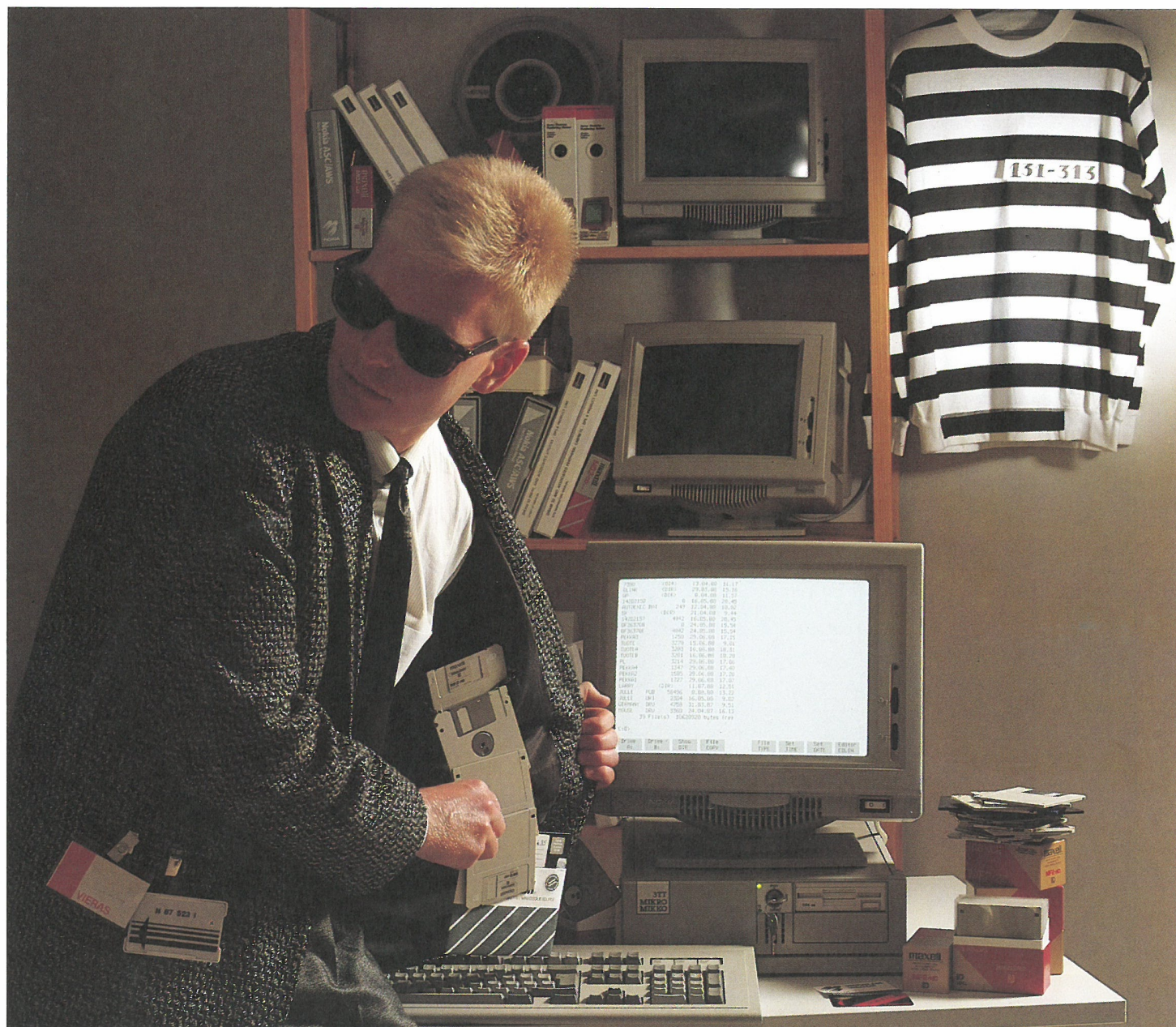


Softari

NOKIA DATAN OHJELMISTOLEHTI 3/88



**Ohjelmistojen kopiosuojaus
on moraalinen kysymys**

Softari

Nokian ohjelmistolehti

Päätoimittaja:

Anne Typpö puh. (90) 567 2981

Softaritiimi:

Kari Kaipainen, Jukka Oksanen,
Hannu Markus, Tapio Sulkava,
Ossi Syrjä, Anne Typpö,
Esko Uotila, Raija Visamo

Ulkoasu ja kuvitus:

Taina Sonninen puh. (90) 567 3260

Toimituksen osoite:

Nokia Data
Softari-lehti
PL 780
00101 HELSINKI

Painosmäärä: 11 000

Kirjapaino: Vientipaino

Julkaisija:

Nokia Data

ISSN 0782-8209

Kansikuva: Utte Ekholm

Sisällysluettelo

Pääkirjoitus	2
Ohjelmoinnin viehäty	
säilyy	3
Suojata vai ei, kas siinä	
ohjelmistokysymys	4
Käyttäjän ja suunnittelijan	
Paradox	6
Tietokantapalvelin jakaa	
yhteiset tiedot	8
Exploring the OS/2	
Multitasking	11
OS/2:n ohjelmaliitos ja	
muistinhallinta	16
Opettele Windows-	
ohjelmointi	20
Sano se kuvin	22
"Löylyä"	24
Lähiverkolla varmaan	
lentokonehuoltoon	26
Muisti pettää,	
EMS apuun	28
MikroPOS valvoo kaupan	
kannattavuutta	30
Learning Ada	32
Uutisia	37
Pakina	40

NOKIA DATA

Softari uudistuu

Syksyn alkajaisiksi jo neljä vuotta ilmestynyt Softari sai uuden ulkomuodon. Lehden sisältö ei oleellisesti muutu, mutta se keskittyy entistä tiukemmin mikro-maailman tapahtumiin ja ennen kaikkea ohjelmistoista kertomiseen. Joka numerossa on myös tarkoitus esittää nokialainen kannanotto johonkin ajankoh- taiseen kysymykseen kuten nyt kopiosuojaukseen.

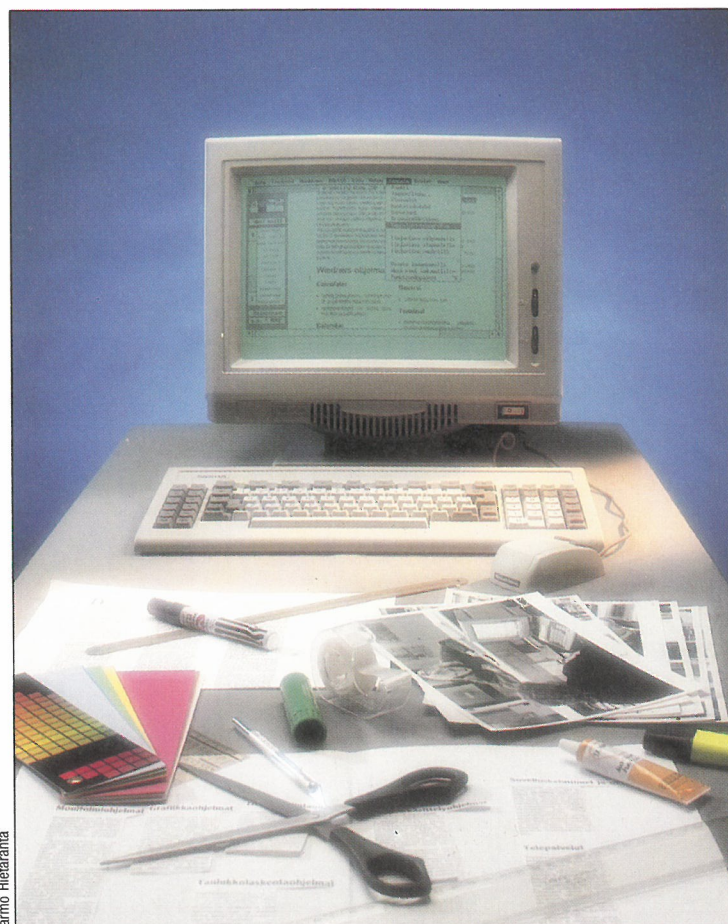
OS/2-käyttäjärjestelmään on tullut ensimmäinen sovellus, Paradox/386. Seuraavia odotellaan innolla. Itse uutta käyttäjärjes- telmää on tutkittu myös Nokia Datan Tuotekehityksessä. Sitä varten tehtiin TreadScope-ohjel- ma, jolla saadut tulokset on esi- tetty artikkelissa "Exploring OS/2 Multitasking". OS/2-ohjel- moinnin piirteistä kerrotaan al- kavassa artikkelisarjassa, jossa tutustutaan nyt ohjelmaliitok- seen ja muistinhallintaan. Leh- dessä on esitetty ytimekkäästi myös Windows-ohjelmoinnin pe- riaatteet.

Nokia Data edustaa yksinoi-

keudella Gupta Technologies, Inc.:n SQLBase-tietokantajär- jestelmää Suomessa ja Ruotsis- sa. SQLBase on DOS-maailmas- sa erittäin edistysellinen tietokantajärjestelmä, joka toimin- nallisesti vastaa mini- ja suurko- neiden järjestelmiä. SQLBasen uusin osa on SQLWindows-so- velluskehitin. Sillä ohjelmoija voi kehittää sovelluksia Micro- soft Windows-ympäristöön il- man C-ohjelmointia tai Window- sin sovelluskehitysvälineitä.

Muita lehden teemoja ovat muun muassa lisämuistin han- kinta, Designer-grafiikkaohjel- ma, MikroPOS-järjestelmä ja Finnairin lähiverkko. Kun Ada toimii nykyisin myös mikroissa, voivat kaikki asiasta innostuneet opetella Ada-kielellä ohjelmoin- tia alkavan kurssisarjan oppien mukaan.

Tervetuloa viihtymään uudis- tuneen Softarin pariin.



Jarmo Hietaranta

A man with glasses and a beard, wearing a striped short-sleeved shirt, a dark tie, and light-colored trousers, is walking on a green golf course. In the background, there are trees and a clear sky.

Tom Törnwall

Ohjelmoinnin viehätys säilyy

OS/2-käyttöjärjestelmä ja WPM eivät vähennä ohjelmointityön viehätystä. Tekniikka pysyy samana, mutta moniajoympäristö muuttaa ohjelmien rakennetta ja loogista toimintaa.

DOS:n seuraaja OS/2 on tulossa työasemien käyttöjärjestelmäksi. Sen yhteydessä mainitaan usein Windows Presentation Manager (WPM), joka taas on Windowsin seuraaja. On kuitenkin syytä pitää nämä erillään, sillä OS/2 on käyttöjärjestelmä ja WPM on graafinen käyttöliittymä. Työaseman käyttäjille on tietysti WPM, silloin kun se on mukana, järjestelmän näkyvin osa.

Loppukäyttäjille ja järjestelmien rakentajille OS/2 ja WPM avaavat aivan uusia mahdollisuuksia. Työasemissa voidaan ajaa samanaikaisesti useita sovelluksia, siirtyä välillä toiseen ja palata taas keskeytettyyn. Kun taito kehittyy, näytön layout voidaan rakentaa mielen mukaan. Lisäksi periaatteessa sovelluksia voi työasemassa olla

rajaton määrä, sillä rahalla saa lisää muistia.

WPM taas on graafisen käyttöliittymän viimeisin sana. Sen avulla pystytään tekemään todella upeita sovelluksien käyttöliittymiä ja siksi se tietysti kiinnostaa kaikkia ohjelmistotaloja. OS/2:n ja WPM:n menestymisen viimeinen tae on se valtava tuki, jonka ne saavat järjestelmätoimittajilta ja ennen kaikkea Microsoftilta ja IBM:ltä.

Toteutuuko tämä loppukäyttäjän unelma noin vain OS/2:n ja WPM:n myötä? Voidaanko nykyisiä DOS-sovelluksia siirtää moniajoympäristöön? Voidaan, jos tyydytään siihen, että jokainen sovellus on itsenäinen, yhden käyttäjän sovellus. Eli yksittäinen sovellus ei tiedä muiden sovelluksien olemassaolosta, vaan toimii ikään kuin kaikki resurssit ja tiedot olisivat vain sen käytössä. Ehkä tämä aluksi tyydyttääkin. Myöhemmin kuitenkin varmasti halutaan, että ympäristö olisi todellinen moniajoympäristö, jossa useat sovellukset ja loppukäyttäjät voivat käyttää samoja resursseja ja tietoja.

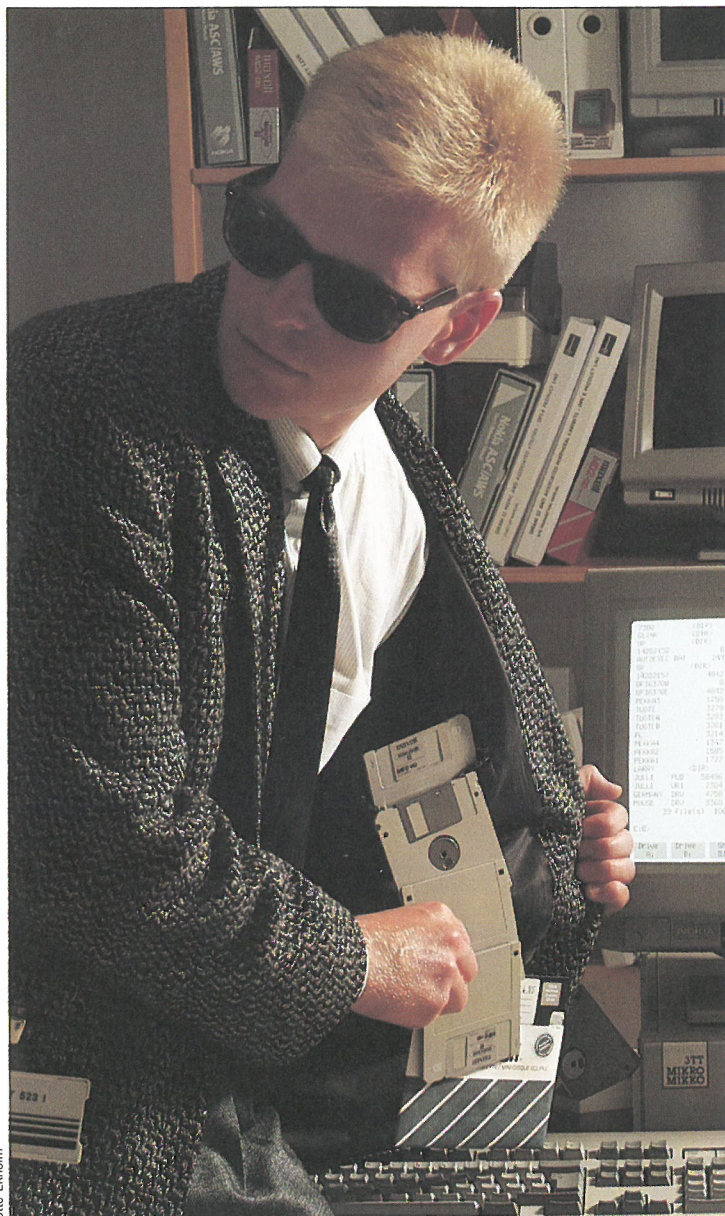
Moniajoympäristö muuttaa ohjelmoinnin työtä, sillä ohjelmointi ei enää olekaan ihan samanlaista kuin DOS-ympäristös-

sä. Kaikkia ohjelmointia rakastavia voidaan kyllä rauhoittaa. Itse ohjelmoinnin viehättävyys kyllä säilyy, mutta työhön tulee lukuisia lisäpiirteitä. Ohjelmoinnin täytyy nyt ottaa huomioon, että ohjelman käyttämä tieto voi olla samanaikaisesti jonkun toisen käytössä. Tämähän ei itse asiassa kuulosta niin merkittävältä, onhan sitä harrastettu jo kauan isoissa koneissa. Erona on kuitenkin, että ohjelman ei ole tarvinnut samanaikaisesti palvelulla satunnaisesti käyttäytävää käyttäjää.

Siis, kun siirrytään OS/2-käyttöjärjestelmään ja WPM:n käyttöön niin ohjelmointitekniikka if-lauseineen säilyy, mutta ohjelmien rakenne ja looginen toiminta muuttuvat. Ohjelman eteneminen täytyy kuvata tapahtumasarjana. Koko ajan täytyy ottaa huomioon, että ohjelman toiminta on loppukäyttäjän kannalta järkevää myös silloin, kun toimitaan monikäyttäjä- ja moniajoympäristössä. Kun ohjelmoijat tulevat sinuiksi uuden ympäristön kanssa, meillä on syytä odottaa entistä järkevämmin toimivia ja käyttäjäystävällisempiä järjestelmiä. OS/2 ja WPM antavat siihen kaikki mahdollisuudet. ■

Suojata vai ei, kas siinä ohjelmistokysymys

Uffe Ekholm



**Ohjelmistojen kopio-
suojaus on kaksijakoi-
nen, osin ristiriitainen-
kin kysymys. Vastak-
kain ovat ohjelmiston
tekijän tai tuottajan ja
ohjelmiston käyttäjän
edut – näin ainakin
yleisesti ajatellaan.
Asiaa moraalisesta nä-
kökulmasta tarkastel-
taessa voi kuitenkin to-
deta, että kyseessä on
yhteisen edun havaitse-
miseen liittyvä
probleema.**

Ohjelmistojen käytön tulisi olla mahdollisimman helppoa. Toisaalta ohjelmiston tekijälle kuuluu oikeutetusti korvaus tuotetusta hyödykkeestä. Kopioimisen estämisellä varmistetaan, että ohjelmiston tekijä saa palkan vaivastaan. Samalla sitten osittain menetetään sovelluksen joustavat käyttömahdollisuudet ja käytön helppous.

