins-valikko

Tähän mennessä pitäisi jo olla selvää, että ihannoin BBBS:ää! No, ei aivan sentään, mutta BBBS:ssä on monia hyviä ominaisuuksia, ja niitä olen tuonut TechnoBBS:ään useita. Yksi näistä on tuki bulletiineille eli "tekstikirjastolle". Bulletiinit ovat tekstitiedostoja, joita voi lukea linjalla ja imeä yksitellen:

```
Command "bulletins" (N:4 G:) {
    rexx("LogEntry Bulletins");
    $read = arg(1);
    $bull = "?";
    while ($bull != "") {
        if($read == "") $bull = AskInput(
            "\(27)[32mBulletin number (? for list): \(27)[0m","",40);
            else {
                $bull = $read;
                $read = "";
            }
        if($bull == "?") $bull = "list";
        if(str($bull,1,1) == "q") $bull = "";
            else if($bull == "d") {
            rexx("ClearFiles");
            while($bull != "") {
                rexx("SendModem \(13)\(10)");
                $bull = AskInput(
                    "\(27)[32mBulletin number to download: \(27)[0m","",40);
                if($bull != "") {
                    rexx("LogEntry Downloading bulletin \($bull)");
                    rexx("AddFile BBS:Bulletins/\($bull)");
                }
            };
            $dl = 1;
            while($dl) {
                if(SendFiles()) $dl = 0; else {
                    rexx("SendModem \(13)\(10)\(13)\(10)\(13)\(10)\(27)[31m
```

```
                        Transfer failed.\(13)\(10)\(13)\(10)");
                    if(!(GetYesNo("\(27)[36mSend again? \(27)[0m",1,1)))
                        $dl = 0;
                }
            }
        } else if($bull != "") {
            rexx("LogEntry Reading bulletin \($bull)");
            rexx("SendModem \(12)");
            rexx("SendASCII BBS:Bulletins/\($bull)");
            rexx("SendModem \(13)\(10)");
        };
    }
};
```

Tämä komento käyttää hyväkseen ClearFiles/AddFile/SendFiles()-toimintosarjaa. Olen ajatellut kirjoittaa koodin uusiksi Rexx-kielellä, koska nykyisellään bulletiineja ei voi konvertoida. Online-Saku-lukijassani olisi valmis konvertointikoodi sekä pakkaus. Ne voisi helposti kopioida myös bulletiinitukeen, jolloin myös imettävät bulletiinit voisi konvertoida oikealle merkistölle sekä pakata ennen siirtoa.

Komennolle voi antaa parametriksi bulletiinin numeron, joka luetaan välittömästi, minkä jälkeen tulostetaan prompti. Muuten sen saa heti. Parametriksi voi antaa bulletiinin numeron tai komentoja. Tällä hetkellä hyväksyttäviä komentoja ovat d, jolla voi imeä bulletiineja, tai ?, joka tulostaa bulletiinilistan, sekä q, jolla pääsee takaisin päävalikkoon, mikä onnistuu myös enterin painamisella tyhjälle riville.

Bulletiinit pidetään hakemistossa BBS:Bulletins. Bulletiinilista on tiedostossa BBS:Bulletins/list. Kun promptiin vastataan kysymysmerkillä, se muutetaan "list":ksi, jolloin lähetetään tuo lista. Käyttäjälle lähetetään suoraan sen niminen tiedosto, minkä hän on promptiin kirjoittanut. Lisäksi siitä tehdään lokimerkintä. Listan tapauksessa tiedoston nimi on "list", joten lokientryksi tulee näppärästi Reading bulletin list... Tässä on esimerkiksi oma bulletiinilistani:

```
       Bulletin List

    0   Tietoa bulletiineista
    1   Uploading Instructions
    2   Star Fleet Top Callers
    3   Star Fleet Top Message Writers
    4   Star Fleet Top Uploaders
    5   Star Fleet Top Downloaders

       Alaryhmät

    20  SAKU
    30  Amigan tulevaisuus
    40  Amigan laitteisto ja ohjelmisto
    60  Tee-se-itse
    80  Sekalaiset
    100 Star Trek

    enter to quit bulletins, cls to clear screen, d to download
```

Lopussa oleva ylimääräinen ohje on syytä pitää mukana, koska noita tietoja

ei anneta muualla. Bulletiinit ovat hyvä paikka pitää TopTechin tuottamat top-listat. Niiden generointi lennossa on hidasta ja epäsuotavaa, koska TopTech voi sekoittaa TechConin toimintaa. Alaryhmien tekeminen on myös mahdollista. Bulletiinien ei ole pakko olla numeroituja. Niiden nimenä voi olla myös jokin sana, mutta sanavälejä siinä ei ole suotavaa olla.

Alaryhmien hakemistot pidetään vastaavissa tiedostoissa, esimerkiksi SAKU-bulletinien hakemisto on tiedostossa BBS:Bulletins/20:

```
       Bulletin List - SAKU

    21  Amiga-käyttäjät perustavat rekisteröidyn yhdistyksen
    22  Uusi HTML-formaatin SAKU-lukija, ideoijana Jukka Aho (30k)
    23  SAKU-kokous IRC:ssä
    24  Seuraavan Sakun uusin tilannekatsaus
    25  Esa Heikkinen perustaa SAKU-netin!
    26  SAKU-netin nodelista

    enter to quit bulletins, cls to clear screen, d to download
```

Ohje on aina syytä laittaa kaikkiin hakemistoihin. Varsinaiset bulletiinit ovat tietysti tiedostonimiltään BBS:Bulletins/21, BBS:Bulletins/22 ja niin edelleen. Olen nähnyt vastaavan palvelun ainakin Amiga Zonessa, ja se on minun mielestäni erittäin kätevä tapa välittää tekstimuotoista tietoa.

1.67 Paranneltu etälukutuki

Nykyään eräs tärkeimmistä BBS:n palveluista on viestien etäluvun järjestäminen. TechnoBBS tukee QWK:ta ja WWF:ää ja tarjoaa hyvät mahdollisuudet etälukuun, mutta BA/BW/QU/WU-sarja on aivan liian mahdoton muistettavaksi. Näin ollen olen kirjoittanut kaksi komentoa, grab ja put. Grabilla otetaan viestit ja putilla laitetaan vastaukset. Komennot sisältävät alkuperäisten komentojen koodin, mutta ne on ympäröity uudenlaisella liittymällä:

```
Command "grab" (G:) {
    $format = GetUserMisc("OFFLINER");
    if($format == "") {
        rexx("SendModem \(13)\(10)\(27)[31mThere is no Offliner defined.
            \(13)\(10)\(13)\(10)");
        $resp = GetYesNo("\(27)[0mDo you want to use WWF (or QWK) ", 1, 1);
        if($resp == 1) $format = "WWF";
            else $format = "QWK";
        SetUserMisc("OFFLINER",$format);
    }
    if(lower(arg(1)) == "a") SetUserMisc("OFFLINERMODE", 1);
        else if(lower(arg(1)) == "s") SetUserMisc("OFFLINERMODE", 0);
    if(GetUserMisc("OFFLINERMODE") == "") {
        rexx("SendModem \(13)\(10)\(27)
            [31mYou have no offliner operation mode defined. Using default.
            \(13)\(10)\(13)\(10)");
        SetUserMisc("OFFLINERMODE",0);
        SetUserMisc("WWFASYNC",0);
        SetUserMisc("QWKASYNC",0);
    }
```

```
    if($format == "QWK") {
        rexx("SendModem \(13)\(10)\(27)[33mPacking QWK messages");
        if(GetUserMisc("OFFLINERMODE")) {
            rexx("LogEntry Packing QWK messages asynchronously");
            rexx("SendModem  asynchronously, you will be informed on
                completion\(13)\(10).\(13)\(10)\(27)[0m");
            dos("Run <>NIL: TechQWK \(node()) O BBS:Cfg/TechQWK.Cfg");
        } else {
            rexx("LogEntry Packing QWK messages");
            rexx("SendModem .\(13)\(10)\(13)\(10)");
            dos("TechQWK \(node()) O BBS:Cfg/TechQWK.Cfg");
            rexx("SendModem \(13)\(10)");
            GetHotkey("\(27)[32mPress any key to start sending...\(27)[0m");
            rexx("SendModem \(13)\(10)\(13)\(10)");
            rexx("MarkAnyFile \(GetUserPath(GetUserName()))/SFL.QWK");
            rexx("Download MARKEDONLY");
        }
    } else {
        rexx("SendModem \(13)\(10)\(27)[33mPacking WWF messages");
        if(GetUserMisc("OFFLINERMODE")) {
            rexx("LogEntry Packing WWF messages asynchronously");
            rexx("SendModem  asynchronously, you will be informed on
                completion\(13)\(10)\(13)\(10)\(27)[0m");
            dos("Run <>NIL: TechWWF \(node()) O BBS:Cfg/TechWWF.Cfg");
        } else {
            rexx("LogEntry Packing WWF messages");
            rexx("SendModem .\(13)\(10)\(13)\(10)");
            dos("TechWWF \(node()) O BBS:Cfg/TechWWF.Cfg");
            rexx("SendModem \(13)\(10)");
            GetHotkey("\(27)[32mPress any key to start sending...\(27)[0m");
            rexx("SendModem \(13)\(10)\(13)\(10)");
            rexx("MarkAnyFile \(GetUserPath(GetUserName()))/SFL.WWF");
            rexx("Download MARKEDONLY");
        }
    }
};

Command "put" (G:) {
    $format = GetUserMisc("OFFLINER");
    if($format == "") {
        rexx("SendModem \(13)\(10)\(27)[31mThere is no Offliner defined.
            \(13)\(10)\(13)\(10)");
        $resp = GetYesNo("\(27)[0mAre you using WWF (or QWK) ", 1, 1);
        if($resp == 1) $format = "WWF";
            else $format = "QWK";
        SetUserMisc("OFFLINER",$format);
    }
    if(lower(arg(1)) == "a") SetUserMisc("OFFLINERMODE", 1);
    else if(lower(arg(1)) == "s") SetUserMisc("OFFLINERMODE", 0);
    if(GetUserMisc("OFFLINERMODE") == "") {
        rexx("SendModem \(13)\(10)\(27)[31mYou have no offliner operation
            mode defined. Using default.\(13)\(10)\(13)\(10)");
        SetUserMisc("OFFLINERMODE",0);
        SetUserMisc("WWFASYNC",0);
        SetUserMisc("QWKASYNC",0);
    }
    if($format == "QWK") {
```

```
        rexx("SendModem \(13)\(10)\(27)[33mUploading QWK messages.
            \(13)\(10)\(13)\(10)");
        rexx("LogEntry Uploading QWK messages");
        dos("Delete \(GetUserPath(GetUserName()))/SFL.REP quiet");
        rexx("SendModem \(13)\(10)\(27)[33mStart sending SFL.REP
            \(13)\(10)\(13)\(10)\(27)[0m");
        if(ReceiveFiles()) {
            if(GetUserMisc("OFFLINERMODE")) {
                dos("Run >NIL: TechQWK \(node()) I BBS:Cfg/TechQWK.Cfg");
                rexx("SendModem \(27)[33mProcessing replies asynchronously
                    \(13)\(10)\(27)[0m");
            } else dos("TechQWK \(node()) I BBS:Cfg/TechQWK.Cfg");
        } else rexx("SendModem \(13)\(10)\(13)\(10)\(27)[31mTransfer failed
            \(13)\(10)\(27)[0m");
    } else {
        rexx("SendModem \(13)\(10)\(27)[33mUploading WWF messages.
            \(13)\(10)\(13)\(10)");
        rexx("LogEntry Uploading WWF messages");
        dos("delete \(GetUserPath(GetUserName()))/SFL.RRF quiet");
        rexx("\(13)\(10)\(27)[33mStart sending SFL.RRF
            \(13)\(10)\(13)\(10)\(27)[0m");
        if(ReceiveFiles()) {
            if(GetUserMisc("OFFLINERMODE")) {
                dos("Run >NIL: TechWWF \(node()) I BBS:Cfg/TechWWF.Cfg");
                rexx("SendModem \(27)[33mProcessing replies asynchronously
                    \(13)\(10)\(27)[0m");
            } else dos("TechWWF \(node()) I BBS:Cfg/TechWWF.Cfg");
        } else rexx("SendModem \(13)\(10)\(13)\(10)\(27)[31mTransfer failed
            \(13)\(10)\(27)[0m");
    }
};
```

Tämä koodi meni varsin risaiseksi, koska jouduin jakamaan useita rivejä. Jos haluat hyödyntää sitä, parasta lienee pyytää minulta suoraan kunnollinen versio. Komennoissa on paljon päällekkäistä koodia, joten tuota voisi parannella vielä. Ideana on, että viestien ottaminen ja laittaminen onnistuu samalla komennolla riippumatta siitä, mitä formaattia käyttää. Käytän käyttäjämuuttujaa OFFLINER välittämään tiedon siitä, missä formaatissa viestien tulee olla. Vaihtoehdot ovat tuetut QWK ja WWF.

Toinen käyttämäni muuttuja on OFFLINERMODE. Se sisältää periaatteessa tiedon siitä, pakataanko linjalla vaiko taustalla. Lisäksi sen avulla voidaan aktivoida automaattipakkaus loginissa. Lisäksi täytyy kuitenkin asettaa myös QWKASYNC ja WWFASYNC, jotta bundlerit osaavat toimia oikein.

OFFLINERMODE voi olla 0, 1, 2 tai 3. Nolla tarkoittaa pakkausta linjalla ja yksi taustalla. Myös arvoilla kaksi ja kolme pakataan taustalla. Lisäksi arvo kaksi aktivoi automaattipakkauksen. Arvo kolme toimii muuten samoin, mutta asia varmistetaan vielä käyttäjältä. Näitä toimintoja varten tarvitaan koodia Login.rexxiin:

```
    Mode = GetUserMisc(ln,"OFFLINERMODE")
    if Mode = 3 then do
        Mode = Mode - GetYesNo(ln, "Engage bundler? ", 1, 1)
        SendModem CRLF
        end
    if Mode = 2 then do
```

```
        Format = GetUserMisc(ln,"OFFLINER")
        if Format = "WWF" then address command 'Run >NIL: TechWWF '||ln
            ||' O BBS:Cfg/TechWWF.Cfg'
        if Format = "QWK" then address command 'Run >NIL: TechQWK '||ln
            ||' O BBS:Cfg/TechQWK.Cfg'
        LogEntry "Autopacking messages using "||Format
        end
```

Tämä koodi tulee Login.rexxin loppuun juuri ennen LoginExtra.rexxin ajamista. Koodi ajaa bundlerin, jos OFFLINERMODE on kaksi. Jos se on kolme, asiaa kysytään käyttäjältä. Koodi vähentää ovelasti vastauksen moodiarvosta, jolloin siitä tulee kaksi, kun käyttäjä vastaa myöntävästi, eli pakkaus käynnistetään! Ja lokiin tehdään tietysti merkintä unohtamatta mainita, mitä formaattia käytetään...

Olen koonnut etälukuvalinnat omaan asetusvalikkoonsa, Offliner settings. Koodi on tiedostossa BBS:Rexx/OS.rexx. En laita tähän koko koodia, koska sen pituus on yhdeksän kilotavua, mutta otan mukaan olennaisimman. Suurin osa koodista onkin samaa, joka on jo vakiona. Tässä tulee kuitenkin koodi, jolla valitaan offlinerin moodi, paketin formaatti ja merkistökonversio. Mikään näistä ei ole TechnoBBS:n vakiotoiminto. Ensiksi tulee aina asetusohjelman alkuun tuleva asetuksen näyttävä koodi ja sitten sen muuttamiseen tarvittava koodi.

```
    des = GetUserMisc(ln, "OFFLINERMODE")
    select
        when des = 0 then des = "Online"
        when des = 1 then des = "Background"
        when des = 2 then des = "Auto"
        when des = 3 then des = "Query"
        otherwise des = "<NONE>"
        end
    SendModem CRLF||PURPLE||"Offliner Settings"||CRLF||CRLF||WHITE
    SendModem "1   "||CYAN||"Packing mode ..... "||
        WHITE||des||CRLF

        when cmdid = "1" then do
            des = GetUserMisc(ln, "OFFLINERMODE")
            select
                when des = 0 then des = 1
                when des = 1 then des = 2
                when des = 2 then des = 3
                when des = 3 then des = 0
                otherwise des = 0
                end
            call SetUserMisc ln, "OFFLINERMODE", des
            if des = 0 then do
                call SetUserMisc ln, "WWFASYNC", 0
                call SetUserMisc ln, "QWKASYNC", 0
                call SetUserMisc ln, "WWFTASKPRI", 0
                call SetUserMisc ln, "QWKTASKPRI", 0
                end
            else do
                call SetUserMisc ln, "WWFASYNC", 1
                call SetUserMisc ln, "QWKASYNC", 1
                call SetUserMisc ln, "WWFTASKPRI", -1
                call SetUserMisc ln, "QWKTASKPRI", -1
```

```
            end
        end

des = GetUserMisc(ln, "OFFLINER")
if des = "" then des = "<NONE>"
SendModem "2   "||CYAN||"Bundle format ..................... "||
    WHITE||des||CRLF

    when cmdid = "2" then do
        dez = GetUserMisc(ln, "OFFLINER")
        if dez = "" then dez = "WWF"
        des = "<NONE>"
        do while des = "<NONE>"
            SendModem CRLF
            SendASCII "Text/Offliner.txt"
            des = AskInput(ln, "Select: ", "", 2, "NUMERIC");
            select
                when des = "1" then des = "WWF"
                when des = "2" then des = "QWK"
                when des = "" then des = dez
                otherwise des = "<NONE>"
                end
            end
        call SetUserMisc ln, "OFFLINER", des
        end

des = GetUserMisc(ln, "QWKOUTCSET")
select
    when des = "BBS:CharSet/ISO" then des = "ISO"
    when des = "BBS:CharSet/IBM" then des = "IBM"
    when des = "BBS:CharSet/SF7" then des = "SF7"
    when des = "BBS:CharSet/ISOSF7" then des = "ISOSF7"
    when des = "BBS:CharSet/IBMSF7" then des = "IBMSF7"
    otherwise des = "<NONE>"
    end
SendModem "8   "||CYAN||"QWK Outbound Character Conversion ... "
    ||WHITE||des||CRLF

    when cmdid = "8" then do
            CSet = ""
            do while CSet = ""
                SendASCII "Text/CharSet.txt"
                CSet = AskInput(ln, "Select: ", "0", 2, "NUMERIC")
                select
                    when CSet = "0" then CSet = "BBS:CharSet/ISO"
                    when CSet = "1" then CSet = "BBS:CharSet/IBM"
                    when CSet = "2" then CSet = "BBS:CharSet/SF7"
                    when CSet = "3" then CSet = "BBS:CharSet/ISOSF7"
                    when CSet = "4" then CSet = "BBS:CharSet/IBMSF7"
                    otherwise CSet = "0"
                    end
                end
            call SetUserMisc ln, "QWKOUTCSET", CSet
            SendModem CRLF
            end
```

Tässä vielä esimerkiksi Offliner.txt, joka lähetetään formaattia valittaessa:

```
    The following offliners are available in this system:

       1 WWF
       2 QWK

Select one or press enter for no change or default selection.
```

1.68 Telmex Mouse - heikko halpishiiri?

Telmex Mouse - heikko halpishiiri?

Janne Siren

SAKU seiskassa testattu Telmex Mouse on osoittautunut alkutuntumaa heikommaksi. Hiiren sisällä oleva jousi alkoi jo lyhyen ajan käytön jälkeen naksuttaa heitellen hiiren osoitinta sivusuunnassa liikuttaessa. Puolen vuoden käytön jälkeen hiiren liikkuminen vaakasuunnassa oli täysin jumissa. Hiiren sisällä täysin suojatta olevien tunnistinledien painaminen lähemmäksi toisiaan korjasi ongelman hetkeksi, mutta parin viikon jälkeen ongelma palasi, eikä korjausta tällä kertaa enää löytynyt.

Hyvästä alkuvaikutelmasta huolimatta hiiri on varsin heikko rakenteeltaan, ja olenkin palannut Amigan uskolliseen perushiireen, vaikka uuden mallin nappulat hieman heikot ovatkin - Telmex Mousessa ainoastaan napit toimivat paremmin.

Hiiri kun on amigistille niin kovin tärkeä työväline, olisi mukava saada hieman kommentteja julkaistavaksi Amigan hiiristä. Mikä on omasta mielestäsi paras hiiri ja miksi, entä mikä on huonoin ja miksi? Kirjoitelkaapa toimitukseen!

1.69 Warp Engine 68040/40 MHz -turbokortti

Warp Engine 68040/40 MHz -turbokortti

Jyrki Saarinen

Warp Engine

- Sisältää Motorolan MC68040-prosessorin joko 28, 33 tai 40 MHz:n kellotaajuudella. 28 MHz malli ei sisällä prosessoria, vaan malli on tarkoitettu A4000/040:n omistajille, jotka voivat käyttää vanhaa prosessoriaan.

- MC68040:ssa on sekä MMU (virtuaalimuisti pelaa erittäin nopeasti SC-

SI2-kovalevyllä) että FPU. Matematiikkaprosessori on todella nopea, vastaavantaajuista 486-prosessoria kolme kertaa nopeampi. 40 MHz 68040 vastaa siis noin 120 MHz 486-prosessoria liukuluvuissa, joten ollaan siis Pentium-tasolla - paitsi että Pentium laskee väärin. :-)

Kokonaisluvuissa 68040 voittaa vastaavantaajuisen 486:n kaksinkertaisesti, eikä tässä ole vielä huomioitu sitä, että Motorolan MC680x0-sarja on kaikin puolin paremmin suunniteltu, siinä on enemmän rekistereitä ja muutenkin ohjelmoijalle MC680x0-sarja on kuin taivas.

- Neljä SIMM-paikkaa, maksimimuisti 32 Mt SIMM-palikoilla siis 128 Mt.

- Markkinoiden nopein SCSI2-ohjain. Käyttää DMA:ta, vie siis ERITTÄIN vähän prosessoriaikaa. Mitattu Seagaten Barracudalla 8 Mt/s muutaman prosentin CPU-kuormituksella.

- Erittäin nopea muistisysteemi, tukee burst-osoitusta, 4-2-2-2 muistijaksot eli ensimmäisen 32-bittisen sanan lukemiseen menee neljä kellojaksoa, seuraavien kaksi. Tämä tekee 40 MHz mallissa jopa 64 Mt/s. A4000/040:n muistijaksot vertailun vuoksi: 7-7-7-7. Muistinopeus noin viisinkertainen A4000/040-malliin verrattuna.

- Sopii A3000T/4000/4000T:hen, A3000:lle oma malli, muuten sama mutta sisältää vain kaksi SIMM-paikkaa.

- Muisteiksi käyvät 4 Mt, 8 Mt, 16 Mt ja 32 Mt 72-piikkiset SIMM-palikat, 60ns suositeltava nopeus 40 MHz kortilla. Erikokoisia palikoita saa sekoittaa vapaasti. Sekä 32- että 36-bittiset muistit käyvät kuten muissakin Amiga-tuotteissa, koska Amiga ei käytä ylimääräistä neljää bittiä mitenkään.

- Micropoliksen 1 Gt SCSI2-kovalevyllä siirtonopeus lähelle 5 Mt/s muutaman prosentin kuormituksella (Suikilla on tällainen levy).

- Hintaa USA:sta tilattuna 6700 markkaa, jota pidän varsin kohtuullisena hintana tällaisesta tehosta. Jo vastaavantasoinen SCSI2-ohjainkin maksaisi jonkin verran.

- Valmistaja MacroSystem Development, USA.

- Tällä kortilla varustettuna A4000/EC030 on halvempi mutta paljon tehokkaampi kuin A4000/040, bonuksena on vielä SCSI2-ohjain.

Asennus

Ensimmäinen ajatus oli pienen paketin saavuttua pitkän odotuksen jälkeen postilaatikkooni: "Eihän se näin pieni voi olla." Mutta kyllä se oli, eiväthän A3000/4000:n prosessorikortit ole fyysisesti isokokoisia.

Asennus A4000/EC030-koneeseeni sujui mutkattomasti: avasin koneen, otin levyasemakehikon ja ZorroII/III-backplanen pois, poistin vanhan prosessorikortin ja painoin Warp Enginen paikalleen. Asetin SCSI-ohjaimen kokonaan pois hidastamasta käynnistystä, kun en omista yhtään SCSI-laitetta. Siirsin kaksi 4 Mt SIMM-palikkaa emolevyltä Warp Enginelle ja asetin jumpperit kohdalleen. Minulla oli yksi 60ns ja yksi 80ns palikka. 40 MHz Warp Engine vaatisi kyllä 60ns muistia ilman odotustiloja, mutta 80ns muistikin näyttää

toimivan 60ns muistina! Minulla on ilmeisesti käynyt hyvä tuuri... Hitaampiakin muisteja voi käyttää, tällöin tosin muistiosoitukset hidastuvat hieman, jolloin kellojaksot ovat 5-3-3-3. Mukana tulleelta levyltä kopioin Libs:-hakemistoon 68040.library-tiedoston, jota AmigaOS tarvitsee 68040:n kanssa. (Sisältää FPU-emuloinnin ym.)

Kokemukset kortista ovat pelkästään positiivisia, A4000/EC030 nopeutui kuusinkertaiseksi, ja se myös näkyy. Ruudunpäivityskin nopeutui huomattavasti, kun Kickstart on siirretty Warp Enginen muistiin, saati sitten tavalliset, jokapäiväisessä käytössä olevat ohjelmat.

Imagine 3.0:lla renderointi nopeutui ainakin kymmenkertaiseksi, vaikka minulla oli 68EC030:n kanssa kyllä matematiikkaprosessorikin.

Vanhan 210 Mt ATID-kovalevyn nopeus ei muuttunut odotettavasti miksikään. Vielä kun saisi SCSI2-kovalevyn, niin kaikki olisi hyvin.

SysInfo v3.24

```
CopyBack Mode................................ ON
Instruction Cache............................ ON
Instruction Burst............................ ON
Data Cache................................... ON
Data Burst................................... ON
Central Processing Unit Type................. 68040
Memory Management Unit Type.................. 68040  (IN USE)
Floating Point Unit Type..................... 68040+6882
Vector Base Register (VBR) Address........... $0802DD9C
Ramsey Chip Revision (A3000)................. N/A
Gary Chip Revision (A3000)................... N/A
DMA/Gfx Chip................................. AGA ALICE - 2Meg
Display Mode................................. DBLPAL: High-res
Display Chip................................. AGA LISA CHIP
VBlank Frequency in Hz....................... 56
Power Supply Frequency in Hz................. 50
Horizontal Frequency in KHz.................. 29.96
Card Slot Installed.......................... NO
Hardware Clock installed..................... CLOCK FOUND
EClockFrequency in Hz........................ 709379

SPEED COMPARISONS AGAINST KNOWN MODELS & PERIPHERALS
A500 512k or A600 with 1MB CHIP ONLY.........  55.18
B2000, A2000, A1000 or A500 with fast ram....  41.76
A1200/14 68EC020 ICACHE 2MB CHIP ONLY........  23.98
A2500/14 A2620 68020 card....................  14.19
A3000/25 68030 ICACHE IBURST DCACHE NODBURST.   6.30
A4000/25 68040 ICACHE DCACHE COPYBACK........   1.60
CPU Million Instructions per Second..........  30.47
FPU Million Floating Operations per Second...   7.73
Speed of Chip Memory vs A600 Chip Memory.....   5.14
Dhrystones per second........................  29195
Nics Comment............................ Numero Uno
```

1.70 TRA1200-turbokortti

TRA1200-turbokortti

Jari Jokivuori

TRA1200 on EC020-prosessorilla varustettu 28 MHz turbokortti, jossa on kello/kalenteri, paikka PLCC-tyypiselle MC68881/68882-matematiikkaprosessorille ja paikka yhdelle 32-bittiselle 2/4/8 Mt SIMM-muistipalikalle. Matematiikkaprosessorin nopeus voi olla 14 MHz, 28 MHz tai omalla oskillaattorilla suurempikin.

Kuva: TRA1200-turbokortti

Kortin EC020 on mallia 25 MHz, joten kortti ylikellottaa sen 28 MHz:iin. Koska muutkin valmistajat käyttävät vastaavissa korteissaan samaa tekniikkaa, huolta ei ilmeisesti ole. (Ainakaan takuuaikana!)

Korttiin asennettavan muistin nopeuden on oltava vähintään 70ns tai nopeampaa. Itselläni sattui RCA-muistikortista jäämään 4 Mt:n 70ns SIMM-palikka, jonka TRA1200 hyväksyi mukisematta. Kortin perussiltauksetkin olivat oikeat tähän kokoonpanoon.

Kortti asennetaan A1200:n pohjassa olevaan laajennuspaikkaan. Asennus on melko helppoa, vaikka sormille onkin vaikea löytää paikkaa, mikä on ongelmana muidenkin täyspitkien korttien kanssa.

Piuhat paikalleen ja kone tulille. Nopeus oli aistittavissa. DiskMaster, Workbench yms. tuntuivat nopeammilta kuin entisellä FAST-muistikortilla. Seuraavaksi testasin konetta erilaisilla ohjelmilla saadakseni selville kortin todellisen nopeuden.

SysInfo:

TRA1200 ja FAST RAM	5.05 MIPS
A1200 ja FAST RAM	2.86 MIPS
A1200	1.36 MIPS
A4000/030	4.46 MIPS

SysInfon mukaan TRA1200-kortilla varustettu Amiga 1200 oli siis vähän yli 10% nopeampi kuin Amiga 4000/030! (Testissä tulee myös hyvin esille, kuinka paljon pelkkä FAST-muisti jo nopeuttaa A1200:aa, toim. huom.)

AIBB 6.1:n mukaan kaikissa prosessoria testaavissa testeissä kokoonpano oli 1,71-1,95 kertaa nopeampi kuin FAST-muistilla varustettu Amiga 1200. AIBB:n memtest antoi tuloksen 12 Mt/s, joka oli huomattavasti nopeampi kuin A3000, A4000 tai A1200 FAST-muistilla.

Samoin E:llä tehty MandelBench piirsi mandelbrot-kuvan 2,9:ssä sekunnissa aiemman 5,5 sekunnin asemasta (A1200 ja FAST RAM).

JPEG-kuvien katselu vauhdittui myös (TEST.JPG):

A1200	41s

```
A1200 ja 4 Mt FAST RAM            24s
A1200, TRA1200 ja 4 Mt FAST RAM   13s
```

Käyttötuntuman ja testien yhteenvetona voi sanoa, että nopeus lähes tuplaantui verrattuna FAST-muistilla varustettuun A1200:een.

Koska kortin hinta on ilman muistia ja matematiikkaprosessoria 990 markkaa ja pelkän RCA-muistikortin hinta 640 markkaa, tämän lisänopeuden saa vaivaisella 350 markan lisäsijoituksella. Verrattuna "oikeisiin" 030:lla varustettuihin turbokortteihin teho/hintasuhde vain kasvaa. Eli jos haluaa pienellä rahalla mahdollisimman paljon tehoa A1200:een, vastaus on TRA1200. Markkinoilla on myös muita vastaavia turbo/muistikortteja, mutta TRA on tähänastisista halvin. Halpaan hintaan sisältyy yllättäen myös 18 kuukauden takuu!

Amiga 1200:n omistajilla on harmina pienitehoinen virtalähde. Jos koneessa on 3.5" kovalevy ja tarkoituksena on hankkia lisämuisti- tai turbokortti, kannattaa varautua tilanteeseen, jossa virtalähteen teho yksinkertaisesti loppuu ja on tarpeen ostaa tehokkaampi virtalähde. Koneessani on A600:n virtalähde, joka on jotenkin onnistunut ruokkimaan 3.5" kovalevyn ja TRA1200-kortin 4 Mt:n muistilla.

```
        TRA1200

        Edustaja:       Karelia Computer Ky
                        Puh. (973) 897 088

        Hinta:          990 mk (ilman muistia ja matematiikkaprosessoria)

        Nopeus:         ****
        Asennus:        ***
        Rahanvaste:     *****
```

PS. Mainitkaa SAKU, kun kyselette myyjältä tuotteista.

1.71 SerMouse v2.0 - PC:n hiiri Amigaan

```
                SerMouse v2.0 - PC:n hiiri Amigaan
                ----------------------------------

                            Janne Siren
```

Amigan käyttöliittymän hallinta perustuu paljolti hiiren käyttöön. Hiirtä voidaan simuloida myös näppäimistöltä, mutta käytännössä ilman hiirtä ei Amigisti pärjää. Koneen mukana toimitetaan hiiri, mutta kaikkia se ei miellytä ja joskus sekin hajoaa. Amigalle on koko joukko hyviä hiiriä, mutta PC-koneille sitäkin enemmän. Esimerkiksi erinomaisesti käteen istuva Microsoftin hiiri saa toivomaan, että sen saisi kiinni myös Amigaan.

Mutta saahan sen. Patrick van Beem on julkaissut PD-levitykseen paketin, joka sisältää kaiken tarvittavan PC-hiiren liittämiseksi Amigaan.

PC-koneita ei alunperin suunniteltu hiiren kanssa käytettäväksi, ja hiiri lisättiin järjestelmään vasta jälkeenpäin serial-portin välityksellä. Hii-

ret kytketään PC:ssä samaan porttiin kuin modeemit. Teknisesti tämä ei ole kovin järkevä ratkaisu, mutta serial-hiiret ovat levinneet kaikkein yleisimmäksi ryhmäksi, koska kaikissa PC-mikroissa on serial-portti, yleensä kaksikin.

Amigaan serial-hiiren liittäminen on yhtä helppoa. Ongelmaksi muodostuu lähinnä 25-piikkinen serial-portti. PC:n hiiret käyttävät lähes poikkeuksetta 9-piikkistä liitintä, joten väliin tarvitaan adapteri. Sellaiset eivät paljon maksa, ja niitä löytynee kaikista tietokonetarvikkeita myyvistä liikkeistä. Virittelijöille SerMouse-paketissa on ohjeet sellaisen rakentamiseen.

Kun adapteri on kiinnitetty hiireen, voidaan adapterin 25-piikkinen liitin liittää Amigaan. Sitten ajetaan SerMouse-ajuri (ajuri tukee Microsoft- ja Mouse System -yhteensopivia hiiriä), ja PC:n hiiren pitäisi toimia kuin Amigan oma. Ainakin Super Mouse II toimi Amigassa moitteetta sitä kokeillessani. Amigan ja PC:n hiiriä voi käyttää molempia samaan aikaan. Oma hiiri kummallekkin kädelle. :-)

Tässä menetelmässä on kuitenkin varjopuolensa. Ne, joilla on koneessaan vain yksi serial-portti, joutuvat uhraamaan sen hiirelle, eikä serial-ajuri toimi niiden ohjelmien kanssa, jotka kytkevät moniajon pois päältä. Pelaamisen saa siis serial-hiirellä unohtaa. SerMouse-ajuri tarvitsee lisäksi AmigaOS 2.0:n tai uudemman.

Vaan eipä hätää: SerMouse-paketissa on myös ohjeet adapterin rakentamiseen ns. bus-hiirelle. Esimerkiksi IBM PS/2 -koneissa on erillinen liitin hiirelle, joka toimii samalla periaattella kuin Amigan ja Atarin hiiret, sen liitin on vain erilainen.

Bus-hiiret eivät ole yhtä yleisiä PC-maailmassa kuin serial-hiiret, mutta useimmista hiiristä on olemassa molemmat mallit. Bus-hiiren liittäminen Amigaan onkin huomattavasti suositeltavampaa. Adapterilla sen saa kiinni joystick-porttiin Amigan oman hiiren tavoin, eikä se tarvitse erillistä ajuria toimiakseen.

SerMousen löydät hyvin varustetuista purkeista (sermou20.lha, 7 kt).

1.72 Overdrive CD

Overdrive CD

Jari Jokivuori

CD-ROM-asema tuo käyttäjälleen suunnattomat tietomäärät käden ulottuville. Musiikkia, animaatiota ja ohjelmia riittää täydessä levyssä päivä- ellei jopa viikkokausiksi selailtavaksi. Myös peleissä on alettu hyödyntää CD-levyn mahdollisuuksia. Yleensä romppupelien musiikit tai suuret animaatiot lisäävät mukavasti pelifiilistä. CD-ROM myös vapauttaa pelaajan levykkeenvaihtorumbasta, ja pelitalotkin ovat mielissään lähes piraattivapaasta mediasta.

Kuva: Overdrive CD

CD-ROM-aseman voi liittää Amigaan joko SCSI- tai IDE-liitäntään. SCSI:iin liittäminen on ollut tavallisinta, IDE:n kohdalla puutetta on ilmeisesti sopivista ajureista. Erillisen ohjaimen/sovittimen avulla tämäkin ongelma väistyy. A600:aan ja A1200:aan tällaisen sovittimen saa kätevästi PCMCIA-liitäntään.

Overdrive CD (asema tunnetaan myös nimellä Zappo CD-ROM) on ranskalaisen Archos-yhtiön valmistama ulkoinen romppuasema A1200- ja A600-koneisiin. Laite on tuplanopeuksinen (300 kt/s), ja se liitetään Amigan vasemmassa kyljessä olevaan PCMCIA-väylään. A1200-koneissa Overdrive CD pyrkii toimimaan CD32-yhteensopivasti, mutta muutamat pelit vaativat AmigaOS version 3.1 ja kieltäytyvät toimimasta ilman sitä.

Hankin Overdrive CD:n jo viime vuonna, ja silloin koneistona oli Mitsumi. Valitettavasti sen kanssa ilmeni muutamia ongelmia, esimerkiksi äänen läpiviennin taso ja laitteen toimintavarmuus oli huono. Jälkimmäinen on voinut olla muuntajan syytä, koska koneisto vaihdettiin jo kerran tuloksetta. Pienen odottelun jälkeen sain sitten uuden version Overdrive CD:stä. Koneisto oli vaihtunut BTC:hen ja ajurin versiokin muuttunut. Nyt poistuivat ääni- ja toimintaongelmat. Version vaihdoksessa aseman mounttaus ja tunnistus nopeutuivat huomattavasti, eikä kone enää jumiutunut niiden ajaksi. Koneiston vaihdoksen myötä aseman etupaneeliin ilmestyivät musiikki-CD:n soittoon painikkeet, joten soitto-ohjelmaa ei tarvitse välttämättä käyttää.

Overdrive CD -ohjelmistoon kuuluu CD-ROM-ajuri, musiikki-CD-levyjen kuunteluohjelma ja PhotoCD-levyjen katseluohjelma. Ohjelmat asennetaan kätevästi Commodoren Installerilla. Musiikkilevyjen kuunteluohjelma on hyvin toimiva, mutta hieman kolhon näköinen. PhotoCD-levyjen katseluohjelma näyttää kuvat lomitetussa Hires HAM8 -tilassa, mutta osaa tallentaa ne 24-bittisenä, HAM8:na ja 256-värisenä. Lisäksi kuvakokoja on 197x128 - 768x512 asti.

Rompulla on Amigalle saatavana kohtuullinen määrä erilaisia PD-ohjelmia, demo-, kuva- ja musiikkikokoelmalevyjä. Lisäksi CD32-emulaation ansiosta Overdrive CD kykenee ajamaan suuren osan CD32-peleistä, ja myöskin jotkut CDTV-ohjelmat toimivat. CD32-pelien ostajalle on hyvä hankinta Jukka Kauppisen ylläpitämä lista CD32-peleistä, jotka ovat yhteensopivia Overdrive CD:n kanssa. Lista on saatavana hyvinvarustetuista purkeista. Overdrive CD lukee myös PC:n ja Macintoshin CD-ROM-levyjä, paria MacFormat-lehden levyä se ei tosin suostunut lukemaan.

CD32:ssa olevan Akiko-piirin toiminnot suoritetaan ohjelmallisesti, samoin kuten AmigaOS 3.1:ssä. Overdrive CD:n CD32-emulointiin kuuluu myös joypadin simulointi näppäimistöltä, tosin mukavuussyistä on parempi hankkia se oikea.

CD32-pelithän ovat suoraan rompulta käynnistyviä, mutta koska Overdrive CD tulee systeemin alaisuuteen vasta mounttauksen jälkeen, joudutaan käynnistyksen alussa assignoimaan CD-ROM-asema siten, että CD:llä olevat ohjelmat hakevat tarpeelliset tiedot sieltä, eivätkä varsinaisesti boottaavalta kiintolevyltä tai levykkeeltä. Tämä kaikki hoituu painamalla oikeaa hiirennappia Amigan käynnistyksen yhteydessä.

CD32-ohjelmista olen testannut Super Stardustin, joka toimii moitteettomasti, sekä kolme CD32-Gamer-lehden mukana tullutta levyä, joissa on demoja CD32:lle saatavista peleistä. Toimimattomia demoja olivat Simon the Sorcerer ja Kidchaos. Useat pelitalot tekevät nykyisin CD32-pelinsä Overdrive CD

-yhteesopivana, esim. US Gold, Electronic Arts, Team 17 ja Mindscape. Overdrive CD on siis turvallinen ratkaisu pitemmällekin katsottuna.

Kaiken kaikkiaan uudistunut Overdrive CD on erittäin mukava hankinta A1200:n omistajalle. Overdrive CD paikkaa myös aukon, joka jäi jo esitellyltä CD1200-laajennukselta Commodoren jouduttua selvitystilaan. CD1200:sta poiketen Overdrive CD ei valloita A1200:n sisäistä laajennusväylää, mikä on tietysti pelkästään positiivinen asia esim. turbokorttien omistajille.

PCMCIA-liitäntää käytettäessä on hyvä muistaa, että silloin vakio EC020-prosessorilla varustettu Amiga 1200 voi käyttää vain 4 Mt:n FAST-muistia, koska PCMCIA-liitäntä varaa neljän megatavun muistialueen.

Overdrive CD (ohjelmistoversio 11)

Testattu: Amiga 1200, EC020/28Mhz, 4 Mt FAST, kovalevy

Valmistaja: Archos

Edustaja: Broadline Oy
Puh. (90) 874 7900

Hinta: 2190 mk

1.73 Professional Filing System

Professional Filing System

Sami Klemola

Paremmin lyhenteellä PFS tunnetaan hollantilaisen Michiel Peltin shareware-tiedostojärjestelmä. PFS:n luvataan kasvattavan levykkeen kapasiteettia, nopeuttavan lukemista 50 prosentilla ja kirjoittamista peräti 3-5 kertaa sekä hakemistolistausta ja muita yksittäisiä levytoimintoja peräti 10-20 kertaa! Nopeampi hakemiston listaus olisikin varmasti usean mielestä tarpeen. Lunastaako ohjelma lupaukset vai onko kyseessä floppi? Nopeutuuko kirjoittaminen? Kasvaako kapasiteetti? Alatko pärjätä paremmin koulussa vai lähteekö kissaltasi karva? Menitkö sekaisin? Kaikki selviää tässä artikkelissa!

1.74 Levynkäyttöjärjestelmän tasot

Amigan käyttöjärjestelmä on massamuistien kannalta kolmitasoinen. Ylimmällä tasolla on DOS. Se lataa ja ajaa ohjelmat ja tarjoaa käynnistämilleen prosesseille ajonaikaisen ympäristön, jonka avulla ne voivat suoraan päästä käsiksi alemman tason järjestelmiin. DOS tarjoaa prosesseilleen sekä rajoitetusti Execin tehtäville korkean tason liitynnän tiedostojärjestelmiin. Ohjelmat voivat DOSin kautta suorittaa kaikki levytoimintonsa. DOS tarjoaa myös puskuroidut luku- ja kirjoitusrutiinit.

DOS tarjoaa myös mahdollisuuden kommunikoida suoraan tiedostojärjestelmän kanssa. Siihen ei kuitenkaan useimmiten ole tarvetta. Tiedostojärjestelmät ovat DOSin alla toimivia järjestelmiä, jotka toimivat levytoimintojen alihankkijoina sille. DOS itse ei suorita luku- ja kirjoitustoimenpiteitä, vaan antaa ne tiedostojärjestelmän käsiteltäväksi. Ohjelma voi myös, kuten jo todettu, tehdä suoraan bisnestä sen kanssa. Tällöin ovat hyödynnettävissä myös toiminnot, joita DOSin korkeamman tason liittymä ei välitä ohjelmalle. Yleisimmät Amigassa käytetyt tiedostojärjestelmät ovat OFS ja FFS.

Alimmalla tasolla on laiteohjain, esimerkiksi trackdisk.device (sisäänrakennettuun MFM-väylään liitetyt levyasemat) tai scsi.device (Commodoren SCSI-väylään liitetyt kovalevyt tai CD-asemat). Laiteohjain on toiminnaltaan "tyhmä". Se vain tekee, mitä sen käsketään tehdä. Tiedostojärjestelmä lähettää komennot laiteohjaimelle. DOS ei ole koskaan suoraan tekemisissä laiteohjaimien kanssa. Ohjelma voi toimia niin, mutta massamuistien kohdalla se ei ole suotavaa.

1.75 Tiedostojärjestelmä

Levynkäyttöjärjestelmän keskus on tiedostojärjestelmä eli filing system, joka tunnetaan myös lyhenteellä FS. DOS välittää sille ohjelmien käskyt ja se taas käskee laiteohjainta. Tiedostojärjestelmä tekee kaiken ajattelutyön. Se pitää muistissa, mitkä sektorit levyllä ovat käytössä ja missä tiedostot siellä sijaitsevat. Luettaessa tiedosto tiedostojärjestelmä lukee ensin hakemiston, josta usein on jo osia muistissa. Siitä se katsoo, millä sektoreilla tiedoston blokit sijaitsevat ja komentaa sen mukaan laiteohjainta hakemaan ne kaikki sisään.

Kirjoitettaessa tiedosto tiedostojärjestelmä etsii vapaita sektoreita ja täyttää ne kirjoittamalla dataa tiedostosta niille. Lopuksi kirjoitetaan tiedoston tiedot hakemistoon. Amigan levyasemat toimivat urapohjaisesti. Niiltä ei koskaan lueta eikä niille kirjoiteta yksittäisiä sektoreita. Vaikka tarvittaisiin vain yksi sektori, koko ura on silti luettava tai kirjoitettava aina. Yhtä tiedoston lukemista tai kirjoittamista varten voi laiteohjain joutua lukemaan jopa kymmeniä uria, jos tiedoston sektorit ovat levällään levyllä. Sektorit voi järjestää ulkoisilla ohjelmilla.

Tiedostojärjestelmä siis on levynkäyttöjärjestelmän tärkein osa. Kuitenkin se on toteutettu jokseenkin kehnosti. Etenkin hakemiston lataamista joutuu odottamaan ikuisuudelta tuntuvan ajan. Tämä johtuu siitä, että hakemisto ei ole kiinteä kokonaisuus levyllä, vaan se kirjoitetaan dynaamisesti sinne, missä on vapaata aina kun sille tulee tarve laajentua. Lisäksi pitkillä tiedostoilla (yli 34.3 kilotavua OFS:n alla ja yli 36 kilotavua FFS:n alla) tulee tarve käyttää nk. FileExt-blokkeja. Jostain syystä OFS:n tarvitsee lukea FileExt-blokit myös hakemistoa luettaessa, ja kun niitä voi olla le-

vyllä suuri määrä ja ne sijaitsevat hajautetusti, on hakemiston listaaminen OFS-levyltä todella hidasta. FFS tuo tähän pienen parannuksen, mutta edelleen hakemiston lataaminen on aika hidasta.

OFS (Old tai Original FS) on Amigan ensimmäinen tiedostojärjestelmä. Fast Filing System eli FFS julkaistiin käyttöjärjestelmäjulkaisussa 1.3. Silloin sitä voitiin hyödyntää kovalevyillä sekä pienin virittelyin ja rajoituksin levykkeilläkin. Julkaisussa 2.04 tuli FFS, joka toimii täysin myös levykkeillä. FFS nopeuttaa levytoimintoja ja kasvattaa levykkeen kapasiteettia. DD-levykkeen kapasiteetti formatoituna on OFS:n alla 837 kilotavua ja FFS:n alla 879 kilotavua. Julkaisu 3 tuo mukanaan kaksi uutta versiota kummastakin tiedostojärjestelmästä. Nyt sekä OFS:stä että FFS:stä on vanhojen versioiden lisäksi myös Intl- ja DC-versiot. Intl-versio osaa käsitellä kansainväliset merkit oikein ja DC-versio (Directory Cache) nopeuttaa hakemiston lukemista moninkertaisesti. DC-versio ylläpitää kahta kopiota levyn tiedoista, ja toinen näistä on nopeasti luettavissa. Tästä syystä tosin tiedostojen kirjoittaminen on hitaampaa kuin standardilla FFS:llä, ja levytilan kulutus on hieman suurempi.

1.76 PFS:n toiminta

Edellä kuvatut FFS:n tuomat parannukset eivät PFS:n tekijälle ↩
riittäneet,
vaan hän päätti kirjoittaa oman tiedostojärjestelmänsä, joka olisi vieläkin parempi. Hakemistolistauksen luvataan olevan vielä kolme kertaa niin nopea kuin FFS-DC:llä! Ja tottahan toki, niin se onkin. Hakemiston lukemisen suuri nopeus on saatu aikaan niin, että koko levyn hakemisto pidetään RAMissa. Tällöin hakemistolistauksia otettaessa ei tarvitse käynnistää levyä ollenkaan. Haittapuolena tässä on muistinkulutus, joka on levykkeillä 30-60 kilotavua ja kovalevyllä jopa 300 kilotavua.

Toinen vielä merkittävämpi hyöty on PFS:ään implementoitu mekanismi, jonka avulla levykkeiden invalidoituminen on saatu estettyä. Normaalistihan levy sekoaa, jos kone kaatuu tai levykkeen poistaa asemasta kesken kirjoittamisen ("You MUST replace volume..."). Käyttöjärjestelmään on kyllä aina kuulunut disk-validator, joka korjaa levykkeen seuraavalla käyttökerralla, mutta aina se ei onnistu, vaan levy on käyttökelvoton ja muutenkin siihen kuluu aikaa, parisataamegaisella kovalevypartitiolla minuuttikaupalla, koska koko levy on käytävä läpi sektori kerrallaan.

PFS tekee kaikki muutokset muistissa olevaan kopioon levykkeen hakemistosta. Ne päivitetään levylle vasta, kun mitään ei ole tapahtunut puoleen sekuntiin. Merkityksellisin on kuitenkin mekanismi, että kun levylle kirjoitetaan olemassaolevan tiedoston päälle, tiedostoa ei tuhota ensin, vaan mikäli levyllä on tilaa, uusi versio kirjoitetaan vapaille sektoreille ja vasta sen jälkeen päivitetään hakemisto eli vapautetaan vanhan tiedoston käyttämät sektorit ja laitetaan hakemistoon uuden tiedoston tiedot.

Tästä saatava hyöty on se, että jos kone sekoaa kirjoittamisen aikana, hakemisto on edelleen sitä edeltäneessä kunnossa, samoin kuin tiedosto, koska uutta tiedostoa oltiin kirjoittamassa eri paikkaan levyllä. Näin PFS-levy ei koskaan invalidatoidu! Jotta tämä toimisi, levyllä täytyy olla vapaata tilaa vähintään tiedoston pituuden verran plus 5 kilotavua. Tämä järjestely auttaa myös pitämään levykkeen fragmentoitumattomampana. Myös hakemisto ja allokaatiotaulukko päivitetään samaan tapaan. Näin tiedot eivät koskaan tu-

houdu kuten muilla tiedostojärjestelmillä, jos kirjoitus keskeytyy kesken päivittämisen.

Toinen merkittävä hyöty levyn tietojen pitämisestä muistissa ja edellä kuvatusta päivitysmekanismista on hakemistotoimintojen uskomaton nopeus. Yksittäiset toiminnot, kuten tiedoston tuhoaminen (Delete), statusbittien muuttaminen (Protect) ja tiedoston uudelleennimeäminen (Rename) ovat äärettömän nopeita, koska muutokset tehdään muistissa ja päivitetään hetkessä levylle. Kokeilumielessä tuhosin levykkeeltä 7 hakemistoa ja 175 tiedostoa. Aikaa siihen kului mukaanlukien tietojen päivitys levylle alle sekunti! PFS:n hakemistotoiminnot ovat aivan uskomattoman nopeat.

PFS-asemalla ei AddBuffers-komento tee mitään. PFS käyttää dynaamista hakemisto- ja allokaatiotaulukkovälimuistia, mutta ilmeisesti se ei ollenkaan puskuroi varsinaista levydataa. List-komento kertoo käytettyjen blokkien määrän väärin, koska se olettaa, että jokaista tiedostoa kohti kuluu yksi blokki sen headeriin. PFS ei kuitenkaan tuhlaa levytilaa sillä tavalla, vaan tiedot tallennetaan huomattavasti tehokkaammin. Tästä syystä PFS-levyn kapasiteetti on suurempi kuin FFS-levyn. Käsittelen toisaalla tässä lehdessä

DiskSpare II:n

, joka on myös levynkäyttöä tehostava ohjelma. Samassa yhteydessä annan tarkempaa tietoa PFS:n suorityskyvystä ja kerron, kuinka levyasemista saa vielä lisää irti käyttämällä PFS:ää yhdessä DiskSpare II:n kanssa!

1.77 Loppuarvostelu

PFS, kuten monet kolmansien osapuolien tiedostojärjestelmät, ei tue filenotifikaatiota, linkkejä ja recordlockeja. PFS-levyltä ei myöskään voi bootata. Muistia kuluu ja PFS toimii kovalevyllä vain alle 32 megatavun partitioilla. Siinä olivat PFS:n huonot puolet. Kun hyvät puolet ovat reilusti nopeampi luku, moninkertaisesti nopeampi kirjoitus, äärettömän nopeat hakemistotoiminnot, kasvanut kapasiteetti, invalidatoitumattomuus ja fragmentoitumattomampius, on PFS mahtava tiedostojärjestelmä! Kaikki huimat lupaukset osoittuivat paikkansapitäviksi.

Kieltämättä 32 megatavun rajoitus partition koossa on aika suuri haitta. Rajoitus johtuu siitä, että PFS käyttää 16-bittisiä sektorioffsettejä. Parannus asiaan on tulossa. PFS:n seuraavassa versiossa ne tulevat olemaan 32-bittisiä, ja silloin partition maksimikoko nousee kahteen teratavuun. Suurilla partitioilla muistinkulutus voi myös tulla ongelmaksi. Siihenkin on jonkinlaista parannusta tulossa. Ilmeisesti hakemistosta pidetään vain osia muistissa.

Uusin tänne asti kulkeutunut PFS:n versio on 9.5.4. Tekijä lupasi toimittaa uuden version vuodenvaihteeseen mennessä, mutta en ole sitä vielä saanut. PFS:n distribuutioon kuuluu 020+-versio FS-segmentistä. Sen voi asentaa suoraan kovalevylle RDB:n avulla, joten sitä ei tarvitse ladata L:-hakemistosta. Mukana tulee myös MFS, MultiFileSystem, jonka avulla voidaan käyttää samalla asematunnuksella useita tiedostojärjestelmiä. Eräiden versioiden mukana tuli myös aiemmin mainittu DiskSpare II, mutta sen tekijä ei varmaan pitänyt siitä, että sitä sanottiin "PFS-tooliksi" ja poistatti sen.

PFS on sharewarea ja tällä hetkellä sen rekisteröimismaksu on 30 guldenia.

Suomen rahassa kuluja tulee postimaksu ja muut kulut (mm. valuutanvaihdosta aiheutuvat kulut) mukaanlukien siinä satasen pintaan, ja se on ehdottomasti hinta, joka PFS:stä kannattaa maksaa. Tällä hetkellä rekisteröidyn ja rekisteröimättömän version välillä ei ole mitään eroja, mutta niitä on ollut ja ilmeisesti tulee taas olemaan. Itse olen PFS:n rekisteröinyt, ja suosittelen sitä kaikille muillekin. Kerran kun on PFS-levykkeitä käyttänyt, ei niistä voi luopua.

1.78 DiskSpare II

DiskSpare II

Sami Klemola

Amigan vakiona käyttämä Trackdisk ei hyödynnä levykkeitä ultimaalisesti. Saksalaisen Klaus Deppischin DiskSpare II ottaa levyasemistasi enemmän irti. DiskSpare II, tästä eteenpäin DSP, tallentaa sektorit tiukemmin uralle muunmuassa jättämällä tarpeettomat sektoriheaderit pois. Näin uralle mahtuu 12 sektoria Trackdiskin 11 sijaan. Tuloksena on 80 kilotavua suurempi kapasiteetti! Myös lisänopeutta luvataan. Mutta toimiiko se ja onko se luotettava? Näihin kysymyksiin saat vastauksen aivan pian.

Käsittelen tässä artikkelissa myös DSP:n käyttämistä yhdessä PFS:n kanssa, sekä annan tarkempaa tietoa myös PFS:n suorituskyvystä. PFS eli Professional Filing System on FFS:n korvaava tiedostojärjestelmä, jonka esittelen toisaalla tässä lehdessä.

1.79 Mitä DSP tekee?

Levykkeestä tulee 960-kiloinen DSP:n alla. Lisäksi on mahdollista käyttää kaksi lisäsylinteriä (80 ja 81), jolloin levykkeen kapasiteetti kasvaa 984 kilotavuun. En suosittele kuitenkaan tätä mahdollisuutta käytettäväksi, koska levykkeiltä on suunniteltu käytettäväksi vain 80 sylinteriä, joten 82 sylinterin käyttäminen on levykkeen suorituskyvyn ylittämistä, ja levykkeestä tulee riskialtis tietovarasto. DSP tukee myös HD-asemia. HD-levyk-

keen kapasiteetti nousee uskomattomaan 1920 kilotavuun!

DSP vaatii toimiakseen käyttöjärjestelmäjulkaisun 2.0. DSP ei ole varsinainen ohjelma vaan laiteohjain, diskspare.device, joka korvaa vakiolevyohjain trackdisk.devicen. DSP:n pitäisi olla A3000:ssa 15 prosenttia nopeampi kuin Trackdisk, mutta ainakin minun koneessani nopeutuminen on minimaalista. DSP-levykkeeltä ei voi bootata.

1.80 Miten levyasemaohjain toimii?

DSP on standardin Exec-devicen liityntätavan sisältävä laiteohjain. Normaalisti tiedostojärjestelmä komentaa sitä. Ohjelmat voivat myös käyttää sitä suoraan normaalien asynkronisten tai synkronisten I/O-funktioiden avulla. Laiteohjain ei pidä kirjaa levykkeen sisällöstä, vaan sen tekee tiedostojärjestelmä, esimerkiksi FFS tai PFS. Levyaseman laiteohjain lukee sisään uria ja dekoodaa niillä sijaitsevat sektorit. Vastaavasti kirjoitettaessa se koodaa datan.

Koodaaminen on tarpeen, koska tieto tallennetaan levylle synkronisesti sarjamuotoisena, eikä luettaessa ulkoista kelloa ole saatavilla, joten datassa ei saa olla liian montaa samaa bittiä peräkkäin, jottei ajastus mene pieleen. Koodaamalla data sopivalla tavalla varmistetaan, ettei näin pääse käymään. Amigan levyasemat käyttävät MFM-koodausta. Normaalisti uralla on 11 sektoria, mutta DSP jättää sektoriheaderit pois saadakseen uralle mahtumaan yhden sektorin enemmän.

Uria on levykkeellä sylinteriä kohti kaksi, yksi levykkeen yläpuolella ja toinen alapuolella. Näin sylinterillä on 24 sektoria. Sylintereitä on 80. Näin ollen DSP-levykkeellä on 1920 sektoria. Sektorin koko on yleensä 512 tavua, joten kapasiteetiksi saadaan 860 kilotavua. Sektorin kokoa voidaan kasvattaa, mutta levykkeillä se ei ole järkevää. Kovalevylläkin käytettynä se on hyödyllistä vain, kun käsitellään suuria tiedostoja, jolloin niiden lukeminen nopeutuu.

Amigan vakiotiedostojärjestelmä FFS tuhlaa aina yhden kokonaisen sektorin tiedoston headerille, vaikka sen koko olisi kahdeksan kilotavua. Tästä tulee nopeasti jopa megatavujen ylimääräinen kulutus. PFS toimii järkevämmin ja tallentaa tiedot älykkäämmin. Se ei käytä tilaa läheskään niin paljon kuin FFS. PFS-asemalla sektorin koon kasvattamista voi jo harkita. Esimerkiksi yhden kilotavun kokoiset sektorit olisivat käteviä. Silloin Infon näyttämät sektorilukemat olisivat suoraan kilotavuja.

1.81 HD-aseman käytön tekniikkaa

Koska suurempaa tallennustiheyttä tukevissa levyasemissa voidaan käyttää keskenään erikapasiteettisia levykkeitä, täytyy tiedostojärjestelmän tietää, minkäkokoinen levyke asemassa milloinkin on. Se tapahtuu monen mutkan kautta. Ensimmäiseksi trackdisk.device havaittuaan, että asemaan on laitettu levyke, tiedustelee disk.resourcelta aseman tyypin. Disk.resource lähettää asemalle tiedon, että se haluaa tietää, minkätyyppinen levyke on. Vastineeksi asema lähettää 32-bittisen ID-koodin, josta disk.resource saa tietää levykkeen kapasiteetin. HD-levykeasema asettaa itsensä syötetyn le-

vykkeen mukaan DD- tai HD-tilaan. Jos HD-asemaan laitetaan DD-levyke, levyasema käyttäytyy aivan kuin se olisi DD-asema.