Asiakas etusijalla

Nokia Datan Tuotekehitys-yksikköön kuuluvan Tuotelinjat-osaston päällikkö Pentti Mäkipää katsoo, että painopisteen tulee ehdottomasti olla asiakkaan edun puolella. ”Sekä käyttöönoton että jokapäiväisen käytön aloittamisen helppous tulee asettaa etusijalle. Kopioinnin estämistä ei voi tehdä käytön helppouden kustannuksella.”

Yleinen trendi tuntuu olevan, että sovellusten käyttö pyritään tekemään mahdolli-

simman vaivattomaksi ja yksinkertaiseksi. Kopiosuojalla saavutettu hyöty katsotaan haittoja pienemmäksi.

Mäkipään mukaan kopiosuojaongelma koskee lähinnä viihdekäyttöön tarkoitettuja ohjelmistoja, ei niinkään tuotannollisiin tarkoituksiin suunnattuja sovelluksia. "Ongelma on silti todellinen, vaikkei se mitenkään laajasti asiakkaitamme koskekaan." Etenkin mikromarkkinoille suunnatut, yksittäisen käyttäjän hyödynnettävissä olevat ohjelmistot ovat kopioinnille otollisia kohteita.

Keinoja riittävästi

Erilaisia keinoja kopioinnin estämiseen tai vaikeuttamiseen on olemassa useita. On mahdollista varustaa joko itse ohjelma tai levyke mekanismeilla, jotka suojaavat sovelluksen reprodusoinnilta. Ohjelmaan voidaan esimerkiksi sisällyttää tekijän "allekirjoitus" tai "sormenjälki", joilla toisaalta estetään kopiointi, toisaalta varmistetaan sovelluksen tuottajan tekijänoikeus. Hankaluutena on, että tällöin estetään varmuuskopioiden ottaminen ja samalla myös tuotantoprosessi vaikeutuu.

Ohjelmistot voidaan myös rakentaa siten, että niitä voidaan käyttää vain erillisen elektronisen avaimen eli tietokoneeseen liitettävän piirikortin kanssa. Kopiointi on tällöin vapaata, mutta käyttö mahdollista vain avaimen kera. Tällä modulisuojausjaksella on Mäkipään mukaan merkitystä lähinnä vain kotimikroissa.

"Tässä sovellussuojakysymyksessä on muistettava, että kehitys kulkee ohjelmistojen verkkokäyttöä kohti. Tällöin ohjelman on oltava helposti asennettavissa", muistuttaa Mäkipää. Kiustausta verkkoversioiden kopiointiin vähentäneen Nokia Datan hintapolitiikka, jonka mukaan ohjelmiston kustannus ei kasva samassa suhteessa verkon käyttäjien lukumäärän kanssa. "Ohjelmistojen verkkokäyttöä voidaan rajoittaa määrittelemällä käyttäjien maksimilukumäärä.

Muunlaisia suojauksia ei nähdäkseen ole tulossa."

Moraali hyödyttää

Olennaista koko ongelmassa on yhteisen edun vaatiman moraalisuuden vastuun havaitseminen. "On sekä asiakkaan että ohjelmiston valmistajan etu, että sovelluksesta ja sen käytöstä suoritetaan asianmukainen korvaus", Mäkipää painottaa. "Mikäli sovelluksen tekijän ohjelmaan panostamat sekä henkiset että aineelliset sijoitukset jätetään ottamatta huomioon ja kompensoimatta, vaaraantuu myös ohjelmiston käyttöturva." Sovelluksen elinikä, kehittyminen ja ohjelmistotoimittajan tukipalvelut turvataan vain jokaisen käyttäjän suorittamilla palkkioilla.

Onko muita keinoja estää kopiointi kuin käyttöä vaikeuttavat tai puhtaasti omaantuntoon perustuvat menetelmät? Nokia Datan käyttämät menetelmät ovat oikeastaan omaatuntoa tukevia – Mäkipään mukaan yhtiön virallinen kanta on, ettei kopioinnin tiukasti estäviä mekanismeja lähdetä edistämään. "Pikemminkin edetään toisesta suunnasta eli asiakkaasta päin. Ohjelmistot suojaus pyritään takaamaan esimerkiksi hyvillä käyttöohjeilla." Käytön helppous kulminoituu käyttöohjeissa, sillä ilman niitä käytön aloittaminen voi olla varsin työlästä. "Toisaalta kyse on myös luottamusasiasta. Luotamme siihen, että asiakkaat noudattavat ohjelmistojen käyttöohjeita", jatkaa Mäkipää.

Asiakasrekisterillä voidaan kontrolloida, onko esimerkiksi tukipalveluja kaipaava todella hankkinut sovelluksensa Nokialta. Vain asiakasrekisteriin kuuluva voi lisäksi olla varma siitä, että tieto ohjelman mahdollisesti sisältämästä virheestä saavuttaa käyttäjän ja tulee korjatuksi. "Joissakin yksittäisissä tukipyyntötapauksissa vertailu rekisteriin on osoittanut, ettei ohjelmaa ole hankittu virallisilla keinoilla", paljastaa Mäkipää. Asiattomalta kopioinnilta pidättäytymisen merkitystä pyritään muun ohella tuomaan esille erilaisissa asiakas- ja koulutustilaisuuksissa.



Outi Kuusela

Luottamus luo perustan

Ohjelmistosuojakysymystä voi hyvin verrata yhteiskunnallisten etujen kustannusten jakamiseen liittyvään problematiikkaan. Melko varmasti useimmat raitiovaunu- tai junamatkustajat ovat matkansa maksaneet, vaikkei riski tarkastajan kohdalle osumisesta olekaan järin suuri. Mikäli riittävän moni jättäisi lipunsa lunastamatta, palvelujen jatkaminen ja kehittäminen olisi ennenpitkää mahdotonta. Samantapaista moraaliseen selkärankaan perustuvaa itsekontrollia tarvitaan myös sovellusten käytön korvausten suorittamisessa.

"Yleisesti ottaen voi todeta, ettei kyseessä ole mikään suuri ongelma. Nokialla panostetaan, kuten todettu, käytön helppouteen, hyviin käyttöohjeisiin ja luottamukseen. Uskomme, etteivät itseään arvossa pitävät asiakkaat laittomia kopioita teekään", päättää osastopäällikkö Mäkipää. ■

"Ohjelmistojen suojaus kopioinnilla on asia, joka on hoidettava niin, ettei ohjelmistojen käyttö muodostu asiakkaalle hankalamaksi", sanoo Pentti Mäkipää Nokia Datasta. Nokia Data ei lähde edistämään tiukasti kopioinnin estäviä mekanismeja. Ohjelmistosuojaus pyritään takaamaan esimerkiksi hyvillä käyttöohjeilla.

Paradoxissa tehty kysely, jossa on haluttu tietää mitä asiakas on tilannut ja kuinka paljon. Asiakas- ja tilaustiedot on yhdistetty asiakasnumeron avulla.

View Ask Report Create Modify Image Forms Tools Scripts Help Exit
Get a query form to ask questions about a table.

ASIAKAS	Asiakasno	Sukunimi	Etunimi	Puhelin
	asnro	J	J	

TILAUS	Tilausno	Asiakasno	Tuote	Määrä
		asnro	J	J

ANSWER	Sukunimi	Etunimi	Tuote	Määrä
1	Ahonen	Martti	Uusara	1
2	Kuparinen	Kauko	Saha	1
3	Mäkinen	Uuno	Ruohonleikkuri	1
4	Mäkinen	Uuno	Uusara	5
5	Nieminen	Pekka	Saha	3
6	Suokko	Sakari	Saha	2

Uffe Ekholm

Käyttäjän ja suunnittelijan Paradox

Paradox-relaatiotietokantaohjelmisto on suunniteltu nimenomaan käyttäjän hyödynnettäväksi, mutta se sisältää myös hyvät sovelluskehitysvälineet.

Paradox-ohjelmiston hienoutena on sen kyky käsitellä joustavasti ja helposti relaatiotietokantoja. Tällaiset järjestelmät, joissa kootaan yhteen useissa eri taulukoissa (kortistoissa) olevia tietoja, ovat varsin hankalia toteuttaa muilla vastaavilla ohjelmistoilla ja siinä vaaditaan ohjelmointia. Paradox taas tekee useiden taulukoiden yhtäaikaisen käytön helpoksi eikä edellytä käyttäjältä ohjelmointia.

Toinen Paradoxin erikoisuus on älykäs verkkotuki. Tietueiden lukitus tapahtuu automaattisesti toisin kuin dBasessa, jossa se on tehtävä ohjelmoimalla ja erityisillä komennoilla. Paradox päivittää myös tietokantaa automaattisesti, jolloin yhdeltä työasemalta kortistoon tehty muutokset näkyvät välittömästi kaikkien muiden työasemien näytöillä. Taulukot voidaan suojata käyttäjäkohtaisin oikeuksin.

Periaatteessa Paradoxin verkkoversiolla voi olla rajattomasti käyttäjiä, mutta yhtäaikaisten käyttäjien enimmäismäärä riippuu sovelluksesta ja verkon laitteista. Käytännössä raja kulkee jossain 5-15 käyttäjän tienoilla. Suurimmissa suomalaisissa sovelluksissa on toistakymmentä käyttäjää.

PC-ympäristön lisäksi Paradoxista on vaativimpien ympäristöjen erityispiirteet hyödyntäviä versioita. Nyt ovat saatavissa versiot 80386-suorittimelle ja OS/2:lle, joka on ensimmäinen todellinen OS/2-sovellusohjelma.

Valmis loppukäyttäjän hyödynnettäväksi

Paradoxia voi hyödyntää tehokkaasti ilman erillisiä sovelluksia. Tiedon rakenteen määrittely, syöttö, selaus, haku, lajittelu ja raporttien tulostus tehdään valitsemalla komennot Lotus-tyyppisistä valikoista ja käyttämällä taulukoiden selaus- ja muokausnäppäimiä.

Tiedot esitetään näytöllä joko taulukko- tai lomakemuodossa. Muodosta toiseen siirrytään yhdellä näppäimenpainalluksella. Taulukoita voi olla näytöllä useita samanaikaisesti. Taulukko-muodossa tietueet näytetään riveittäin. Kaikki kentät eivät useinkaan mahdu näytölle samanaikaisesti, mutta kenttien järjestystä näytöllä voi vaihtaa. Lomakemuodossa näytöllä näkyvät vain yhden tietueen tiedot kerrollaan. Laajoja tietueita varten voi lomakkeen jakaa useille näyttösiiville. Paradox luo peruslomakkeen, josta käyttäjä voi näytöllä maalata omia lomakkeitaan.

Taulukon katselutilasta siirrytään yhdellä näppäimenpainalluksella muokkaustilaan, jossa voi myös poistaa tai lisätä tietueita. Yhden lomakkeen kautta on mahdollista syöttää tietoa moniin taulukoihin. Paradoxin nerokas virhekontrolli estää tietojen muuttamisen vahingossa.

Tietojen haku tehdään esimerkkien avulla (Query-By-Example, QBE). Paradox luo

taulukon kenttien mukaisen hakulomakkeen, josta osoitetaan mukaan otettavat kentät. Lisäksi jokaiseen kenttään voi kirjoittaa hakuehdon. Esimerkiksi henkilötaulukon hakulomakkeen etunimikenttään kirjoitettu teksti "Matti" rajaa haun koskemaan henkilöitä, joiden etunimi on Matti. Valmis testattu hakulomake voidaan tallentaa myöhempää käyttöä varten.

Raportin ulkoasu kuvataan raporttipohjassa, joita voi yhdellä taulukolla olla useita. Paradox luo joko taulukko- tai vapaa- muotoisen perusraporttipohjan, josta käyttäjä voi suunnitella lopullisen raportin muodon näytölle maalaamalla. Vapaamuotoinen raportti soveltuu esimerkiksi laskujen, joukkokirjeiden tai osoitetarjosten tulostukseen.

Tietueet voidaan ryhmitellä kentän arvon tai tietueiden määrän perusteella. Raportille voidaan määrittää alku- ja loppu-tekstit sekä ylä- ja alatunnisteet. Raporttipohjaan voidaan lisäksi sijoittaa laskukaavoja. Osoitetarjoja voidaan tulostaa monia rinnakkain. Raportti voidaan määrätä tulostuvaksi näytölle, kirjoittimelle tai tiedostoon.

Paradoxissa on monipuolinen avustustoiminto. Yhdellä näppäimenpainalluksella saa apua meneillään olevasta toiminnosta. Avustusta voi hakea myös hakusanan perusteella.

Paradoxin käyttöä automatisoidaan käyttämällä hyväksi käskyjen nauhoitus- ja toisto-ominaisuuksia. Automatisointia voi viedä pidemmälle tekemällä omia sovelluksia Paradoxin mukana tulevalla valikko-ohjatulla Personal Programmer -sovelluskehittimellä. Tällöin sovellukseen saa esimerkiksi oman valikkorakenteen.

Sovelluskehittäjän Paradox

Personal Programmer tuottaa PAL-kielisen (Paradox Application Language) ohjelman, jolla myös sovelluskehittäjät voivat kirjoittaa ohjelmia. Tällöinkin kannattaa käyttää Personal Programmeria ja ohjelmoida vain toiminnot, joita ei pysty valikko-ohjatusti kuvaamaan.

Integroituun sovelluskehitysympäristöön kuuluu sisäänrakennettu muokkain ja virheen jäljitysohjelma. Ympäristöön voi liittää myös oman suosikku- muokkaimensa.

Data Entry Toolkit on sovel-

luskehittäjän apuväline tiedon syötön hienosäätöön. Sen avulla voidaan syötettäviin kenttiin liittää omia tarkastustoimintoja.

Developers Toolkit -lisäosa sisältää valmiita aliohjelmarutiineja, joita voi liittää omiin ohjelmiin.

Paradox-tietokantoja voi käyttää Turbo C ja Turbo Pascal -ohjelmista lisäosana saatavan liittymän kautta.

Hyödyntää 80386-suorittimen ominaisuudet

Paradox/386 on vakuuttava osoitus Ansa Softwaren kyvystä pysyä mukana alan kiivaassa teknisessä kehityksessä. Paradox/386-versio murtaa MS-DOS:n 640 kilotavun muistirajan. Ohjelma osaa käyttää 16 megatavun muistia ja 80386-suorittimen koko käskykanta. Käyttäjälle tämä näkyy vaativien toimintojen jopa viisinkertaisena nopeutumisen verrattuna perusversioon.

Vain murto-osan Paradox/386:n käyttämästä työmuistista täytyy sijaita kriittisellä 640 kilotavun alueella. Loppu muisti otetaan lisämuistikortilta, jonka on oltava extended-tyyppinen. Tämä helpottaa muun muassa paljon muistia vaativan Personal Programmerin käyttöä.

Paradox/386 vaatii laitteelta 80386-suorittimen, jota se käyttää nk. suojatussa tilassa. Tämän vuoksi ohjelmaa ei voi käyttää samanaikaisesti Windows/386:n kanssa eikä myöskään OS/2:n DOS-tilassa.

Paradox/386:n käyttöliittymä ja sillä tehdyt taulukot sekä sovellukset ovat yhteensopivia Paradoxin perusversion kanssa.

Moniajoa OS/2-versiolla

Paradox OS/2:n avulla on mahdollista hyödyntää uuden käyttöjärjestelmän moniajo-ominaisuuksia. Esimerkiksi suuren taulukon tulostuksen ajaksi voi käynnistää uuden Paradox-istunnon tiedon syöttöä varten. Paradox OS/2 osaa käyttää 16 megatavun työmuistia.

Nyt Paradoxin tekijä kehittää ohjelmasta versioita OS/2 Presentation Managerille ja UNIXille. Tulossa on myös erillinen SQL-moduli, jonka avulla Paradoxista päästään suoraan käyttämään mm. OS/2-palvelimessa olevia SQL-tietokantoja. ■

Paradox-taulukko voi sisältää:

2 miljardia tietuetta
255 kenttää
4 000 merkkiä tietueessa
255 merkkiä kentässä
15 15-sivuista lomakepohjaa
15 raporttipohjaa

Tietotyytit:

Número: 15 merkitsevää numeroa
Lyhyt número: -32767 - 32767 (vain kokonaisluvut)
Merkkijono: 1 - 255 merkkiä
Päiväys
Raha

Tiedon siirto:

Siirto molempiin suuntiin seuraavista muodoista:

Lotus 1-2-3 1A, 2.0
Symphony 1.0, 1.1
dBASE II, III, III Plus
pfs:file
DIF
ASCII

Laitevaatimukset:

Paradox 2.0

Teollisuusstandardin mukainen mikro
Vähintään 640 kilotavua työmuistia
DOS 2.0 tai uudempi
Kiintolevy
EMS/EEMS lisämuistikortti (optio)
8087, 80287 matematiikkasuoritin (optio)

Paradox/386

80386-suorittimella varustettu mikro
Vähintään 1,5 megatavua työmuistia
DOS 3.0 tai uudempi
Kiintolevy
80287, 80387 matematiikkasuoritin (optio)

Paradox OS/2

80286- tai 80386-suorittimella varustettu mikro
Vähintään 3 megatavua työmuistia
OS/2 1.0 tai uudempi
Kiintolevy
80287, 80387 matematiikkasuoritin (optio)

Tero Koskivirta työskentelee Nokia Datan Tuotekehityksessä PC Applications -ryhmässä ohjelmistoasiantuntijana.



saari/foto

Hajautettuna työryhmäympäristönä mikrot yhdistävä lähiverkko on erinomainen ratkaisu. Kaikkien saavutettavissa olevissa verkon palvelimissa sijaitsevat yhteiset tiedostot ja resurssit. Tiedosto-, kirjoitin- ja tietoliikennepalvelimien rinnalle tulee nyt tarjolle myös tietokantapalvelin. SQLBase on DOS- ja OS/2-pohjaisissa lähiverkoissa toimiva tietokantapalvelinohjelma.