Tämä jälkeen trackdisk.device tietää levykkeen tyypin, mutta tiedostojärjestelmä ei. Kun tiedostojärjestelmä havaitsee uuden levykkeen, se lähettää trackdisk.devicelle komennon TD_GETGEOMETRY. Komentoon liittyy datastruktuuri, johon trackdisk.device täyttää aseman senhetkiset attribuutit eli urien, sektorien ja päiden määrän sekä sektoreiden koon ja paljon muutakin tietoa. Trackdisk.deviceltä saamiensa tietojen mukaan FFS toimii jatkossa.

Tämä mekanismi on periaatteessa toimiva ja käytännöllinen, mutta ongelma on olemassa. Useat kolmansien osapuolien tekemät laiteohjaimet eivät tue GetGeometryä. Kaiken lisäksi ne eivät edes palauta oikeaa virhekoodia tai jopa toimivat täysin väärin, esimerkiksi formatoiden levyn! Tuloksena olisi joko levykkeen toimimattomuus tai jopa koneen kaatuminen. Tästä syystä FFS kutsuu GetGeometryä vain trackdisk.devicen tapauksessa. Näin varmistetaan, että saatu tieto on aina oikeaa. Tämä on valitettava takaisku, mutta pakollinen, koska kolmannet osapuolet tekevät yleensä ohjelmansa ihan miten sattuu.

Koska FFS ei suostu kutsumaan GetGeometryä muilla laiteohjaimilla, on diskspare.deviceen ollut pakko tehdä hacki, että HD-levykkeiden käyttäminen olisi mahdollista. DSP joutuu itse muuttamaan FFS:n sisäisen datastruktuurin sisältöä, mikä on vaarallista ja jopa kiellettyä. Tässä voi piillä vaaroja. Privaattien struktuurien rakenne voi muuttua, jolloin DSP ei enää toimisi, vaan voisi aiheutta vakavia tuhoja. Toistaiseksi kaikki toimii, eikä tästä pitäisi olla nykyisellään vaaraa.

1.82 Kaikki irti levyasemista

Meillä on kaksi levyasemien toimintaa tehostavaa ohjelmaa, PFS ja DSP. Saisiko ne toimimaan yhdessä vieläkin paremman suorityskyvyn aikaansaamiseksi? Kyllä! Pienen yrittämisen ja virittämisen jälkeen onnistuin tekemään MountList-entryn, jolla sain tehtyä itselleni aseman, jolla Professional Filing Systemin alla toimii diskspare.device. Myöhemmin myös PFS:n tekijä toimitti vastaavan, mutta enää sitä ei ole saatavilla. Tässä on entry, jolla PFS ja DSP saadaan tekemään yhteistyötä:

```
PS0:
FileSystem      = l:ProfFileSystem
Device          = diskspare.device
Priority        = 10
Unit            = 0
Flags           = 1
Surfaces        = 2
BlockSize       = 512
BlocksPerTrack  = 12
Reserved        = 2
Interleave      = 0
LowCyl          = 0
HighCyl         = 79
StackSize       = 5000
Buffers         = 32
BufMemType      = 0
```

```
Mount           = 1
GlobVec         = -1
DosType         = 0x50465300
MaxTransfer     = 2097152
Mask            = 0x7ffffffe
#
```

Jos haluat käyttää ensimmäisen sisäisen aseman asemasta esimerkiksi ensimmäistä ulkoista (toista sisäistä) asemaa, vaihda aseman tunnukseksi PS1 ja Unitiksi 1. Mikään ei tietysti estä kutsumasta tätäkään asemaa PS0:ksi, mutta se voisi aiheuttaa sekaannusta, ja jos sinulla jo on PS0, se on mahdotonta, koska DOS-asemilla pitää olla ainutlaatuinen asematunnus. Mihin PFS ja DSP sitten yhdessä pystyvät, selviää seuraavassa luvussa. Siinä kerron tekemieni testien tulokset.

1.83 Ohjelmien suorituskyky

Seuraavana on taulukko, joka havainnollistaa PFS:n ja DSP:n suorituskykyä. Testissä kopioin RAM-asemalta levykkeelle 173 tiedostoa 73 alihakemistossa. Käytetty laitteisto on A3000-25, jossa ei testauksen aikana ajettu muita ohjelmia. Luettaessa tiedostot kopioitiin takaisin RAM-asemalle. Hakemistolistaus ohjattiin >NIL:-asemalle, jolloin tulostus- tai siihen liittyvä tiedostotoiminta ei kuluttanut ylimääräistä tehoa, vaan tuloksiin saatiin oikeammat luvut.

Käytetyt lyhenteet:

FFS	Fast Filing System
PFS	Professional Filing System
TRK	Trackdisk
DSP	DiskSpare II

	FFS/TRK	PFS/TRK	FFS/DSP	PFS/DSP	
Lukeminen	44	28	40	26	sekuntia
Kirjoittaminen	201	34	197	32	sekuntia
Hakemistolistaus	26	7	22	7	sekuntia
Tuhoaminen	21	2	22	1	sekunti(a)
Vapaata tilaa	464	630	624	790	blokkia
Koko/käytössä	879/2	879/9	959/2	959/9	kt/blokkia

Taulukko 1. PFS:n ja DSP:n vertailu vakiojärjestelmiin.

Taulukko kertoo meille paljon ohjelmien suorituskyvystä. Ensiksi pistävät silmään tilaluvut. DSP-levyke on jo alunalkaen 80 kilotavua suurempi kuin Trackdisk-levyke, koska jokaisella uralla on yksi sektori enemmän. Levyllä, jolla käytettiin sekä PFS:ää että DSP:tä, vapaan levytilan määrä on miltei kaksinkertainen vakiolevykkeeseen verrattuna.

Nopeuksissa erityisesti pistää silmään se, että FFS lukee viidesosassa siitä ajasta, jonka se tarvitsee samojen tietojen kirjoittamiseen. Vertailtaessa nopeuslukemia Trackdiskin ja DSP:n välillä havaitaan, että DSP ei ole paljon nopeampi. Tiedostojen tuhoaminen oli jopa hitaampaa DSP-levykkeeltä kuin Trackdisk-levykkeeltä. Lukeminen levyltä tuntuisi olevan tuol-

laiset 10 prosenttia nopeampaa, mutta kirjoittaminen ei ole sanottavasti nopeampaa kuin Trackdisk-levykkeelle.

Erot tiedostojärjestelmien välillä taas ovat huomattavat. PFS:n lupaus 50 prosenttia nopeammasta lukemisesta ei kyllä täyttynyt, vaan testissäni lukemisen nopeuskasvu jäi noin 35 prosenttiin, mutta huomattava kasvu se on sekin. Hurjilta kuulostavat lupaukset 3-5 kertaa nopeammasta kirjoituksesta eivät pettäneet. PFS kirjoittaa melkein yhtä nopeasti kuin lukee. Testissäni FFS kirjoitti tietoja kuusi kertaa niin kauan kuin PFS. Tuollaisen tietomäärän jälkeen tulee tosin paljon päivitettävää hakemistotietoa. PFS päivittelikin levyä vielä kymmenisen sekuntia kirjoittamisen jälkeen, joten itse asiassa kirjoittaminen ei ollut aivan noin nopeaa.

Hakemistolistaus tulostui PFS-levykkeellä seitsemässä sekunnissa. Poistin levykkeen asemasta ennen tämän testaamista, jotta tietoa ei olisi valmiiksi muistissa. Seuraavilla listauskerroilla levyä ei tarvinnut lukea, koska tiedot olivat jo RAMissa. Näin ollen listausaika laski alle sekuntiin. Käytännössä listaukseen kului se aika, jonka tietojen siirtäminen PFS:n taulukoista List-komennon tulostukseen vei.

Kaikkien 73 hakemiston ja 173 tiedoston tuhoaminen PFS-levykkeeltä kävi noin sekunnissa. Hakemiston päivittämiseen riitti yhden uran kirjoittaminen. Erityisesti PFS:n hakemistotoimintoihin, mutta myös DSP:n toimintaan vaikuttaa käytettävissä oleva prosessoriteho. Lukemat ovat varmaankin hieman pienempiä, kun koneena on A500 tai A1200. Vastaavasti A4000:lla saadaan vieläkin parempia lukemia.

1.84 Loppuarvostelu

DSP on freewarea, joten sen käyttämisestä ei tarvitse maksaa mitään! Käytössäni olen havainnut sen luotettavaksi eikä ongelmia ole esiintynyt. Päin vastoin, diskspare.device korjaa jopa joitakin trackdisk.devicen bugeja. Esimerkiksi Trackdiskin yleistä ongelmaa, lukuvirheisiin johtavaa ajoitusbittien kääntymistä, ei DSP:n kanssa ole. Sen tuoma lisäkapasiteetti on mukava piriste levykkeiden käyttöä ajatellen.

Kummatkin esittelemäni levynkäyttöä tehostavat ohjelmat ovat erittäin hyviä, ja ne kannattaa ehdottomasti hankkia. PFS ja DSP kuuluvat kaikkien varteenotettavien BBS:ien valikoimiin. Ainakin omassani kummatkin ovat saatavilla, ja voin myös toimittaa ne postitse kympin kolikkoa (kerään niitä) tai kahta korppua (saat kummatkin takaisin ohjelmin) ja palautuskuorta ja -postimerkkiä vastaan. Myös koordinaattoreilta ohjelmia kannattaa kysellä.

1.85 Hetki ohjelmoijalle?

Hetki ohjelmoijalle?

Heimo Laukkanen

"Tälläkin hetkellä joku istuu näppäimistön edessä suunnitellen ohjelmaa

juuri sinulle." - Täysin tarpeeton tieto

Mikä ajaa ihmisen kiusaamaan itseään ja menettämään yönunensa vapaaehtoisesti, uhraamaan pitkiä ajanjaksoja ja tuhottomasti aikaa harjoitteluun, jolle muut eivät näe mitään järkeävää tarkoitusta ja käyttämään viimeiset pennosensa alan uusimpiin tietoihin ja välineisiin?

Ei, kyse ei ole triathlonista tai edes Cooper-testiin valmistautumisesta, vaan ohjelmoinnista. Siitä maagisesta asiasta, jota pidetään jossain määrin tietokoneen käytön jaloimpana muotona ja sen asiantuntijoita, ohjelmoijia, kaikkein fiksuimpina - tai tyhmimpinä.

Ohjelmointi on monille harrastus toisten joukossa. On mukavaa pystyä sanomaan osaavansa assembleria, C:tä ja Pascalia, vaikka taidot eivät olisikaan mitään huippuluokkaa. Tärkeintä on siis se, että tekee jotain.

"Joo kato tiiätsä. Unixit, GPS:ät, Slipit, Dipit, hipit ja kaikki muu. Kyllä ne menee. Hei mä oon guru..."

Mikä saa ihmisen istumaan tuntitolkulla ruudun edessä muutaman bitin epämääräistä tilaa pohtien ja yönunet menettäen murehtimaan kahden tai kolmen tavun salaperäistä häviämistä? Onko se ihmisen suunnaton kunnianhimo vai halu osata jotain, mitä muut eivät osaa?

Tottahan on, että se tunne, minkä onnistumisesta saa, lähentelee jossain määrin orgasmia. Ajattele, kun viimeinkin rankan uurastamisen jälkeen linkkeri ilmoittaa, että ohjelma on linkattu ja valmiina ajettavaksi ja sormet täristen käynnistät sen huomataksesi viimeinkin, että se toimii! Kuinka huojennuksen aalto kulkeekaan varpaista otsaluuhun - aina siihen asti, kunnes ohjelma kaatuu ilman mitään merkkiä vian syystä ja huojennus vaihtuu rasittavaksi raivoksi, tykytykseksi otsaluussa.

Voithan toki ohjelmoida rahasta: tehdä suurta softaa, josta ei koskaan tule mitään tai työkaluja shareware-levitykseen siinä toivossa, että muutama omantunnontuskainen käyttäjä jopa rekisteröi ohjelmasi mieluummin kuin käyttää kräkättyä versiota.

"Joo, mulla on Porsche, Lotus ja Ferrari. Huvikseni mä vaan ajelen naapureita hämätäkseni bussilla. On mulla huvila ja neljä kuumaa rakastajatarta. Hei mä teen miljoonia."

Olisiko kunnia ja kuolemattomuus kaikkien bitinvääntäjien ikuinen unelma? Olisihan se hienoa, jos sinut muistettaisiin siitä, että teit sen ja sen ohjelman. Kieltämättä siinä olisi jotain hehkua, mutta onko sellainen kunnia vaivansa arvoista? Kokeile ravintolassa iskeä naisia sanomalla, että ohjelmoit juuri uuden muistinhallintajärjestelmän. Se siitä. Tapahtumat puhuvat varmasti puolestaan.

"Mut kato, tää on sikanerokas juttu, tosin ethän sä voi sitä ymmärtää, kun sä et ohjelmoi..."

Sanotaan, että ohjelmointi parantaa ongelmanratkaisukykyä ja logiikan tajua. Totta. Mainitsematta kuitenkin jätetään, että samalla se aiheuttaa suurimmassa osassa tapauksia harmaita hiuksia ja hermoromahduksen oireita. Pitempiaikaisina seurauksina on nähty laskujen ratkaisemista matematiikan tenteissä konekielisin ohjelmin.

"Isketään kato nää rutskut tänne ja dumpataan inkut tonne. Sitten voidaan kato rundata nää chunkystä planariksi ja silloin meillä on jo kaikki, mitä me tarvitaan..."

Ohjelmointi on itsensä toteuttamisen muotona varsin omituinen valinta. Ohjelmoidakseen joutuu ensiksi opettelemaan kielen ja sitten vielä systeemit, joille haluaa ohjelmoida. Toki olisi vielä hyvä tietää miten ohjelmoida, ettei ala vahingossa tehdä uusia Microsoft-ohjelmistoja. Paljon helpommalla pääsee ryhtymällä taiteilijaksi. Ostat kangasta, muovia ja maalia. Sulatat muovin kankaan pintaan ja maalaat komeuden viskomalla maalia todella extra-terrestriaalin atmosfäärisellä otteella teoksesi päälle. Sitten vain lainaat Ave Koskelan sarjakuvasankarin Hemmo-paskiaisen albumin ja kuolemattomin sanoin siteeraat vielä kuolemattomampaa, kriitikoille tuntematonta idoliasi. Todistettuhan on, että taiteesta tekee taidetta vasta selitys, mikä teoksessa on taidetta.

"Oisit hei soittanut mulle. Mä oon tehny näitä hommia jo ennen kuin Bill Gates tiesi mikä taskulaskin on. Itse asiassa mä opetinkin sille kaiken, mitä se tietää."

Entä sosiaaliset vaikutukset? Kuka jaksaa yrittää seurustella puolikuolleen zombien kanssa, joka päivisin mumisee mustien silmänalustensa lomasta epämääräistä mantraa rutiineista, jotka pitää korjata?

Fysikaaliset ongelmat ilmaantuvat vasta myöhemmin, muutamien vuosien ohjelmoinnin jälkeen. Näkö huononee, kun pimeässä huoneessa yömyöhällä ohjelmoijat yrittävät tihrustella tuhat kaksisataa eri lohkoa käsittävästä ohjelmasta sitä pientä virhettä. Alkoholiongelmat ilmaantuvat uniongelmien ja mustien peikkojen, bugien, rinnalle. Tuoppi tai kaksikymmentä saavat maailman näyttämään hieman enemmän Pentiumin liukuluvuilla lasketulta, kunnes seuraavana aamuna joku kertoo armottoman päänsäryn ja kohmelon johtuvan pienestä laskuvirheestä.

"Ohjelmointi on kuin rakastelua. Ensin tehdään hirveästi työtä ja sitten lopulta vain muistellaan, kuinka ihanaa se olikaan."

Kunto ja mielenterveys laskevat suoraan verrannollisesti bugien määrän kasvuun. Silmistä heijastuva silmitön raivo, epätoivo, angsti yhteiskuntaa ja koko maailmaa kohtaan ovat tuntomerkkejä, joista nikotiinin ja kofeiinin tuhottoman kulutuksen lisäksi voi tunnistaa ammattiohjelmoijan. Viimeinen testi ohjelmoijan tunnistamiseksi on laulaa Ismo Alangon Nuorena syntynyt -kappaletta: "No eletään, kun vielä ehditään." Perään voi mainita rauhaisat yöunet ja nokkaunet keskipäivällä. Mikäli epäilty ohjelmoija ei hyökkää kimppuun, ei hän ole ohjelmoija eikä mikään.

"Kato työaika on kaikkein paras. Mä teen duunia silloin, kun haluan ja rahaa tulee."

Ohjelmoijien yksi puolustelu omalle ammatinvalinnalleen on myös työn joustavuus. Saat tehdä töitä, kun itse haluat ja niin pois päin. Kukaan ei vain muista mainita, että kun homma ei toimi, niin kummasti ’tahtoo’ tehdä töitä yömyöhään ja hieman ylikin.

"No hei... Mä en tiedä... Tai no joo..."

Eli mitä me voimme tästä kaikesta oppia?

1) Älkää ruvetko ohjelmoijiksi, tai jos rupeatte, niin muistakaa tämä:

"Ohjelmointi on yksi kehittävimmistä, ärsyttävimmistä, mukavimmista, palkitsevimmista, aliarvostetuimmista ja addiktiivisimmista asioista, mitä voit tehdä."

2) Nostakaa hattua taitavalle ohjelmoijalle, kun näette sellaisen. Masokismi on katoava luonnonvara.

3) Jos teistä tuntuu siltä, ettei teistä ole aivan yhtä hyväksi ohjelmoijaksi kuin monet muut, älkää ruvetko sellaiseksi, vaan kirjoittakaa ko. hommaa kritisoiva pakina.

1.86 Järjestelmäohjelmoinnin alkeiskurssi - Osa 2

Järjestelmäohjelmoinnin alkeiskurssi - Osa 2

System Executive - ohjelmointi

Sami Klemola

Kurssin ensimmäisessä osassa tutustuttiin tietokoneen toimintaan yleisellä tasolla sekä alustavasti Execin tehtäviin. Tässä ainakin toistaiseksi viimeisessä osassa selviää, miten ohjelmia tehdään Amiga-ympäristöön. Artikkelissa sivutaan myös konekieliohjelmointia, mutta pääpaino on C-kielessä. Tässä osassa käydään myös läpi Execin osat yksi kerrallaan, ja jokaisesta tulee esimerkkikoodia. Ensiksi kuitenkin katsotaan ohjelmoinnin vaiheita.

Ohjelmoinnin työkalut ja kääntäminen

Ohjelmointi käytännössä

Signaalit

Listat ja jonot

Portit ja viestit

Kirjastot

Laiteohjaimet ja Execin I/O-toiminnot

Muisti

Tehtävät

Tulevaisuutta kohti

1.87 Ohjelmoinnin työkalut ja kääntäminen

Tarvitset kääntäjän ja linkkerin. Jos ohjelmoit konekielellä, tarvitset makroassemblerin. Jos kirjoitat C-koodia, tarvitset lisäksi C-kääntäjän, mutta myös edelleen makroassemblerin, koska C-kääntäjä ei tuota valmista koodia, vaan ASCII-muotoista assemblyä eli konekielen lähdekoodia. C-kääntäjän mukana voi tulla erityisesti sen tuottamaa koodia kääntämään tarkoitettu makroassembleri. Sitten käännetään vielä makroassemblerilla eli konekielikääntäjällä, aivan kuin C-kääntäjän tuottama tiedosto olisi itse kirjoittamaasi konekielilähdekoodia. Makroassembleri ei sekään tuota vielä ajettavaa ohjelmaa, vaan objektikooditiedoston. Tämän jälkeen tulee vielä yksi vaihe, linkkaus. Linkkeri tulee usein makroassemblerin mukana, mutta niitä on saatavilla erillisinäkin.

Linkkeri yhdistää makroassemblerin tuottaman objektikoodin sekä tarvittaessa linkkerikirjastosta mukaan rutiineja, joita se kutsuu. Linkkeri hakee myös kirjastokutsujen osoituksien arvot koodiin ja C-koodin tapauksessa liittää mukaan tarvittavan startup-koodin, joka mm. tulkitsee komentorivin valmiiksi ja alustaa I/O:n. Kaikki nämä ajetaan usein yhdellä komennolla, joka voi ajaa skriptin tai frontend-ohjelman. Esimerkki tällaisesta tulee myöhemmin. Ohjelmaan voi myös kuulua useita lähdekoodeja, joista jokainen käännetään omaksi objektikooditiedostokseen. Linkkeri yhdistää silloin nämäkin tiedostot ja asettaa niistä toisiinsa tehdyt osoitukset eli cross referencet, ristiviittaukset.

Muuta tarpeellista

Includet ovat olennainen osa ohjelmointia. Ne ovat tiedostoja, jotka määrittelevät käyttöjärjestelmän sekä muiden ohjelmien käyttämiä datarakenteita ym. Autodocit ovat kuvauksia esimerkiksi käyttöjärjestelmän kirjastofunktioista. Niistä käy ilmi kullekin funktiolle annettavat parametrit, mitä funktio tekee ja mitä se palauttaa. Myös rekisterit, joihin parametrit sijoitetaan, näkyvät autodoceissa. Tieto siitä hyödyttää erityisesti konekieliohjelmoijaa. Palautusarvo tulee aina D0:ssa. Jotkin funktiot voivat palauttaa kaksikin arvoa, jolloin toinen palautetaan yleensä D1:ssä, ja sen saa myös DOS:n kautta kyselemällä jälkikäteen.

Lisäksi tarvitaan amiga.lib, joka on nk. linker library. Tähän asti on puhuttu vain ajonaikaisista kirjastoista, jotka ohjelma avaa kutsuakseen niiden funktioita. Ajonaikaiset kirjastot sijaitsevat LIBS:-hakemistossa. Linker library on aivan erilainen kirjasto. Ohjelman käyttämät rutiinit, jotka sijaitsevat kirjastossa, linkataan ohjelmatiedostoon mukaan, joten kirjastoa ei tarvita ohjelmaa ajettaessa. Tällaisten funktioiden käyttäminen tietysti pidentää ohjelmaa huomattavasti. Siksi onkin aina syytä käyttää vastaavaa ajonaikaisen kirjaston rutiinia, jos sellainen vain on olemassa.

Lisäksi amiga.lib sisältää ajonaikaisten kirjastojen offset-arvot. Näitä arvoja tarvitaan kutsuttaessa kirjaston funktioita. Offset lisätään kirjaston kantaosoitteeseen, jolloin saadaan hyppyosoite funktioon. Osoite on tosin vain hyppytaulukkoon, jossa on yleensä suoraan hyppy varsinaiseen koodiin. Offset-arvot ovat negatiivisia, koska hyppytaulukko sijaitsee ja kasvaa kirjaston kantaosoitteesta alaspäin. Kirjastofunktioiden käyttämisestä tulee esimerkkejä ja lisää tietoa myöhemmin.

Ohjelman kääntäminen

Kuten on jo mainittu, ohjelma käännetään yleensä frontendilla. Itselläni on omatekoinen cc-ohjelma, joka ajaa edelleen dcc:n, joka on varsinainen DICE:n frontend. Käytän cc:tä yksinkertaistamaan kääntämistä entisestään. Kirjoitan vain "cc ohjelma", jolloin source-hakemistossa oleva lähdekoodi ohjelma.c käännetään ja linkitetään valmiiksi ajettavaksi ohjelmaksi Progs-hakemistoon. Konekielipuolella käytössäni on niin ikään itse kirjoittamani Compile-ohjelma, joka ajaa ensin makroassemblerin, joka kääntää source-hakemistossa (luonnollisesti pidän C- ja konekielikoodit eri hakemistoissa, nämä ovat c/source ja asm/source) olevan lähdekoodin object-hakemistoon objektikoodiksi, ja sen jälkeen linkkerin, joka edelleen linkkaa tämän objektikoodin ajettavaksi ohjelmaksi. Sekä konekieli- että C-ohjelman tapauksessa linkataan mukaan amiga.lib:ssä määriteltyjä arvoja, joihin tehdään ristiviittaus ohjelmassa. Lisäksi mukaan linkataan mahdollisesti muuta objektikoodia sekä rutiineja omista kirjastoistani.

Käännöstyön ohjelmat

Olen jo maininnutkin DICE:n. Se on Matthew Dillonin C-ympäristö, joka on eräs tämän hetken varteenotettavimmista Amigalla. Toinen suosittu paketti on gcc eli GNU C, mutta se on massiivinen niin olemukseltaan (kaikkiaan yli kahdeksan megatavua vieläpä pakattuna) kuin muistinkulutukseltaankin (4-8 megatavua). SAS C:n eli entisen Latticen tuki on lopetettu. Myös DICE on kaupallinen, mutta se ei ole ollenkaan niin hinnakas kuin SAS C oli. DICE maksaa $150 normaalille ihmiselle ja $90 opiskelijalle. GNU C:n hyvä puoli on se, että se on ilmainen! DICE ja gcc ovat kattavia paketteja, ja niiden mukana tulee kaikki tarpeellinen ohjelmisto ja kaikkea ylimääräistä vielä lisäksi. Myös makroassembleri ja linkkeri kuuluvat kauppaan.

Konekieltä ohjelmoidessa kiinnitetään tietysti suurempi huomio makroassembleriin. Aikansa hyvin palvelleet MC ja a68k on jo syytä unohtaa. Markkinoilla on monia hyviä moderneja kääntäjiä, jotka osaavat myös uusimpien prosessoreiden osoitusmuodot ja erikoisuudet. Itse olen varsin mieltynyt suomalaiseen SNMA:aan, joka on erittäin nopea ja tehokas. Assemblerin kanssa joutuu myös itse kiinnittämään huomiota linkkeriin. Vanhat alink ja blink ovat historiaa. Itse olen käyttänyt varsin menestyksekkäästi DICE:n mukana tulevaa dlink:iä myös omien konekieliohjelmieni kääntämiseen. Ainakaan SNMA:n tuottaman koodin kanssa dlinkillä ei ole ollut mitään ongelmia, kun taas blink kaatuili aika ajoin, samoin kuin MC. Dlink on hoidellut esimerkillisesti myös omat kirjastoni.

Kääntäminen käytännössä

C-kielinen ohjelma kääntyy näin:

```
dcc source/prg.c -o Progs/prg -2.0
```

Välivaiheet eli konekielinen lähdekoodi ja objektikoodi kirjoitetaan tilapäishakemistoon, joka on normaalisti RAM-asemalla sijaitseva T:. Optio "-2.0" määrää kääntäjän käyttämään käyttöjärjestelmäversion 2.0 includeja.

Konekielinen ohjelma prg kääntyy seuraavasti:

```
SNMA source/prg.asm OBJ /object/prg.o INCLUDE /include
dlink object/mystartup.o object/prg.o lib/amiga.lib -o Progs/prg
```

SNMA haluaa välttämättä asettaa current diriksi sen hakemiston, jossa lähdekoodi on, joten object- ja include-hakemistoihin on viitattava taak-

sepäin. SNMA käyttää AmigaDOS-tyylistä komentoriviteknikkaa, kun taas dlink soveltaa Unix-tyylisiä "-"-merkillä alkavia yksimerkkisiä "avainsanoja".

1.88 Ohjelmointi käytännössä

Tässä luvussa käsittelen yleisesti ohjelmointia Amigalla. Otan puheeksi joitakin asioita, jotka jokaisen ohjelmoijan tulee tietää.

Ohjelman rakenne, C-ohjelma

C-ohjelman pituus on heti ainakin viisi kilotavua. Se johtuu siitä, että ohjelmalle tehdään kaikkea pientä valmiiksi. C-ohjelma saa valmiina parsetun argumenttijonon ja alustetut I/O-streamit stdin, stdout ja stderr. Alustuskoodi myös avaa kirjastoja valmiiksi. Ainakin DOS avataan aina, mutta esimerkiksi DICE osaa avata myös monet muut kirjastot riippuen siitä, mitä niistä ohjelmassa käytetään! Usein kuitenkin on parasta avata ne itse ohjelmassa, koska esimerkiksi koodi voidaan kääntää toisella kääntäjällä, sellaisella, joka ei automaattisesti avaa kirjastoja, jolloin tuloksena olisi ohjelma, joka ei avaa käyttämiään kirjastoja!

Lisäksi startup-koodi alustaa DATA- ja BSS-alueet. Tämä tulee lähinnä kysymykseen residentattavan ohjelman kanssa, koska dos.libraryn LoadSeg() huolehtii jo niiden alustuksesta ladattaessa ohjelma DOS-asemalta. Residentattu ohjelma ladataan muistiin kiinteästi, ja kaikki prosessit, jotka sitä ajavat, suorittavat samaa koodia muistissa. Tällöin data-alueet joudutaan alustamaan itse ohjelmassa, mutta C-kääntäjät osaavat ottaa sen huomioon. C-kääntäjät tarjoavat myös muita käteviä toimintoja, kuten pinon seurannan. Jos se on päällä, kun vapaan pinon määrä laskee liian pieneksi, ohjelmaan sisällytetty koodi varaa lisää pinomuistia!

Ohjelman rakenne, konekieliohjelma

Konekieliohjelmalle ei tehdä mitään valmiiksi, vaan se ajetaan aivan suoraan DOSista. Konekieliohjelma saa osoittimen argumenttijonoon A0:ssa ja sen pituuden D0:ssa. Useimmiten sen joutuu vieläpä itse päättämään komennolla clr.b -1(a0,d0.l). Jos ohjelma haluaa käyttää jotakin muuta kirjastoa kuin Execiä, on se avattava. Jos se haluaa tulostaa tai lukea tietoa esimerkiksi näppäimistöltä, on filehandle haettava itse.

Konekieliohjelman tekeminen onkin huomattavasti hankalampaa kuin C-kielisen. Sen lisäksi, että konekielellä ohjelmointi on hitaampaa sen yksityiskohtaisemman luonteen takia, on tehtävä enemmän työtä. Tietysti on mahdollista kirjoittaa geneerinen oma startup-koodi käytettäväksi konekieliohjelman kanssa, jolloin työmäärä pienenee ratkaisevasti. Konekieli tulee kysymykseen erittäin pienien ohjelmien kanssa sekä C-kielisen ohjelman funktioiden kirjoituskielenä. Usein C-kieliset ohjelmat sisältävät konekielisiä osia. Jotkin osat on kätevää ja tarkoituksenmukaista kirjoittaa konekielellä C:n sijaan. Kaikkea ei C:llä voi tehdäkään. Esimerkiksi keskeytykset edellyttävät konekielen käyttöä, ja myös kirjastot on parasta kirjoittaa konekielellä.