SQLBase

Tietokantapalvelin

jakaa yhteiset tiedot

SQLBase on mikroverkossa sekä yksittäismikrossa toimiva reaaliomallinen tiedonhallintaohjelmisto. Muista mikrojen tiedonhallintaohjelmista se eroaa siinä, että tietokantojen käsittely tapahtuu keskitetysti tietokantapalvelimissa sijaitsevassa SQLBase-ohjelmassa eikä jokaisessa työasemassa erikseen. Kuitenkin varsinaiset käyttäjäsovellukset käyttäjäsovellukseen ja tietokantatyökalut toimivat itsenäisesti jokaisen käyttäjän työasemassa kuten työryhmäverkon ideologiaan sopii. Sovellukset keskustelevat palvelimissa toimivan SQLBasen kanssa suoraan lähiverkon alemman tason NetBios-kommunikointiprotokollan välityksellä. Toiminnallisesti prosessointi on siis hajautettua.

Mikroverkkoon kehitetty tietokantajärjestelmä

Tiedonhallinta on ollut pitkään sovellusalueella, jossa mini- ja suurtietokoneet keskitettyine ratkaisuineen olivat huomattavasti mikrotietokonepohjaisia järjestelmiä kehittyneempiä. Mikrojen tietokantajärjestelmistä puuttui kyky suorittaa monimutkaisia tietokantatapahtumia tehokkaasti. Nytemmin on kuitenkin mikrotietokoneiden kehittyminen ja tehostuminen mahdollistanut täydellisen tietokantajärjestelmän rakentamisen myös mikroverkko-ympäristöön. Niinpä vaikka SQLBase on kehitetty nimenomaan mikroverkkoon, se on toiminnoiltaan ja ominaisuuksiltaan kilpailukykyinen suuren koneiden tietokantajärjestelmien kanssa.

Suorituskyvyn loppuessa mikroverkon tiedonhallintaan saadaan lisää palvelukykyä lisäämällä tietokantapalvelimien määrää ja jakamalla verkossa käytössä olevat tietokannat tasaisesti eri tietokantapalvelimiin. Tämä poikkeaa suuresti perinteisistä mini- ja suuren kone-ratkaisuista, joissa ainoa suorituskykyongelmien ratkaisukeino on hankkia entistä järeämpi tietokone tietokantaohjelmaa suorittamaan.

SQLBase-ohjelmisto pystyy käsittelemään kaikkia samassa palvelimessa olevia tietokantoja rinnakkain. Syytä on kuitenkin muistaa, että DOS:n alaisuudessa toimivia tietokantapalvelimia ei voida käyttää samanaikaisesti työasemina tai tiedosto- ja kirjoitinpalvelimina. Vasta OS/2:n moniajoympäristö tarjoaa mahdollisuuden usean palvelun palvelimiin.

Tietokannan käsittelykielenä SQL

SQLBasen komentokielenä on standardien mukainen täydellinen SQL (Structured Query Language). Muita ohjaus- tai komentokieliä ei tarvita, sillä SQL sisältää kaikki tietokantojen käsittelyssä tarvittavat komennot aina tietokannan rakenteen muokkauksesta käyttöoikeuksien määrittelyyn.

Vaikka SQL on kehitetty luonnollisen kielen muotoon ("english like"), ei se kuitenkaan tarkkojen syntaksisääntöjensä takia ole loppukäyttäjälle soveltuva kyselykieli. SQL on pikemminkin työasemasovellusten ja SQLBasen välinen keskustelukieli. Niinpä työasemien sovellukset muodostavat käyttäjien antamien syötteiden perusteella vastaavia SQL-lauseita ja välittävät ne sellaisenaan tekstimuodossa tietokantapalvelimen SQLBase-ohjelmalle käännet-

täväksi ja suoritettavaksi.

SQL:n merkitys kasvaa jatkuvasti, koska se on valittu SQLBasen lisäksi yhä useamman mikro-, mini- ja suurtietokoneen tietokantajärjestelmän kyselykieleksi. Myös monelle omaa komentokieltä käyttävälle tietokantaohjelmistolle on jätetty lisäty SQL-tuki. SQL:stä on muodostunut sekä markkinoiden "de facto" että kansainvälisten ANSI- ja OSI-standardointijärjestöjen standardi. Vielä on kuitenkin pieniä eroja eri "murteiden" välillä.

Tietokanta- eikä tiedostopalvelin

Useimmissa nykyisissä mikroverkon tiedonhallintaohjelmissa tietokannan yhteiskäyttö perustuu yhteisen tiedostopalvelimen ja siten yhteisten tiedostojen käyttöön. Vaikka itse tiedot sijaitsevat jaetuissa tiedostopalvelimen tiedostoissa, niin varsinaisen tietokannan käsittely tapahtuu kuitenkin jokaisessa työasemassa erikseen.

SQLBasen yhteiskäyttö ei perustu tiedostopalvelimeen, vaan siinä tietokantoja käsitellään keskitetysti yhdessä tietokantapalvelimessa. Se pystyy palvelemaan samanaikaisesti useita työasemasovelluksia. Niinpä yhden tietokannan fyysisiä tiedostoja käsittelee aina ainoastaan yksi tietokantaohjelma käyttäjien määrästä riippumatta. Tietokantaohjelma on siten koko ajan täysin tietoinen tietokannan tilasta, koska mikään muu ohjelma ei tietokantaa käsittele. Etuna seuraa esimerkiksi, että tietokannan yhteiskäytön hallinta luokitukseen ja tietokannan puskurointi nopeaan välimuistiin (cache-muisti) vievät vain minimaalisesti suoritusaikaa.

Toinen tietokantapalvelimen käytön etu verrattuna tiedostopalvelimen käyttöön on kommunikoinnin minimoituminen lähiverkossa. Tietokantapalvelin lähettää työasemille ainoastaan työaseman pyytämät tiedot eikä mitään ylimääräistä. Sama tietokantapalvelin hoitaa haluttaessa tietojen ryhmittelyyn, järjestämisen ja suorittaa tietokannan tietojen pohjalta laskutoimituksia.

Esimerkiksi haettaessa useita tietueita esimerkiksi 10 000 tietuetta sisältävästä asiakastaulukosta pienellä paikkakunnalla sijaitsevat asiakkaat (5 tietuetta), niin tiedostopalvelimeen perustuva tiedonhallintaohjelmisto hakee verkon läpi työasemaan kaikki 10 000 asiakastietuetta, joista työasemassa sitten seuloa halutut viisi tietuetta. Sitä vastoin tietokantapalvelimeen perustuva ohjelma lähettää työasemaan suoraan ainoastaan pyydyt 5 tietuetta.

nen kopio alkuperäisestä tietokannasta.

Myös väärinkäytösten ehkäisemiseksi SQLBasessa on monia suojakeinoja. Ensinnäkin tietokantapalvelimen levymuistissa sijaitsevaa tietokantaa ei käyttäjä pääse käsittelemään muuten kuin SQLBasen kautta käyttämällä SQL:ä. Toisaalta SQLBaseen kuuluva käyttöoikeuksien tarkastaja huolehtii, että ainoastaan käyttöoikeudet omaavat

tion Manager -ympäristöön tuleva SQLWindows. Siinä sovellusten graafinen käyttöliittymä piirretään hiiren avulla ja sovellusten logiikka kirjoitetaan SQLWindowsin omalla korkean tason ohjelmointikielellä.

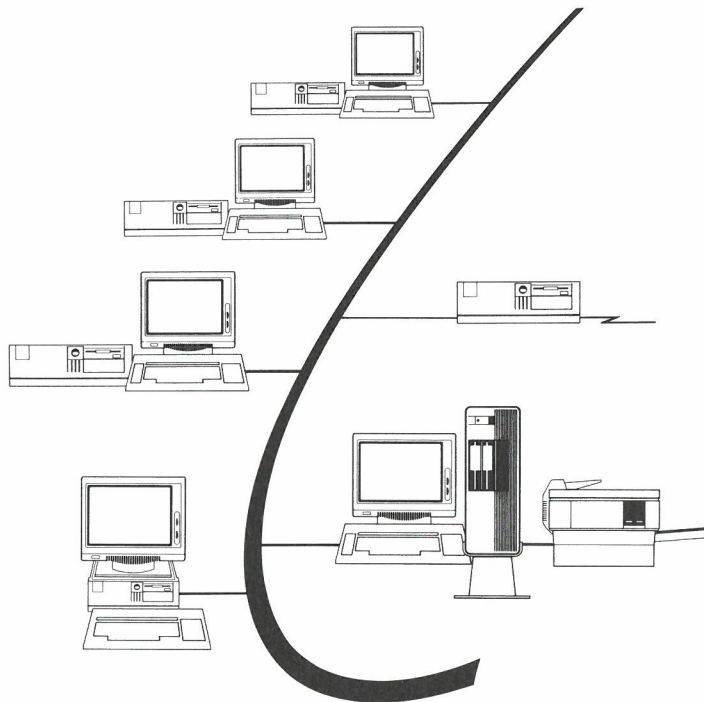
Raporttien tuottamiseen on erikseen vielä oma ohjelmansa: SQR Report Writer. Sen yksinkertaisella mutta tehokkaalla komentokielellä voidaan rakentaa monimutkaisiakin raportteja.

jien avulla voidaan upotetut SQL-lauseet muuttaa käytettävän SQL-tietokantaohjelmiston määräämään funktiomuotoon.

Esikäntäjiä käytettäessä minimoituu muutostyö, kun siirretään sovelluksia eri ympäristöihin. Jopa siinä määrin, että yleensä pelkkä koodin uudelleen kääntäminen riittää. Koska eri SQL-ohjelmistot eroavat toisistaan, niin jokaisella on oltava omat esikäntäjät. SQLBaseen on saatavilla C-kielen esikäntäjä ja tulossa on myös Cobolin esikäntäjä.

Toimintaympäristö

- * SQLBasen saa joko työasema-kohtaisena tai verkkoversiona
- * Vähintään 640 kilotavua muistia
- * Lisämuisti suositeltavaa
- * MS-DOS 3.1 tai uudempi
- * Verkkoversio vaatii lisäksi NetBiosin
- * Osaa käyttää suoraa (extended) lisämuistia aina 16 megatavuun saakka



käyttäjät pääsevät tietokantaa käsittelemään. Tietokannan ylläpitäjä voi antaa käyttäjille taulukko-kohtaisesti luku-, kirjoitus-, lisäys- ja poisto-oikeuksia.

Sovellusten kehittämiseen omat työkalut

SQLBase on pelkkä tietokantaohjelmisto, eikä se siten sisällä sovelluskehitysohjelmia. SQLBasen toteuttamaan SQL-komentokielen ei kuulu muuttujien käsittelyyn, käyttöliittymän muodostamiseen eikä ohjelmarakenteen määrittelyyn tarvittavia komentoja. Niinpä tarvitaan erilliset työkalut työasemissa toimivien sovellusten rakentamiseen.

Sovellukset voidaan tehdä ohjelmointikielellä ja SQLBasen työasemien SQL/API-ohjelmajapintaa käyttämällä. Vaihtoehtoisesti voidaan sovellukset tehdä tarjolla olevilla SQL-sovelluskehittimillä. Ensimmäinen tällainen yleinen SQL-sovelluskehitin on Windows 2 -ympäristöön kehitetty ja myös Presenta-

Komentokieli sisältää ohjelmarakenteet, SQL-lauseet, muuttujien käsittelyn ja tulostuksen ohjauksen. Valmiista komentokieliestä tiedostosta SQR-ohjelma muodostaa halutunlaisen raportin tulostiedostoon, joka voidaan sitten joko jatkokäsitellä tai tulostaa suoraan kirjoittimelle.

Esikäntäjillä varmistetaan siirrettävyys

Jos halutaan varmistaa SQL-sovellusten siirrettävyys toisiin ympäristöihin, niin tulee pitäytyä SQL:n standardoiduissa ja yleisesti hyväksytyissä ominaisuuksissa ja komennoissa. Eri toimittajien tekemät SQL:n laajennukset voivat poiketa suurestikin toisistaan.

Ohjelmointikielellä tehtyjen sovellusten siirrettävyyttä vaikeuttaa se, että ohjelmointikielten liittymät ovat eri SQL-ohjelmistoihin erilaiset. Olemassa on kuitenkin standardi ohjelmointikieleen upotetusta SQL:stä (Embedded SQL). Esikäntä-

SQLTalkilla ad hoc -raportteja

SQLBase-toimitus sisältää vuorovaikutteisen tietokannan käsittelyohjelman SQLTalkin, jolla käyttäjä voi muodostaa SQL-lauseita SQLBasen suoritettavaksi. SQLTalkia voidaan käyttää esimerkiksi tietokantojen rakenteen määrittelyyn, sovelluksissa käytettävien lauseiden testaamiseen ja ennen kaikkea ennalta määrittelemättömien raporttien tekemiseen. Kaikki SQLBasen SQL-komennot ovat SQLTalkissa käytettävissä.

Ad hoc -raporttien muodostaminen SQLBase-tietokannasta edellyttää ainakin toistaiseksi SQL-osaamista, koska ainoa vaihtoehto on SQLTalkin käyttäminen. Apunaan SQLTalkin käyttäjillä on vuorovaikutteisessa SQL-lauseiden muokkaamisessa ainoastaan joko käyttäjärjestelmän oletusmuokkain (DOS:ssa EDLIN) tai käyttäjän itse valitsema muokkain.

Vaikka SQLTalk suorittaa kerrallaan ainoastaan yhden SQL-lauseen, käyttäjä voi tehdä useasta peräkkäisestä SQL-lauseesta koostuvan ajotiedoston (script), jonka suoritusta voidaan jälkepäin käynnistää yhdellä komennolla. Ajotiedostoja voidaan käyttää esimerkiksi tietokantojen ylläpitoon kuuluvien toimintojen kuten varmuuskopion oton automatisointiin.

SQLTalkista on merkkipohjaisen version lisäksi graafinen Windows 2 -pohjainen versio, joka on hankittavissa erikseen.

SQLBase tehostaa ohjelmoijien ja systeemisuunnittelijoiden työtä

Jos on tarvetta rakentaa keskikokoinen tai iso sovelluskokonaisuus mikroverkkoon, niin SQLBase on hyvä vaihtoehto tietokantajärjestelmäksi. Sovelluskehityksessä tarvitaan vielä kuitenkin systeemisuunnittelijoita ja mahdollisesti ohjelmoijia, koska sovellukset on tehtävä joko kolmannen sukupolven tai SQLWindowsin ohjelmointikielellä. ■

Marko Nyroos työskentelee Nokia Datan Tuotekehityksessä Application Services-ryhmässä tietokantaohjelmistojen tuotepäällikkönä.



Exploring the OS/2 Multitasking

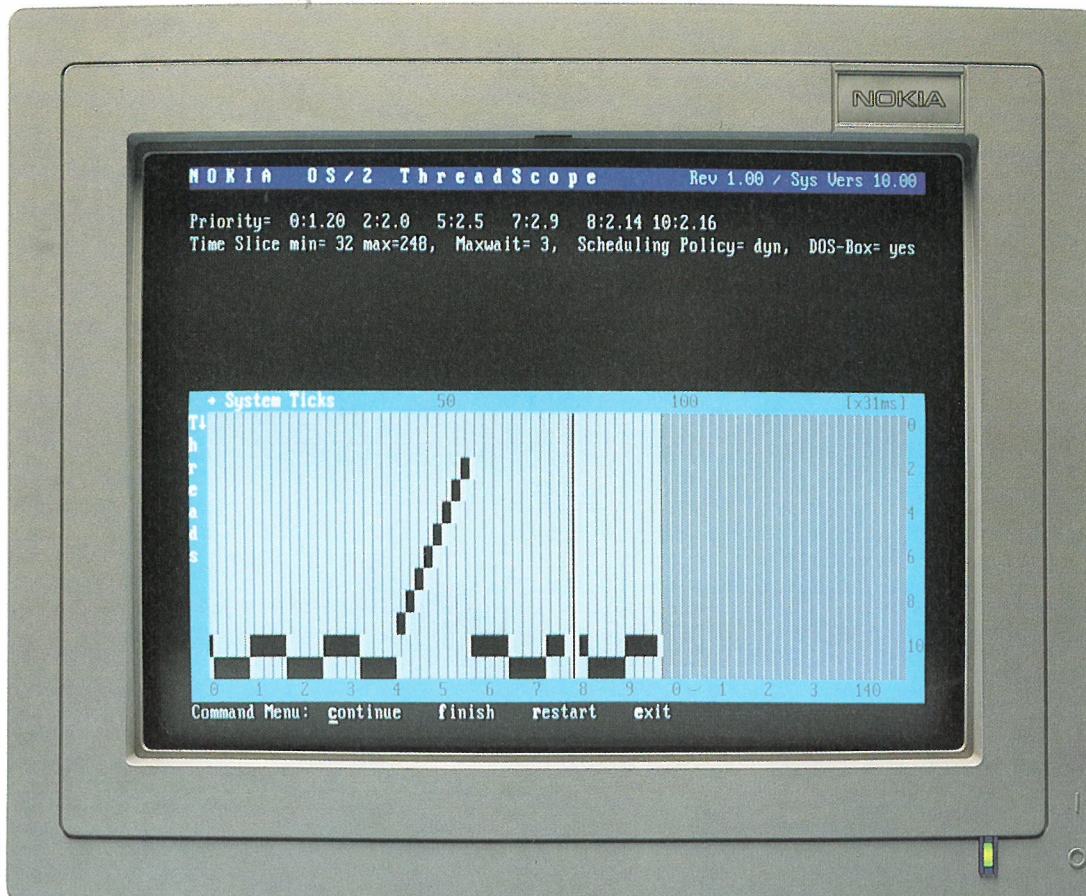
Besides other contributions, OS/2 brings with it a sophisticated multitasking facility that offers a new type of functionality in the PC environment. As with many other dynamic processes, this multitasking is best understood by observing its behavior. For that purpose, a program was designed and implemented to facilitate a view of the working OS/2 threads via an attractive and easy to understand graphic user-interface. Some of the most interesting results are presented and discussed.