Yleistä ohjelmista

Ohjelma saa käyttää kaikkia prosessorin rekisterejä, mutta pinoa ja pino-

osoitinta ei saa sotkea. Ohjelman tulee palauttaa validi arvo (katso alla). Aliohjelmat saavat käyttää rekisterejä D0/D1/A0/A1. Tämä pätee myös kirjastofunktioihin sekä C-ohjelman omiin konekielialiohjelmiin. Käyttäjä odottaa saavansa tietää ohjelmasta sen versionumeron. Tiedon saa Version-komennolla, mutta vain, jos se on sisällytetty ohjelmaan. Versiotieto annetaan erityisellä merkkijonolla, joka alkaa "$VER:".

C-kielellä:

```
unsigned char versionstring[] = "$VER: Ohjelma 1.00";
```

Konekielellä:

```
        dc.b    "$VER: Ohjelma 1.00",0
```

Nyt Version-komento osaa löytää versiotiedon ja tulostaa "Ohjelma 1.00". Versiostringin tulisi olla juuri tässä formaatissa eikä siinä saisi olla mitään ylimääräistä. Jotkut ohjelmoijat laittavat siihen itsekeskeisyydessään omia nimiään, copyrighteja ja muuta siihen kuulumatonta roskaa, mikä voi olla harmittavaa.

Lisäksi käyttäjä odottaa voivansa breikata ohjelman <CTRL>-C:llä. C-kääntäjä voi tarjota automaattisen breikkisysteeminkin, mutta mielestäni se kannattaa tehdä itse:

```
        if(SetSignal(0,0) & SIGBREAKF_CTRL_C) error("user break");
```

Tämän kun laittaa sellaiseen paikkaan, että se suoritetaan tarpeeksi usein, voi käyttäjä keskeyttää halutessaan ohjelman suorituksen.

Esimerkkejä konekielellä

Tässä on lyhyt konekieliohjelma:

```
        moveq   #18,d0
        addq.l  #2,d0
        rts
```

Ohjelma ei tee muuta kuin laskee yhteen 18 ja 2 ja palauttaa arvon. Palautusarvo 20 vastaa FAIL-tilaa, joten ajettaessa tämä ohjelma saadaan ilmoitus sen "failaamisesta" eli jokin olisi mennyt vakavasti pieleen, jos kyseessä olisi ollut oikea ohjelma. Ohjelman tulee onnistuessaan palauttaa arvo 0. Tässä ovat tärkeimmät arvot:

```
        0       OK      Kaikki meni kuten suunniteltua
        5       WARN    Varoitus, jotain pientä meni vikaan
        10      ERROR   Virhe, kaikki ei onnistunut
        20      FAIL    Epäonnistuminen, kaikki meni päin seiniä
```

Tässä on vähän järkevämpi ohjelma, mutta vain vähän:

```
        include "exec/types.i"              ; yleisiä määrittelyjä
        include "exec/libraries.i"          ; kirjastojen määrittelyjä

        SECTION prg_code,CODE

        movea.l 4,a6                        ; Haetaan Execin kantaosoite
```

```
        movea.l  a0,a2                  ; Argumenttijonon osoitin
        move.l   d0,d3                  ; ja pituus talteen
        lea      Dos(pc),a1             ; Osoitin dos.libraryn nimeen
        moveq    #LIBRARY_VERSION,d0    ; kirjastoversio, mikä tahansa
        Lib      OpenLibrary            ; versio kelpaa meille tässä
        tst.l    d0                     ; testataan, saatiinko
        bne      DosOK                  ; kirjasto auki
        moveq    #20,d0                 ; failataan, jos ei saatu
        rts
DosOK   movea.l  d0,a6                  ; DOSBase a6:een
        Lib      Output                 ; Haetaan "stdout"
        move.l   d0,d1                  ; annetaan sen fh Writelle
        move.l   a2,d2                  ; argumenttijonon osoitin
        Lib      Write                  ; pituus on valmiiksi d3:ssa
        movea.l  a6,a1                  ; DOSBase a1:een, josta
        movea.l  4,a6                   ; kirjasto
        Lib      CloseLibrary           ; suljetaan
        clr.l    d0                     ; ja poistutaan, kaikki OK
        rts

        SECTION  prg_data,DATA

Dos     dc.b     "dos.library",0

        END
```

Ohjelma tulostaa sille annetun argumenttijonon, mutta ilman LF-koodia, joten prompti tulostuu sen perään. Ohjelma kaivannee selvennystä. A6 on rekisteri, jossa tulee olla aina kulloinkin kutsuttavan kirjaston kantaosoite, koska sitä kutsutaan sen kautta. Lisäksi kirjaston rutiinit saattavat myös hyödyntää sitä - Execin tapauksessa harvemmin, mutta varsinkin Intuition käyttää sitä usein. Kirjastot käsitellään yksityiskohtaisesti myöhemmin.

LIBRARY_VERSION on exec/libraries.h-includetiedostossa määritelty kiinteä muuttuja, jonka arvo tällä hetkellä on 33. Se tarkoittaa vanhinta tuettua käyttöjärjestelmäversiota, ja suositus on avata kirjasto sillä, ellei tarvitse uudemman kirjaston ominaisuuksia. Kaikki merkkijonot tulee päättää nollalla. Output palauttaa filehandlen (fh) tulostuskanavaan, joka on normaalisti Shellin, josta ohjelma käynnistettiin, ikkuna. Se annetaan Writelle, kuin myös ohjelman parametreinä saamat argumenttijonon osoitin ja pituus.

Tässä tapauksessa merkkijonoa ei tarvitse päättää nollaan, koska Write kirjoittaa tietyn määrän merkkejä sen sijaan, että se tulostaisi niitä nollaan asti, kuten esimerkiksi C-kielen printf(). Pituus Writelle annetaan D3:ssa, johon se siirretään D0:sta jo heti ohjelman alussa. Tämä on yksi syy, miksi konekielellä ohjelmointi on C:tä tehokkaampaa. Siinä on mahdollista tehdä tuollaisia ennakoivia toimenpiteitä, jotka lyhentävät ohjelmaa. C-kääntäjän tuottamassa koodissa pituus olisi kirjoitettu ties minne ja sitten vasta D3:een, kun sitä olisi siellä tarvittu, mutta me osaamme laittaa sen sinne jo valmiiksi.

Ohjelmassa käytetty Lib on makro, joka kutsuu kirjastofunktiota:

```
Lib MACRO * routinename[,basereg]
      xref _LVO\1                   ; _LVO#? eli offset-arvo haetaan
```

```
      ifnc '\2',''                  ; jostain muualta eli amiga.lib:stä
      movea.l \2,a6                 ; jos basereg on määritelty, siirretään
      endc                          ; base sieltä A6:een, kuten kuuluu
      jsr _LVO\1(a6)                ; ja kutsutaan itse funktiota
      ENDM
```

LVO eli Library Offset Vector on arvo, jonka mukaan funktiota kutsutaan. Writen tapauksessa offsetin nimi on _LVOWrite. Se asetetaan xref:llä ulkoiseksi muuttujaksi. Linkkeri täyttää muuttujan arvon hakemalla sen amiga.lib:stä. Ohjelmassa tarkistetaan OpenLibraryn palauttama arvo, joka on kirjaston kantaosoite tai nolla, jos kirjasto ei jostain syystä auennut, jolloin emme voi jatkaa ja ohjelman tulee failata. Kaikki palautusarvot pitää aina tarkistaa. Jos funktio vain voi failata, on mahdollinen virhekoodi löydettävä, koska useimmiten virheen jälkeen ei voida tehdä sitä, mitä pitäisi ja sen yrittäminen ehkä kaataa koneen tai ainakin saa aikaan suuria vaikeuksia.

Esimerkkejä C-kielellä

Tässä tulee äskeistä konekieliohjelmaa pidemmälle viety C-kielinen ohjelma:

```
#include "proto/exec_protos.h"
#include "exec/types.h"
#include "dos/dos.h"

main(ac,char*av[]);

main(ac,char*av[]) {
    printf("Ensimmäinen parametri on \"%s\".\n",av[1]);
    return(RETURN_OK);
};
```

Ohjelma on huomattavasti yksinkertaisempi, vaikka se tulostaa jopa johdannon ja rivinvaihdon sekä nimenomaisesti ensimmäisen parametrin. RETURN_OK vastaa arvoa 0. Paluuarvojen määrittelyt ovat dos.h-tiedostossa. Itse asiassa edellisessä konekieliohjelmassakin olisi pitänyt failauskohdassa olla RETURN_FAIL. Ensimmäinen include, "exec_protos.h", sisältää prototyypit kaikille Execin funktioille. Se on syytä aina ladata, samoin kuin muidenkin käytettävien kirjastojen prototyypit, jottei tulisi hankaluuksia.

Tässä tulee uusi versio:

```
#include "proto/exec_protos.h"
#include "proto/dos_protos.h"
#include "exec/types.h"
#include "dos/dos.h"

_main(void);

unsigned char *as;

_main(void) {
    as=GetArgStr();
    Write(Output(),as,strlen(as));
    return(RETURN_OK);
};
```

Nyt käytän _main()-entrypointia. Normaalisti _main():ksi tulee DICE:n oma _main()-funktio, mutta nyt korvaan sen omallani. Varsinainen _main() tekee ohjelman alustukset, joita olen jo muutamaan kertaan luetellut, minkä jälkeen se kutsuu main()-funktiota, joka on normaalisti ohjelman pääfunktio. Nyt se kuitenkin on _main(). Hyöty on se, että tässä tapauksessa tarpeeton alustus jää pois ja ohjelman pituudeksi tuli vain 616 tavua. Tällöin kuitenkin printf() on käyttökelvoton, samoin argc ja argv, mutta yllä kuvatuilla tavoilla ne voi kiertää. Tosin argumenttijonon parseamisen joutuu tekemään itse, mutta usein se itse tehdäänkin. Tämä ei kuitenkaan ole suotava tapa toimia. DICE on hyvin riippuvainen omasta alustuskoodistaan, joten sitä ei pidä mennä jättämään pois, jollei tiedä varmasti, mitä tekee.

Tähän loppuu artikkelin yleinen ohjelmointiosuus. Uusia asioita ohjelmien tekemiseen liittyen voi tulla vielä esille, mutta nyt aletaan käydä läpi Execin osia yksi kerrallaan. Jokainen osa kuvataan yksityiskohtaisesti ja esimerkkikoodia tulee, tästä lähtien kaikki C-kielellä, joka on *se* kieli, jolla ohjelmat tulee kirjoittaa, paitsi erityisesti konekielistä koodia edellyttävissä tilanteissa. Mukana on myös edistyneempää ohjelmointia.

1.89 Signaalit

Signaalit ovat Execin tarjoama keino välittää yksinkertaista tietoa esimerkiksi tehtävien välillä. Signaalit ovat tehtävienvälisen kommunikaation kaikkein alin taso, jonka varassa lepäävät kaikki muut järjestelmät. Useat signaalit voivat olla aktiivisia yhtä aikaa. Jokaisella tehtävällä on 32 signaalibittiä, joista 16 on varattu käyttöjärjestelmän käyttöön. Loput 16 ovat vapaasti ohjelmoijan käytettävissä. Harvemmin niitä kuitenkin tarvitsee suoraan itse käyttää, joten voit hypätä tämän kappaleen yli. Pääasia on, että tiedät, mitä signaalibitit ovat, koska et todennäköisesti tule koskaan tarvitsemaan niiden suoraa hyödyntämistä.

Viestiportin käyttöön voidaan signaalibitti varata automaattisesti, mutta jos sitä tarvitaan johonkin muuhun tarkoitukseen, se tehdään näin:

```
UBYTE signal;

if((signal = AllocSignal(-1)) < 0 )
    printf("Ei vapaita signaalibittejä"); else {
    printf("Varatun signaalibitin numero on %ld.\n",signal);
    FreeSignal(signal);
}
```

Signaalia odotetaan Wait()-funktiolla. Sille ei kuitenkaan anneta signaalibitin numeroa vaan maski, joka muodostetaan kaikista odotettavista signaalibiteistä. Wait() odottaa, kunnes yksi signaaleista saadaan ja palauttaa maskin, josta on saatavissa selville aktivoitunut signaali. Usea signaali voi aktivoitua samalla kertaa, joten kaikki odotetut signaalit tulee tutkia:

```
firstsigmask = 1L << firstsigbit;
secondsigmask = 1L << secondsigbit;

signalmask = Wait(firstsigmask | secondsignalmask | SIGBREAKF_CTRL_C);

if(signals & firstsigmask)
```

```
    printf("Ensimmäinen signaali tuli aktiiviseksi\n");
if(signals & secondsigmask)
    printf("Toinen signaali tuli aktiiviseksi\n");
if(signals & SIBREAKF_CTRL_C)
    printf("User break\n");
```

Tässä odotamme kahta signaalia, joiden bittinumerot ovat muuttujissa firstsigbit ja secondsigbit. Niistä muodostetaan maskit, jotka yhdistetään ja annetaan Wait():lle, joka sitten aikanaan palauttaa maskin, jossa ovat päällä kaikki aktiiviset signaalit. Tämän jälkeen ne kaikki tutkitaan yksitellen. Mukana on vielä SIGBREAKF_CTRL_C, joka on yksi käyttöjärjestelmälle kuuluvista kiinteistä signaalibiteistä. Tehtävä saa tällaisen signaalin, kun käyttäjä painaa <CTRL>-C:tä.

Signaalibittejä voi itse manipuloida funktiolla SetSignal(). Sen käyttämiseen on kiinnitettävä erityistä huomiota, koska se suorittaa erittäin matalan tason toimintoja. SetSignal():lle annetaan kaksi parametriä, signaalibittien ja maskin uudet arvot. Maski on luku, jossa ovat ne signaaleja vastaavat bitit päällä, joiden arvo asetetaan arvoluvun vastaavien bittien mukaisesti. Tämä ei ole erityisen yksinkertainen asia, joten tässä on muutama esimerkki:

```
SetSignal(0,-1)   nollaa kaikki signaalit
SetSignal(6,3)    nollaa nollabitin ja asettaa ykkösbitin
SetSignal(0,0)    ei muuta mitään, vaan ainoastaan palauttaa signaalit
```

Näistä viimeistä voidaan käyttää signaalien tilan tarkistamiseen. Se ei nollaa mahdollista aktiivista signaalibittiä, kuten Wait(), joten se on tehtävä käsin signaalin saamisen jälkeen. Parhaiten SetSignal():n toimintaa selventänee keskimmäinen esimerkki. Arvosta 6 ei käytetä ollenkaan bittiä 2, arvoltaan 4, koska maskissa ovat päällä bitit 0 ja 1, joten vain ne asetetaan arvon mukaisesti. Arvossa bitti 0 on pois päältä ja bitti 1 päällä, joten myös vastaavat signaalit saavat nämä tilat.

Exec tarjoaa viisi funktiota signaalien hyödyntämiseen:

```
ULONG SetSignal( unsigned long newSignals, unsigned long signalSet );
```

Asettaa ja/tai tarkistaa halutut signaalibitit. Tästä olikin jo esimerkki aikaisemmin <CTRL>-C:n tunnistamisessa. Tällä funktiolla voidaan tutkia tehtävän signaalien tilaa antamalla kummaksikin parametriksi nolla, mutta yleensä signaaleja tulisi odottaa Wait()-funktiolla.

```
ULONG Wait( unsigned long signalSet );
```

Odottaa haluttuja signaalibittejä ja palauttaa aktiiviset signaalit ja nollaa kaikki aktivoituneet signaalit. Tästä syystä on ehdottomasti tarkistettava kaikki palautetut bitit, jotta signaali ei pääsisi livahtamaan ohi huomaamatta, koska seuraavalla kerralla se ei enää ole päällä. Wait(0) aiheuttaisi loppumattoman odottamisen.

```
void Signal( struct Task *task, unsigned long signalSet );
```

Signaloi toista tehtävää. Signaalibitti tulee tällöin signaloitavan tehtävän signaaleista. Tällaista toimintatapaa ei voi hyödyntää tietämättä, mitä signaalia toinen tehtävä odottaa. Kyseeseen tuleekin lähinnä ohjelman alatehtävä, joka on varannut signaalibitin ja kertonut päätehtävälle sen

numeron, jotta se osaa aktivoida oikean signaalin.

```
BYTE AllocSignal( long signalNum );
```

Varaa signaalin käyttöä varten. Numero voi olla -1, jolloin varataan seuraava vapaana oleva. Signaalit eivät ole globaaleja, vaan jokaisella tehtävällä on oma joukkonsa signaaleita.

```
void FreeSignal( long signalNum );
```

Vapauttaa varatun signaalin.

1.90 Listat ja jonot

Amigan ympäristön olennainen ominaisuus on dynaamisuus. Myös järjestelmän datastruktuurit ovat dynaamisia rakenteeltaan. Järjestelmä on joustava eikä rajoja juuri ole. Dataa tallennetaan dynaamisesti luotaviin struktuureihin (joukko tietoa ennalta määrätyssä muodossa), joita pidetään listoissa. Lista voi olla tyhjä, mutta ei koskaan täysi. Lista voi myös olla järjestetty, jolloin sitä kutsutaan jonoksi (queue). Käytän siitä hämäännyksen vähentämiseksi vastaisuudessa englanninkielistä nimeä.

Exec pitää kaiken järjestelmään liittyvän tiedon listoissa. Lista koostuu headerista ja kaksoislinkatusta ketjusta elementtejä, joita kutsutaan nodeiksi. Header sisältää osoittimen listan ensimmäiseen ja viimeiseen nodeen. Header toimii handlena koko listaan. Listoihin voidaan liittää nodeja ja niitä voidaan poistaa niistä. Listaa käsitellessä ei tarvitse tietää, millaista tietoa se sisältää. Exec tarjoaa listojen käsittelyyn joukon funktioita, joita voidaan käyttää kaikkien listojen kanssa.

Node on ryhmä toisiinsa liittyvää tietoa, joka kuvaa jonkin asian. Itse asiassa nodet ovat erilaisia struktuureja, jotka alkavat Node-struktuurilla, jonka avulla listaa ylläpidetään. Nodet voivat sijaita missä tahansa muistissa toisistaan riippumatta. Ne pitävät kiinni toisistaan kahden osoittimen avulla. Kaksoislinkattu lista tarkoittaa sitä, että jokaisessa nodessa on osoitin sitä edeltävään ja seuraavaan nodeen listassa. Näitä kutsutaan nimillä predecessor ja successor.

Listan ensimmäisen noden eli Head-noden edeltäjä on listan header. Vastaavasti listan viimeisen noden eli Tail-noden jäljittäjä on niin ikään listan header. Kuten sanottu, headerissa on osoittimet listan Head- ja Tail-nodeen. Tyhjässä listassa nämä osoittavat toisiinsa. Listasta on olemassa myös supistettu versio, MinList. Nodejen määrittelyt tulevat tässä:

```
struct Node {
    struct  Node *ln_Succ;  /* Osoitin seuraavaan nodeen (successor) */
    struct  Node *ln_Pred;  /* Osoitin edelliseen nodeen (predecessor) */
    UBYTE   ln_Type;        /* Noden tyyppi, sama kuin listan tyyppi */
    BYTE    ln_Pri;         /* Prioriteetti, listan järjestämistä varten */
    char    *ln_Name;       /* ID-stringi, päättyy nollaan */
};

struct MinNode {
    struct MinNode *mln_Succ;
    struct MinNode *mln_Pred;
```

```
};
```

Tyyppejä on monia. Listassa voi olla vain keskenään samantyyppisiä nodeja. Lista voidaan järjestää nodejen prioriteettien mukaiseen järjestykseen. Tällöin sitä kutsutaan jonoksi eli queueksi. Nimeä harvemmin käytetään. Yleisimmät nodejen tyypit ovat NT_TASK, NT_INTERRUPT, NT_MSGPORT ja NT_MESSAGE. Yksi esimerkki nimen hyödyntämisestä on kirjasto. Kirjastot alkavat nodella, jolloin nimi on kirjaston nimi, esim. exec.library. Tällöin noden tyyppi on vastaavasti NT_LIBRARY. Exec pitää kirjastot kirjastolistassa, josta ne voidaan helposti löytää. Tässä tulevat headerien määrittelyt:

```
struct List {
   struct  Node *lh_Head;           /* Head-node */
   struct  Node *lh_Tail;           /* Nolla */
   struct  Node *lh_TailPred;       /* Tail-node */
   UBYTE   lh_Type;
   UBYTE   l_pad;
};

struct MinList {
   struct  MinNode *mlh_Head;
   struct  MinNode *mlh_Tail;
   struct  MinNode *mlh_TailPred;
};
```

Minimaalinen lista käy yksiin täyden listan alun kanssa, mutta sen tyyppiä ei voida testata. Tässä on kaavio, joka selventää headerin rakennetta:

```
         Head-node         Tail-node         Headeri

         ln_Succ                             lh_Head

         ln_Pred = 0       ln_Succ = 0       lh_Tail = 0

                           ln_Pred           lh_TailPred
```

Header on siis tavallaan listan Head- ja Tail-nodejen yhteensulautuma, kun niiden ajatellaan menevän lomittain päällekkäin. Header alustetaan kuvaamaan tyhjää listaa asettamalla lh_Head osoittamaan lh_Tailiin ja lh_TailPred lh_Headiin sekä nollaamalla lh_Tail. Myös tyyppi tulee asettaa oikeaksi, jos käytetään täyttä headeria. Amiga.lib sisältää rutiinin listan alustamiseen - C:llä NewList() ja konekielellä NEWLIST, joka on lists.i-tiedostossa määritelty makro. Tässä on kuitenkin vastaava koodi kummallakin kielellä:

```
/* c */

struct List list;

list.lh_Head = (struct Node *) &list.lh_Tail;
list.lh_Tail = 0;
list.lh_TailPred = (struct Node *) &list.lh_Head;

; assembly (a0 = osoitin alustettavaan headeriin)

         move.l   a0,LH_HEAD(a0)
         addq.l   #4,LH_HEAD(a0)
```

```
        clr.l    LH_TAIL(a0)
        move.l   a0,LH_TAILPRED(a0)
```

Listan tyhjyyden voi tarkistaa. Siihen on monia tapoja, mutta tässä on eräs kummallakin kielellä:

```
/* c */

if(list->lh_TailPred == (struct Node *)list) printf("Lista on tyhjä\n");

; assembly

        cmp.l    LH_TAILPRED(a0),a0
        beq      List_is_empty
```

Lista skannataan helposti ottamalla aina seuraavan noden osoitin:

```
struct List *list;
struct Node *node;

for(node = list->lh_Head; node->ln_Succ; node = node->ln_Succ)
    printf("Node nimeltä %s on osoitteessa %lx.\n",node->ln_Name,node);
```

Tässä ovat Execin listojen käsittelyyn tarjoamat funktiot:

```
void Insert( struct List *list, struct Node *node, struct Node *pred );
```

Lisää noden listaan haluttuun paikkaan. Tässä list on tietysti lista, johon lisätään, ja node on node, joka lisätään. Lisäksi pred on osoitin nodeen, jonka jälkeen listassa uusi node sijoitetaan. Uuden noden predecessor osoittaa lisäämisen jälkeen siihen ja successor siihen, joka ennen lisäämistä oli sen perässä listassa. Insert():llä voidaan lisätä node myös listan alkuun tai loppuun, mutta se ei ole tehokasta. Siihen kannattaa käyttää alla olevia erityisfunktioita, ja Insert():iä vain silloin, kun node sijoitetaan tiettyyn paikkaan keskelle listaa.

```
void AddHead( struct List *list, struct Node *node );
```

Lisää noden listan alkuun eli siitä tulee sen Head-node. Lisäämisen jälkeen headerin lh_Head osoittaa tähän nodeen.

```
void AddTail( struct List *list, struct Node *node );
```

Lisää noden listan loppuun eli siitä tulee sen Tail-node. Lisäämisen jälkeen headerin lh_TailPred osoittaa tähän nodeen.

```
void Remove( struct Node *node );
```

Poistaa noden listasta. Poistamisen jälkeen noden successor ja predecessor ovat invalideja.

```
struct Node *RemHead( struct List *list );
```

Poistaa listan Head-noden.

```
struct Node *RemTail( struct List *list );
```

Poistaa listan Tail-noden.

```
void Enqueue( struct List *list, struct Node *node );
```

Lisää noden listaan kuten Insert(), mutta sen paikka listassa määräytyy prioriteettien mukaan. Listan ensimmäinen node on suuriprioriteettisin, ja prioriteetti laskee loppua kohden. Samanprioriteettiset nodet laitetaan listaan FIFO-periaatteella eli node lisätään viimeisen samanprioriteettisen noden perään. Lista, johon nodet lisätään Enqueue():lla on jono eli queue.

```
struct Node *FindName( struct List *list, UBYTE *name );
```

Etsii listasta halutunnimisen noden ja palauttaa osoittimen ensimmäiseen nodeen, jonka nimi täsmää annetun merkkijonon kanssa. Erikoisuutena on se, että funktion avulla voidaan etsiä kaikki listassa olevat tietynnimiset nodet, vaikka niitä olisi useita. Tällöin headerin sijaan funktiolle annetaankin sen palauttaman noden osoitin, jolloin se jatkaa etsimistä eteenpäin listassa ja palauttaa mahdollisesti seuraavan täsmäävän noden:

```
struct List *list;
struct Node *node;

unsigned char name[];

if(node = FindName(list,name)) while(node) {
    printf("Node nimeltä %s löytyi osoitteesta %lx.\n",node->ln_Name,node);
    node = FindName((struct List *)node,name);
} else printf("Nodea ei löytynyt nimellä %s.\n",name);
```

1.91 Portit ja viestit

Seuraava askel syvemmälle Execin toimintaan on "interprocess communication" eli tehtävienvälinen kommunikaatio. Tästä olikin puhetta jo kurssin ensimmäisessä osassa. Nyt katsotaan, miten tieto siirtyy ohjelmien välillä käytännössä.

Ideana on, että tehtävät lähettävät viestejä toisilleen. Viestit ovat samoja dynaamisesti varattavia struktuureja, joista oli puhetta edellisessä luvussa. Myös keskeytys voi lähettää viestin tehtävälle tai päin vastoin. Viesti on kaksiosainen. Ensimmäinen osa on vakio kaikilla viesteillä ja pitää sisällään tietoa viestistä, mm. sen koon. Edelleen, se alkaa nodella, joihin tutustuttiin edellisessä luvussa. Toinen osa on viestin sisältö, data, joka siirretään. Sitä voi olla jopa melkein 64 kilotavua.

Viesti lähetetään aina kohdeporttiin. Siihen kuuluu viestilista, johon saapuvat viestit niiden noden avulla lisätään. Kun viesti otetaan vastaan, se poistetaan portin listasta. Myös porttistruktuuriin kuuluu node. Sen avulla Exec pitää portit listassa. Näin tietty portti voidaan helposti etsiä edellisessä luvussa kuvatuin keinoin.

Viestiä ei kopioida, vaan se sijaitsee staattisesti muistissa. Käytännössä vain osoitin viestiin siirretään, mutta teoriassa katsotaan myös käyttöoikeus viestistruktuuriin siirtyväksi viestin vastaanottajalle. Kun viesti on lähetetty, lähettäjä ei saa koskea siihen, ennen kuin saa viestin takaisin vastauksena, minkä jälkeen se taas kuuluu lähettäjälle.

Kun viesti saapuu viestiporttiin, se lisätään sen viestilistan perään. Kun viesti vastaanotetaan portista, se otetaan listan alusta. Näin viestit saadaan aina saapumisjärjestyksessä. Viestin saapuminen viestiporttiin voi aikaansaada signaalin sen omistajalle tai aiheuttaa ohjelmallisen keskeytyksen.

Tässä tulevat määrittelyt viestiportille ja viestille:

```
struct MsgPort {
    struct  Node mp_Node;       /* Node */
    UBYTE   mp_Flags;           /* liput */
    UBYTE   mp_SigBit;          /* signaalibitin numero */
    void   *mp_SigTask;         /* kohde, jota signaloidaan */
    struct  List mp_MsgList;    /* viestilista  */
};

struct Message {
    struct  Node mn_Node;           /* Node */
    struct  MsgPort *mn_ReplyPort;  /* vastausportti */
    UWORD   mn_Length;              /* viestin pituus */
};
```

SigTask sisältää osoittimeen ohjelman, jota signaloidaan, Task-struktuuriin. Mikäli tarkoitus on aiheuttaa keskeytys, se onkin osoitin Interrupt-struktuuriin. Nämä käsitellään myöhemmin. ReplyPort on osoitin lähettäjän omaan viestiporttiin. Viesti linkataan tämän portin viestilistaan, kun vastaanottaja vastaa siihen. Pituus on Message-struktuurin pituus plus datablokin koko.

Viestiportin voi luoda varaamalla tarpeeksi muistia ja alustamalla sen kentät. Viestilista tulee ehdottomasti alustaa NewList()-funktiolla tai NEWLIST-makrolla. V36 tarjoaa myös funktion CreateMsgPort(), jolla portin voi luoda automaattisesti. Tämän jälkeen portin voi tehdä julkiseksi, mutta usein se ei ole tarpeen. Julkinen portista tulee silloin, kun se lisätään Execin porttilistaan, josta toiset tehtävät voivat saada osoittimen siihen ja lähettää tehtävälle viestejä.