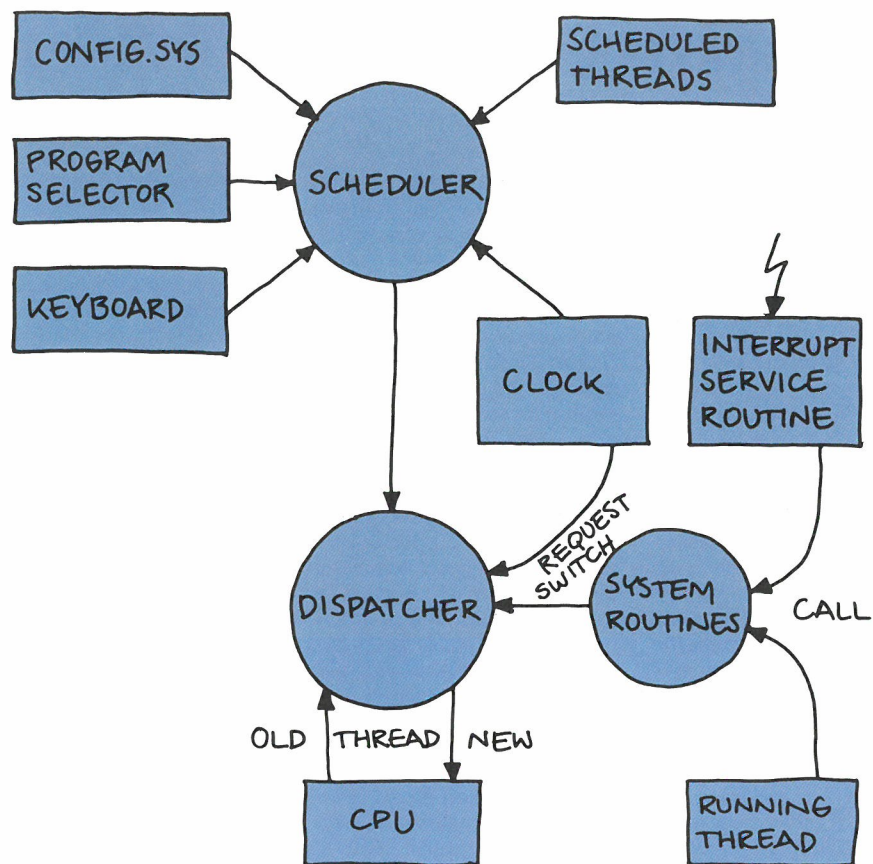
Obviously, somewhere inside OS/2 exists a built-in mechanism for performing the multitasking that gives the appearance of executing more than one task simultaneously. But bear in mind that the personal computer physically has only one processor. Thus, OS/2 has to switch rapidly from one instruction sequence, called a thread, to the next. This switching or multiplexing of the CPU is the principle of any multitasking system (see box for more details). But from that simple generic concept arise the questions: which one is the next thread? and at what time intervals should a switch be performed? This is what characterises any multitasking system and OS/2, as no exception, uses its own fairly sophisticated strategy, too.

Much of the current OS/2 literature has been devoted to this subject (see enclosed references for some examples). But when reading such descriptions you will probably notice the trade-

off each author has to make between the precision and readability of the text. Despite the simple concept of multitasking, the detailed design of the election algorithm is much more complex, making it difficult to describe completely in one sentence. Yet there should be a broad knowledge of it among programmers and users. Instead of reading ambiguous explanations, such a subject is often best understood by experimenting with it, by observing the reality through a graphic display.

The graphic representation of threads in action exposes some insight that is otherwise hard to derive; moreover, it becomes most powerful if the user interacts with the system by controlling some parameters, thus exploring unknown territory. That is the intention of the approach taken in the following, the outcome of which was the implementation of an experimental or demonstration tool called the ThreadScope program.





OS/2 Multitasking Figure 1 represents a typical, simplified model of the OS/2 multitasking mechanism in the standard notation of a Data Flow Diagram. This view starts at the top with the set of actually scheduled threads as logical objects in a data storage element. Applications running in the OS/2 system will frequently put in several threads ready for execution at any time. A thread in OS/2 is a particular path of instructions through an individual application which, from a user's point of view, is executed by one of a multiple of virtual CPUs. Any application program consists of at least one thread; but, typically, there are more of them to perform multiple operations overlapping during execution. However here, in the microscopic view of this model, a thread is the manageable object of the multitasking code itself. A thread is scheduled and placed logically in the storage of scheduled threads if it is ready to execute. One out of this set will get the CPU.

At the bottom, the diagram ends by dispatching the CPU. That is typically a rather simple, straightforward piece of assembly code, called the Dispatcher. On an order from a system routine, or at periodic ticks of the clock, the Dispatcher switches the CPU from one thread to another. If triggered by the tick or another interrupt, the switch is invisible for the current running thread, and it may occur in any unprotected instruction sequence. This is referred to as the preemption ability. The decision making procedure in between, called the Scheduler, is the critical design factor of any multitasking system. At the request of the Dispatcher, the Scheduler has to identify, using a defined deterministic algorithm, the thread presently on top of the scheduled set, including that currently being executed. Scheduling in OS/2 is performed according to a complex, stratified scheme which is driven by the following inputs: - The individual priority value of each

thread. Some of the system configuration parameters, which are Maxwait, Priority and Timeslice. - The Program Selector status, which means the session currently selected for the foreground as opposed to those in the background, also called the foreground/background subcategory. - The actual keyboard assignment, which means the threads currently interacting with the keyboard, called the interactive subcategory.

The scheduler's decision is mainly priority-based. All threads have a priority assigned to them and that priority - besides the configuration parameters and system status - is used to decide which thread to pick. The priority is a composite variable consisting of an assignment for a class out of a set of three classes (also called categories) and, within each class, a level value in the range of 0 to 31. Classes 1 to 3 are sometimes also described as low or idle, general or normal and time-critical or force-run.

The Tool

The program used for the experiments yields to a 100% CPU bound situation. All threads created for the purpose of an experiment are competing for the CPU. None of them blocks itself. Their only task is to be a member of the set of scheduled threads such that the system has to handle them. They all execute the one code that creates the real-time display seen by the user. Each of them consumes the time allowed it by the system totally under the assumption there is no other activity interrupting such a running thread. That means it is up to the system to decide within the set you created which thread to run for each system tick. All threads so far preempt each other. And that behavior is what you will see in real-time on the screen.

The major feature of the program is the real-time display of the running threads versus time, as shown in photo 1. The horizontal axis represents a time axis of system ticks, with two ticks for each strip. Alongside the vertical axis, each thread has its own row in which to display a black box for any tick it is consuming. A vertical cursor (or sweep), running from left to right, represents real time and updates the past scheduling image in terms of the actual situation. That way the outcome of the OS/2 thread scheduling mechanism can be seen at a glance. This display, while a living one, can be frozen at any instance to allow a detailed look – as was done for all photos shown.

Because of the fixed conditions of such an experiment, there is always a periodically repeated scheduling sequence. In some cases, this period is smaller than the width of the display frame. By interactively decreasing the actually used length of the time axis down to the period, it is possible to get a stable but living screen. That synchronization feature was used in photo 1, as you can see from the shadowed, clipped area to the right. Such a display very much resembles the image of the ordinary oscilloscope used to show voltage versus time. The chain of black boxes represents something like a waveform generated by the system. That is why the program is called ThreadScope.

Experimental Parameters

To experiment, you have to know the parameters that affect the behavior. The experiments carried out mainly cover the capabilities of the thread priority and the configuration parameters (see box for further discussion). The user may enter values for Maxwait and/or Timeslice into the CONFIG.SYS file. After reboot, he or she will see the system's reactions to such a variation. A number of threads of any priority class or level can be created interactively by the ThreadScope program, and after observation of the scheduling results the set of threads may be modified immediately for a new experiment.

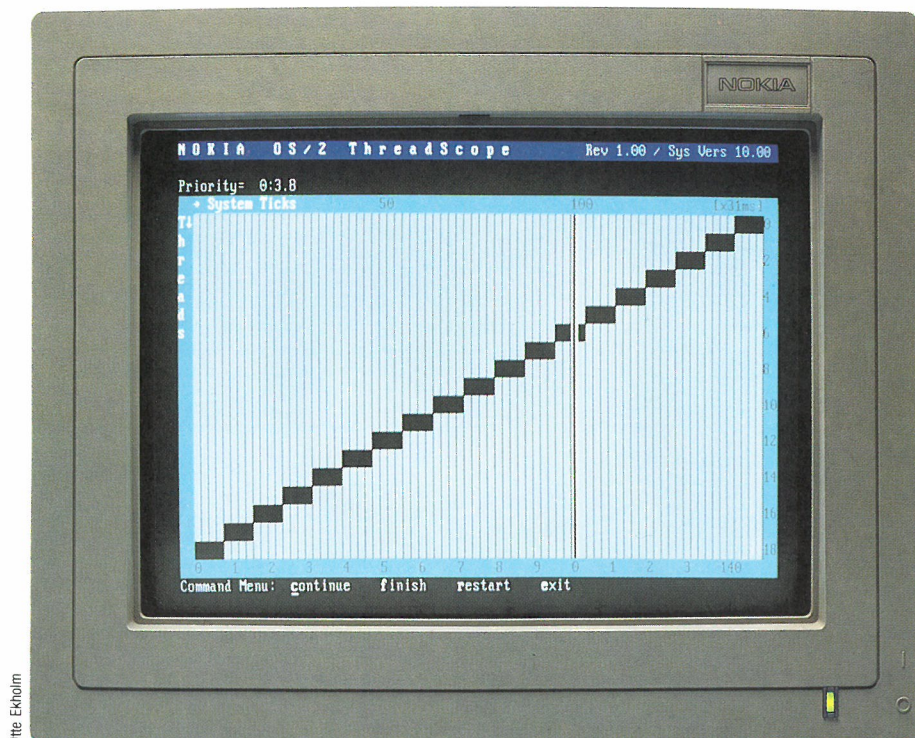
The effects of the foreground and background status can be observed to a limited extent by running the program in several sessions. However, the program provides no means of showing the scheduling of the sessions in the background. A fast switch to such a session simply pops up on the virtual screen flashing a small portion of its background life for an instant before being overwritten. The keyboard assignment is bound by the program structure and it can not be experimented with. Thread 1, the master thread created immediately by the system at program startup, is used as the experiment supervisor, which interacts with the user, receiv-

ing all keyboard inputs. If running in the foreground, the process belongs to the interactive subcategory.

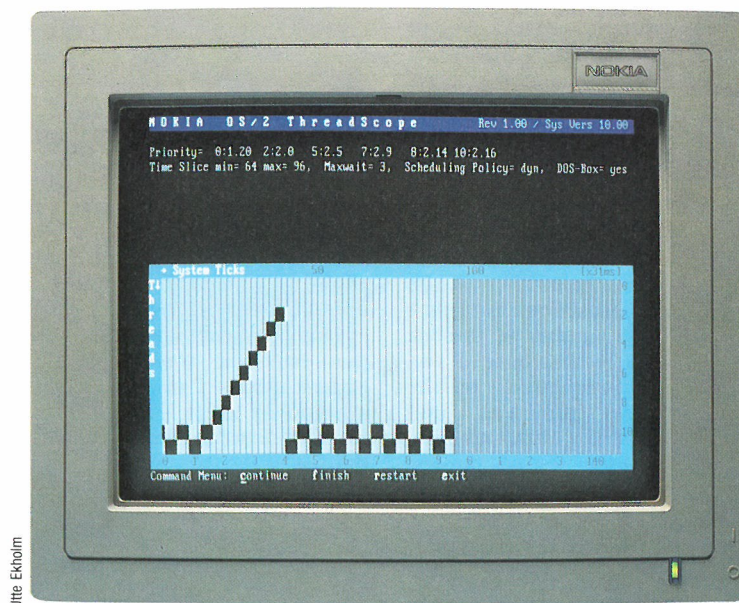
Threads in Operation

The primary factor in executing a thread is its priority class in comparison with the other scheduled threads. That can easily be checked by running a mix of threads in different classes. Such experiments will show that only the threads in the highest class – 1 to 3 in ascending priority order – are of interest. For example, the two class 1 threads of the experiment shown in photo 1 never get the CPU. Their rows on the display are completely blank. Next, a mix of threads within each class can be explored. In class 1 or 3, all the highest level threads – 0 to 31 in ascending priority order – take turns in a simple round-robin fashion, as demonstrated in photo 2. The next one started is the one suspended the longest. All lower level threads are totally starved of CPU time.

Of more interest are mixes of threads in class 2. For example, the experiment in photo 1 is set up with 10 threads of varying levels in class 2. The ramp of boxes in the center of photo 1 is formed by 8 threads in class 2, but with a priority level less than the highest one. Let us call this the low cluster of threads, as op-



Uffe Ekholm



posed to the threads of highest priority, called the high cluster. The latter are shown in the lowest 2 rows of the display. It is an exceptional feature of class 2 that all threads in this class are assured of a chance to run. However, the user can disable that by placing the entry Priority=Absolute in the CONFIG.SYS file. Referring to the OS/2 manual, each thread in the low cluster should get a CPU share every Maxwait period. A quick glance at photo 1 shows a period of 96 ticks, which was the size the active area was squeezed to in order to get a stable, synchronized image. This results, when

multiplied by the tick time of 31.25 ms, in a time of exactly 3 seconds, which is precisely the Maxwait default value as used here.

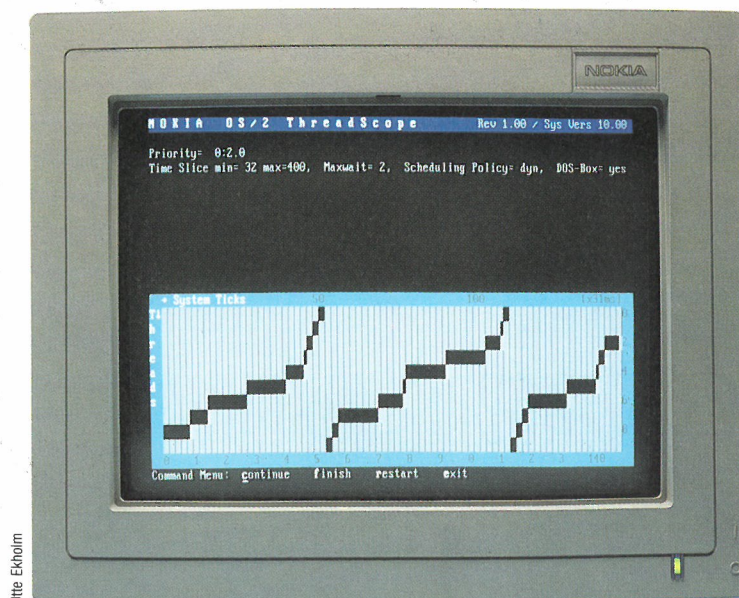
In the fourth line of photo 1, the Timeslice default values are reported as 32 and 248 for the minimum and maximum value respectively. The minimum value represents the continuous slice any member of the low cluster will get, with the maximum value defining the slice length of the high threads. From photo 1, you can take 8 ticks for the slice of each thread in the high cluster which means the

system assigns as many ticks as needed to exceed the defined minimum and maximum values. E.g. the slice of 8 ticks, with a resulting duration of 8 times 31.25, or 250 ms, satisfies the maximum setting of 248. But 1 tick for the low slice would not be enough for the minimum of 32 requested, so 2 ticks are used. However, note that all threads of the low cluster get 2 ticks, with no account taken of their different priority levels. To cross-check the way Timeslice affects the system, photo 3 was created with Timeslice set to 64 for min and 96 for max. Now the low slice is prolonged to 3 ticks, whereas the high slice is cut to 4 ticks, since 4 times 31.25 exceeds 96.

The slice size of high cluster threads does not always equal the specified max value of Timeslice. In photo 4, the system is under stress from the large number of high cluster threads, the only type of threads running. The Timeslice maximum value was set to 400, which means at least 13 ticks ($13 \times 31.25 = 406.25\text{ms}$) would be needed. As you see, the size of the slices varies widely. A regular, full 13-tick slice for any of the 10 threads would result in a turn-around of $13 \times 10 \times 31.25\text{ ms}$, a value exceeding the specified Maxwait (set to 2 for the purpose of this experiment). To avoid that, the system cuts the slices of some threads, and in this way any thread is re-executed within the Maxwait interval. That means the Maxwait value has a higher importance than Timeslice and is applied not only to boost the low cluster threads. If the slices are different in length, especial care must be taken that the long term CPU share is nevertheless equal. As you can see from photo 4, OS/2 is alternating between the different slices of all threads, thereby achieving an equal share in the long run.