V36:lla portti tehdään ja tuhotaan näin:

```
struct MsgPort *mp;

if(mp = CreateMsgPort()) {
    mp->mp_Node.ln_Name = "Portin nimi";   /* tarpeen vain julkisissa */
    mp->mp_Node.ln_Pri = 2;                /* erityisen tärkeä portti */
    AddPort(mp);                           /* lisätään portti listaan */

    /* Portin käyttöä */

    RemPort(mp);                           /* poistetaan portti listasta */
    DeleteMsgPort(mp);                     /* ja tuhotaan se */

} else printf("Porttia ei saatu tehtyä.\n");
```

Nimeä ja prioriteettia tarvitaan vain, jos portista tehdään julkinen eli se lisätään Execin porttilistaan. Porttilistaan lisätty portti pitää aina muistaa myös poistaa ennen sen tuhoamista. Tällaiseen porttiin lähetetään

toisesta ohjelmasta viesti näin:

```
struct MsgPort *port;
struct Message *msg;

if(port = FindPort("Portin nimi")) PutMsg(port,msg);
```

Portin löytyminen täytyy aina tarkistaa, eikä antaa FindPort():n palauttamaa osoitinta suoraan PutMsg():lle. Nyt siirrytään taas vastaanottavaan ohjelmaan. Viesti otetaan vastaan ja siihen vastataan seuraavasti:

```
struct MsgPort *mp;
struct Message *msg;

while(!(port = GetMsg(mp))) WaitPort(mp);

/* viestin käsittelyä */

ReplyMsg(msg);
```

Tässä viestiä odotetaan WaitPort()-funktiolla. Mikäli tarvitsee odottaa jotakin muutakin, joudutaan käyttämään Wait()-funktiota, kuten yleensä on asian laita. Tällöin kyseeseen tuleva toimintatapa on selvitetty aiemmin tässä artikkelissa, mutta tässä on vielä esimerkki, jossa odotetaan viestiä ja breikkiä:

```
portsigmask = 1L << mp->mp_SigBit;

Wait(portsigmask | SIGBREAKF_CTRL_C);
```

Lopuksi tässä ovat vielä Execin viesteihin liittyvät funktiot:

```
void AddPort( struct MsgPort *port );
```

Lisää viestiportin porttilistaan tehden siitä julkisen, jolloin mikä tahansa järjestelmässä ajettava ohjelma voi etsiä sen ja lähettää siihen viestin.

```
void RemPort( struct MsgPort *port );
```

Poistaa julkisen portin Execin porttilistasta. Tämä on toimenpide, joka on muistettava aina tehdä ennen portin tuhoamista tai poistumista ohjelmasta.

```
void PutMsg( struct MsgPort *port, struct Message *message );
```

Lähettää viestin eli laittaa sen kohdeporttiin.

```
struct Message *GetMsg( struct MsgPort *port );
```

Vastaanottaa viestin portista.

```
void ReplyMsg( struct Message *message );
```

Vastaa viestiin eli lähettää vastaanotetun viestin takaisin lähettäjälle.

```
struct Message *WaitPort( struct MsgPort *port );
```

Odottaa viestiä saapuvaksi porttiin ja palaa vasta sitten, kun sellainen tulee. Yleensä on kuitenkin tarve odottaa useita signaaleja, jolloin joudutaan käyttämään WaitPort():n sijasta Wait():ia, jolle on silloin rakennettava maski portin signaalibitistä.

```
struct MsgPort *FindPort( UBYTE *name );
```

Etsii porttia porttilistasta ja palauttaa osoittimen porttiin, jonka nimi täsmää annettuun merkkijonoon, tai nollan, jos halutunnimistä porttia ei ollut listalla.

1.92 Kirjastot

Moneen otteeseen on jo ollut puhetta kirjastoista. Ne ovat funktiokasautumia, jotka liittyvät tiettyyn osa-alueeseen. Kirjastojen funktioita kutsutaan käyttäen niiden kantaosoitetta, joka saadaan avaamalla kirjastot. Käytettävät kirjastot on aina avattava ja lopuksi suljettava. DICE osaa avata tärkeimmät kirjastot automaattisesti, kun se havaitsee viittauksen niiden kantaosoitteen nimeen.

Tässä on tärkeimpien kirjastojen osoittimien (library base pointer) nimiä:

```
asl.library               AslBase
commodities.library       CxBase
dos.library               DOSBase
exec.library              SysBase
expansion.library         ExpansionBase
graphics.library          GfxBase
icon.library              IconBase
iffparse.library          IFFParseBase
intuition.library         IntuitionBase
layers.library            LayersBase
utility.library           UtilityBase
version.library           <salainen>
workbench.library         WorkbenchBase
```

Kirjasto koostuu funktioiden hyppykäskytaulukosta ja kirjastostruktuurista:

```
Kantaosoite + sizeof Lib   kirjastokohtaista tietoa
Kantaosoite                struct Library
Kantaosoite - 6            JMP Func1
Kantaosoite - 12           JMP Func2
Kantaosoite - 18           JMP Func3
...
```

Kantaosoitteessa on siis kirjaston node, Library-struktuuri, jonka perässä yleensä tulee lisää kirjastokohtaista tietoa. Hyppytaulukko, kuten jo mainittu, on kantaosoitteesta taaksepäin. Ensimmäiseen funktioon hyppäävä käsky on -6:ssa, joka on sen LVO eli vektorioffset. Funktiota kutsutaan hyppäämällä tähän osoitteeseen:

```
jsr      -6(a6)
```

Kantaosoitteen tulee olla a6:ssa. Ensimmäiset neljä funktiota on varattu:

```
        LVO      Funktio  Tarkoitus

        -6       Open     Avaa kirjaston (OpenLibrary() kutsuu)
        -12      Close    Sulkee kirjaston (CloseLibrary() kutsuu)
        -18      Expunge  Poistaa kirjaston muistista, jos ei käyttäjiä
        -24      Res.     Ei toimintoa, varattu paikka, palauttaa nollan
```

Näitä funktioita ohjelman ei tavallisesti tarvitse koskaan kutsua. Exec kutsuu niitä (lisää tietoa myöhemmin). Ensimmäinen varsinainen funktio on offsetissä -30, seuraava on -36 ja niin edelleen. C-kielellä ohjelmoitaessa näistä asioista ei tarvitse huolehtia ollenkaan. Kirjasto avataan näin:

```
struct Library *IntuitionBase;

if(!(IntuitionBase = OpenLibrary("intuition.library",LIBRARY_VERSION))) {
    printf("Intuition ei auennut\n");
    exit(RETURN_FAIL);
}
```

Rakenteellisesti parempi tapa olisi:

```
if(IntuitionBase = OpenLibrary("intuition.library",LIBRARY_VERSION)) {

    ...<Intuitionin käyttöä>...

} else printf("Intuition ei auennut\n");
```

Intuition on siinä määrin keskeinen kirjasto, että jos se ei aukea, on järjestelmä jo aika nurin. Kirjasto suljetaan lopuksi näin:

```
CloseLibrary(IntuitionBase);
```

Jos kutsut cleanup-koodia myös virhetilanteissa, on tarkistus tarpeen:

```
if(IntuitionBase) CloseLibrary(IntuitionBase);
```

Jos et ole kiinnostunut kirjastojen tekemisestä, katso luvun lopusta kirjastojen käyttöön liittyvien funktioiden selitykset ja hyppää seuraavaan lukuun.

Kirjastojen tekeminen

Jos olet kirjoittanut joukon funktioita, joita käytetään useissa ohjelmissa, kannattaa harkita niiden sijoittamista kirjastoon. Kerron nyt lyhyesti, miten kirjasto tehdään. Ensin katsotaan kantaosoitteessa olevaa struktuuria:

```
struct Library {
    struct  Node lib_Node;
    UBYTE   lib_Flags;
    UBYTE   lib_pad;
    UWORD   lib_NegSize;            /* Kirjaston koko alas päin */
    UWORD   lib_PosSize;            /* Kirjaston koko ylös päin */
    UWORD   lib_Version;            /* versionumero */
    UWORD   lib_Revision;           /* merkityksettömämpi revisionumero */
    APTR    lib_IdString;           /* Kirjaston tunniste */
    ULONG   lib_Sum;                /* Kirjaston tarkistussumma */
```

```
    UWORD   lib_OpenCnt;            /* Käyttäjien määrä */
};
```

Tätä kirjastonodea ei sisällytetä segmenttiin, vaan se tehdään lennossa kirjastoa ladattaessa. Tiedot toki saadaan kirjastotiedostosta. IdString on kirjaston tunniste, joka on muodossa "nimi versio.revisio (päiväys)". Lopussa tulee olla ensin CR ja sitten LF. Merkkijonon tulee päättyä nollatavuun. Tässä on esimerkki kirjaston idauksesta:

```
graphics 40.8 (15.3.93)
```

Tässä tapauksessa kirjaston versionumero on 40 (lib_Version) ja revisio 8 (lib_Revision). Exec ylläpitää kirjastosta tarkistussummaa (lib_Sum). Kirjaston koko pidetään kahdessa muuttujassa, erikseen koko kantaosoitteesta alaspäin (hyppytaulukko) ja ylöspäin (datastruktuuri). Kirjaston koodi on eri paikassa. Kirjasto on aina tehtävä konekielellä - ainakin sen runko. Tässä on lähtökohta siihen:

```
 STRUCTURE RT,0
    UWORD RT_MATCHWORD              ; tunniste
    APTR  RT_MATCHTAG               ; osoitin siihen...
    APTR  RT_ENDSKIP                ; lopun osoitin
    UBYTE RT_FLAGS                  ; lippuja
    UBYTE RT_VERSION                ; versionumero
    UBYTE RT_TYPE                   ; moduulin tyyppi (kirjasto)
    BYTE  RT_PRI                    ; prioriteetti
    APTR  RT_NAME                   ; osoitin moduulin nimeen
    APTR  RT_IDSTRING               ; osoitin moduulin idaukseen
    APTR  RT_INIT                   ; alustusosoitin
    LABEL RT_SIZE
```

Tämä Resident-struktuuri määrittää moduulin. Moduuli on tässä tapauksessa kirjasto, mutta se voi olla muukin, esimerkiksi laiteohjain. RomTag (RT) alkaa matchwordilla, jonka arvon tulee olla $4AFC. Siitä se tunnistetaan. Seuraavaksi tulee osoitin tuohon sanaan ja sitten osoitin, jota käytetään lähinnä moduulien skannaukseen ROMissa. Sen tulisi osoittaa koodin loppuun. Muut kentät ovat itsekuvaavia.

Liput ovat yleensä RTF_AUTOINIT. Tämä on erityinen alustusmoodi. Jos AUTOINIT-lippu ei ole päällä, Exec ei alusta kirjastoasi automaattisesti. Sen sijaan kutsutaan funktiota, jonka osoitin on RT_INIT:ssä. Tämän funktion tulee tehdä myöhemmin kerrottavat alustustoimenpiteet. Kannattaa kuitenkin hyödyntää AUTOINIT-toimintoa. Kun se on päällä, onkin RT_INIT:ssä osoitin taulukkoon:

```
        dataSize        Kirjaston data-alueen koko
        vectors         Osoitin funktiotaulukkoon
        structure       Osoitin InitStruct()-dataan
        initFunction    Osoitin alustusfunktioon
```

Data-alueen koko käsittää Library-struktuurin sekä mahdolliset omat lisät. Funktiotaulukko koostuu joko longword-osoitteista kirjaston funktioihin, tai ensimmäisen wordin ollessa -1, word-offseteista funktioiden alkuun suhteellisina taulukon alkuun. Jälkimmäinen tapa voi olla hieman hankalampi implementoida, mutta se on tehokkaampi.

Exec kutsuu myös InitStruct()-funktiota, jolla alustetaan Library-node ja

muut mahdolliset omat datat. Alustusfunktiota kutsutaan sen jälkeen, kun kirjasto on kunnossa. Sillä ei ole enää paljon tehtävää, sille jää lähinnä vain kirjanpidollisia tehtäviä sekä tietysti tarvittaessa alustettavat kirjaston omat datat.

Aikaisemmin mainitut neljä ensimmäistä funktiota ovat pakollisia jokaisessa kirjastossa. Open():n tehtävä on lähinnä kasvattaa opencountia ja varata avaajalle käyttäjäkohtaisia data-alueita, jos niille on tarvetta tms. Close() vähentää opencountia ja vapauttaa varaukset ym. Expunge() poistaa kirjaston muistista. Sitä varten pitää vapauttaa muistialueet ja poistaa kirjaston segmentit muistista.

Reserved()-funktio on myös pakko implementoida. Sen tulee palauttaa nolla. Olen nähnyt kirjaston lähdekoodin, jossa Reserved()-funktion paikalle hyppytaulukkoon laitettiin nolla. Tuloksena on varmuudella koneen kaatuminen tai muu häiriötilanne, kun tälle funktiolle keksitään käyttöä, ja tulevaisuuden käyttöjärjestelmä kutsuu sitä - ja tällaisilla kirjastoilla hyppää tuohon osoitteeseen.

Kirjaston tekeminen ei välttämättä ole helppoa, joten laitan tähän hieman koodia esimerkiksi. Tämä tulee suoraan sh.libraryn lähdekoodista. Poistin tosin funktiomäärittelyt - laitoin vain muutaman esimerkiksi, jotta ne eivät sotkisi tässä. Segmentti alkaa pienellä koodinpätkällä, jonka tarkoitus on estää kirjaston ajaminen ohjelmana. RTC_MATCHWORD on laiton käsky, jolloin sen ollessa tiedoston alussa kone vain kaatuisi nätisti, mutta pienellä ylimääräisellä koodinpätkällä sekin voidaan välttää.

```
VER   equ   10    ; versionumero
REV   equ   297   ; revisionumero

start moveq #-1,d0
      rts

RTAG  DC.W RTC_MATCHWORD
      DC.L RTAG,FINAL
      DC.B RTF_AUTOINIT,VER,NT_LIBRARY,0
      DC.L LNAME,IDSTR,INITD

INITD DC.L sh_base_sizeof,funcs,datat,initc

funcs dc.w -1,open-funcs,close-funcs,expunge-funcs,extfunc-funcs
      dc.w func1-funcs,func2-funcs,-1

; Tämä on InitStruct():n datataulukko. INIT#?-makroilla tehdään sille
; "komentojono", jolla alustetaan noden kentät. Lisää tietoa InitStruct():n
; toiminnasta on include-tiedostossa "exec/initializers.i". Se käsitellään
; ehkä joskus.

datat INITBYTE LN_TYPE,NT_LIBRARY
      INITLONG LN_NAME,LNAME
      INITBYTE LIB_FLAGS,LIBF_SUMUSED!LIBF_CHANGED
      INITWORD LIB_VERSION,VER
      INITWORD LIB_REVISION,REV
      INITLONG LIB_IDSTRING,IDSTR
      DC.W 0

; Tämä on alustusfunktio, jota OpenLibrary() kutsuu sen jälkeen, kun
```

```
; kirjasto on pantu pystyyn. Tallennan segmentin osoitteen (BCPL-muodossa)
; sekä avaan tarvittavat kirjastot. OpenLib on yksinkertainen makro,
; joka kutsuu OpenLibrary():ä ja moveaa basepointterin data-alueelle
; kirjastonoden perään (sh-struktuuri pitää sisällään Library-noden
; sekä kirjaston omia muuttujia). ASL:sta ja Commoditiesista edellytetään
; versio 37 tai suurempi.

initc movem.l a4/a6,-(sp)
      movea.l d0,a4
      move.l a0,sh_SegList(a4)
      movea.l 4,a6
      move.l a6,sh_SysBase(a4)
      OpenLib Dos,sh_DosBase(a4)              ; DOS
      OpenLib intuition,sh_IntuitionBase(a4) ; Intuition
      OpenLib AName,sh_AslBase(a4),37         ; ASL
      OpenLib CName,sh_CxBase(a4),37          ; Commodities
      move.l a4,d0
      movem.l (sp)+,a4/a6
      rts

; Nämä ovat Open() ja Close(). Ne vain päivittävät opencountin. Open()
; palauttaa kirjaston kantaosoitteen, joka on sitä kutsuttaessa a6:ssa.
; Close() tarkistaa lisäksi DELEXP-bitin arvon. Se tarkoittaa, että
; "delayed expunge" on päällä, eli joku on kutsunut Expunge():a sillä
; aikaa, kuin kirjastoa on käytetty (katso alla). Jos DELEXP on päällä,
; suoritus lasketaan läpi Expunge()-funktioon.

open  addq.w #1,LIB_OPENCNT(a6)
      bclr.b #LIBB_DELEXP,LIB_FLAGS(a6)
      move.l a6,d0
      rts

close subq.w #1,LIB_OPENCNT(a6)
      btst.b #LIBB_DELEXP,LIB_FLAGS(a6)
      beq extfunc

; Expunge() poistaa kirjaston muistista. Ensin varmistetaan, että kirjasto
; ei ole kenelläkään auki. Jos näin on, ei sitä tietenkään poisteta, vaan
; asetetaan DELEXP-lippu viivästetyn expungen merkiksi. Kirjasto expungetaan
; heti, kun viimeinen sen käyttäjä sulkee sen. Muussa tapauksessa suljetaan
; kirjastot, vapautetaan data-alueen käyttämät muistialueet ja poistetaan
; segmentti muistista. Minun mielestäni tämä on kyllä hieman vaarallista,
; mutta näin se neuvotaan tekemään.

expunge

      movem.l d2/a5/a6,-(sp)
      clr.l d2
      movea.l a6,a5
      movea.l 4,a6
      tst.w LIB_OPENCNT(a5)
      beq expu0
      bset.b #LIBB_DELEXP,LIB_FLAGS(a5)
      bra expu2
expu0 CloseLib sh_DosBase(a5)
      CloseLib sh_IntuitionBase(a5)
      move.l sh_CxBase(a5),d0
```

```
      beq expu3
      movea.l d0,a1
      Lib CloseLibrary
expu3 move.l sh_AslBase(a5),d0
      beq expu1
      movea.l d0,a1
      Lib CloseLibrary
expu1 move.l sh_SegList(a5),d2
      movea.l a5,a1
      Lib Remove
      clr.l d0
      movea.l a5,a1
      move.w LIB_NEGSIZE(a5),d0
      suba.l d0,a1
      add.w LIB_POSSIZE(a5),d0
      Lib FreeMem
expu2 move.l d2,d0
      movem.l (sp)+,d2/a5/a6
      rts

extfunc

      clr.l d0
      rts
```

Kirjaston lataaminen

Kirjaston voi ottaa käyttöön käsin lataamalla segmentin LoadSeg():llä (DOS-funktio), valmistelemalla sen MakeLibrary():llä ja lisäämällä sen Execin kirjastolistaan AddLibrary():llä. Helpompaa on kuitenkin käyttää yllä kuvattua RomTagia (Resident-struktuuria), jolloin OS osaa ladata segmentin automaattisesti, kun se on sijoitettu LIBS:-hakemistoon, aina kun joku sitä haluaa käyttää (kutsuu OpenLibrary():ä).

Kirjasto voi liittyä myös Zorro-korttiin tai muuhun laajennukseen, jolloin se voidaan sijoittaa SYS:Expansion-hakemistoon. Tällöin startup-sequencessa ajettava BindDrivers-komento lataa sen. Kortin toiminnasta huolehtiva kirjasto voidaan myös sijoittaa kortilla olevaan ROMiin. Mutta tämä kuuluu Expansion-kirjaston toimialaan, joten siitä ei enempää tässä.

Nämä ovat Execin tarjoamat kirjastofunktiot:

```
void AddLibrary( struct Library *library );
```

Lisää kirjaston Execin kirjastolistaan. Normaalisti Exec itse kutsuu tätä funktiota, kun joku haluaa avata levyllä olevan sen jälkeen, kun se on ensin kutsunut MakeLibrary():ä. Viimeksi mainittua en käsittele tässä ollenkaan, koska se kuuluu moduulinluontifunktioihin. Näihin palataan ehkä myöhemmin kurssin aikana.

```
void RemLibrary( struct Library *library );
```

Poistaa kirjaston Execin kirjastolistasta.

```
struct Library *OldOpenLibrary( UBYTE *libName );
```

Vanha avaamisfunktio. Ei tule käyttää.

```
struct Library *OpenLibrary( UBYTE *libName, unsigned long version );
```

Avaa kirjaston. Parametreinä annetaan kirjaston nimi ja versionumero. Versionumeron tulee olla sellainen, että kirjastossa varmasti on kaikki funktiot, joita tullaan kutsumaan. Mikäli kyseessä on käyttöjärjestelmän kirjasto, tulee versionumeron olla LIBRARY_VERSION, joka on tätä nykyä 33. Se on vanhin virallisesti tuettu versionumero.

```
void CloseLibrary( struct Library *library );
```

Sulkee kirjaston. Parametrinä annetaan kirjaston basepointer.

```
APTR SetFunction( struct Library *library, long funcOffset,
    unsigned long (*newFunction)() );
```

Vaihtaa kirjaston funktion osoittimen. Tällä funktiolla voidaan kirjaston alkuperäinen funktiokoodi korvata omalla koodilla. Vektori vaihdetaan niin, että tarkistussumma on aina oikein (katso alla). Tällä funktiolla ei voi vaihtaa epästandardien kirjastojen (esimerkiksi dos.library) vektoreita.

Funktiolle annetaan osoitin uuteen funktiokoodiin, ja sen jälkeen, kun kyseistä funktiota kutsutaan, suoritetaan uusi koodi. Mikäli uusi koodi ei jää muistiin, vaan lähtee pois esimerkiksi ohjelman suorituksen loputtua, tulee alkuperäinen vektori palauttaa. SetFuntion() palauttaa vanhan koodin osoitteen, joten se on helppo toteuttaa.

```
void SumLibrary( struct Library *library );
```

Laskee kirjaston noden tarkistussumman. Funktio päivittää tarkistussumman kirjaston nodestruktuuriin. Funktio myös tarkistaa vanhan summan, ja mikäli se ei täsmää, näyttää alertin. Aiheettomien alertien välttämiseksi kirjasto on merkittävä muutetuksi aina, kun sitä muutetaan. Tätä varten on lippu CHANGED. Ohjelmat eivät yleensä kutsu tätä funktiota.

1.93 Laiteohjaimet ja Execin I/O-toiminnot

Amigassa I/O-toiminnoista vastaavat Execin laiteohjaimet eli devicet. Laiteohjaimet ovat suoraan tekemisissä hardwaren kanssa. Ohjelmat käyttävät hardwarea laiteohjaimien kautta. Exec tarjoaa helpon liittymän laiteohjaimien hyödyntämiseen. I/O-toiminnot tehdään lähettämällä laiteohjaimelle komentoja ja mahdollisesti dataa sekä otetaan niitä vastaan siltä.

Käyttöjärjestelmään kuuluu 14 laiteohjainta:

audio.device	äänet
clipboard.device	leikkuulauta
console.device	konsoli (näppis ja näytin)
gameport.device	ilotikku, paddlet ja hiiri
input.device	syöte (monista lähteistä)
keyboard.device	näppis (matalalla tasolla)
narrator.device	puhe, selostus
parallel.device	rinnakkaisportti
printer.device	kirjoitin
scsi.device	Commodoren SCSI-väylä

```
        serial.device             sarjaportti
        timer.device              ajastin
        trackdisk.device          sisäänrakennettu MFM-levyasemaväylä
```

Laiteohjaimiin tutustutaan tarkemmin joskus muulloin. Tässä on tarkoitus vain nopeasti katsoa, miten niitä käytetään. Laiteohjain on avattava ennen käyttöä aivan kuin kirjastokin. Laiteohjaimen käyttöä varten tarvitsemme viestiportin ja IORequestin:

```
struct IORequest {
    struct  Message io_Message;
    struct  Device  *io_Device;     /* Device-struktuuri            */
    struct  Unit    *io_Unit;       /* Unit-struktuuri (privaatti)  */
    UWORD   io_Command;             /* I/O-komento                  */
    UBYTE   io_Flags;               /* Liput (IOF_QUICK)            */
    BYTE    io_Error;               /* Virhenumero                  */
};
```

Tällä struktuurilla voidaan lähettää vain komentoja, joihin ei liity datan lähettämistä tai vastaanottamista. Jos dataakin liikkuu, tarvitaan pidempi versio:

```
struct IOStdReq {
    struct  Message io_Message;
    struct  Device  *io_Device;     /* Device-struktuuri            */
    struct  Unit    *io_Unit;       /* Unit-struktuuri (privaatti)  */
    UWORD   io_Command;             /* I/O-komento                  */
    UBYTE   io_Flags;               /* Liput (IOF_QUICK)            */
    BYTE    io_Error;               /* Virhenumero                  */
    ULONG   io_Actual;              /* Siirtyneiden tavujen määrä   */
    ULONG   io_Length;              /* Tavujen määrä                */
    APTR    io_Data;                /* Osoittaa dataan              */
    ULONG   io_Offset;              /* Offset laitteella            */
};
```

Laiteohjainta avatessa tarvitsee huolehtia vain siitä, että viestiosuus struktuurista on alustettu kunnolla. Tietysti kenttien tulee olla nollattu. Laiteohjain alustaa Device- ja Unit-kentät, ja muut alustetaan myöhemmin komentoja lähetettäessä. Käyttis V36 tarjoaa käteviä apufunktioita viestiportin ja IORequestin tekemiseen. Laiteohjain aukeaa näin:

```
struct MsgPort *port;
struct IOStdReq *request;

if(!(request = CreateIORequest(port = CreateMsgPort(),
    sizeof(struct IOStdReq)))) {
    printf("IORequestin tekeminen epäonnistui\n");
    DeleteMsgPort(port);
}

if(OpenDevice("trackdisk.device",0,request,0)) printf("Ei auennut\n");
    else {

    ....<trackdisk.devicen käyttöä>...

}
```

Huomattavaa tässä on erityisesti se, että OpenDevice() palauttaa nollan silloin, kun laiteohjaimen avaaminen onnistui. Yleensähän nolla tarkoittaa virhettä, mutta tässä se on toisin päin. Virheen tapauksessa palautetaan laiteohjainkohtainen virhenumero. Toinen parametri on yksikön numero. Tässä tapauksessa se tarkoittaa levyasemaa, nolla on (ensimmäinen) sisäinen levari. Viimeisenä voidaan antaa lippuja. Lopuksi laiteohjain pitää sulkea:

```
CloseDevice(request);
DeleteIORequest(request);
DeleteMsgPort(port);
```

Jokainen laiteohjain tunnistaa kahdeksan vakiokomentoa:

CMD_RESET	Resetoi laitteen alkutilaansa
CMD_READ	Lukee dataa laitteelta
CMD_WRITE	Kirjoittaa dataa laitteelle
CMD_UPDATE	Päivittää tietoja
CMD_CLEAR	Tyhjentää laiteohjaimen puskurit
CMD_STOP	Pysäyttää laitteen
CMD_START	Käynnistää laitteen
CMD_FLUSH	Puhdistaa komentojonon

Kaikki laiteohjaimet eivät välttämättä tue kaikkia näitä, mikä on minusta varsin kummallista. Tuntemattoman laiteohjaimen käyttäminen voi olla vaarallista. Komennon lähettäminen laiteohjaimelle on helppoa:

```
request->io_Command = CMD_CLEAR;
DoIO(request);
```

Nämä peruskomennot toteuttavat yksinkertaisia toimintoja. Esimerkiksi FLUSH abortoi kaikki laiteohjaimella jonossa olevat I/O-toimintopyynnöt - myös muiden ohjelmien, joten sen kanssa kannattaa olla varovainen. CLEAR tyhjentää sarjaportin puskurit. UPDATE päivittää levylle trackdisk.devicen muistissa olevan uran, jos sitä on muutettu. STOP pysäyttää laitteen toiminnan. Laiteohjain lopettaa kyseisen yksikön jonossa olevien pyyntöjen käsittelyn ja jatkaa toimintaansa vasta, kun saa START-komennon. Moniyksikköisessä laitteessa komennot vaikuttavat vain siihen yksikköön (esimerkiksi nimenomaiseen levyasemaan), jolle komennot annetaan.

Kun halutaan esimerkiksi kirjoittaa dataa, täytyy requestiin liittää tiedot kirjoitettavasta datasta, missä se on ja paljonko sitä on. Levykkeen tapauksessa pitää myös ilmoittaa, minne se kirjoitetaan. Kun kyseessä on massamuistiasema, muuttuja Offset kertoo datan sijaintipaikan levyllä:

```
APTR data;

request->io_Data = data;            /* osoitin dataan */
request->io_Length = 0x200;         /* kirjoitetaan vain yksi sektori */
request->io_Offset = 0x20000;       /* paikka levyllä tavuina */
request->io_Command = CMD_WRITE;
DoIO(request);
```

Tämä koodi kirjoittaa 512 tavua dataa levyllä kohtaan $20000. Sektorin numero, jonka sisältö korvautuu uudella datalla, saadaan jakamalla Offset 512:lla. Offsetin ja pituuden tulee olla jaollisia sektorin koolla, joka tässä oletuksena on 512 tavua. Otin tässä trackdisk.devicen vain yksinkertaiseksi esimerkiksi - massamuistin käyttäminen laiteohjaintasolla ei ole

suotavaa.

DoIO() on synkroninen funktio. Se jää odottamaan vastausta ja ottaa viestin sitten portista. SendIO() vain lähettää viestin, joten vastausta on itse odotettava ja otettava viesti vastausportista. Sitä odotellessa voi myös kysellä, onko pyyntö jo käsitelty. Komennon voi myös abortoida. Tässä on esimerkki:

```
SendIO(request);

while(TRUE) {

        ...<muuta juttua, jossa kuluu hieman aikaa>...

    if(CheckIO(request)) {
        GetMsg(port);
        break;
}
```

Voisihan tuota GetMsg():tä tietysti kutsua suoraankin, mutta näin se näyttää kivemmalta... Jos on tarve keskeyttää operaatio:

```
if(!(CheckIO(request))) {
    AbortIO(request);
    WaitIO(request);
}
```

Tämä koodinpätkä tutkii, vieläkö komento on kesken. Jos se on, abortoidaan se ja odotetaan, että se todellakin on abortoitu.

Laiteohjaimen tekeminen

Tähän asiaan en paneudu nyt tässä. Laiteohjaimet tullaan käsittelemään omana osanaan tällä kurssilla. Siinä yhteydessä teemme myös oman laiteohjaimen. Voin tässä kuitenkin johdantona valottaa prosessia hieman. Laiteohjain on toiminnaltaan melko lailla samanlainen kuin kirjasto. Myös sillä on kantaosoite, io_Device osoittaa Device-struktuuriin, joka on aivan samanlainen kuin Library-struktuuri.

Myös laiteohjaimilla on hyppytaulukko, ja jotkin laiteohjaimet tarjoavat kirjastojentyylisen funktioliittymän. Pääasiassa toimet kuitenkin hoidetaan IORequestien avulla. Laiteohjaimen on yleensä tarpeen laukaista lapsitehtävä erikseen jokaista yksikköä varten. Laiteohjaimen tekeminen on hankalampaa kuin kirjaston. Perehdymme siihen hamassa tulevaisuudessa.

Nämä ovat Execin I/O-funktiot:

```
void AddDevice( struct Device *device );
```

Lisää laiteohjaimen Execin listaan.

```
void RemDevice( struct Device *device );
```

Poistaa laiteohjaimen.

```
BYTE OpenDevice( UBYTE *devName, unsigned long unit,
    struct IORequest *ioRequest, unsigned long flags );
```

Avaa laiteohjaimen. Parametreinä annetaan sen nimi ja yksikkö, osoitin asianmukaisesti alustettuun requestblokkiin ja liput.

```
void CloseDevice( struct IORequest *ioRequest );
```

Sulkee laiteohjaimen.

```
BYTE DoIO( struct IORequest *ioRequest );
```

Lähettää komennon laiteohjaimelle synkronisesti.

```
void SendIO( struct IORequest *ioRequest );
```

Lähettää komennon laiteohjaimelle asynkronisesti.

```
BOOL CheckIO( struct IORequest *ioRequest );
```

Tarkistaa komennon suorituksen. Palauttaa nollan, jos komento ei ole vielä valmis.

```
BYTE WaitIO( struct IORequest *ioRequest );
```

Odottaa komennon valmistumista. Voidaan kutsua esimerkiksi AbortIO():n jälkeen tai suoraan SendIO():n jälkeen toiminnan synkronoimiseksi. Huomaa, että WaitIO() myös ottaa viestin vastausportista. Sitä voidaankin käyttää myös jo valmiiksi tiedetyn I/O-requestin valmisteluun seuraavaa tehtävää varten. WaitIO() odottaa viestiä porttiin kutsumalla Wait():iä. Mikäli komento on jo valmis, ei Wait():iä kutsuta, vaan Wait() ottaa viestin portista ja palaa välittömästi.