Beyond the Limits

So far, the observed experiments produced proper results. With a tool like this ThreadScope program, you can demonstrate the system's behavior in stressing or unusual situations. Some such experimental settings show up special effects. For example, in photo 5 a Timeslice value of 63 for min and max was set up and 3 threads in the low and 3 in the high cluster are run-



ning. In contrast to the expected equal share of the threads within both clusters a disparity can be observed. Each cluster contains one thread which gets only 2 ticks for its slice, instead of the three the others get. That yields to an unbalanced CPU share ratio of 3:3:2, which is some way off the specification. Of course, in this case the remedy is simple. You have only to avoid such a critical Timeslice value as 63 and use 64 instead to produce proper 3-tick slices.

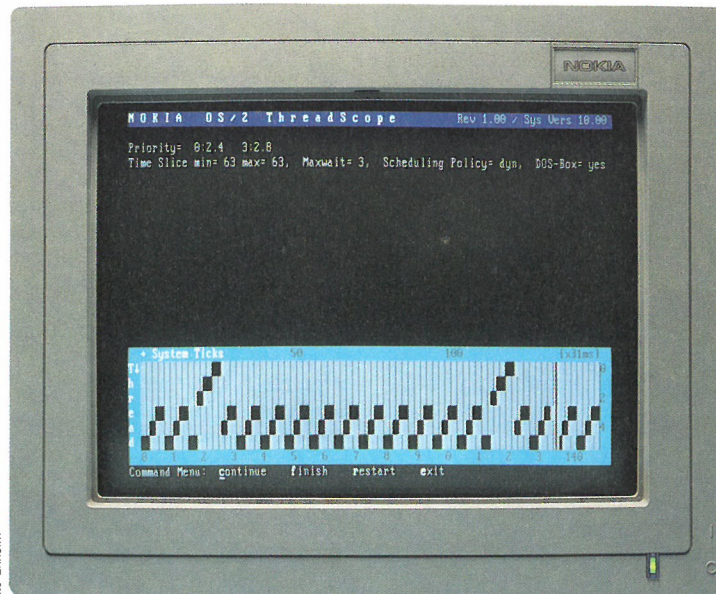
Another share disparity can arise from long slices. For this experiment, the Timeslice max value was set to 800. The execution sequence of the high cluster of 16 threads stabilizes after some time, as shown in photo 6. But as you can readily see, 3 of the 8 threads are preferred by the system. They get a full long slice each time, whereas the slices of the others are cut to 2 ticks. The share is dramatically different now. In this case, the setting is probably beyond the working limits of the system, and some specification of the allowed range for Timeslice – like 32 up to 500 – would avoid the problem.

Outlook

With the help of such a tool, as presented above, multitasking becomes a visible reality; and, by playing with it, you would come to understand its behavior better than any document could explain. Of course, the scheduling situation of an experiment is created artificially; but nevertheless, it gives some impression of how real OS/2 applications would work.

For traditional PC programmers, multitasking is a completely new dimension, and it is highly recommended that they familiarize themselves with all its aspects before starting an OS/2 application development project. Multitasking, originally evolved as a mechanism for optimizing the system throughput, is also – and perhaps more important for a PC – an extremely powerful approach to simplifying the structure of a program. With multitasking, the programmer can organize its code by implementing a separate thread for each independent activity. But at that point he or she has to assign priorities to them, and to do this properly their impacts should be known. ■

Uffe Ekholm

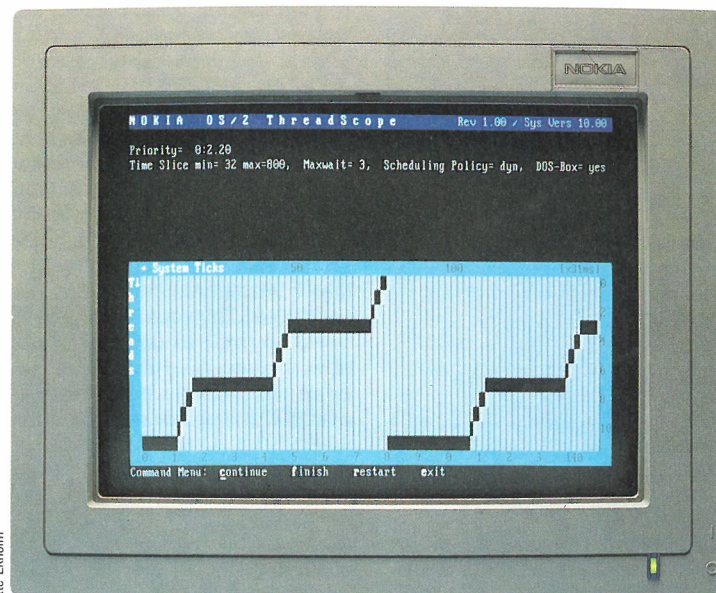


OS/2-moniajoympäristö

OS/2-moniajoympäristössä (multitasking) liikutaan Program Selectorin avulla näytöstä toiseen. Samanaikaisia istuntoja (session) voi olla käytössä 12. Istunnolla tarkoitetaan omaa näyttöä. Lisäksi on omat istunnot DOS-ohjelmille, Program Selectorille (istunnon valitsin), tausta-ajolle ja virheenkäsitelylle. Jokaisessa istunnossa voi ajaa yhtä tai useampaa sovellusta (application) ja jokainen sovellus sisältää yhden tai useamman tehtävän (thread, säie). Tehtävä on yksikkö, joka jakaa CPU-ajan.

Moniajoympäristössä on prioriteetteja (priority), koska kaikki sovellukset eivät ole samanarvoisia. OS/2:ssa prioriteetit jaetaan kolmeen pääryhmään ja niistä jokainen 32 alaryhmään. Ensimmäisen ryhmän sovellukset ovat esimerkiksi aikakriittisiä. Toisen ryhmän sovellukset ovat tavallisia sovelluksia (regular). Alimmassa ryhmässä on nk. joutoajan sovellukset (idle time).

Uffe Ekholm



References

1. Ed Iacobucci, "OS/2 Programmer's Guide", Osborne/McGraw-Hill 1988
2. Gordon Letwin, "Inside OS/2", Microsoft Press 1988
3. Steven Armbrust and Ted Forgeron, "Multiple Tasks", PC Tech Journal, Nov 87, pp. 90
4. Ray Duncan, "OS/2 Multitasking: Exploiting the Protected Mode of the 80286", Microsoft System Journal, May 87, pp. 27
5. Eva White and Richard Grehan, "Microsoft's New DOS", Byte, June 87, pp. 116

OS/2:n ohjelmaliitos ja

Alkavassa neliosaisessa artikkelisarjassa koulutus-suunnittelija Markku Mäkinen selvittää OS/2:n ytimen toimintaa suojatussa tilassa. Ensimmäisessä osassa kerrotaan ohjelmaliitoksesta ja muistinhallinnasta.

OS/2:n arkkitehtuuri voidaan karkeimmillaan jakaa ytimeen ja alijärjestelmiin. Ydin on järjestelmän osa, joka viime kädessä hallitsee ja omistaa kaikki järjestelmän resurssit. Järjestelmän ytimen pääasiallisimmat tehtävät ovat moniajon, muistinhallinnan ja I/O:n ylläpito sekä kontrollointi. Ytimen palveluja hyödynnetään ohjelmaliitoksen välityksellä. OS/2:n ohjelmaliitos poikkeaa selvästi MS-DOS:n ohjelmaliitoksesta.

I OHJELMALIITOS

OS/2 versiossa 1.0 käyttöjärjestelmän ydin tarjoaa yli 200 erilaista palvelua sovellusohjelmille. Sovellusohjelmien on pyydyttävä käyttöjärjestelmältä palveluja ohjelmaliitoksen eli API:n (Application Program Interface) välityksellä. Ohjelma saa käyttöönsä resursseja kuten muistia, levyjä ja oheislaitteita vain, jos käyttöjärjestelmä sallii. Ohjelmaliitosta ei voi ohittaa. Ero MS-DOS:n ohjelmaliitokseen on selvä, sillä MS-DOS tarjoaa ohjelmille valmiita rutiineja, joita voidaan haluttaessa hyödyntää. Ilman MS-DOS:n ohjelmaliitosta tulee myös hyvin toimeen ja käytännössä useat ohjelmat tehokkuussyistä ohittavatkin MS-DOS:n.

Koska monen prosessin järjestelmässä ei voida luottaa pelkäs-

tään prosessien väliseen yhteistyöhön, käyttöjärjestelmä kontrolloi resursseja tiukasti. Varsinaisessa OS/2:n moniajo-ympäristössä (suojatussa tilassa) käyttöjärjestelmä hallitsee resursseja lähes täydellisesti. Käyttöjärjestelmä kykenee virtualisoimaan laitteiston tarjoamalla ohjelmille palveluja pelkästään käyttöliittymän välityksellä. Ohjelma voi itse asiassa olettaa laitteistosta ainoastaan, että suoritin tuntee 80286-suorittimen käskykannan.

OS/2:n ohjelmaliitoksessa on hyödynnettävä dynaamista linkitystä (käsitellään tarkemmin artikkelisarjan toisessa osassa). Dynaamisen linkityksen keskeinen piirre on, että ohjelma voi kutsua aliohjelmaa, jota ei ole liitetty ohjelmaliitokseen (exe-tiedostoon). Ohjelmalle käyttöjärjestelmäpalvelut näkyvät kutsuttavina kirjastoituina aliohjelmina.

Palvelujen kutsuminen FAR CALL -käskyllä

Kun käytetään symbolista konekieltä, käyttöjärjestelmäpalveluja kutsutaan FAR CALL -käskyllä. Tässäkin kutsumenettelystä OS/2:n palvelut nähdään ulkoisina aliohjelmina.

Parametrien välitys perustuu seuraaville periaatteille: parametrit välitetään pinon kautta, viedään pinon ilmentymäjärjestyksessä vasemmalta oikealle (pascal-järjestys), kutsuttava rutiini poistaa parametrit pinosta ennen kontrollin palauttamista kutsuvalle ohjelmalle, parametrien määrä on aina kiinteä, virhekoodi palautetaan AX-rekisteriin.

DOS:n ja OS/2:n palvelukutsujen erot ovat nähtävissä malliohjelmista 1, 2 ja 3. Kaikissa ohjelmissa tehdään tiedoston luku-pyyntö. Taulukossa 1 esitetään palvelupyyntö MS-DOS:ssa. Esimerkistä selviää, että MS-DOS-kutsu vaatii assembly-kielen käyttöä. Välitettävät parametrit viedään rekistereihin ja palvelupyyntö käynnistetään ohjelmakeskeytyksellä. Tämä ei onnistu korkean tason kielistä ilman välittäviä kirjastomoduuleja.

Standardikirjastot parantavat siirrettävyyttä

Ohjelmaa koodattaessa kannattaa käyttää mahdollisimman paljon valmiita standardikirjastoja. Standardikirjastojen käyttö parantaa ohjelman siirrettävyyttä OS/2:sta toiseen käyttöjärjestelmään ja helpottaa päivitystä.

Kun hyödynnetään OS/2:n moniajo-ominaisuuksia, on kuitenkin syytä varmistaa, että käytettävä ajonaikainen kirjastorutiini on vapaakäyttöinen. Vapaakäyttöisyydellä tarkoitetaan mahdollisuutta käyttää tiettyä kirjastorutiinia samanaikaisesti monessa tehtävässä ts. samanaikaisesti etenevien ohjelman osien kesken.

Ei vapaakäyttöiset kirjastorutiinit saadaan vapaakäyttöisiksi erillisellä käännöksen ohjauksella.

Family API-palveluilla DOS- ja OS/2-ohjelmia

Kun rajoitutaan tiettyihin järjestelmäkutsuihin, OS/2-sovelluksia voidaan ajaa myös DOS-tilassa tai puhtaassa MS-DOS-ympäristössä. Kaikkien OS/2:n tarjoamien palvelujen joukosta voidaan erottaa alijoukkona FAMILY API -palvelut (FAPI). Kun käytetään vain FAPI-palveluja, voidaan tehdä sekä OS/2- että MS-DOS-ympäristössä toimivia ohjelmia.

FAPI-sovellus tunnistaa käyttöjärjestelmän ja suorittaa järjestelmäkutsut ympäristön mukaisesti. FAPI-sovelluksen syntymiseen ei kuitenkaan riitä pelkästään pitäytyminen FAPI-kutsuissa. Käännöksessä syntyy edelleen OS/2-sovellus, joka kieltäytyy toimimasta DOS-ympäristössä. Sovellus saadaan DOS-yhteensopivaksi BIND-ohjelman avulla. BIND ottaa syöteenä käännöksessä tuotetun exe-tiedoston ja muokkaa sen FAPI-sovellukseksi.

```
; Datasegmentissä:
; ...
fhandle DW ? ; Tiedoston kahva
nbytes DW ? ; Luettavien tavujen lukumäärä
buffer LABEL BYTE ; Puskuri luetulle datalle
; ...
; Koodisegmentissä:
; ...
mov bx,fhandle ; Ladataan tiedoston kahva
mov cx,nbytes ; Ladataan luettavien tavujen lukumäärä
lea dx,buffer ; Ladataan puskurin poikkeama (offset)
mov ah,3fh ; Ladataan Read File -funktionumero
int 21h ; Kutsutaan Read File -funktiota
jc READ_ERROR ; Käsitellään virheet, muuten jatketaan
; ...
```

Taulukossa 1 on esitetty palvelupyyntö MS-DOS:ssa. Esimerkistä selviää, että MS-DOS-kutsu vaatii assembly-kielen käyttöä. Välitettävät parametrit viedään rekistereihin ja palvelupyyntö käynnistetään ohjelmakeskeytyksellä.

muistinhallinta

II MUISTINHALLINTA

Intelin 80286-suoritin voi toimia joko suojatussa tai real-tilassa. Toimiessaan jälkimmäisessä, suoritin emuloi 8086-suoritinta. Tämä on MS-DOS:n toimintatila.

OS/2 käyttää 80286-suoritinta suojatussa tilassa eli suorittimen varsinaisessa toimintatilassa. Suojatussa tilassa suoritin kykenee käyttämään 16 megatavun suoraan osoitettavaa muistia ja yhden gigatavun (1000 MT) virtuaalimuistia. OS/2 hyödyntää nämä suorittimen ominaisuudet täydellisesti. Toimiakseen moitteettomasti, OS/2 vaatii versiossa 1.0 vähintään kaksi megatavua ja versiossa 2.0 vähintään kolme megatavua työmuistia.

DOS-tila välillä 0-1024 kilotavua

OS/2-käyttöjärjestelmä sisältää tavallisesti ns. DOS-yhteensopivan tilan. DOS-tila voidaan konfiguroida pois. DOS-tila sijoittuu muistiavaruudessa MS-DOS:a vastaavalle kohdalle, toisin sanoen välille 0-1024 kilotavua. Muistiosoitteet 640 kilotavun ja 1024 kilotavun välissä on MS-DOS-järjestelmän tavoin varattu oheislaitteille ja ROM-BIOS:lle. Alin muistialue 0-640 kilotavua voidaan konfiguroida kokonaan DOS-tilalle tai mahdollisesti varata sen yläpäästä (<640 kT) osa suojatun tilan käyttöön.

MS-DOS:sta OS/2:en siirtymistä helpottava OS/2:n DOS-tila emuloi MS-DOS-käyttöjärjestelmää. Kun ohjelmaa suoritetaan DOS-tilassa suoritin vieään real-tilaan, joka kutistaa muistiavaruuden yhteen megatavuun. Real-tilassa suoritin ei tarjoa järjestelmälle suojausmekanismeja. Tällöin MS-DOS-ohjelman kaatuminen saattaa lukita näppäimistön ja estää DOS-tilasta poistumisen. Yhteensopivuustila on koko OS/2-järjestelmän akilleen kantapää, sillä huonosti käyttäytyvä DOS-ohjelma kykenee halvaannuttamaan koko järjestelmän.

Suojattu tila megatavusta alkaen

Yhden megatavun yläpuolinen muistiavaruus on kokonaan va-

rattu suojatun tilan käyttöön. Käyttöjärjestelmän ydin ja I/O-segmentit ovat kyseisen alueen alapäässä ja näiden yläpuolella oleva muistiavaruus on aina 16 megatavuun asti sovellusohjelmien käytössä. Suojatussa tilassa toimivat ohjelmat on eristetty toisistaan laitetasolla. Niin ohjelmat eivät häiritse toisiaan.

Työmuistia käsitellään segmentteinä

Suoritin käsittelee työmuistia segmentteinä, joiden koko vaihtelee 1 kilotavusta 64 kilotavuun. Real-tilassa suoritin saadaan viittaamaan suoraan fyysiseen muistiosoitteeseen, kun määrätään segmentin alkuosoite ja ilmoitetaan poikkeama (offset) kyseisestä alkuosoitteesta. Suorittimen suojatussa tilassa segmenttirekisterissä sijaitseva arvo ei kerro segmentin fyysistä alkuosoitetta. Sen sijaan arvo on indeksi todellisen osoitteen sisältävään segmenttien kuvaaja-työmuistia tauluun (descriptor table).

Ohjelmalle segmenttirekisterin sisältö on lähinnä segmentin nimi, jota yleisemmin kutsutaan valitsimeksi. Erityisen tärkeää on huomata, ettei tavallinen ohjelma kykene lukemaan segmenttien sijoittelutaulua. Taulun lukemiseen kykenee ainoastaan käyttöjärjestelmä. Assemblerilla ohjelmointiaessa on lisäksi pidettävä mielessä, ettei suoritin hyväksy dataa ja koodia samassa segmentissä. Datasegmentin sisältöä ei voi ajaa ohjelmakoodina. Vastaavasti suoritin estää koodisegmenttiin kirjoittamisen.

Käyttöjärjestelmä voi tarvittaessa mielivaltaisesti muuttaa kuvaajataulun sisältöä, käytännössä siis siirrellä segmenttejä. Ohjelmalla ei ole mitään tietoa sen omien ja muiden ohjelmien segmenttien sijainnista työmuistissa.

Suoritin huolehtii siitä, etteivät segmentit vahingossa joudu päällekkäin. Jos suojatussa tilassa ajettava ohjelma yrittää lukea tai kirjoittaa sellaiseen muisti- paikkaan, johon sillä ei ole oikeuksia, suoritin antaa järjestelmälle virheilmoituksen. Virheilmoituksesta käyttöjärjestelmä keskeyttää ohjelman suorituksen. Suojatussa tilassa huonosti käyttäytyvä ohjelma ei näin ollen voi häiritä muita ohjelmia.

```
; Datasegmentissä:
; ...
fhandle    DW    ?           ; Tiedoston kahva
bufflen    DW    ?           ; Puskurin koko
nbytes     DW    ?           ; Luettavien tavujen lukumäärä
buffer     LABEL BYTE        ; Puskuri luettulle datalle
; ...
```

; Koodisegmentissä:

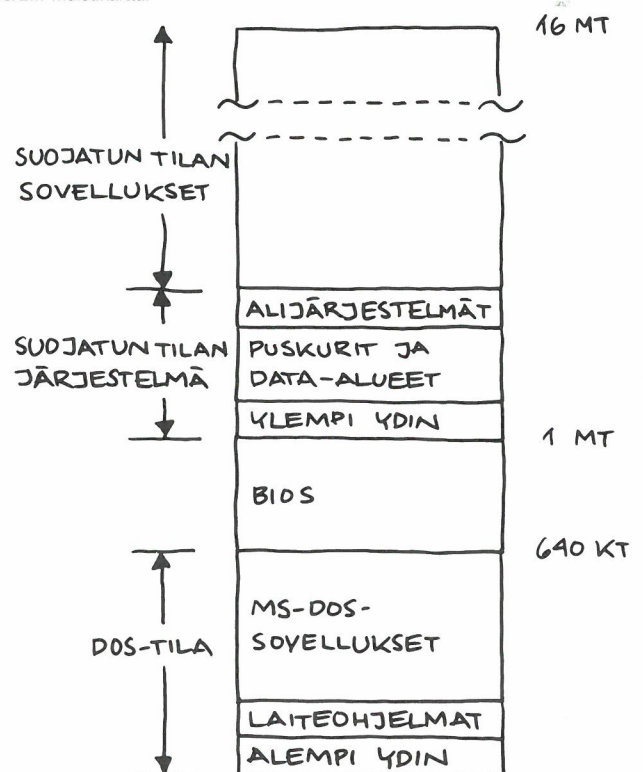
```
; ...
EXTRN      DosRead:FAR
push       fhandle          ; Vieään tiedoston kahva pinoon
push       ds               ; Vieään puskurin osoite pinoon
lea        ax,buffer
push       ax
push       bufflen          ; Puskurin koko pinoon
push       ds               ; Palautettavien lukumäärän
lea        ax,nbytes        ; osoite pinoon
push       ax
call       DosRead          ; Kutsutaan DosRead-funktiota
or         ax,ax             ; Onnistuiko?
jnz        read_error       ; Käsitellään virheet, muuten jatketaan
```

Taulukon 2 assembly-kielisessä esimerkissä pyydetään palvelua OS/2:lta. Parametrit vieään pinoon ja palvelupyyntö käynnistetään kutsumalla rutiinia nimeltä.

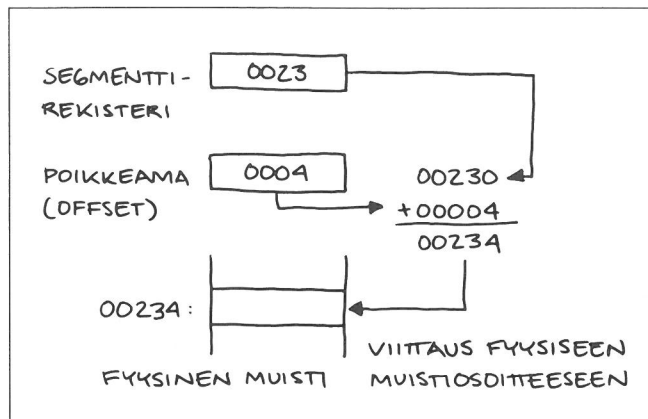
```
#include <os2.h>
...
BYTE       buffer [256];    /* puskuuri */
HFILE      handle;          /* tiedoston kahva */
USHORT     nbytes;          /* luettavien tavujen lukumäärä */
...
main ()
...
    ret = DosRead (handle, buffer, sizeof(buffer), &nbytes;
    if (ret)
        ReadError;
    else ...
```

OS/2:n palvelujen kutsuminen korkean tason kielistä on selkeää. Kutsuissa ei tarvita välittäviä kirjastorutiineja. Käyttöjärjestelmän tarjoamaa palvelua kutsutaan kuten mitä tahansa ulkoista aliohjelmia taulukon 3 mukaisesti.

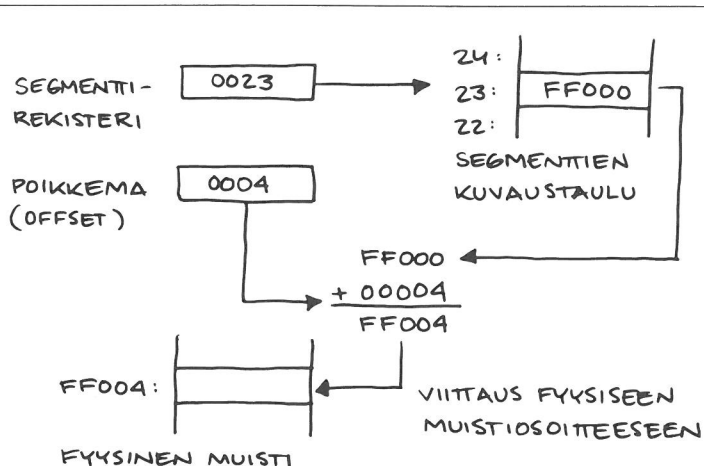
OS/2:n muistikartta.



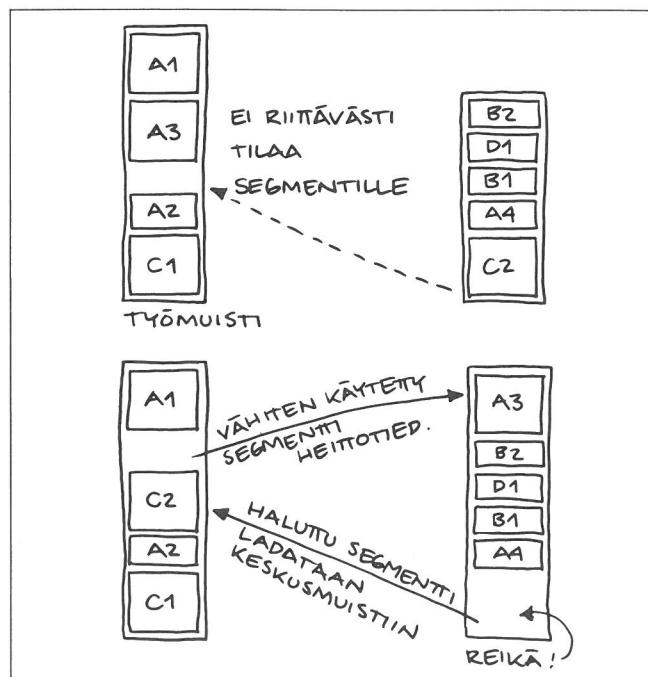
Muistin osoitus suorittimen real-tilassa.



Muistin osoitus suorittimen suojatussa tilassa.



Esimerkki heittovaihdosta. Segmentti A3 vietään heittotiedostoon tilan saamiseksi ladattavalle C2-segmentille.



Virtuaalinen muisti ja heittovaihto

Muistiavaruus on ohjelman kannalta täysin virtualisoitu, tehty näennäiseksi. Käyttöjärjestelmä piilottaa fyysisen muistiavaruuden ohjelmilta ja antaa niiden ymmärtää muistiavaruutta olevan 1 000 megatavua. Virtualisointi on toimiva silmankääntötempu, koska laajastakaan ohjelmasta ei fyysisessä työmuistissa tarvita samanaikaisesti kuin korkeintaan neljää segmenttiä. Käytännössä gigatavun virtuaalivähyys merkitsee nykyisillä kiintolevyillä ja niiden osoitusrajoituksilla sitä, että kiintolevylle mahtuva ohjelma voidaan aina käynnistää.

Käyttöjärjestelmä huolehtii ohjelman tarvitsemien segmenttien lataamisesta työmuistiin. Jos työmuistissa ei ole riittävän laajaa yhtenäistä muistialuetta segmenttiä varten, käyttöjärjestelmä aloittaa tilan järjestelyn. Segmenttejä siirtämällä käyttöjärjestelmä pyrkii luomaan ladattavalle segmentille riittävän ison yhtenäisen muistialueen. Työmuistin järjestelyllä ei ole haitallista vaikutusta ohjelmien toimintaan. Suojatussa tilassa ohjelma ei milloinkaan voi viitata muistiosoitteeseen segmentin osoitteella.

Segmenttien siirtely ei kuitenkaan aina takaa riittävän muistitilan syntymistä. Tällöin käyttöjärjestelmä aloittaa heittovaihdon (swap). Keskeytyneinä olevien prosessien segmenttejä siirretään kiintolevylle ja näin saadaan ajettavalle prosessille tarvittava tila työmuistissa. Levykäsitteilyn minimoimiseksi käyttöjärjestelmä pyrkii pitämään eniten käytetyt segmentit työmuistissa.

Järjestelmä siirtää kiintolevylle sijaitsevaan vaihtotiedostoon (swap file) ainoastaan datasegmenttejä. Koodisegmenttien siirto vaihtotiedostoon on tarpeetonta, koska ne pysyvät työmuistissa muuttumattomina. Työmuistista poistettavat koodisegmentit yksinkertaisesti tuhoataan. Tarvittaessa koodisegmentti löytyy kiintolevyltä olevasta ohjelmatiedostosta (.exe). Poikkeuksena ovat levykkeeltä ladattavat ohjelmat. Ne ladataan kokonaisuudessaan työmuistiin ja tarvittaessa koodisegmenttejä siirretään järjestelmän vaihtotiedostoon SWAPPER.DAT. Tämä johtuu siitä, että levyke saataan vaihtaa tai poistaa kesken prosessia.

OS/2 versiossa 1.0 vaihtotiedoston koko voi ainoastaan kasvaa, eikä milloinkaan pienentyä. Huonosti suunnitellusta datan käsittelystä seuraa vaihtotiedoston reikiintyminen ja kasvu. Tiheä heittovaihto erikokoisilla da-

tasegmenteillä johtaa viimein levytilan loppumiseen. Vaihtotiedoston alustaminen onnistuu vain järjestelmän uudelleen käynnistyksellä. Reikiintyminen voidaan välttää suunnittelemalla ohjelmien datasegmentit yhtä suuriksi, esimerkiksi neljän kilotavun kokoisiksi.

Heittovaihto hidastaa aina järjestelmän toimintaa. Vaihdon todennäköisyyteen voidaan onneksi vaikuttaa. Yksittäisen koodisegmentin tulisi sisältää vain toiminnallisesti yhteenkuuluvia rutiineja. Heittovaihdon mahdollisuus kasvaa aliohjelman kutsussa toistuvasti eri segmenteissä olevia rutiineja. Globaali koodi ja data on järkevä sijoittaa omiin segmentteihin. Liian pienet segmentit kasvattavat järjestelmän kuormitusta ja liian suuret heikentävät heittovaihdon tehokkuutta.

Kontrollirakenteiden suunnittelussa kannattaa ottaa huomioon suurten tietorakenteiden vaikutus. Jos alla olevan ohjelman taulukot ovat suuria (satoja kilotavuja), ne eivät välttämättä mahdu samanaikaisesti työmuistiin. Joskus käyttöjärjestelmä saattaa joutua suorittamaan heittovaihdon yli tuhat kertaa.

```
for (i = 0; 999; i++)
    taulukko1 = <lauseke1>;
    taulukko2 = <lauseke2>;
```

Pienellä kontrollirakenteen muutoksella saattaa tilanne parantua. Edelliseen esimerkkiin verrattuna seuraavalla rakenteella saattaa heittovaihtojen määrä vähetä jopa tuhannella.

```
for (i = 0; 999; i++)
    taulukko1 = <lauseke1>;
for (i = 0; 999; i++)
    taulukko1 = <lauseke2>;
```

Haku kiintolevyltä kestää noin 10 000-kertaisen ajan verrattuna hakuun työmuistista. Heittovaihto kestää noin 30 millisekuntia työmuistihakua kauemmin. Esimerkkitapauksessa silmukoiden nopeusero on suurimmillaan noin viisi minuuttia.



Seuraavat osat käsittelevät moniajoa, dynaamista linkitystä, prosessien välistä kommunikointia ja tiedostojen käsittelyä.

OS/2-käyttöjärjestelmä on esitelty Softarissa 2/1988.

Nokian lähiverkolla pitkäjänteisyyttä oppilaitoksen atk-ratkaisuihin



Kaupalliset ja tekniset oppilaitokset, ammattikoulut, kurssikeskukset sekä monet erikoisoppilaitokset asettavat tietotekniikalle varsin erilaisia vaatimuksia. Yhteisenä pyrkimyksenä on pienentää opiskelun ja käytännön välistä kuilua simuloimalla työelämän todellisia tilanteita. Suomalaisissa yrityksissä noin joka kolmas työasema on nokialainen – ohjelmista puhumattakaan. Työelämässä koulutuksesta tutut välineet antavat tuulta siipien alle alusta asti.

Jatkuvasti taloudellinen ratkaisu myös monitoimittajaympäristöissä

Tällä hetkellä oppilaitoksissa joko ollaan sitouduttu yhteen laitetoimittajaan tai kahlataan yksittäisten laitteiden kirjavuudessa. Kumpikin vaihtoehto voi merkitä uhkaa jatkuvuudelle, taloudellisuudelle ja tekniselle suorituskvyylylle.

Nokian lähiverkko on valmis, käytännössä testattu ratkaisu näihin ongelmiin. Siihen voidaan liittää kaikki standardeihin perustuvat laitteet merkistä riippumatta. Keskuslaitteistosta riittää käsittelyvoimaa työasemille, joten esimerkiksi osa mikroista voidaan korvata yksinkertaisilla päätteillä. Mikroihin voidaan tallentaa harjoitustyöt, opetusohjelmat taas järeämpään keskuslaitteistoon. Yhteydet oppilaitoksen ulkopuolisiin tietopankkeihin ym. ovat käytettävissä kaikilta työasemilta.

Joka laitteesta saadaan maksimaalinen hyöty koko käyttöiän. Laajentaminen käy joustavasti, ja valinnanvapaus säilyy myös tulevaisuuden hankinnoissa.

Nokia Datan uutta vahvuutta ovat Unix-ympäristöt

Unix-käyttöjärjestelmä tuo Nokian lähiverkolle uudet ulottuvuudet. MikroMikot, Unix-pohjaiset SUN-tietokoneet sekä tekniset työasemat ovat korkealaatuisia, tehokkaita ja alun perin myös verkkokäyttöön suunniteltuja.

Halpoja laitteita vai toimiva ratkaisu?

Nokia Datalla on ainutlaatuista monitoimittajaosaamista – ammattitaitoa keskustella ja ratkaista kokonaiskysymyksiä. Me emme jätä asiakkaitamme yksin ongelmineen. Kokonaisuuden toimivuus merkitsee enemmän kuin laitteiden hankintahinta. Yhtä tärkeätä on jatkuva tuki ja palvelu, jotka Nokia Data varmistaa. Pyydä esitteitä, kysy lisää.