Yleensä paras vaihtoehto I/O-toiminnan päättymisen odottamiseen on kutsua itse Wait():iä. Silloin voidaan odottaa myös esimerkiksi breakia ja ajastinta toteuttaen I/O:lle timeout-featuren. Jos aikaa kuluu liikaa, voi olla, että I/O:ssa on jotain vikaa, joten se abortoitaisiin. Syynä voisi olla vaikkapa, että kahden tietokoneen keskinäisessä tiedonsiirrossa toinen kone ei lähetä mitään, vaikka sen pitäisi. Mikäli käytetään WaitIO():ta odottamiseen (tai DoIO:ta), koko ohjelma jumittuu, koska pyyntö ei koskaan täyty eikä funktio palaa.

```
void AbortIO( struct IORequest *ioRequest );
```

Abortoi komennon. Ei odota abortin tapahtumista.

1.94 Muisti

Muistinvaraukseen on olemassa monimutkaisia funktioita, joita on myös muissa kirjastoissa kuin Execissä. Käsittelen tässä kuitenkin vain Execin muistinhallintafunktiot. Yksinkertaisimmillaan muistia varataan näin:

```
APTR mem;

if(!(mem=AllocMem(BUFFER_SIZE,MEMF_ANY))) printf("Ei saatu muistia\n");
```

Muisti vapautetaan lopuksi näin:

```
if(mem) FreeMem(mem,BUFFER_SIZE);
```

Halutun muistin tyypin määräävät AllocMem():lle annettavat liput:

MEMF_ANY	Mikä vain käy
MEMF_PUBLIC	Julkinen muisti
MEMF_CHIP	CHIP RAM
MEMF_FAST	FAST RAM
MEMF_LOCAL	Paikallista muistia, ei laajennuskortilla
MEMF_24BITDMA	24-bittisen DMA:n ulottuvissa oleva muisti
MEMF_CLEAR	Tyhjennä alue ennen palaamista
MEMF_LARGEST	Suurin alue (katso alla)
MEMF_REVERSE	Etsi sopiva alue muistin yläpäästä
MEMF_TOTAL	Koko muisti (katso alla)

Jos ei ole mitään tarvetta pyytää mitään erityistä muistia, tulee käyttää MEMF_ANY:ä. Tällöin varataan ensin FAST-muistia, jos sitä on vapaana, ellei, saat CHIPiä. PUBLIC tarkoittaa muistia, joka on toisten tehtävien saatavilla. Kaikkiin systeemistruktuureihin (esimerkiksi julkisiin viestiporteihin) käytetty muisti tulee varata PUBLIC:lla. PUBLIC-lipulla ei ole vielä merkitystä, mutta tulevaisuuden laajennuksiin tulee varautua käyttämällä sitä. Käyttöjärjestelmän ulkopuoliset virtuaalimuistiohjelmat tosin käyttävät PUBLIC-lippua - ne eivät hukkaa PUBLIC-muistia levylle, mutta se ei ole tämän lipun suunniteltu käyttötarkoitus.

LOCAL on muistia, joka on suoraan prosessorin väylällä, yleensä emolevyllä. Laajennuskorteilla oleva muisti menetetään resetin yhteydessä, eikä sitä voida käyttää resetintakaisiin viritelmiin. 24BITDMA-lippu pyytää muistia, joka on Zorro II -muistiavaruudessa. Tämä lippu on tarkoitettu ainoastaan Zorro II -korttien ohjaimien käytettäväksi. Ohjelmien ei tulisi käyttää sitä koskaan. CLEAR-lipulla muistialue voidaan pyytää nollattavaksi ennen palaamista AllocMem():stä. Normaalisti halutun suuruista aluetta etsitään muistin alapäästä, mutta REVERSE muuttaa toiminnan niin, että aluetta aletaankin hakea muistilistan lopusta.

LARGEST ja TOTAL ovat lippuja, joita ei koskaan käytetä muistia varatessa. Ne voidaan antaa AvailMem():lle, joka kertoo vapaan muistin määrän:

```
printf("Vapaata CHIP-muistia on %ld tavua\n",AvailMem(MEMF_CHIP));
```

Tätä funktiota ei yleensä tarvitse käyttää, ellei halua kertoa käyttäjälle tietoa muistista. Muistin riittävyys selviää kyllä, kun sitä yritetään varata. AvailMem():n palauttama arvo ei välttämättä ole oikein. On hyvin todennäköistä, että muistia on jo varattu ja vapautettu monta kertaa, ennen kuin palautusarvoa päästään tutkimaan. Sen tarkkuuteen ei pidä luottaa liikaa.

AvailMem() osaa kertoa vapaan muistin kokonaismäärän, jonka se palauttaa, kun sille annetaan lippu MEMF_TOTAL. Suurimman yhtenäisen vapaan tietyntyyppisen muistialueen koon saa näin:

```
printf("Suurin yhtenäinen CHIP-muistin palanen on kooltaan %ld tavua\n",
    AvailMem(MEMF_CHIP|MEMF_LARGEST));
```

Exec tarjoaa versiosta 37 alkaen lisäksi funktiot AllocVec() ja FreeVec().

Ne toimivat aivan kuin AllocMem() ja FreeMem(), mutta lisäksi AllocVec() säilöö alueen pituuden, joten sitä ei tarvitse antaa ollenkaan aluetta vapautettaessa. AllocVec() varaa aina neljä tavua enemmän kuin pyydetään ja tallentaa alueen ensimmäiseen longwordiin sen pituuden.

Muistin varaaminen ja vapauttaminen AllocVec():iä ja FreeVec():iä käyttäen käy näin:

```
APTR mem;

if(!(mem=AllocVec(BUFFER_SIZE,MEMF_ANY))) printf("Ei saatu muistia\n");

if(mem) FreeVec(mem);
```

Näillä funktioilla pärjää hyvin. Käsittelen vielä lisää muistifunktioita, mutta voit hypätä suoraan luvun loppuun, jossa on kooste aihepiirin funktioista.

Muistin siirtäminen paikasta toiseen

Joskus tulee tarve siirtää dataa paikasta toiseen. Siihen on olemassa funktioita:

```
void CopyMem(source,dest,size);
void CopyMemQuick(source,dest,size);
```

Funktioiden toiminta on identtinen, mutta CopyMemQuick() on optimoitu versio, ja sitä käytettäessä kaikkien parametrien on oltava jaollisia neljällä (data kopioidaan longwordeja siirtelemällä). CopyMem() ja CopyMemQuick() ovat tehokkaita funktioita datan siirtämiseen, vaikka ne eivät tuekaan päällekkäisten alueiden siirtoa. Toisin sanoen kohdealue ja lähdealue eivät saa olla osittain samassa paikassa. Osittain päällekkäin sijaitsevien alueiden kopioimiseen käyviä funktiota on esimerkiksi omassa kirjastossani.

Useat yhtäaikaiset muistinvaraukset

Jos tarvitset useita vieläpä erityyppisiä muistialueita, voit varata ja vapauttaa ne helposti kerralla. Siihen tarkoitukseen ovat olemassa funktiot AllocEntry() ja FreeEntry(). Näitä käyttämällä muisti myös pysyy hyvin järjestyksessä omassa listassaan. Jokaisella tehtävällä on matalalla tasolla tällainen lista, mutta on kuitenkin syytä tehdä oma, koska se on lähinnä systeemin sisäiseen käyttöön. Tosin sitä voidaan hyödyntää automaattisen muistinvapautuksen aikaansaamiseksi. Tehtävät-luvussa on lisää tietoa tästä.

Muistilista ei ole standardi Execin lista, vaan noden sisältävä yksinkertainen struktuuri, johon kuuluu yksi tai useampi MemEntry:

```
struct  MemList {
    struct  Node ml_Node;
    UWORD   ml_NumEntries;      /* Listassa olevien entryjen lukumäärä */
    struct  MemEntry ml_ME[1];  /* ensimmäinen entry */
};

struct  MemEntry {
union {
    ULONG   meu_Reqs;           /* AllocMem()-liput */
```

```
    APTR    meu_Addr;           /* Tämän alueen osoite */
    } me_Un;
    ULONG   me_Length;          /* Tämän alueen pituus */
};
```

Varsinkin MemEntry vaatinee selvittämistä. Sen pituus siis on kahdeksan tavua. Ensimmäisenä on longword, joka sisältää AllocMem():lle annettavat liput - tämän alueen vaatimukset, mikäli kyseessä on vasta muistinvaraus, joka tullaan antamaan AllocEntry():lle. Seuraavana on toinen longword, joka sisältää alueen pituuden. Osoitin MemListiin annetaan AllocEntry():lle.

Mitä seuraavaksi tapahtuu, onkin hieman erikoista. Tälle alkuperäiselle muistilistalle ei tehdä mitään, vaan AllocEntry() varaa uuden, johon se kopioi tiedot tästä listasta. Kuitenkaan tässä uudessa listassa MemEntryn ensimmäinen muuttuja ei sisällä lippuja, vaan se korvataan osoittimella varattuun muistialueeseen! AllocEntry() palauttaa osoittimen tähän uuteen listaan, ja se annetaan vapautettaessa FreeEntry():lle.

Virhetilanteessa kaikki jo mahdollisesti varatut muistialueet vapautetaan, ja AllocEntry() palauttaa epäonnistuneen varauksen liput bitti 31 asetettuna, joka testaamaalla saadaan tietää onnistuiko varaus. Varaus, sen onnistumisen tarkistus ja alueiden vapauttaminen onnistuu näin:

```
#define ALLOCERROR 0x80000000
struct MemList *ml;             /* valmiiksi alustettu MemList + MemEntryt */
struct MemList *memlist;        /* tässä vaiheessa vielä NULL */

memlist = AllocEntry(ml);

if(memlist & ALLOCERROR) printf("Ei saatu muistia\n"); else {

    ...<muistin käyttöä>...

FreeEntry(memlist);
}
```

Tehtävää vaikeuttaa MemEntry-struktuurissa käytetty unioni. Sen saamiseksi määriteltyä oikein tai yleensä ollenkaan kannattaa lukea lehden C-kurssi!
Lisää muistifunktioita

Muistia voi varata myös absoluuttisesta osoitteesta, mutta se ei missään nimessä ole suotavaa. Älä tee niin kuin ehdottomasta pakosta. AllocAbs() toimii kuten AllocMem(), ensimmäisenä parametrinä annetaan alueen pituus, mutta toisena alueen osoite:

```
APTR mem;

if(!(mem=AllocAbs(0x10000,0x40000))) printf("Ei saatu muistia\n");
```

Muisti vapautetaan aivan normaalisti FreeMem()-funktiolla. AllocAbs():lle voi olla käyttöä joillakin erityissovelluksilla, esimerkiksi järjestelmän toimintaan liittyvillä ohjelmilla tms. Normaalien ohjelmien ei tulisi koskaan käyttää tätä funktiota.

Edelleen löytyy vielä yksi pari funktioita, Allocate() ja Deallocate(). Ne käyttävät MemHeaderia ja MemChunkia:

```
struct  MemHeader {
    struct  Node mh_Node;
    UWORD   mh_Attributes;      /* ei käytetä                   */
    struct  MemChunk *mh_First; /* ensimmäinen vapaa alue       */
    APTR    mh_Lower;           /* muistin alaraja              */
    APTR    mh_Upper;           /* muistin yläraja + 1          */
    ULONG   mh_Free;            /* vapaiden tavujen määrä       */
};

struct  MemChunk {
    struct  MemChunk *mc_Next;  /* osoitin seuraavaan chunkiin  */
    ULONG   mc_Bytes;           /* tavujen määrä chunkissa      */
};
```

Käyttöjärjestelmä pitää kirjaa vapaasta muistista käyttäen näitä struktuureja. Execin muistilistan lisäksi muistia voidaan manageroida myös paikallisesti. Ideana on alustaa ensin MemHeader- ja MemChunk-struktuurit kuvaamaan käytössäsi olevaa muistialuetta. Tämän jälkeen tästä paikallisesti ylläpidetystä muistilammikosta voidaan varata muistia Allocate():lla ja vapauttaa sitä Deallocate():lla.

Tarvetta tälle ei juuri ole - itse en ole hyödyntänyt sitä koskaan. Yksi mahdollinen sovellus on se, että ohjelmasi laukaisee muita tehtäviä, ja haluat seurata niiden muistinkäyttöä ja ylläpitää omaa muistijärjestelmää. Mielenkiintoinen yksityiskohta systeemin muistinhallinnassa on, että kirjaa pidetään vain vapaista muistialueista - kun muistialue varataan, Execillä ei ole hajuakaan, kenellä se on!
Execin muistifunktiot ovat tässä:

```
APTR Allocate( struct MemHeader *freeList, unsigned long byteSize );
```

Varaa muistia yksityisesti ylläpidetystä muistilammikosta. Parametreinä funktiolle annetaan osoitin MemHeaderiin ja varattavan alueen koko.

```
void Deallocate( struct MemHeader *freeList, APTR memoryBlock,
    unsigned long byteSize );
```

Vapauttaa yksityisesti ylläpidetystä muistilammikosta varatun muistialueen. Parametrit ovat osoitin MemHeaderiin, muistialueen osoite ja koko.

```
APTR AllocMem( unsigned long byteSize, unsigned long requirements );
```

Varaa muistia, parametreinä annetaan alueen koko ja vaatimukset.

```
APTR AllocAbs( unsigned long byteSize, APTR location );
```

Varaa muistia absoluuttisesta osoitteesta. Parametreinä annetaan alueen koko ja osoite, josta muistia halutaan.

```
void FreeMem( APTR memoryBlock, unsigned long byteSize );
```

Vapauttaa AllocMem():llä tai AllocAbs():lla varatun muistialueen. Funktiolle annetaan osoitin alueeseen sekä sen koko.

```
ULONG AvailMem( unsigned long requirements );
```

Kertoo, kuinka paljon määritellyntyyppistä muistia on vapaana. Voi myös

kertoa kokonaismuistin määrän sekä suurimman yhtenäisen alueen koon.

```
struct MemList *AllocEntry( struct MemList *entry );
```

Varaa yhden tai useita muistialueita käyttäen muistilistaa. Palauttaa osoittimen uuteen muistilistaan, jossa on osoittimet varattuihin alueisiin, tai yhdenkin varauksen epäonnistuessa, epäonnistuneen varauksen vaatimukset bitti 31 asetettuna.

```
void FreeEntry( struct MemList *entry );
```

Vapauttaa yhden tai useamman muistialueen, jotka ovat muistilistassa. Listan tulee olla AllocEntry():n palauttama - ei alkuperäinen lista, jossa on osoittimien sijaan vaatimukset.

1.95 Tehtävät

Amiga ajaa ohjelmakoodia prosesseina ja tehtävinä. Myös ↩ prosessit ovat
tehtäviä, mutta ne ovat laajempia kokonaisuuksia. Käsittelen tässä nyt tehtäviä Execin näkökulmasta. Tehtävät pidetään tehtävälistoissa tällaisten datastruktuurien avulla:

```
struct Task {
    struct  Node tc_Node;
    UBYTE   tc_Flags;               /* Liput                        */
    UBYTE   tc_State;               /* Tehtävän tila                */
    BYTE    tc_IDNestCnt;           /* Keskeytysestot               */
    BYTE    tc_TDNestCnt;           /* Tehtävänvaihtoestot          */
    ULONG   tc_SigAlloc;            /* Varatut signaalit            */
    ULONG   tc_SigWait;             /* Signaalit, joita odotetaan   */
    ULONG   tc_SigRecvd;            /* Saadut signaalit             */
    ULONG   tc_SigExcept;           /* Poikkeuttavat signaalit      */
    UWORD   tc_TrapAlloc;           /* Varatut ansat                */
    UWORD   tc_TrapAble;            /* Sallitut ansat               */
    APTR    tc_ExceptData;          /* Poikkeustiladataosoitin      */
    APTR    tc_ExceptCode;          /* Poikkeustilakoodi            */
    APTR    tc_TrapData;            /* Ansadataosoitin              */
    APTR    tc_TrapCode;            /* Ansakoodi                    */
    APTR    tc_SPReg;               /* Pinorekisteri                */
    APTR    tc_SPLower;             /* Pinon alaraja                */
    APTR    tc_SPUpper;             /* Pinon yläraja + 1            */
    VOID    (*tc_Switch)();         /* Kun CPU lähtee...            */
    VOID    (*tc_Launch)();         /* Kun CPU tulee...             */
    struct  List tc_MemEntry;       /* Varattu muisti               */
    APTR    tc_UserData;            /* Käyttäjän dataosoitin        */
};
```

Tehtävään liittyy paljon tietoa. Näistä tärkeimmät ovat tuolla lopussa. Tehtävää vaihdettaessa on syytä tietää, missä sen pino on ja missä kohtaa siinä mennään (SPReg ladataan SP-rekisteriin). Switch ja Launch osoittavat koodiin niihin kohtiin, joihin hypitään, kun tehtäviä vaihdetaan. Ne tietysti vaihtuvat koko ajan, kun koodia ajetaan. Seuraavalla kerralla jatketaan siitä, mihin viimeksi jäätiin.

Tämän enempää ei välttämättä ohjelmoijan tarvitse tehtävien teknisestä toteutuksesta tietää. Käsittelen tässä luvussa vielä lisää tehtäviin liittyviä toimintoja, mutta en usko, että niitä koskaan tarvitset. Kiinnostuksesta voit toki lukea loppuunkin...

Task Creation

Task Exclusion

Task Exceptions

Task Traps

Tehtävänhallintafunktiot Execissä ovat nämä:

```
APTR AddTask( struct Task *task, APTR initPC, APTR finalPC );
```

Lisää taskin Execin ajovalmiiden tehtävien listaan. Tehtävää ruvetaan ajamaan joko heti, jos sen prioriteetti on suurempi kuin tehtävä, jota jo ajetaan, tai sitten, kun sen aika tulee. Parameterinä annetaan osoitin alustettuun Task-struktuuriin, tehtävän koodin osoite ja tehtävän cleanup-koodin osoite.

```
void RemTask( struct Task *task );
```

Poistaa tehtävän. Keskeyttää tehtävän ajamisen, vapauttaa sen varaaman muistin ja poistaa kaiken siihen liittyvän tiedon muistista.

```
struct Task *FindTask( UBYTE *name );
```

Etsii tehtävän tehtävälistoista. Tehtävä voi ottaa selville oman Task-struktuurinsa osoitteen kutsumalla FindTask(NULL):ia.

```
BYTE SetTaskPri( struct Task *task, long priority );
```

Asettaa tehtävän prioriteetin.

```
ULONG SetExcept( unsigned long newSignals, unsigned long signalSet );
```

Asettaa signaalit, joiden halutaan aiheuttavan poikkeustila.

1.96 Task Creation

Tehtäviä voi tehdä itsekin. Monet ohjelmat laukaisevat lapsitehtävän tai jopa useita. Myös laiteohjaimet ajavat jokaiselle yksikölle oman tehtävän huolehtimaan erikseen sille tulevista komennoista. Tehtävän voi tehdä joko käsin tai käyttämällä apufunktioita. Erittäin tehokas tehtävänkäynnistysfunktio löytyy omasta sh.librarystäni. Operaatio ei ole kovinkaan monimutkikas. Meidän tarvitsee alustaa Task-struktuuri ja varata tehtävälle muistia pinoa varten.

Tehtävää käynnistettäessä Exec täyttää alustamattomat kentät oletusarvoilla esimerkiksi tuoden sisään oletuskoodin poikkeustiloista selviämiseen ja ansoihin joutumiseen ym. Alustettavia kenttiä ovat vain pinomuuttujat ja node sekä muistilista, jota kannattaa tässä ehdottomasti hyödyntää. Kaikki

tehtävään liittyvä muisti (Task-struktuuri itse, pinoalue tms.) kannattaa varata AllocEntry():llä ja liittää muistilista tc_MemEntryyn, jolloin Exec vapauttaa automaattisesti kaikki muistialueet, kun tehtävän ajaminen päättyy!

Laitan tähän nyt lyhyen konekielisen ohjelman, joka tekee meille tehtävän. Ensiksi se varaa muistialueet valmiina olevan MemListin mukaan ja alustaa tehtävän listan, jossa se pitää MemEntryt, ja lisää AllocEntry():n palauttaman listan siihen. Pinoa tehtävälle varataan huimat 256 tavua, joka riitää, kun se ei tee mitään. Oikealle ohjelmalle pinoa tarvitaan 4000 tavua.

Ohjelma myös kopioi tehtävän nimen Task-struktuurin perään sekä koodin pinoalueen perään! Näin tämä isäntäohjelma voi exitoida ja jättää lapsensa pyörimään. Jos lapsen koodia ajettaisiin isännän sisällä, ei sitä voitaisi lopettaa ennen lapsen tappamista. Tietysti tällä tavalla kopioitava koodi on oltava PC-relatiivista. Kirjastossani oleva tehtävänkäynnistysfunktio toimii samaan tapaan, ja se osaa myös varata data-alueen tehtävälle.

Lopuksi tehtävä käynnistetään kutsumalla AddTask()-funktiota. Sille annetaan osoitin Task-struktuuriin sekä initialPC ja finalPC. Näistä ensimmäinen osoittaa tehtävän koodin alkuun, ensimmäiseen käskyyn, jonka uusi tehtävä suorittaa. Jälkimmäinen osoite työnnetään pinoon eli siihen hypätään, jos tehtävä joskus suorittaa RTS-komennon. Jos se on nolla, Exec tuo siihen oletuskoodin osoitteen. Oletuskoodi vain kutsuu RemTask()-funktiota, joka poistaa tehtävän, mutta finalisaatiokoodi voi suorittaa muitakin cleanup-toimenpiteitä.

Ohjelma on kauan sitten opetustarkoituksessa kirjoittamani esimerkki.

```
;
; Task v1.02! Written by Sami Klemola.
;             Finished on Sunday the 3rd of March at 19:40.
;             Commented on Sat 25-May-91 at 12:40.
;             Final adjustments made on Mon 17-Jun-91 at 13:00.
;
; Copyright 1991 by Sami Klemola. All Rights Reserved.
;

      include "Macros"
      include "exec/memory.i"
      include "exec/tasks.i"

      movea.l 4,a6
      cmpi.b #'0',(a0)
      beq.s RTask
      cmpi.b #'1',(a0)
      bne.s exit
      lea MList(pc),a0
      Lib AllocEntry                    ; varataan muisti
      bmi.s exit                        ; pois, jos ei saatu
      movea.l d0,a4
      movea.l ML_ME(a4),a5
      move.l ML_ME+ME_LENGTH(a4),d2
      lea TC_MEMENTRY(a5),a0            ; alustetaan tc_MemEntry
      NEWLIST a0
      movea.l a4,a1                     ; ja lisätään MemList siihen
```

```
      Lib Enqueue
      move.l d2,TC_SPLOWER(a5)          ; pinon alaraja (varatun muistin
      addi.l #$100,d2                   ; alkuosoite, pino kasvaa alaspäin)
      move.l d2,TC_SPUPPER(a5)          ; pinon yläraja (alkuosoite + 256)
      move.l d2,TC_SPREG(a5)            ; pino-osoitin (sama kuin yläraja)
      lea Start(pc),a0                  ; tehtävän koodi
      movea.l d2,a1
      movea.l d2,a2                     ; koodin osoite a2:een AddTaskille
      moveq #MList-Start-1,d2
tcopy move.b (a0)+,(a1)+                ; kopioidaan koodi pinoalueen perään
      dbf d2,tcopy
      move.b #NT_TASK,LN_TYPE(a5)       ; nodetyyppi task
      move.b #$80,LN_PRI(a5)            ; prioriteetti -128
      lea TName(pc),a0
      lea TC_SIZE(a5),a1
      move.l a1,LN_NAME(a5)
ncopy move.b (a0)+,(a1)+                ; kopioidaan tehtävän nimi
      bne.s ncopy                       ; struktuurin perään
      movea.l a5,a1                     ; task-struktuurin osoite
      suba.l a3,a3                      ; oletuslopetuskoodi
      Lib AddTask                       ; lisäätän tehtävä tehtävälistaan
exit  rts
RTask lea TName(pc),a1                  ; etsitään tehtävä
      Lib FindTask
      movea.l d0,a1
      beq.s error
      Lib RemTask                       ; ja lopetetaan sen ajaminen
error rts

Start clr.l d0                          ; tehtävän ohjelmakoodi
Cont  move.w d0,$dff180                 ; kirjoitetaan d0 COLOR00:aan
      addi.w #$1,d0                     ; lisätään d0:aan 1
      bra.s Cont                        ; ja uudestaan

MList dc.l 0,0                          ; MemList
      dc.b NT_MEMORY,0
      dc.l TName
      dc.w 2                            ; MemEntryjen määrä

      dc.l MEMF_PUBLIC!MEMF_CLEAR       ; Task-struktuuri + tila nimelle
      dc.l TC_SIZE+10

      dc.l MEMF_CLEAR                   ; Pino + koodi
      dc.l $100+MList-Start

TName dc.b 'ExtraTask',0                ; tehtävän nimi
```

1.97 Task Exclusion

Edistynyt järjestelmäohjelma voi joskus havaita tarvitsevansa pääsyä johonkin globaaliin datastruktuuriin. Moniajosta johtuen datat voivat kuitenkin muuttua kesken kaiken. Tarvitaan jonkinlainen keino sen estämiseksi. Tehtävä voi hetkellisesti estää moniajon, mutta se tulee palauttaa välittömästi, kun datat on luettu.

Moniajo estetään kutsumalla funkiota Forbid(). Muita tehtäviä ei ajeta, enen kuin kutsut funktiota Permit(), joka jälleen sallii moniajon. Mikäli kutsut Wait()-funktiota suoraan tai epäsuorasti Forbid():n jälkeen, odottaminen katkaisee eston, ja muita tehtäviä ajetaan odottaessasi. Kun odotettu signaali saadaan, Wait() palaa ja moniajo estetään uudelleen. Keskeytykset ajetaan normaalisti myös moniajon ollessa kiellettynä.

Toinen mahdollisuus on disablointi kutsumalla funktiota Disable(). Disable() kieltää keskeytykset, joten moniajokaan ei toimi. Keskeytykset sallitaan jälleen funktiolla Enable(). Keskeytyksiä ei pitäisi kieltää yli 250 mikrosekunnin ajaksi kerrallaan, koska Amigan käyttöjärjestelmä on hyvin riippuvainen ajallaan tapahtuvista keskeytyksistä. Erityisesti serial.device tykkää kyttyrää, ellei se pääse lukemaan merkkejä ajoissa - ne menetetään ikuisesti.

Forbid() ja Disable() ovat kasaantuvia funktiota. Jos kutsut niitä useamman kerran, joudut kutsumaan myös Permit():iä tai Enable():a yhtä monta kertaa. Kutsuista pidetään kirjaa Task-struktuurin NestCount-muuttujissa. Koskaan ei ole tarvetta kutsua sekä Forbid():iä että Disable():a. Disable() estää keskeytykset, joten se kieltää myös moniajon, koska Execin tehtävänvaihto tapahtuu keskeytyksessä.

Joskus on tarpeen käyttää näitä funktiota, mutta ne tilanteet ovat harvassa. Normaaleilla ohjelmilla niitä ei pitäisi tulla ollenkaan. Useimmat kirjastot tarjoavat erityisiä funktioita toisten tehtävien sulkemiseksi pois tietystä järjestelmästä. Näitä ovat #?Lock#?()-funktiot, jotka hetkellisesti lukitsevat halutun datan, jotta se voidaan rauhassa lukea, estämättä moniajoa tai keskeytyksiä ja näin ollen systeemiystävällisemmin.

On olemassa vielä yksi tapa, jota voidaan käyttää, kun kyseessä ei ole systeemidata, vaan jonkin ohjelman omat datat. Jos jotkin muutkin tehtävät haluavat päästä käsiksi dataan, voidaan käyttää opastimia. Semaphoret ovat käteviä ohjelmien kesken tapahtuvaan tiedon ristiosoitukseen. En kuitenkaan käsittele niitä tässä, koska niille ei yleensä ole tarvetta.

1.98 Task Exceptions

Poikkeustilat ovat ohjelmallisia keskeytyksiä, jotka aiheutuvat tiettyjen signaalien aktivoituessa. Execin "exception":lla ei ole mitään tekemistä Motorolan "exception":n kanssa. Tällä tasolla jälkimmäiset tunnetaan "Task Trap":eina. Normaalisti ohjelma odottaa signaaleja Wait()-funktiolla, mutta SetExcept()-funktiolla voidaan asettaa tietyt signaalit aikaansaamaan poikkeustila.

Tällä tavalla toimiessa signaaleja ei tarvitse odottaa, vaan suoritus siirtyy automaattisesti poikkeustilakoodiin, kun haluttu signaali aktivoituu. Tätä tarkoitusta varten tulee toimittaa erityinen poikkeustilakoodi (tc_ExceptCode). Tähän koodiin hypätään poikkeustilan sattuessa. Koodille annetaan D0:ssa signaalimaski, jossa ovat päällä saatuja signaaleja vastaavat bitit. Nämä tulee myös palauttaa D0:ssa. A6:ssa on SysBase ja A1 osoittaa poikkeustiladataan (tc_ExceptData).

Varsinainen keskeytys poikkeustila ei ole. ExceptCode ajetaan normaalissa tilassa ja tehtävän pinoa käyttäen. Ennen koodiin hyppäämistä Exec työntää kaikki rekisterit pinoon. Koodin tulee tarkistaa, mitkä signaalit aiheutti-

vat poikkeustilan, ja toimia sen mukaan käsitellen kaikki saadut signaalit. Poikkeustilakoodin ajaminen päättyy RTS-käskyyn. Tämän jälkeen Exec palauttaa alkuperäisen tilanteen, ja tehtävän suoritus jatkuu normaalisti.

Poikkeustilojen käyttäminen on vaarallista! ExceptCodeen hypätään välittömästi, kun poikkeustilan aiheuttavaksi määrätty signaali aktivoituu, mikä tarkoittaa sitä, että näin voi käydä kesken kriittisen koodin. Seurauksena voi olla tehtävän virhetoiminta. Mikäli tehtävä suorittaa juuri esimerkiksi jotain systeemifunktiota, se voi olla lukinnut jonkin resurssin. Jos kutsut samaa tai vastaavaa funktiota myös ExceptCodesta, voi tuloksena olla deadlock. ExceptCodesi odottaa saavansa resurssin, joka on keskeytetyllä tehtävälläsi, joka ei sitä siten voi palauttaa, eikä se sitä koskaan saa.

1.99 Task Traps

Ansat ovat prosessorin "exception":eja. Ansat toimivat samalla tavalla kuin poikkeustilatkin. Kun trappi aiheutuu, hypätään koodiin, johon osoittaa tc_TrapCode. Execin oletustrappikoodi näyttää alertin kertoen, mikä oli trapin aiheuttaja. Mikäli trappinumeron bitti 31 on asetettu, on kyseessä korjaamaton virhe, joka johtaa väistämättä reboottiin.

Tehtävä voi toimittaa oman koodin hoitamaan trapit, jolloin vakavistakin virheistä voidaan selvitä. Trapit voivat olla myös ohjelmiston virheitä tai tarkoituksella koodista TRAP-komennolla aikaansaatuja hyppyjä. Trappikoodi ajetaan Supervisor-moodissa SysStackissa. Pinoon työnnetään "exception":n numero sekä prosessorikohtainen "exception frame".

TrapCodesta palataan RTE-käskyllä. Huomaa poistaa trappinumero pinosta ennen palaamista. Ovelalla manipuloinnilla voidaan trappihandler tehdä niin, että se siirtää kontrollin oletushandlerille, jos trappi ei ole se, jota se odottaa. Minun mielestäni olisi kyllä järkevintä tehdä handler, joka osaa selvitä kaikista trapeista.

Trapeista selviäminen ei välttämättä aina ole helppoa, mutta useimmiten se on mahdollista. En kuitenkaan tässä käsittele aihetta enempää, koska se ei varsinaisesti kuulu tämän alkeiskurssin piiriin. Hardwaretason "exception":t on mahdollista hoidella myös matalammalla tasolla koukkimalla prosessorin vektorit. Se ei sitten enää kuuluu minkäänlaisen järjestelmäohjelmointikurssin aiheisiin.

1.100 Tulevaisuutta kohti

Kurssin tämä osa loppuu valitettavasti jo tähän juuri, kun kaikki alkoivat päästä vauhtiin. Aivan kaikkea ei ehditty käsitellä. Tuleviin osiin jäivät vielä esimerkiksi semaphoret ja keskeytykset, jotka eivät välttämättä aivan alkeiskurssin asiaa olekaan. Jotkin asiat tässä osassa on käsitelty aika ylimalkaisesti. Kurssiohjelmaan kuuluu jatkossa täydentäviä osia.

Kovin vähälle jäi lähdekoodin osuus. Esimerkkejä tuli paljon, mutta kunnollista ohjelmakoodia ei yhtään. Tilaa sille ei oikeastaan jäänyt. Kannattaa haeskella BBS:istä lähdekoodeja ja ohjelmointiohjeita. Ainakin The Spitissä

on paljon ohjelmien lähdekoodeja. Ohjelmoinnista on myös olemassa ihan opetusmielessä julkaistuja ohjelmanpätkiä, joissa on ohjeita mukana.

1.101 Halpoja ja ilmaisia

Halpoja ja ilmaisia

Janne Siren

PD-pelejä on Amigalle kuin sieniä sateella, joten arvosteltavaa riittää. Edellisen numeron tapaan tutustumme jälleen muutamiin peleihin. Kaikki artikkelin pelit on testattu neljän megan lisämuistilla ja 68881-matikkalutikalla varustetussa Amiga 1200:ssa. Ne kaikki voidaan asentaa myös kovalevylle.

Battle Cars

Car v2.00

Egyptian Run v1.1

Giger Tetris AGA

Legend of Lothian v1.02

Tomcat

Kaikki Sakun koordinaattorit eivät voi rajallisten resurssiensa takia lähetellä ja etsiskellä PD-pelejä lukijoille, joten ennen kuin saamme Sakun PD-palvelun toimintaan, turvautukaa mieluummin modeemiin tai PD-pelejä myyviin postimyyntiliikkeisiin, jos näitä pelejä haluatte päästä pelaamaan.

1.102 Battle Cars

Battle Cars

Pudotetaan kaksi autoa ajajineen suureen kumiseen laatikkoon, jonka sisällä on kaksi pienempää laatikkoa. Asennetaan autoihin vielä kaksi järeää asetta ja annetaan palaa. Lopputuloksena toinen autoista tuhoutuu ja toinen selviää kuskeineen voittajaksi. Näin voisi Battle Carsia lyhyesti kuvata.

Battle Cars on ajajan näkökulmasta vektorigrafiikalla kuvattu autopeli, jossa ajetaan autoa ja ammuskellaan toista autoa yksinkertaista sokkeloa kiertäen. Auton ohjaus on muuten kohdallaan, mutta kumma kyllä ajoneuvo kimpoilee seinistä kuin ne olisivat kumia. Kunnon törmäyksiä seiniin jää kaipaamaan.

Peliä voi pelata joko tietokonetta tai toista pelaajaa vastaan. Kahden pe-

laajan peliin tarvitaan kaksi Amigaa ja niiden väliin nollakaapeli. Tietokonetta vastaan pelatessa kone tyytyy ajelemaan ennalta määriteltyä reittiä aseet hiljaisina, ja pelaajan tehtäväksi jää yrittää saada sen auto tuhottua ennen aikarajan umpeutumista.

Grafiikka on yksinkertaista mutta erittäin nopeaa ja toimivaa. Battle Cars ei moniaja, ja AGA-koneissa hiiren osoitin pitää vaihtaa lores-tilaan ennen pelaamista tähtäimen sekoilujen välttämiseksi. Mukavaa ajanvietettä.

Battle Cars

Tekijä: David Jameson

Pelityyppi: Toiminta/Autopeli
Pelaajia: 1-2

Yleisarvosana: 85%

1.103 Car v2.00

Car v2.00

Vroom! DTM-vakioautojen näillä näkymin hurjastellessa Helsingin kaduilla kesän alussa autourheilu on entistäkin enemmän muodissa. Car on ylhäältäpäin kuvattu autopeli. Pieni auto kaahaa hurjaa vauhtia pelaajan ohjaksissa usean ruudun alueella pehmeästi vierivällä radalla läpi nopeiden suorien ja jäisten kurvien.

Carin vahvuus on sen erinomainen pelattavuus. Auto tottelee mukisematta joystickin heiluttelua, ja pelaaminen on nautinto. Pelattavaakin riittää, kun pelissä on useita erilaisia ratoja valittavana. Tietokoneen ohjaamia vastapelaajia jää silti kaipaamaan.

Car, kuten Legend of Lothian, kärsii iästään. Peli ei kyllä kaatuile tai muutenkaan ole yhtään sen epäpelattavampi uudemmalla koneella, mutta A1200:lla Car aiheuttaa kummia väliaikaisia sekoiluja, joita ei vuosia sitten A500:lla näkynyt.

Car v2.00

Tekijä: Anders Bjerin

Pelityyppi: Autopeli
Pelaajia: 1

Yleisarvosana: 88%

1.104 Egyptian Run v1.1

Egyptian Run v1.1

Egyptian Run on hyvin yksinkertainen toiminta- ja autopelin risteytys. Pelaaja ohjaa takaa yläviistosta kuvattua autoa, joka huristelee hurjaa vauhtia läpi erämaan väistellen pieniä pyramideja ja pomppien hyppyreistä. Tarkoituksena on päästä horisontissa häämöttävälle kontrollipyramidille ja tuhota se ohjuksella. Aikansa tätä pelailee, mutta sitten siihen kyllästyy. Vaikeusastekin on säädetty hivenen liian korkealle, tai sitten nopeammalla koneella on jotain tekemistä asian kanssa...

Egyptian Run on vuodelta 1987, mutta toimii moitteetta uudemmalla kokoonpanollani ja moniajaa. Jos peliä tulee pelailtua paljon, shareware-periaatteen mukaisesti siitä tulisi maksaa tekijälle kymmenen taalan korvaus.

Pelin takana on sama Chris Hames, joka pari vuotta tämän pelin julkaisun jälkeen alkoi niittää mainetta ja kunniaa erinomaisen PC-Task-emulaattorin tekijänä. Noh, jostainhan sitä pitää aloittaa. Egyptian Run löytyy Fish-levyltä 120.

Egyptian Run v1.1

Tekijä: Chris Hames

Pelityyppi: Toiminta/Autopeli
Pelaajia: 1

Yleisarvosana: 75%

1.105 Giger Tetris AGA

Giger Tetris AGA

Giger Tetris oli yksi ensimmäisiä AGA-pelejä. Mitään todella AGA:aa siinä ei kuitenkaan ole. Peli avaa 256-värisen ruudun, jota koristaa yläreunassa jokseenkin värikkäällä fontilla kirjoitettu pelin nimi, ja itse pelilaatikon taustalla on harmaasävykuva. Kovin värikkäältä peli ei siis näytä. Taustalla soi letkeä musiikki.

Peliä ohjataan näppäimistöltä ja pelattavuus on kohdallaan. Peli ei kuitenkaan moniaja eikä tarjoa mahdollisuutta kaksinpeliin, joten paljon parempiakin Tetris-klooneja on tarjolla - pelkkä 256-värinen tausta ei tee pelistä hyvää. En myöskään löytänyt Giger Tetriksestä mahdollisuutta palata Workbenchiin.

Giger Tetris AGA

Pelityyppi: Tetris
Pelaajia: 1

Yleisarvosana: 71%

1.106 Legend of Lothian v1.02

Legend of Lothian v1.02

Mercian kuningaskunnassa ovat asiat sekaisin. Kuningas Lothian on loitsittu pysyvään uneen ja kaaos päässyt valloilleen. Sinut, köyhä paimen, on valittu löytämään avain taian purkamiseen ja kuningaskunnan pelastamiseen.

Legend of Lothian muistuttaa hyvin paljon varhaisia Ultima-pelejä. Perinteisen karttaikkunan lisäksi ruudulla on omat luukkunsa mm. hirviöiden kuville ja komentoikoneille. Tärkeimmät tapahtumat käydään siis vajaan puolen ruudun kokoisessa ikkunassa, jossa pelimaailma näkyy ylhäältäpäin. Mittakaava vaihtelee ulkomaailmakartan ja kaupunkien välillä.

Kartalla näkyvät vain rakennukset ja maasto. Hirviöt ilmestyvät tyhjästä, mikä on hieman ärsyttävä piirre. Flee-napin painaminen on paljon mielenkiinnottomampaa kuin kartalla hirviöiltä pakoon juokseminen... Kaupunkien kaduilla sentään liikuskelee (tai paremminkin seisokelee) myös muita ihmisiä, joiden kanssa voi keskustella. Keskustelut ovat hyvin yksinkertaisia; yleensä ihmisillä on vain pari lyhyttä riviä sanottavaa.

Peliä voi pelata hiirellä tai näppäimistöltä. Näppäimistötoistoa ei ole, joten edessä on armoton nuolinäppäinten hakkaaminen tai hiiren näpytys.

Legend of Lothian on jo muutaman vuoden ikäinen, ja se on kehitetty Amiga 1000:lla. Ikä näkyy teknisissä ratkaisuissa, jotka aiheuttavat ongelmia uusilla malleilla. Esimerkiksi AGA-koneiden hiiren osoittimen ollessa hirestilassa kaikki spritet pelissä kokevat saman ilmiön kuin laajakangaselokuvat televisiossa lopputekstien aikana - ne kaventuvat puoleen. Legend of Lothian moniajaa, mutta muuten mukavan pelin pilaa jatkuva kaatuileminen.

Legend of Lothian v1.02

Tekijä: David W. Meny

Pelityyppi: Roolipeli
Pelaajia: 1

Yleisarvosana: 79%

1.107 Tomcat

Tomcat

Full Motion Video (FMV, videokuvaa tietokoneruudulla) on muodostunut varsinaiseksi muoti-ilmiöksi romppuasemien lyötyä itsensä läpi. Full Motion Video on suuren tilantarpeensa vuoksi aika harvinainen ilmestys levykepeleissä, varsinkin PD-peleissä. Tomcat olikin iloinen yllätys, koska se rakentuu elävän videokuvan ympärille.

Grumman F-14 Tomcat kehitettiin tuhoamaan Yhdysvaltain laivastosaattueita uhkaavat vihollisen lentokoneet, ennen kuin ne pääsisivät tarpeeksi lähelle laukaistakseen risteilyohjuksensa saattueen aluksia kohden. Tomcat-peli on yksinkertaistanut koneen tavalliseksi hävittäjäksi, jonka tehtävänä on tu-

hota ympärillä kieppuvat viholliset.

Ensimmäisenä valitaan joystickilla waypoint. Esittelyversiossa on vain yksi waypoint, suunta taisteluareenalle, mutta rekisteröidyssä versiossa tekemistä löytyy monipuolisemmin.

Kun waypoint on valittu, peli jatkuu sen mukaisesti. Taisteluareenalla valitaan waypointin tyyliin ylhäältä päin katsoen FMV-taustan päälle piirretyistä ikoneista haluttu kohde. Tarkoituksena on tuhota kaikki kohteet eli kuusi Mig-hävittäjää.

Kun kohde on saatu onnistuneesti valittua, päästään hurjaa vauhtia liikkuvaan kuvakulmaan vihollisen taakse. Nyt ohjus pitäisi saada lukittua ja vihollisen kone pudotettua taivaalta. Jos tässä epäonnistuu, joutuu itse riistan osaan ja vihollinen pitäisi saada karkotettua - joko tuurilla tai pudottamalla rajallista flarea.

Sama jatkuu, kunnes kaikki viholliset on tuhottu tai oma kone saa osuman. Peli itsessään, ainakin esittelyversio, on hyvin yksinkertainen ja pidemmän päälle varsin tylsä, mutta videopätkät sopivat peliin hyvin. Ne ovat tyylikästä harmaasävygrafiikkaa ja mikä parasta, noin neljännesruudun kokoisina erittäin nopeita.

Kuten jo yllä tulikin ilmi, Tomcat on sharewarea, ja siitä saa viidellätoista US-dollarilla täydellisen version. Tomcat on ohjelmoitu AMOSilla ja moniajaa. Pelin esittelyversiokin vie useita megatavuja tilaa, joten se vaatii kovalevyn.

Tomcat

Tekijä: John B. Graham

Pelityyppi: Toiminta
Pelaajia: 1

Yleisarvosana: 80%

1.108 Pelivinkkeli

Pelivinkkeli

UFO - The Enemy Unknown

Pelivinkkeli on Sakun uusi vinkkipalsta pelaajille. Pelivinkkeliin ↩ lukijat voivat lähettää omia vinkkejään ja myös avunpyyntöjä pulmallisiin tilanteisiin. Lähetä vinkkisi seuraavaan osoitteeseen postitse.

Jani Höglund / SAKU
Tanotorventie 14-16 D 19
00420 Helsinki

Ensimmäisessä Pelivinkkelissä on niukanlaisesti vinkkejä, mutta eiköhän

vauhtiin päästä, kun myös te lukijat pääsette vinkkejänne postittelemaan.

1.109 UFO - The Enemy Unknown

UFO - The Enemy Unknown

Vinkkipalsta aloittaa täysin ufoilla vihjeillä. Muokkaamme Microprosen UFO - The Enemy Unknown -pelin talletustiedostoja. Varoitus: Jos muuttelu menee liiallisuuksiin, peli-ilokin karisee...

Tarvitset sopivan heksaeditorin, kuten Timo Rossin Zapin. Taulukossa paikka tarkoittaa muutettavan arvon sijaintia tiedostossa.

Varastot: lataa esim. Game_1-hakemistosta (tallennuspaikka 1) tiedosto base.dat ja katso oheisesta taulukosta, miten edetä.

Engineer- ja Scientists-kohtiin voit laittaa arvoksi korkeintaan FF (255 kpl). Muihin kohtiin, joissa on kaksi lukua, voi laittaa korkeintaan 7F FF eli 32 767 kappletta. Luku FF FF antaa -1 kappaletta tuotteita, eli hieman liian vähän. Esineitä voi näin saada enemmän kuin mitä varastoissa on tilaa, mutta sitten et voi enää kyseiseen tukikohtaan siirtää mitään tavaraa. Jos esim. lastaat avarusalukseen vääriä aseita, et saa niitä takaisin, koska varastotilaa ei ole.

Taulukon sijainnit ovat ensimmäiselle tukikohdalle.

Paikka	Tuote
5E	Engineers
5F	Scinetists
60-61	Stingray Launchers
62-63	Avalanche Launchers
64-65	Cannons
66-67	Fusion Ball Launchers
68-69	Laser Cannons
6A-6B	Plasma Cannons
6C-6D	Stingray Missiles
6E-6F	Avalanche Missiles
70-71	Cannon Round
72-73	Fusion Balls
74-75	Tank/Cannon
76-77	Tank/Rocket Launcher
78-79	Tank/LAser Cannon
7A-7B	Hovertank/Plasma Gun
7C-7D	Hovertank/Rocket Launcher
7E-7F	Pistols
80-81	Pistol Clips
82-83	Rifles
84-85	Rifle Clips
86-87	Heavy Cannon
88-89	Heavy AP Ammo = AP Armour Piercing (lävistävät haarniskat)

```
8A-8B   Heavy He Ammo = HE Heavy Explosive
8C-8D   Heavy I Ammo = Incinerator (liekehtivää)
8E-8F   Auto Cannon
90-91   Auto AP Ammo
92-93   Auto HE Ammo
94-95   Auto I Ammo
96-97   Rocket Launchers (raketinheittimiä sotilaille)
98-99   Rokects - Small
9A-9B   Rockets - Large
9C-9D   Rockets - Incendiary
9E-9F   Laser Pistols
A0-A1   Laser Rifles
A2-A3   Heavy Laser
A4-A5   Grenades
A6-A7   Smoke Grenades
A8-A9   Proximity Grenades
AA-AB   Explosives (110 vauriota, tappaa kaiken)
AC-AD   Motion Scanners (turhia kapistuksia)
AE-AF   Medi-Kit
B0-B1   PSI-Amps
B2-B3   Stun Rod (turhia, Stun Bomb on tehokkaamoi ja turvallisempi)
B4-B5   Electro-Flares (yötaisteluihin)
C2-C3   Heavy Plasmas (tehokkaimmat käsiaseet, suosittelen)
C4-C5   Heavy Plasma Clip
C6-C7   Plasma Rifles
C8-C9   Plasma Rifle Clips
CA-CB   Plasma Pistol
CC-CD   Plasma Pistol Clips

CE-CF   Blaster Launchers
D0-D1   Blaster Bombs
D2-D3   Small Launcher : Näillä saat kaapattua alienit. Sinulla pitää olla
D4-D5   Stun Bombs     : Alien Contaiment tai mielellään kaksi tukikohdassa
                       : johon taistelun jälkeen palaat.

D6-D7   Alien Grenades
D8-D9   ELERIUM-118 (näitä ei voi itse valmistaa, mutta tarvitsee paljon)
DA-DB   Mind Probes

E2-E3   Sectoid Corpse (ruumiita tarvitset vain yhden jokaista, tutki tämä)
E4-E5   SnakeMan Corpse
E6-E7   Etheral Corpse
E8-E9   Muton Corpse (vihreitä miehiä, vaarallisia)
EA-EB   Floater Corpses
EC-ED   Celatid Corpses
EE-EF   Silacoid Corpses
F0-F1   Chyssalid Corpses (mustia kavereita, älä ole kättelyetäisyydellä)
F2-F3   Reaper Corpses
F4-F5   Sectopod Corpses
F6-F7   CyberDisc Corpses (pikku-ufoja, ammu kaukaa, tuhoutuvat räjähtäen)

100-101 UFO Power Source
102-103 UFO Naviagion
104-105 UFO Construcion
106-107 Alien Food
108-109 Alien Reproducion
10A-10B Alien Entertainement
```

10C-10D Alien Surgery
10E-10F Examination Room

110-111 Alien Alloys (näitä tarvitsee paljon)
112-113 Alien Habitat
114-115 Personal Armour
116-117 Power Suit
118-119 Flying Suit (jokaiselle sotilaalle tälläinen)
11A-11B HWP Cannon Shells (HWP:t ovat tankeille)
11C-11D HWP Rockets
11E-11F HWP Fision Bombs

Jos rahat tuntuvat loppuvan kesken, lataa editoriin tiedosto liglob.dat samasta hakemistosta ja muokkaa neljä ensimmäistä lukusarjaa muotoon 7F FF FF FF, ja rahaa on enemmän kuin tarpeeksi. Tosin tällöin et saa saada enempää rahaa, tai rahasi menevät miinuksille. Suositeltavampi arvo olisi jokin hieman pienempi, kuten 70 00 00 00.

Siinä olikin listaa kerrakseen. Kerrottakoon vielä, että pelin läpäisemiseksi tulee kaapata kolme Etheral Leaders -ötökkää.

Jani Höglund

1.110 Maukasta ja maittavaa, Ison-Wäiskin laittamaa

Maukasta ja maittavaa, Ison-Wäiskin laittamaa

Maaliskuu, 1995

Liha-perunasoselaatikko

6-8 annosta	perunamuusijauhetta (yksi paketti)
500 g	jauhelihaa
3 kpl	sipulia silppuna
loraus	ketsuppia
pinnalle	juustoraastetta

Laita uuni kuumenemaan 175-asteiseksi. Ruskista sipulit ja kaada sitten loraus ketsuppia niiden päälle. Sekoita tasaiseksi seokseksi ja siirrä johonkin astiaan. Ruskista jauheliha ja valmista perunamuusi. Peitä uunivuoan pohja noin puolella perunamuusista. Kaada jauheliha ja sipuli-ketsuppiseos päälle tasaisesti. Peitä lopulla muusilla jauheliha ja kuorruta pinta kevyesti juustoraasteella. Laita uuniin noin 15 minuutiksi, jotta juusto sulaisi ja saisi kunnon rusketuksen.

Jani 'Wäiski' Höglund

1.111 Amigazen tuotehinnasto - maalis-huhtikuu 1995

T:MI AMIGAZE TUOTEHINNASTO MAALIS-HUHTIKUU 1995

Pidätämme oikeudet hinnanmuutoksiin. Kaikki hinnat sisältävät ALV:n.

MUKANA MYÖS PC-TUOTTEET!

Hintatakuu (Amiga-tuotteet): jos joku myy halvemmalla,
todista se ja pyrimme myymään laitteen vähintään samaan hintaan.

Jos etsimääsi tuotetta ei löydy, soita ja kysäise. Valikoima
kehittyy koko ajan.

TIETOKONEET

486-mikrot ja Pentiumit tappohinnoilla. Rakennamme paketit
yksilöllisesti, joten soita ja kerro tarpeesi - saat tarjouksen
juuri tarvitsemastasi laitteistosta!

RAJOITETTU ERÄ CD32-PELIKONSOLEITA 2 PELIN KERA! VAIN 1 990,-

Uusien Amigoiden saatavuudesta ei vieläkään ole tietoa. Siksi
Amigazen kautta kulkee käytettyjä koneita. Ostamme, vaihdamme
ja myymme käytettyjä Amiga 1200-, 3000- ja 4000-tietokoneita.
Ostamme myös käytettyjä lisälaitteita ja muuta tavaraa. Tarjoa!

Tavara vaihtuu tiuhaan, mutta tässä pari täkyä:

Princeton Ultra 16" Multisync-monitori, mukana kääntyvä jalusta.
Näyttää kaikki AGA-tilat, hyvä kuva ja ISO putki. Lisäksi kuvan saa
venytettyä niin isoksi, ettei sururaitoja jää! 2 500,-

Lisälevyasema Roctec slim. Paras levari. 290,-

Commodore 1942 Multisync-monitori. Näyttää kaikki AGA-tilat,
vasta n. vuoden vanha, uudenveroinen! VAIN 2 400,-

MONITORIT

Microvitec 1438-monitori 3 150,-
Testeissä hyvät arvostelut saanut huippumonitori. Microvitec on
aito Multisync, pikselikoko 0.29 ja mukana VGA-adapteri ja käännettävä
jalusta. Erittäin korkeatasoinen näyttö. Näyttää luonnollisesti
kaikki AGA-tilat.

TURBOKORTIT + LISÄMUISTIT

TÄMÄN HALVEMMALLA EI AMIGAAN OLE SAANUT POTKUA IKINÄ AIKAISEMMIN!

A1200:
Blizzard 1220/4 Turbo + muistikortti Amiga 1200:lle 2 090,-
Sis. 68EC020/28 MHz, 4 MB RAM, kello, paikka FPU:lle

Blizzard 1220/8 - muistia 8 megaa 3 700,-

RCA 1220/4 2 090,-

RCA 1220/8 3 090,-
Kuin Blizzard 1220/8, mutta:

RCA 1220:lla on ongelmia toimia tiettyjen kiintolevyjen kanssa, esim. Conner, Seagate FastATA ja Toshiba. Blizzardilla ei vastaavia ongelmia ole havaittu. Jos levysi toimii RCA:n kanssa, ei ongelmia ule. RCA käyttää normaaleja 32-bit simmejä, ja Blizzard omaa muistinlaajennustaan. RCA on siis halvempi laajentaa 8-megaiseksi.

Blizzard 1230-III Turbo 2 900,-
Sis. 68EC030/40, 4 MB RAM, paikka FPU:lle, kello.
Ilman muistia 1 900,-

Blizzard 1230-III Turbo 3 300,-
Sis. 68030/50, 4 MB RAM, paikka FPU:lle, kello.
Ilman muistia 2 300,-

Blizzard A1230-III Fast SCSI II ohjain edellisiin 750,-

68882-50 FPU PGA-kannalla 990,-

MYÖS GVP:N JAWS:T SAATAVANA.

RCA-muistikortit Amiga 1200:lle:

RCA/2 MB, sis. kello, paikka FPU:lle 1 300,-

RCA/4 MB 1 700,-

RCA/8 MB 2 650,-

A4000:
Blizzard 4030 Turbo (A4000), 68030/50 1 990,-

Cyberstorm 040/40 7 190,-

Cyberstorm 040/40 + 4 MB RAM 8 190,-

MAALISKUUN JYMYPAUKKU!

Amigan ensimmäiset, huipputehokkaat Cyberstorm 060/50 turbot tulevat maahan maaliskuussa. Tehoa on SysInfon mukaan järisyttävät 39,85 MIPS ja 27,79 MFLOPS! Turbon normaalihinta tulee olemaan FIM 9700,- 4 megatavun muistilla, mutta seuraava tarjous on voimassa 7.3.1995 kello 14:00 asti:

Cyberstorm 060/50, sis. 68060/50 + 4 MB 32-bittistä muistia ennakkomaksulla VAIN -> -> 8990 FIM <- <- !!!!

Tilausohje: soita Amigazen tilausnumeroon (952) 2282 223 lauantaina, 04.03. kello 20:00 jälkeen, sunnuntaina 05.03. tai maanantaina 06.03. Myöhäisille Mateille myös viimeinen tilausmahdollisuus: soita armeijan numeroon (952) 3440 194

tiistaina, 07.03. kello 10:00 - 11:30 tai 12:30 - 14:00
(kysy Kovasta). TÄMÄ TILAISUUS EI TULE TOISTUMAAN!!

Cyberstormille saatavana myös Fast SCSI II-laajennus ja
I/O-module, joka sisältää Fast SCSI-II:n, Ethernetin ja sarjaportin.

MUUT LISÄLAITTEET

Cybervision 64 huipputehokas 64-bit näytönohjain A4000:lle - sis. 4 MB näyttömuistia	3 490,-
Multifacecard, 2 * serial, 1 * parallel, 115200 baud	1 150,-
Z3 Fastlane SCSI-II ohjain	2 800,-
Ariadne Ethernet-kortti, sis. softan	2 490,-
Vidi Amiga 12 videodigitoija	990,-
Vidi Amiga 12 RT, reaaliaikainen videodigitoija	1 490,-
Vidi Amiga 24 RT, reaaliaikainen videodigitoija	2 490,-
Vidi RT:t tarvitsevat virtalähteen	250,-
HD-levyasema sisäinen, kaikki konemallit	790,-
HD-levyasema ulkoinen, kaikki konemallit	930,-

Workbench 3.1 saatavana nyt myös A1200:lle!

Kickstart ROM + WB3.1 A500/A1200/A2000/A3000/A4000	890,-
Overdrive 250 MB kiintolevy Overdrive-kiintolevy liitetään A1200:n PCMCIA-liittimeen, ja se sisältää oman virtalähteen. Voit huoletta käyttää koneessasi samaan aikaan lisälevyasemia ja laajennuskortteja.	2 300,-
Overdrive-CD-ROM-asema "ZAPPO" A1200:lle CD32-emulointi (lähes kaikki pelit toimivat), PhotoCD- ja AudioCD-softat, oma virtalähde, liitetään PCMCIA:han joten jättää laajennuspaikan vapaaksi, lukee PC:n ja Mac:n CD-Romeja...	2 100,-

Amigan varaosia (otettu käytetyistä koneista):

CIA	100,-
A500 levari	250,-
A500 muuntaja	250,-
Emolevy 6A	200,-
Emolevy 5	100,-
A500 kotelo	80,-
Agnus 8372A (Fatter ECS)	250,-
Näppis	150,-
Denise OCS	100,-
Gary	100,-
Paula	100,-
MC68000	50,-

Rikkinäinen 512 kB lisämuisti (kello toimii), sisäinen levari,
40 MB Conner-kiintolevy ~ 50,-

PC-TUOTTEET, KIINTOLEVYT, MUISTIT, PRINTTERIT

Emolevyt (PCI, AMI, UMC, GREEN)

486DX2-66, 128 kB cache, 4 * PCI, 3 * ISA, ZIF	1 690,-
486DX2-80, 128 kB cache, 4 * PCI, 3 * ISA, ZIF	1 790,-
486DX4-100, 128 kB cache, 4 * PCI, 3 * ISA, ZIF	2 390,-
486DX ilman CPU:ta, 128 kB cache, 4 * PCI, 3 * ISA, ZIF	890,-
Pentium PCI 66 MHz, 256 kB cache, 2 * PCI, 3 * ISA, ZIF	3 490,-

Kiintolevyt (IDE)

Seagate ST3491A Medalist 428 MB 14 ms FastATA	1 290,-
Seagate ST3660A Medalist 545 MB 14 ms FastATA	1 390,-
Seagate ST3780A Medalist 722 MB 12 ms FastATA	1 890,-
Seagate ST31220A Medalist 1083 MB 12 ms FastATA	2 450,-
Quantum Prodrive 420 MB 12 ms	1 290,-
Quantum Lightning 730 MB 11 ms	TULOSSA

Myös SCSI-kiintolevyt!