NOKIA DATA

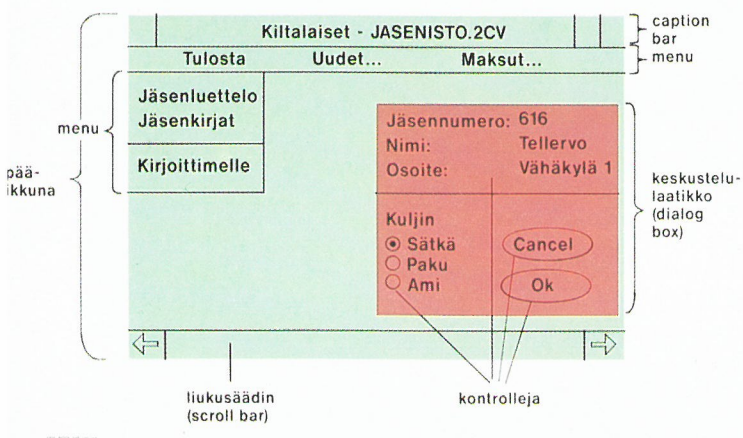
Opetusjärjestelmät

PL 780, 00101 Helsinki, puh. (90) 5671

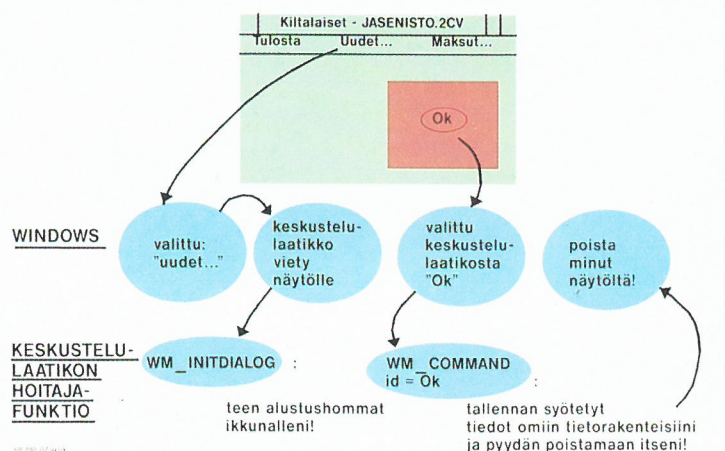
Tampere puh. (931) 509 111, Turku puh. (921) 608 811

Valmistaudu uudistuv

KAAVIO 1: ESIMERKKI WINDOWS-SOVELLUKSEN IKKUNOISTA



KAAVIO 2: KARKEA ESIMERKKI JOISTAKIN KESKUSTELU-LAAITIKKON LIITTYVISTÄ SANOMISTA



Windows-ohjelmointi poikkeaa perinteisestä lohkorakenteisella kielellä ohjelmoinnista. Nyt viimeistään on aika tutustua Windows-ohjelmoinnin periaatteisiin ja näin valmistautua uudistuvaan mikro-maailmaan!

Windowsin ennustetaan olevan tulevaisuudessa de facto -standardi. Käyttäjälle suurimmat edut ovat useiden samanaikaisten sovellusten esilläpito (ikkunointi), sovellusten käytön helppous ja samanlaisuus sekä tiedonsiirtomahdollisuus sovellusten välillä (clipboard).

Ohjelmoijan työssä merkittävää on sovellusten siirrettävyys Windows-ympäristöstä toiseen. Windows-sovelluksen tekijän ei myöskään tarvitse huolehtia oheislaitteiden ohjaamisesta, sillä sen hoitaa Windows. Lisäksi grafiikan piirtämistä varten Windowsissa on sekä funktioita että valmiita välineitä kuten kyniä, siveltimiä ja fontteja.

Windows-ohjelmoinnin vaatimukset

Siirtyminen Windows-ohjelmointiin vaatii melkoisesti opettelua. Jos hallitsee jo C-kielen, joka on ainoa järkevä kieli tässä yhteydessä, ja tuntee Windowsin käytön periaatteet, aikaa harjoitteluun kuluu 2 - 4 kuukautta.

Ohjelmoija tarvitsee Windows Software Development Kitin, Windows Desktopin sekä MS-C-kääntäjän. Näiden on syytä olla versionumeroitaan yhteensopivia (toukokuun alussa: 1.04, 1.04 ja 4.0).

Aluksi Windows-sovelluksen rakentaminen vaikuttaa hankalalta, mutta Windows 1.04 ja 2.0 -ympäristöissä hankittuja taitoja voi sitten suoraan hyödyntää tulevissa OS/2:n Windows Presentation Manager -ympäristössä.

Windows-sovelluksen rakentaminen aloitetaan määrittelemällä ensin erikseen kaikki ikkunat sekä valikot ja vasta sitten toteutetaan ohjelman varsinainen logiikka. Näin sovelluksista saadaan rakenteellisesti erittäin hyviä.

Ikkunat syöttövälineinä

Sovellukset syötetään niiden ikkunoiden kautta. Kaaviossa 1 on esimerkki Windows-sovelluksen ikkunoista. Siinä näkyvät pääikkuna, johon kuuluvat otsake (caption bar), päävalikko (menu) ja liukusäädin (scroll bar) ja

näiden väliin jäävä alue (client area) niiltä osin kuin ei jokin muu ikkuna varaa siitä tilaa.

Kun päävalikosta valitaan "Tulosta", Windows näyttää lisävalikon. Jos valitaan "Uudet...", Windows näyttää keskustelulaatikon (dialog box), jonka kontrolli-ikkunoiden avulla sovellus lukee uuden jäsenen tiedot.

Näillä kahdella ikkunalla on omat ikkunanhoitajafunktiot (aliohjelmasa), pääikkunalla nimeltään "KiltaWndProc" ja keskustelulaatikon nimeltään "UudetDlgBox".

Kohdistimen siirtäminen esimerkiksi hiirtä näpyttämällä jonkin ikkunan kohdalle aiheuttaa sanoman lähettämisen kyseisen ikkunan hoitajafunktiolle ja se ikkuna saa lukuvuoron (input focus). Esimerkiksi, jos "painetaan" Ok-painonappia hiiren tai näppäimistön avulla, Windows lähettää sanoman keskustelulaatikon hoitajafunktiolle: "Ok-valinta tehty". Tyypillisesti tällöin ikkunanhoitaja tallentaa annetut uuden jäsenen tiedot ja lähettää Windowsille sanoman pyytään poistamaan itsensä näytöltä.

Kun jo käynnissä oleva Windows-sovellus käynnistetään toiseksi ilmentymäksi (instance), ilmentymien ikkunat ovat samannäköisiä, mutta täysin erillisiä olioita. Ilmentymät jakavat yleensä saman ohjelmakoodin, mutta niillä on omat data-alueensa.

Syöttö esitetään sanomina

Kaikki syöttö esitetään Windowsissa sanomina sille ikkunalle, jolle syöttö on kohdistettu. Kaaviossa 2 on yksinkertaistettuna esitetty, millaista sanomanvälitystä käydään Windowsin ja keskustelulaatikon ikkunanhoitajafunktion välillä.

Kaaviossa 3 tarkastellaan esimerkkisovelluksen tärkeimpien sanomien kulkua. Siinä on nähtävissä myös Windows-sovelluksen pakolliset funktiot, pääohjelma WinMain ja pääikkunanhoitaja KiltaWndProc. Lisäksi mukana on keskustelulaatikon hoitaja, UudetDlgBox.

WinMain-funktion alusta alkaa ohjelman suoritus. Sen ra-

Vaan mikromaailmaan

kenne on aina samantapainen kuin kaaviosta 3 on nähtävissä. Se rekisteröi ensin ikkunaluokan, joka siis sisältää sekä tiedon pääikkunan ulkonäöstä että tiedon sitä hoitavasta ikkunafunktiosta (tässä esimerkissä KiltWndProc). CreateWindow aiheuttaa vastaavannäköisen ikkunan luomisen keskusmuistiin. Yleensä tämän jälkeen pääohjelmassa on vielä joitakin alustuksia ja ikkunan näyttö päätteelle.

Pääohjelma jää lopuksi aina silmukkaan, jossa se odottaa ja vastaanottaa sanomia, joita Windows on ko. sovelluksen sanomajonoon asettanut. Nämä sanomat se tarkistaa ja ehkä hie- man muokkaa niitä sekä lähettää ne takaisin Windowsille.

Tällaisen WinMainin takaisin- lähettämän sanoman sitten vastaanottaa sovelluksen sanomajonosta se ikkunanhoitajafunktio, jolle sanoma on tarkoitettu. Esimerkiksi pääikkunan luomisesta (CreateWindow) johtuen Windows lähettää pääikkunan hoitajalle sanoman WM-CREATE, jolloin KiltWndProc tietää tehdä alustuksia.

Kun käyttäjä valitsee päävalikosta valinnan "Uudet...", Windows lähettää taas WinMainin kautta KiltWndProc:lle sanoman WM-COMMAND, jonka lisätietona selviää, mikä valinta tehtiin. Saatuaan tämän komen- non, KiltWndProc käskää Windowsia viemään näytölle etukäteen määritellyn keskustelulaatikon.

Näytettyään keskustelulaatikon näytölle, Windows ilmoittaa siitä lähettämällä suoraan keskustelulaatikon hoitajafunktiolle (UudetDlgBox) sanoman INIT-DIALOG. Tällöin UudetDlgBox-funktiosta tiedetään esimerkiksi alustaa keskustelulaatikon kontrolli-ikkunat. Siellä voidaan jättää odottamaan kutakin kontrollia vastaavaa sanomaa. Jos kuvitellaan, että käyttäjä valitsi esim. Ok-painonapin, niin UudetDlgBox:ssa voidaan sanoman saapuessa tallentaa uuden jäsenen tiedot vaikkapa jäsenrekisteriin. Tämän jälkeen UudetDlgBox ilmoittaa Windowsille halunsa poistaa kuvaruudusta. Niinpä Windows lähettää pääikkunan hoitajalle viestin WM-PAINT lisätietonaan sen suora- kaiteen koordinaatit, joka tulisi

piirtää uudelleen keskustelulaatikon poistamiseksi.

Näin sanoman välitys jatkuu, kunnes käyttäjä antaa close-komennon ja pääohjelman GetMessage-funktio palauttaa epä- tosi-arvon.

Luokat ja oliot

Windows-ohjelmoinnin yhteydessä puhutaan usein luokka-käsitteestä ja oliopohjaisesta (objekti-orientoituneesta) ohjelmoinnista. Kaavion 3 esimerkis- sä ikkunaluokan muodostaa siihen liittyvä tietorakenne (ikkunan ulkonäkö-määrittelyt) ja tarvittava ikkunan hoitajafunktio. Oliopohjaista ohjelmointia puolestaan on se, ettei esimerkiksi pääohjelmasta kutsuta ikkunanhoitajafunktioita, vaan ikkunat ovat olioita, jotka aktivoi- tuvat vasta jonkin sanoman ("kohdistin siirretty ikkunaan") vaikutuksesta.

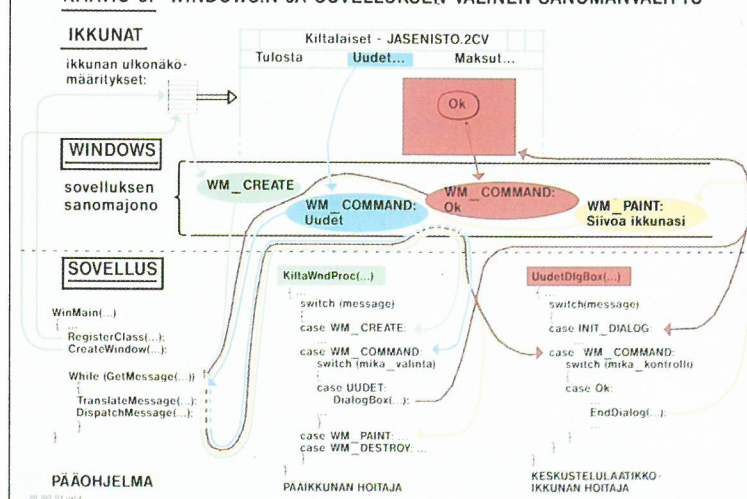
Sovellukseen liittyvät tiedostot

Kaaviossa 4 esimerkissovellus on jaettu tiedostoiksi. Tavallisten lähdekielisten .c- ja .h-tiedostojen lisäksi Windows-sovel- lukseen tarvitaan resurssit (vali- kot, keskustelulaatikat, ikonit jne.) määrittelevä .rc-tiedosto, jonka resurssikäntäjä muokkaa .res-tiedostoksi. Lisäksi tarvitaan mahdolliset .ico-tyyppiset tiedostot, joissa määritellään ikonit, ja koko sovellusta koske- via määrittelyjä varten .def-tyyppinen tiedosto.

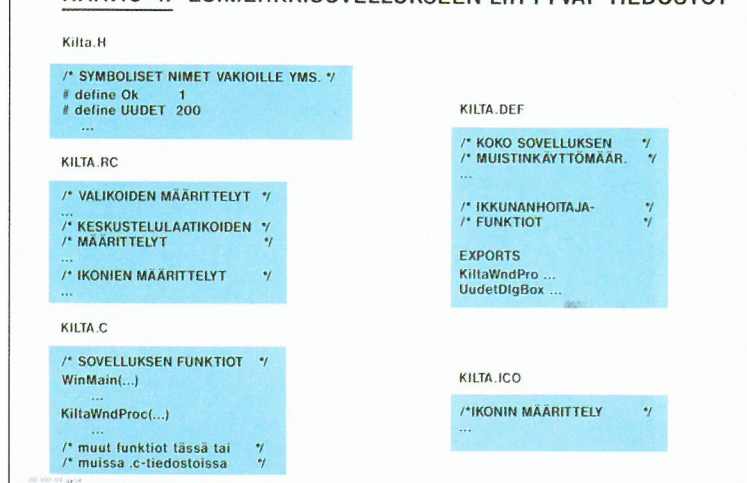
Tiedostojen tuottamista var- ten on olemassa monia apuväli- neitä kuten DIALOG.EXE kes- kustelulaatikoiden muokkaami- seen, FONT.EXE ikonien, font- tien ja kohdistimen muodon piir- tämiseen. Tiedostot kootaan yh- teen Windows-sovellukseksi MS- C-kääntäjän ja SDK:n mukana tulevan link4:n avulla.

Edellä mainitut apuvälineet mahdollistavat sen, että sovel- luksen toteuttaminen aloitetaan ikkunoiden, valikoiden, keskus- telulaatikoiden ja fonttien mää- rittelyllä ja vasta sitten toteute- taan ohjelman varsinainen lo- giikkaosa. Näin saadaan selkei- tä, rakenteisia ohjelmia. ■

KAAVIO 3: WINDOWS:N JA SOVELLUKSEN VÄLINEN SANOMANVÄLITYS



KAAVIO 4: ESIMERKKISOVELLUKSEEN LIITTYVÄT TIEDOSTOT



Tiina Mikonmäki työskentelee Mikonmäki Oy:ssä, p. 915-49 388. Hän toimii mm. vieraile- vana kouluttajana Nitecissä lä- hinnä C-kielen ja Windows-ohjel- mointikurssien vetäjänä.



Designer-grafiikka-ohjelman runsasta piirtoalkiovalikoimaa ja monipuolisia muokausominaisuuksia käyttämällä piirretään mitä mielikuvituksellisempia kuvia. Kuvien tekstit kirjoitetaan sujuvasti omassa editori-ikkunassa tai siirretään muista ohjelmista.

Sano se kuvin

Värejä käyttämällä Designerilla voi tehdä näyttäviä esityksiä. Värisävyjä on kaikkiaan 3,6 miljoonaa ja ne voi käyttää itse määrittä.

Designer-grafiikkaohjelmalla kuvitetaan niin dokumentit kuin yrityslehdet, jotka on tehty esimerkiksi Aldus PageMaker-julkaisujärjestelmällä. Paitsi teknisessä piirtämisessä ja kuvituksessa, Designer on näppärä kaavioiden, piirtoheittinkalvojen, lomakkeiden ja muuta vastaavaa tekstiä sekä kuvia sisältävän aineiston tuottamisessa.

Ohjelma toimii teollisuusstandardin mukaisissa mikroissa, joissa on kiintolevy, työmuistia 640 kilotavua, grafiikkanäyttö ja käyttöjärjestelmänä MS-DOS versio 3.0 tai uudempi. Designer käyttää Windows 2 -käyttöliittymää, mutta sitä voi käyttää myös ilman Microsoft Windows 2 tai Windows/386-ohjelmistoa, sillä toimituspakkaus sisältää suoritusajaisen Windowsin ja laiteohjaimet kaikille Windowsin tukemille kirjoittimille.

Koska käyttöliittymä on kaikilla Windows-sovelluksilla yhtenäinen, valikkojen ja komentojen järjestys on sama kuin muissakin Windows-sovelluksissa. Näin Windowsiin tottuneet käyttäjät tulevat pian sinuiksi myös Designerin kanssa.

Alkioista aloitetaan kuvanteko

Piirtoalkiot valitaan rullaverhovalikosta ja tarvittavat vaihtoehdot annetaan keskustelualaatiokoihin. Rullaverhovalikkojen lisäksi piirtoalkiot voi valita koko ajan näkyvältä työkalualueelta, jossa ne ovat ikoneina.

Kuvat piirretään hiirellä piirtoalkiosta piirtoalueelle. Kuvien paikat voi osoittaa myös näppäilemällä koordinaattien lukuarvot. Designerissa piirtoalkiota on 15 kappaletta ja ne sisältävät In*Visionista puuttuvan paraabelin ja käyrän.

Organisaatio- ja vastaavat kaaviot syntyvät, kun kopio laatikoita haluttuihin paikkoihin. Organisaation osia korostetaan väreillä ja täyttökuviolla. Yritysgrafiikan tuottamiseen Designer ei kuitenkaan sovellu, sillä se ei muodosta kuvaajia lukuarvoista.

Alueita muodostetaan yhdistämällä käyriä tai suoria viivasegmenttejä alueen reunoiksi. Muodostetut alueet voi täyttää ja värittää. Näin tehdään esimerkiksi kuvia esineistä, joissa on reikiä. Tällaisten kuvien tekeminen muilla ohjelmilla on vaikeaa tai jopa mahdotonta.