Muistit

1 MB 3 chip	295,-
4 MB 3 chip	RAJOITETTU 845,-
4 MB 32 Bit JEDEC SIMM	995,-
8 MB 32 Bit JEDEC SIMM	1 950,-
16 MB 32 Bit JEDEC SIMM	2 890,-
32 MB 32 Bit JEDEC SIMM	5 390,-

SVGA-näytöt

14" TARGA TM142PNL, 1024*768	1 690,-
15" TARGA TM3820, 1280*1024	2 390,-
17" TARGA TM4220, 1280*1024	3 890,-

Myös ADI:n näytöt.

Näytönohjaimet

Trident 9000C (ISA) SVGA 256 kB	250,-
Trident 8900C (ISA) SVGA 1 MB	450,-
Avance Logic GL-2228 (VESA) 1 MB	590,-
MiroCrystal S3 10AD (VESA) 1 MB	750,-
Trident 9440 (PCI) 1 MB	590,-
Cirrus 5434 (PCI) 1 MB	690,-
Avance 2301 (PCI) 1 MB	650,-
MiroCrystal 20SD (PCI) 2 MB	1 490,-
Diamond Stealth 64 (PCI) 2 MB näytönohjain	1 990,-

IDE ohjainkortit

Prime2/UMC (ISA) IDE I/O	150,-
Enhanced (VESA) IDE I/O	250,-
Enhanced (VESA) IDE I/O (FAST UART 16550!)	350,-
Vision (PCI) IDE	250,-

CD-Rom

2 x nop CDD-120	790,-
Panasonic 562B ja kaapelit, tuplanopeus	890,-

Panasonic 562B ja kortti + kaapelit, tuplanopeus	950,-
4 x nop CD-ROM Mitsumi FX400, IDE-liitäntä	1 490,-

Äänikortit

Mozart 16 Stereo MultiCD	490,-
Sound Blaster 16 Value Edition	690,-
Sound Blaster AWE 32 Value Edition	1 290,-
Sound Blaster CD16 Game Blaster	2 290,-

Canon-tulostimet:

BJ10SX W	1 390,-
Arkinsyöttölaite edelliseen	550,-
BJ200ex	1 990,-
BJC600	3 390,-
LBP 4i	3 590,-
LBP1260C	8 690,-

SUPERTARJOUS! Canon BJC 4000 huippu väritulostin VAIN 2 890,-!

Canonien mukana tulevat MS Works- ja Entertainment-ohjelmistot!

CD-ROM pelit

Classic Collection (4 CD)	195,-
Gabriel Knight	195,-
New Collection 2 (4 CD)	195,-
Mad Dog McCree	185,-
Mad Dog McCree II	185,-
Megapak (11 CD:tä)	295,-
10 Pak CD-ROM Bundle	295,-
NHL Hockey 95	350,-
Rebel Assault	225,-
Transport Tycoon	395,-
Wing Commander III	450,-

Tässä vain murto-osa CD-rom-ohjelmista.

Muut sälät

Adaptec AHA1542F SCSI-2 ohjain	1 490,-
1.44 MB HD-levari 3,5"	240,-
Keytronic 2000 näppäimistö	250,-
MG Ethernet verkkokortti (NE-2000 yht. sop.)	390,-
3Com Etherlink III BNC/Aui verkkokortti	690,-
Best faxmodeemi 14400 sisäinen, hyväksytty	890,-
Best faxmodeemi 28800 sisäinen, hyväksytty	1 590,-
Microsoft MS 2 hiiri (serial) OEM	190,-

KAIKILLA UUSILLA TUOTTEILLA ON TAKUU (puoli vuotta - kolme vuotta), KÄYTETYILLÄ TOIMIVUUSTAKUU.

T:MI AMIGAZE / PASI KOVANEN
PUH: (952) 2282 223 ARMEIJASTA JOHTUEN (LOPPUU 18.4.) VAIN VIIKONLOPPUISIN + PAIKOITELLEN VIIKOLLA (Puhelinvastaaja kertoo päivittäin tavoitettavuuden). Aina kuitenkin pe 19:00 - su 22:00 lukuunottamatta maaliskuun 1. viikkoa, jolloin tulen kotiin vasta lauantaina, 4.3. kello 19:00.

Postiosoite:

Haltijantie 2 D 31
48350 KOTKA

Alv. Rek.

Trademarks are the property of their respective owners.

1.112 Posti

Posti

CHIP-laajennus toimii

Ongelmia SAKU10.RUN-paketin kanssa

AmigaBasic-listausten suojaus ja yhteensopivuus

Fun Stationin alennukset ja artikkelien muotoilu

Amiga 500:n sisäisen muistin laajentaminen

Artikkeleista, niiden virheistä ja kovalevyistä

Sakusta PC-versio leviämään PC-purkkeihin?

RAD-asemista ja CHIP-muistin laajentamisesta

Suomen Amiga-käyttäjien unionin lehdet pyrkivät ↩
asiantuntijoiden avulla
selvittämään ongelmasi ja vastaamaan palautteeseesi Posti-palstan kautta. Voit lähetellä mielipiteitä ja kysymyksiäsi toimitukseen seuraavaan osoitteeseen postitse. Koordinaattorit voivat myös välittää saamaansa palautetta lehteen.

Janne Siren / SAKU
Oravamäentie 2 F 17
02700 Kauniainen

Internet: jts@krk.fi

1.113 CHIP-laajennus

CHIP-laajennus (ks. Posti, SAKU #10): olen kolvannut kolme konetta C=lehdessä mainitulla tavalla, ja kaikki ovat toimineet ja toimivat edelleen, jos ovat käytössä.

Antti Vähä-Sipilä

1.114 SAKU10.RUN-paketista

SAKU10.RUN-paketista: imuroin sen kovalevylleni ja käynnistin normaalisti Run-käskyllä. Ohjelma purkautui kovalevyni Work-partitioon, mutta hävitti Startup-Sequenceni ja korvasi sen uudella. Sen jälkeen kone latasi vain Sakun käynnistettäessä. Ohjelma on varmasti hyvä luomaan itsekäynnistyvä SAKU-levyke, mutta se näyttää hahmottavan kovalevypartition korpuksi edellämainituin seurauksin. Missä tein virheen?

Aarno Yliselä, Jyväskylä

Toimitus:

SAKU10.RUN on itsestään purkautuva muunnos SAKU10.LHA-paketista. Sitä ei tarvitse käynnistää Run-komennolla, riittää kun ajat sen. Sitä ei kuitenkaan ole suunniteltu mitenkään erityisesti asentamaan Sakua. Se yksinkertaisesti purkaa paketin senhetkiseen työhakemistoon. Paketissa on S-hakemisto ja sen sisällä SAKU-levykettä varten suunniteltu minimaalinen Startup-Sequence. Kun ajoit ohjelman käynnistysosioltasi (boot-partitio), Startup-Sequence korvattiin paketissa oleella. SAKU10.RUN-pakettia ei saa purkaa suoraan kovalevyn juurihakemistoon, vaan sille pitää luoda oma alihakemisto ja purkaa tiedostot sinne. Sama pätee kaikkiin SAKU-paketteihin. Ilmeisesti Run-komennon käyttö aiheutti sen, ettei ohjelma varoittanut tai kysynyt, halutaanko vanha Startup-Sequence korvata uudella.

Virallisesti Sakua levitetään purkeissa LHA-paketteina. Eräät tahot kuitenkin muuttavat sen RUN-muotoon. Itsepurkautuvilla paketeilla on etunsa, mutta myös haittansa.

Janne Siren

1.115 Posti: AmigaBasic-listausten suojaus ja yhteensopivuus

Hyvä SAKU! Minulla olisi yksi pyyntö: voisitteko pistää Sakuun Pascal-kurssin, se kun tulisi tarpeeseen. Sitten kysymykset:

1. Kun tehdään Basicilla ohjelma, kuinka listaus suojataan niin, ettei sitä ihan tavallinen kämmäri saa purettua?

2. Voiko Amigalla tehtyjä Basic-ohjelmia kääntää PC:lle? Jos voi niin miten?

3. Myyttekö Sakun C-kurssia paperilla, kun en omista kirjoitinta? Jos myytte, mistä sitä voisi tiedustella?

"Nimim. Ikuinen Amigan kannattaja, C. Splut"

Toimitus:

Pascal-kurssi ei ole hassumpi ajatus, sitä kirjoittamaan vain pitäisi saada joku. Halukkaat Pascal-kurssin kirjoittelijat ottakaa yhteyttä toimitukseen!

1. Oletan, että puhut AmigaBasicista. Listauksen voit suojata näin:

SAVE Listaus.bas,P

Listaus.bas on esimerkki nimeksi, vain ",P" lopussa on ratkaiseva. Suojatun listauksen voi edelleen ladata ja ajaa, mutta sen näyttäminen tai muokkaaminen AmigaBasicilla ei enää onnistu. Muista tallentaa itsellesi suojaamaton kopio!

2. AmigaBasic on monessa suhteessa yhteensopiva esimerkiksi PC:n GW-Basicin ja QBasicin kanssa, onhan kaikkien kolmen takana Microsoft. Riittää, että siirrät AmigaBasic-ohjelmasi PC:n Basic-tulkkiin ja muutat sitä tarvittavilta osin, jolloin se toimii myös PC:ssä. Basic-tulkkien eroja on mahdoton lähteä tässä kelailemaan, erilaisista komennoista ja muista eroista löydät lisää tietoa vertailemalla ohjekirjoja. Basicin periaatteet ja useat komennot ovat kuitenkin aivan samat.

3. Toistaiseksi Sakulla ei ole tulostuspalvelua tarjolla. Jos joku yksityinen henkilö on valmis auttamaan toista amigistia tulostamalla hänelle kulukorvausta vastaan C-kurssin, ilmoitelkoon toimitukseen.

Janne Siren

1.116 Posti: Fun Stationin alennukset ja artikkelien muotoilu

Koskevatko Fun Stationin alennukset vain yhdistyksen jäseniä? Mainoksessa kun mainittiin yleisesti vain 'Sakun lukijat'. Onko muuten jäseneduista vielä mitään tietoa? SAKU-lehti itsessään on tosin melkoinen etu! Parempaa ja asiallisempaa lehteä saa hakea! Sakun siirtämistä paperille on kaavailtu, mutta eivätkös silloin kärsi juttujen monipuolisuus ja laajuus? Disketillehän ei ole pakko lukea edes mainoksia, jotka hyppivät silmille lähes jokaisesta lehdestä. Suurkiitokset siis hyvästä ja monipuolisesta lehdestä!

Juha Niemimäki

PS. Kun kirjoitan jotain juttua Sakuun, niin saanko käyttää enteriä yms. muotoilumerkkejä?

Toimitus:

Fun Stationin edustaja Heimo Laukkanen kertoi, että riittää pelkästään hänen nimensä ja Sakun mainitseminen ostohetkellä. Alennus Sakun lukijoille on aivan Fun Stationin oma idea, se ei siis liity mitenkään yhdistyksen jäsenetuihin. Jäsenetuja pohditaan tarkemmin mahdollisimman pian järjestettävässä ylimääräisessä ja samalla ensimmäisessä vuosikokouksessa.

Paperilla olevaa lehteä on miellyttävämpi lukea, ja paperijulkaisun levitykseen on enemmän mahdollisuuksia kuin levykelehden. Luonnollisesti lehden sisältöön pitää kiinnittää entistä enemmän huomiota, koska kaikkea ei saa paperille mahtumaan, mutta rinnakkainen levykelehti ei ole sekään pois suljettu ajatus.

Sakuun kirjoitettaessa juttu tulee muotoilla siten, että kaikki rivit päättyvät enteriin ja kappaleiden välissä on yksi tyhjä rivi. Otsikko ja kirjoittajan nimi tulee olla myös selvästi esillä artikkelin alussa. Tavutusta ei saa artikkeleissa käyttää.

Janne Siren

1.117 Posti: Amiga 500:n sisäisen muistin laajentaminen

Mitä muuta on tehtävä kuin kolvattava muistipiirit Amiga 500:n emolevylle saadakseen megatavun CHIP-muistia? Minulla on jo 512 kt lisämuisti, mutten haluaisi käyttää pelkästään sitä.

"Nimetön lukija"

Toimitus:

Jos koneesi emolevyn vasemmassa alanurkassa on neljän muistipiirin välissä vapaita paikkoja, voit juottaa toiset neljä samanlaista muistipiiriä (4x256 kt DRAM, 100 ns tai nopeampia) niihin. (Vanhemmissa Amigoissa emolevyllä ei valitettavasti ole tilaa megatavulle muistia ilman virittelyjä.) Lisäksi tarvitset jokaista muistipiiriä varten suotokondensaattorin (vähintään 220 nF), jonka juotat sille varattuun paikkaan emolevylle muistipiirin alapuolelle, kuten jo tehtaallakin asennetuissa piireissä on tehty. Sitten vielä kerrot koneelle, että lisämuisti on emolevyllä yhdistämällä lisämuistikortin liittimen vieressä olevan JP7A-jumpperin kaksi ylintä nastaa. Tarkemmat ohjeet löytyvät esimerkiksi C=lehdestä 4/89. SAKU tai allekirjoittanut ei vastaa mahdollisista vahingoista, joten käytät näitä ohjeita täysin omalla vastuullasi.

512 kt lisämuistin, oli se sitten juotettuna A500:n emolevyllä oleviin reikiin tai koneen pohjassa olevassa laajennusportissa, muuttamista CHIP-muistiksi käsiteltiin edellisen Sakun (#10) Posti-palstalla.

Amiga 500:n sisäistä RAM-muistia ei ole mahdollista laajentaa yli megatavun ilman virittelyjä. Laajennusportissa olevan lisämuistin käyttöön saaminen emolevyllä olevan megatavun lisäksi onkin jo paljon monimutkaisempaa, ja sen selostaminen vaatisi aivan oman artikkelinsa.

Janne Siren

1.118 Posti: Artikkeleista, niiden virheistä ja kovalevyistä

Lauri Aalto toi ilmi edellisen Sakun palautekanavalla TechnoBBS-artikkelini ensimmäiseen osaan pujahtaneen epäselvyyden. Lauri oli saanut sen käsityksen, ettei TechnoBBS:ää mielestäni saisi käyttämään muita kieliä kuin englantia. Näinhän asia ei ole, enkä niin sanonutkaan. Sanoin artikkelissani, että TechnoBBS tukee käytännössä rajattomasti kieliä, kunhan tekee itse tarvittavat kielitiedostot. Sillä, että sanoin, ettei TechnoBBS tue kuin englantia, tarkoitin virallista distribuutiota, johon ei kuulu ulkoisia kielitiedostoja, vaan ainoastaan englanti sisäisenä kielenä. Jos TechnoBBS:n haluaa tukemaan jotakin muuta kieltä, se on mahdollista, mutta sen joutuu itse tekemään, koska sellaisena kuin TechnoBBS levitetään, se ei tue kuin englantia. Pahoittelen tätä pientä sekaannusta, jonka hieman epämääräinen ilmaisuni nähtävästi sai aikaan.

Haluaisin vielä kommentoida muitakin viime Sakun Posti-palstalla esitettyjä lukijoiden mielipiteitä. Edelleen Lauri Aalto yrittää korjata Piratismi-artikkelia ollen itse väärässä. Hakkerit ovat nimenomaan niitä, jotka murtautuvat tietojärjestelmiin, joihin heillä ei ole oikeuksia. Krakkerit vain murtavat ohjelmien suojauksia. Näillä kahdella ihmistyypillä ei ole yhtään mitään tekemistä toistensa kanssa.

Postissa valitettiin myös Sakun hidasta leviämistä purkkeihin. Omalta osaltani voin sanoa, että kaikissa varteenotettavissa Amiga-purkeissa SAKU on hyvin nopeasti. Itse huolehdin sen usein moniin purkkeihin. Omassani se on luonnollisesti näistä ensimmäisenä. Esimerkiksi SAKU #10 oli purkissani jo aamupäivällä tammikuun ensimmäisenä päivänä. Ehkä olisi aiheellista harkita siirtymistä jonkin toisen purkin asiakkaaksi.

Niin ikään Keskustelu-artikkeli herätti keskustelua, aivan aiheesta. Siinä oli selvä virhe. Kommentoija on aivan oikeassa, sillä massamuistit ovat sektoroituja laitteita, joita ei osoiteta tavuittain vaan sektoreittain. Keskustelun aiheena oleva 32 bitin raja koskeekin sektoreiden määrää. Näin ollen kovalevyn (nimenomaan koko levyn eikä yksittäisen partition) maksimikooksi saamme konventionaalisella blokkikoolla kaksi teratavua. Uusimmat tiedostojärjestelmät antavat myös suurentaa blokkikokoa, jolloin esimerkiksi kahdeksan kilotavun kokoisilla sektoreilla kovalevyn maksimikoko nousee 32 teratavuun. Tämän rajan ei uskoisi tulevan vastaan kovinkaan pian.

Eniten virheitä oli kuitenkin kommentoijan omassa tekstissä. Mikäli asioissa on epäselvyyttä, tulisi ennemmin kysyä ja antaa meidän kertoa, miten asiat ovat kuin arvailla, kuten tässä oli tehty peräti sivun verran. Kommentoijien tulisi myös tarkistaa omat tietonsa ettei käy niin, että artikkeleissa olevia virheitä korjataan uusilla virheillä (esim. hakkerit). Kovalevyistä on jonkin verran tietoa SAKU #10:ssä. Suunnittelin artikkeliin myös jatko-osaa, jossa olisin kertonut tarkemmin kovalevyn toiminnasta Amigassa, mutta koska artikkelissa pyytämääni palautetta en ole saanut, oletin, ettei sille ollut sosiaalista tilausta. Selvitän tässä nyt kuitenkin lyhyesti joitakin kirjoittajan epäselvyyksiä.

RAM-muistin määrä ei millään tapaa riipu osoiteväylän leveydestä. Koneessa voi olla muistia määrä, joka ei ole kiinteässä suhteessa väylien leveyteen. Nykyisin osoiteväylän leveys toki asettaa yleensä muistin määrälle ylärajan, mutta sekään ei ole mikään ehdottomuus. Esimerkiksi jo Commodore 128:ssa on vakiona 48 kilotavua ROMia ja 144 kilotavua RAMia, vaikka siinä onkin vain 16-bittinen osoiteväylä ja 64 kilotavun osoiteavaruus. Todellakin, Keskustelu-artikkelissa valitettiin, että 4 GB:n raja tulee vastaan 10 vuoden kuluttua, mutta kyllähän asia on niin, että silloin 32-bittiset väylät kuuluvat jo museoon! Olisi itsensä petkuttamista uskotella, että vielä 10 vuoden kuluttua koneet olisivat 32-bittisiä. Kun lisätarvetta tulee, väylää tietenkin levennetään. Minusta on kuitenkin erittäin epätodennäköistä, että kotitietokoneessa koskaan tulisi olemaan 9000 gigatavun keskusmuistia - tämä taas kommenttina kommentoijalle.

FAT todellakin on olemassa, mutta vain MS-DOSin tiedostojärjestelmän alla. Sitä ei tule missään nimessä millään tapaa sotkea AmigaDOSin tiedostojärjestelmien käyttämiin metodeihin. Toki Amigan tiedostojärjestelmät käyttävät vastaavaa taulukkoa, mutta FAT tarkoittaa nimenomaan MS-DOSin tiedostojärjestelmän taulukkoa, joka ei kuulu tämän julkaisun aihepiiriin. Tietoa ei haeta kovalevyltä tiedostojen nimiä osoittelemalla vaan kuten jo sanoin, sektoripohjaisesti. Ohjain lukee aina tietyn määrän sektoreita alkaen tietystä sektorista. Tiedostojen nimet ja muut niihin liittyvät tiedot

otetaan selville paljon ylemmällä tasolla.

Levylle tallennetaan tiedostoista paljon muutakin tietoa kuin vain nimi. Pelkällä nimellä ei tee yhtään mitään. Jokaisesta tiedostosta saa halutessaan FileInfoBlockin, josta sen tiedot löytyvät. Tiedostonimelle on varattu 108 tavua, mutta vain 30 (ei 31) on tällä hetkellä käytössä. Tiedot tallennetaan levyllä hakemistoon, jolla ei ole mitään tekemistä allokaatiokartan kanssa, paitsi että kumpaankin päästään käsiksi RootBlockin avulla. Tarkempaa tietoa koneeseensa liitetyistä asemista saa DOSin RootNoden kautta. Kovalevyn toiminnasta kertovat paljon nollasylinterillä sijaitsevat hardblokit, erityisesti RigidDiskBlock.

SCSI itsessään ei aseta mitään rajoituksia väylään liitettäviin oheislaitteisiin. Niiden ei välttämättä tarvitse olla kovalevyjä, vaan myös sektoroimattomia laitteita, kuten lasertulostimia ja skannereita, voidaan liittää. Näin ollen SCSI ei myöskään sisällä mitään erityistoimintoja massamuistilaitteille. Missään vaiheessa raaka data ei liiku SCSI-väylällä, vaan kovalevyn yhteydessä oleva kontrolleri, joka myös mm. ohjaa levyn päitä ja moottoria, koodaa ja purkaa datan. Levyn alemman tason toimintoja voidaan ohjata SCSI-direct-komennoilla. Lisää tietoa saa SCSI-määrittelyistä ANSI-X3T9 (SCSI2) ja ANSI X3.131 (SCSI1).

Sami Klemola, Anjalankoski

1.119 Posti: Sakusta PC-versio leviämään PC-purkkeihin?

Tulipa tässä Sakua lukiessa mieleeni...

Sakusta voisi toimittaa PC-version, vaikka vain yhden numeron, jossa olisi eri sisältö kuin normaaleissa Sakuissa. Kyseessä olisi eräänlainen Amigan piilomainos, jota levitettäisiin PC-bokseissa. Sen sisältö voitaisiin koota aikaisemmista Sakuista ja väliin lisäillä uutta materiaalia.

Samaan tapaan Sakusta voitaisiin taittaa huoliteltu paperilehti. Sitä voitaisiin ujuttaa vaikka kirjastoihin, nehän pursuavat jos jonkinlaista harrastusjulkaisua, jotka lähes kaikki kootaan kuten Saku - siis lukijoilta.

Nämä äskeiset pitäisi siis tehdä täysin markkinointihenkisesti. Jos Commodore (tai mikä se nykyään onkaan) ei markkinointia hoida, niin kyllä "lojaalien" amigistien siitä pitäisi huolehtia.

Ja vielä. Voisiko joku asiantunteva kerätä mahdollisimman kattavan luettelon Suomen Amiga-kauppiaista osoitteineen ja puhelinnumeroineen ja pistää sen sitten Sakuun. Niitä kun on nykyään vähän vaikea löytää.

"IBM - Solutions for a small mind"

Toimitus:

Asiamme ja Amigan markkinointia varten laadittu erikois-SAKU ei ole hassumpi ajatus, varsinkin paperilla siitä olisi hyötyä. Eri asia on kuinka disketille kasattu SAKU leviäisi PC-purkeissa...

Luetteloa Suomen Amiga-kauppiaista yritämme saada jo seuraavaan numeroon!

Janne Siren

1.120 Posti: RAD-asemista ja CHIP-muistin laajentamisesta

1) Halusin avata kaksi 880 kt RAD-levykettä, eli siis kopioin RAD-mountfilen ja muutan nimen ja unit-numeron. Tämä kyllä onnistuu, mutta toinen RAD-levykkeistä käyttää vain CHIP-muistia, jolloin sitä ei juuri muuhun riitäkkään.

2) Kokoonpano: Amiga 500, KS ja WB 3.1, muistia 1 Mt CHIP ja 3.5 Mt FAST (toteutettu 4 Mt sisäisellä Eureka-muistikortilla, josta 0.5 M on käytetty CHIP-muistiksi). Yli 2 Mt menevä osa tarvitsee ohjelmallisen ajurin. Agnus on 1 Mt malli. Saako sen toisenkin RAD-aseman käyttämään jollain määrityksillä FAST-muistia vai estääkö esim. Agnus tuon?

3) Entä tiedätkö saako Suomesta (ja vielä edullisesti) A500:een palikoita, joilla CHIP-muistin kasvattaisi 2 megatavuun. Esim. brittilehdissä on näkynyt Agnuksen alle laitettavaa palikkaa joka tuon tekee.

Kiitos hyvästä lehdestä!

Mika Kiljunen

Toimitus:

1) Itse olen umpilevyn saatuani kokonaan luopunut virtuaaliasemien käyttämisestä. Sähkökatkon sattuessa vain niillä olleet tiedostot ovat mennyttä kalua... Mahdollinen RAD-levyke kannattaa virittää mahdollisimman pieneksi ja nopeasti käyttöjärjestelmän käyttöösi antavaksi.

2) Ilmeisesti FAST-muistista ei heti käynnistyksen jälkeen löydy tarpeeksi suurta yhtenäistä muistialuetta, jotta sinne mahtuisi kaksi lähes megan kokoista virtuaalilevykettä. Koska muistilaajennuksesi konfiguroidaan ohjelmallisesti, se ei säilytä muistialueita ehjänä uudelleenkäynnistyksen yli. Agnuksellakin on toki osuutensa asiaan.

Taitava elektroniikkanikkari rakentaa toki 2 megan CHIP-muistin koneeseesi, mutta sitä tuskin kannattaa antaa RAD-levykkeiden käyttöön vaan antaa grafiikalle ja soundeille kaikki mahdollinen tila.

FAST-alueille laitettavat RAD-levyt ovat itsellänikin tuottaneet runsaasti harmaita hiuksia, kuten keltaista virhesignaalia uudelleenkäynnistettäessä. PD-ohjelmien joukossa on muutama Commodoren omaa virtuaalilevykettä parempaakin vaihtoehtoa. Muistaakseni sellaiset kuin ASDG:n ja StatRam. Näistä voit kysellä lisää Sakun koordinaattoreilta!

3) Käytettynä varmasti löytyisi. Yhdistyksemme sihteerin kautta voisi olla mahdollista tilata tavaraa myös Englannista. Neuvotellaan asiasta.

2 Mt CHIP-laajennusta Englannissa mainostaa Power Computing hintaan £159. CHIP-laajennus, nimeltään MegaChip, sisältää megatavun muistia ja käyttää koneessasi jo olevan megatavun CHIP-muistin toiseksi puoliskoksi. Jos haluat tilata tai tiedustella laajennusta suoraan, tässä ko. yrityksen yhteystiedot. Yksi Euroopan Unionin eduistahan on, ettei erillisiä tulleja sen sisällä ole...

Power Computing
44a/b Stanley St.
Bedford
MK41 7RW
United Kingdom

Puhelin: 990-44-234-273-000
Faksi: 990-44-234-352-207

Tomi Jaskari ja Janne Siren

1.121 Errata

Errata

Errata-palstalla julkaistaan korjauksia aiempien numeroiden kömmähdyksiin. Voit lähettää toimitukseen palautetta seuraavaan osoitteeseen postitse.

Janne Siren / SAKU
Oravamäentie 2 F 17
02700 Kauniainen

Internet: jts@krk.fi

1.122 Sakupörssi

Sakupörssi

Ostetaan ja myydään sekalaista tavaraa

Sakupörssi on tarkoitettu lukijoiden myynti-, osto-, vaihto-, anto ↩
- ja ot-
to-ilmoituksille. Emme julkaise kopioihin liittyviä ilmoituksia, ja toivomme, ettet ilmoita Sakupörssiin asoita, jotka todennäköisesti eivät ole ajankohtaisia enää seuraavan Sakun päästessä jakeluun. Ilmoitusten tulee olla perillä kahta viikkoa aikaisemmin, ja mikäli ilmoituksia tulee liikaa mahtuakseen yhteen lehteen, julkaisemme niitä saapumisjärjestyksessä seuraavissa numeroissa. Voit lähettää ilmoituksesi seuraavaan osoitteeseen.

Tomi Jaskari / SAKU
Rasinkatu 7 C 80
01360 Vantaa

Puhelin: (90) 874 2445 (arkiaamuisin 7-10)
Internet: tomi.jaskari@helsinki.fi

1.123 Sakupörssi: Ostetaan ja myydään sekalaista tavaraa

Myydään:

1. Kovalevy 121 Mb 3.5" Quantum Prodrive, n. 1 vuotta vanha virheetön levy, täynnä Amiga/AGA-tavaraa! Hintapyyntö 700,- + postikulut.

2. RC-autopaketti: Tamiya Avante 4WD sähkömoottoriauto (rattaissa jotain vikaa; varaosista löytyy vaihtorattaita), 2 vir. mottia (huiput yli 40 km/h), 2 akkua, laturi, elektroninen nopeudensäädin, PCM-rattiradio (ilman vastaria ja servoa), uudet kiteet, kolmet renkaat ja paljon varaosia. Hintapyyntö 1000,- + postikulut, tai vaihtaen A1200:n 28 Mhz -turbokorttiin (MC68020) kellokalenterilla ilman muistia.

3. MikroBitti-lehden vuosikerta 1994, yhteensä 11 lehteä + kaupan päälle nro 12/93. Hintapyyntö 100,- + postikulut.

Ostetaan:

1. VideoBackupSystem, uusi/uudehko, mukaan kaikki tarvittava, toimivuustakuu. Maksan 100 markkaa.

2. SIMM-muistia A1200:lle. 2 Mt 32-bittistä 72 pinniä 70 ns. Maksan 300 markkaa.

Soita (964) 484 6477 / Janne, arkisin kello 15 jälkeen.

1.124 Tulossa

Tulossa

Seuraava SAKU ilmestyy 1. toukokuuta, 1995. Luvassa on mm. seuraavaa:

Metallic Nations

Avesoft mainostaa suurin otsikoin uutta kotimaista modeemin välityksellä pelattavaa monen pelaajan strategiapeliä. Janne Siren katsastaa, onko uusi tulokas, Metallic Nations, mainosrahansa arvoinen.

Magic: The Gathering

Magic: The Gathering on fantasia-aiheinen keräilykorttipeli. Jani Höglund tutustuttaa lukijat tähän maailmaa nopeaan tahtiin valloittavaan (ja rahastavaan) peliin.

Viivakoodit

Viivakoodeja näkee nykyään niin muropaketeissa, kuin CD-levyissäkin, mutta miten ne toimivat? Heimo Laukkanen selvittää viivakoodeja.

Ohjelmointikurssit jatkuvat

Ville-Pertti Keinonen palaa ruutuun opastamaan aloittelijoita C-kielen saloihin, ja Sami Klemola jatkaa kurssiaan järjestelmän hallinnasta.

Mukana myös tuoreimmat uutiset Amiga-rintamalta ja paljon paljon muuta. Joko Commodore saadaan myytyä? Entä kuinka Sakun ensimmäinen vuosikokous sujui?