Designerissa voi käyttää kuvanlukijalla luettuja kuvia. Niitä voi kuitenkin muotoilla ainoastaan muuttamalla kokoa ja väriä sekä kiertämällä. Kohdistuspisteet ja kuvan tarkastelu halutussa mittakaavassa auttavat kuvien piirtämisessä. Zoom-komennolla valittu kuvan osa saadaan täyttämään koko piirrosalue ja pienimmät yksityiskohdat viimeisteltä. View Used Pages -komennolla taas saadaan koko kuva näkyviin.

Runsaiden CAD-ominaisuuksien ansiosta Designer sopii hyvin tekniseen suunnitteluun ja dokumentointiin. Esimerkiksi pohjapiirroksista saadaan aina oikean kokoisia, sillä Designer ilmoittaa kuvan koon halutuissa yksiköissä. Valmiit kuvat voi siirtää Aldus PageMaker tai Ventura Publisher -ohjelmilla tehtyihin julkaisuihin tai Windows Write -ohjelmalla tehtyihin dokumentteihin.

Kerran piirretty, aina käytettävissä

Kerran tehtyjen kuvien uudelleenkäyttö on grafiikkaohjelman tärkeimpiä ominaisuuksia. Kun jokin tärkeä symboli on kerran piirretty, sitä on voitava käyttää myös muissa kuvissa, sillä hyvän kuvan piirtämiseen voi mennä tunteja. Designerissa symbolit ja kuvat voi tallentaa kuvakirjastoihin uudelleenkäyttöä odottaen.

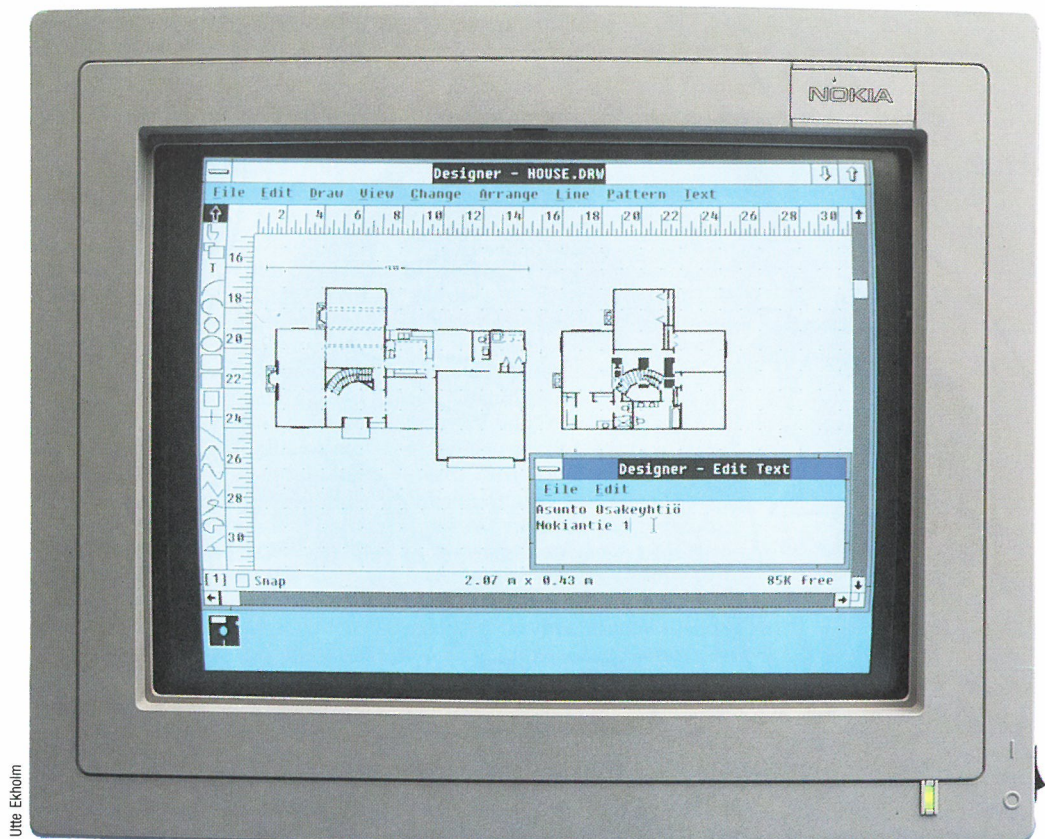
Kuvakirjastot ovat tavallisia kuvatiedostoja, joiden symbolit on nimetty. Kun usein käytettävät symbolit ja kuvat on kerran huolellisesti piirretty, ne voidaan sen jälkeen jakaa kaikille Designerin käyttäjille. Näin esimerkiksi esitettävien kalvojen taso paranee ja muoto yhtenäistyy. Designeriin kuuluu kuvakirjasto, jossa on noin 200 valmista kuvaa.

Symboleja ja kuvia haetaan kirjastoista siten, että kuvakirjasto avataan omaan ikkunaan ja siirretään haluttu kuva varsinaiseen piirroksen tai valitaan haluttu kirjasto ja siitä haluttu symboli sen nimen perusteella ilman, että kirjaston kuvat ovat näkyvissä. Jälkimmäinen tapa on nopeampi, sillä silloin ohjelmaa ei tarvitse näyttöä kaikkia symboleja eikä aktiivista ikkunaa tarvitse vaihtaa.

Designerilla voidaan käsitellä myös muiden Micrografxin grafiikkaohjelmien, kuten Windows Graphin ja In*Visionin kuvia.

Ei kuvia ilman tekstiä

Kuvien ja tekstien yhdistäminen on Designerin vahvin puoli.



Uffe Ekholm

Oy Yritys Ab

Tuotelinjat

Kalusteet

Kodinkalusteet
Toimistokalusteet

Kodinkoneet

Kylmäkalusteet
Pesukoneet

Urheiluvälineet

Jalkineet
Tekstiilit
Sukset
Jääkiekko
varusteet

Designerilla piirretyn talon pohjapiirroksen ympärillä on kuva-alueen sekä yläreunan mittatikka. Näiden avulla kuva kohdistetaan sivulla. Näytön vasemmassa reunassa on piirto-alkioita, joista piirretään monenmoisia kuvia.

Erilaiset kaaviot syntyvät Designerilla helposti. Samanlaisia kuvia voi kopioida Array-komennolla halutulle etäisyydelle. Designerin teksti-, väri- ja alueiden täyttöominaisuuksilla korostetaan eri osia.

Tekstit lisätään kuviin erillisessä editori-ikkunassa, jossa sekä kirjoitetaan että muokataan kuvien tekstejä. Editori-ikkuna käynnistyy, kun valitaan kirjoituskohde kuvassa. Designeriin voi lisäksi lukea ASCII-muotoisia tekstitiedostoja.

Kerralla lisättävät tekstit voivat olla useita riviä pitkiä ja tekstiä voi kiertää mielivaltaiseen asentoon. Kun kirjainten kokoa muutetaan, Designer muuttaa vastaavasti rivinväliä. Kirjainmallit ovat kirjoitinsidonnaisia. Designerilla voi käyttää kaikkia kirjoittimien merkistöjä ja lisäksi käytössä on kolme grafiikkamerkistöä.

Mikäli tulostuksessa käytettävä kirjoitin ei tue riittävän suurikokoisia merkkejä, ne saadaan Designerin grafiikkamerkistöis-

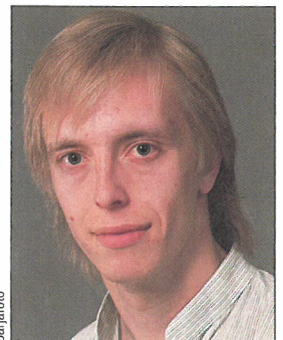
tä. Riittävän suuret ja selkeät kirjaimet ovat tärkeitä etenkin piirtoheitinkalvoissa.

Vaateidenmukaiset tulostuslaitteet

Microsoft Windows-ympäristöön suunnitellut ohjelmat voivat hyödyntää kaikkia Windowsin tukemia oheislaitteita. Laitteistoriippumattomuus onkin yksi Windows-ympäristön hyvistä puolista.

Designerin monipuoliset väriominaisuudet voi hyödyntää piirturilla tai HP PaintJet -värimustesuihkukirjoittimella. Useilla laser- ja matriisikirjoittimilla kuvat voi tulostaa yksivärisinä ja silloin värit erottuvat erilaisina rasterointeina. Esityksiä voi tulostaa myös kalvolle. ■

Jari Luokkala on perehtynyt grafiikkaohjelmiin Nokia Dynat Tuotekehityksessä PC Applications -ryhmässä.



Sarjafoto

Saamisen Tapsan muistelmät osa II "Löylyä"

Asiantuntija Saarinen ja myynti-tykki Mäkinen viruivat puolinu-kuksissa Erneston rantasaunan lauteilla raukeassa lämmössä. Molemmilta oli veto pahasti ka-teissa, Rauta-Yhtiön edustajien kanssa tehdyltä Lontoon matkal-ta oli palattu vasta aamupäiväl-lä. Neljä vuorokautta oli sujunut mukavan sutjakkaasti, mutta johtaja Tasanen oli vaatinut heti illaksi pystyyn sisäisen kritiikki-palaverin, myyntikurssilla oppi-mansa kaavan mukaan. Kave-rukset huokailivat ylätiskillä hil-jaa vuorotellen, Tasanen puus-kutti järvestä takaisin löylyyn täynnä puhtia ja uskoa kinaten Erneston kanssa internaalin kri-tisoinnin merkityksestä tietoko-nemyynnin henkisen dynami-suuden kasvuun, kuten seminaa-reissa oli puhuttu.

– Jos aiomme pysyä kilpailussa mukana, ei pelkkä etukäteisana-lyysi riitä, myös tehdyt virheet pitää tarkkaan ruotia, todisti Ta-sanen tarmokkaana.

– Minun aikamani ei kritiikkiko-kouksia paljon tarvittu. Hyvä, kun ehti uuden kaupan kimp-puun edellisen loputtua. Mu-naukset näki siitä, tuliko keikka kotiin vai ei, vastasi Ernesto ja heitti kopallisen kiukaalle. Väsy-neet pojat pitelivät päätään.

– Ilmankos sorruit niin aikaisin tänne yksinäisen eräkämpän saunamajuriksi, tykitti Tasanen takaisin.

– Johtaja etsii kohta uuden tuki-kohdan tapaamisilleen, ärähti partainen erakko, vanha koke-nut kauppahai. Sitäpaitsi tämä vetäytyminen oli lopulta tietoi-nen valinta, en olisi enää kauan jaksanut. Ja kun sinua, johtaja Tasanen katselee, niin tervetu-loa parin vuoden päästä entisten myyntipäälliköiden kerhoon mu-kaan vaan!

– Olkaa nyt pojat hiljaa, tässä krapulassa on muutenkin kestä-mistä, voivotteli Mäkinen vaa-ka-asennostaan.

Leppeää rauhaa kesti vain het-ken. Kokouksen teeman mukai-esti moitittiin ensin Mäkinen, joka oli menomatalla koneessa hoilannut Porilaisten marssia, kosinut lentoemäntää ja tullissa joutunut ruumiintarkastukseen vaatiessaan äänekkäästi paikalli-sia futishuligaaneja tulemaan Suomeen tappelemaan, jos us-kaltavat. Lisäksi hän oli Tasasen mukaan käsitellyt tiukkaa edus-tusbudjettia turhan helläkätises-ti. Mäkinen kuunteli ripityksen nöyränä kuten monasti ennen-kin, röykytyksen loputtua totesi vain:

– Minulle jäi kuitenkin sellainen tunne, että asiakkaat nauttivat reissusta. Ihmisiä hekin ovat, ei-

vät koneita, vaikka me heille sattumalta nyt sellaisia myymekin. Ihmisen on saatava elää. Vaivainen myyntimieskin on ihminen!

– Aina samanlainen idealisti. Näissä hommissa eivät pehmot pärjää! Ota oppia vaikka tuosta ns. asiantuntija Tapsa Saarisesta, joka ei hymyillyt koko matkan aikana kertaakaan, ärähti Tasanen.

– Itse olet urallasi hymyillyt vain sen kerran, kun pomosi kaatui portaissa, heitti Tapsa.

Niin kävikin. Ketola rynni kuin hirvi. Sen verran päätti Tapsa työtoveriaan auttaa, että neuvoi Ketolan ”Mihin se meni?” kysymykseen tämän risteyksestä väärrään suuntaan.

– Voi Ritu-parkaa, kyllä Ketola sen kiinni saa, vanha estejuoksun piirimestari, huolestui Leena.

– Ei saa, Ketola lähti kiertämään kirkonkylän kautta!

– Ai kun sinä olet ihana!

Mökin pihassa oli aivan hiljaisa. Nuori pari yritti hiipiä hiljaa rantasaunalle mökissä oljoiden huomaamatta. Kun he olivat varovasti aukaisemassa narisevaa ovea, kuului rantakalliolta mahtava huuto.

– Jumalauta jätkä! Neuvoit minut väärrään suuntaan. Nyt tulee turpaan!

– Se on Ketola. Äkkiä veneseen!

Niin kului se kesäyö. Päästyään Ketolan viskomien kivien kanteisyydeltä souteli Saarinen hitain vedoin kohti kaupunkia, jonka silhuetti häämötti parinkymmenen kilometrin päässä. Leena loikoi perätuhdolla lämpimästä suviyöstä nauttien.

Tasanen ja Ketola nukkuivat vierekkäin Erneston vierasvuoteessa sikeää unta. Ritu taivalsi vielä aamulenkkiään kohti kotia, matka kävi hyvin harjoituksesta kohta alkavaan Suomi juoksee tapahtumaan. Hallintojohtaja Reima ja myyntitykki Mäkinen tallustelivat nuorten ystäviensä kanssa sadekuuron kostuttamalla festivaalialueella viinipulloa heiluttaen. Vielä vuosia myöhemmin Reima kertoi retkestä: ”Ikinä minulla ei ole ollut niin hauskaa. Nuorruin ainakin 10 vuotta. Kerrankin sai kuunnella ihmisten nuorta luontevaa puhetta eikä pelkkiä myyntiargumentteja!”

Tapsa souti, souti oikein sydämensä kyllyydestä. Hiljaisuuden katkaisi tuhdoltaan vettä roiskiva Leena.

– Haluatko, että teen yhden paljastuksen?

– Tottakai, riisutko heti?

– Älä viitsi. Te saatte sen kaupan, se on päätetty jo ajat sitten, pojat vaan haluavat vähän pompotella teitä. Te olette kuulemma ihan hyvä toimittaja, alennusmielessä se Reimakin vain kiukuttelee. Reima tykkää Mäkisestä, kun se on niin lämmin ja inhimillinen mokineen ja loppusoinnuttomine runoineen.

– Minä taas voin vastaavasti kertoa, että tämä meidän kaksin joutuminen oli täysin ennalta suunniteltua. Eilen illalla kolme tuntia jaettiin rooleja aina Ritun pakojuoksemista myöten. Siksi Ritu oli selvinpään ja lenkkarit jalassa, saatiin Ketola olemaan ilta hiljaa. Että semmoisia kaupapamiehiä me ollaan.

– Tietysti vanhat ohjelmanvääntäjät taas asialla, yäk!

– Kaikesta huolimatta, minulla on pitkästä aikaa rehellisesti sanottuna onnellinen olo, pehmeni Tapsa.

– Sama täällä perätuhdolla!

Kaupungin työpäivään heräävää taustaa vasten vene näytti hyljättyltä. Airot roikkuivat huolimattomasti vedessä. Paatin vuotanut vettä puolillaan oleva pohja oli kuitenkin täynnä vilkasta elämää.

Kyllä ostohalu syntyy muusta kuin pakosta.

– Älä nyt, Tasanen on kovasti käynyt viime aikoina ihmissuhdekursseja. Hänellä on nyt psykologinen ote myyntityöhön. Kyllä se yrittää parhaansa ja toissisaan.

– Ihminenhan teidän pitäisikin huomioon ottaa. Käyttäjillehän te olette vääntäneet sovelluksia itseänne varten, rahaa on kulu-
nut huikeita määriä ja kaikki ovat valittaneet. Homma ei ole palvellut meitä, jotka sitä todella tarvitsemme.

– Älkää valittako! Pitää alistua meidän viisaampien tahtoon! Tapsaa alkoi jo naurattaa.

– Suu kiinni poika! Se on nyt sinun vuorosi alistua!

Leena kaatoi Saarisin hellästi sohvalle selälleen, kutitteli tämän harvoja viiksiä. Asiantuntija Tapsa Saarinen alkoi selvästi sulaa, irrotteli alleen jääneitä käsiään ja poskensa alkoivat selvästi punottaa.

– Täystunti, tietosuoja- ja turvatakas-
tus. Mitäs täällä touhutaan, on puoliyö, koohotti nuori teollisuusvartija ovenraosta. Tupa tyhjäksi! Miksei mitään tapahdu?

– Otetaan rauhallisesti, olemme talon väkeä, yritti Tapsa maltilla.

– Ei auta. Minulla on nimenomaan määräys epäsiiveellisen toiminnan ehkäisemiseksi yhtiön tiloissa.

– Ei ole ennen ollut. Kuka perkele sellaisen käskyn on antanut?

– Juuri soitettiin, joku johtaja se oli, Loiva tai Laakea tai jotain sinnepäin.

– Tasanen?

– Juuri se, johtaja Tasanen!

– Jos se on suuren intiaanipäällikön viimeinen sana, niin kai se on pakko lähteä, vieläköhän sinne Erneston mökille mahtuu? Saarista houkutti kopsauttaa ke-
sääpulaisten näköistä vartijaa, mutta Tapsa malttoi mielensä.

Taksimatalla Erneston mökki-
tiellä oli vilkasta. Jo tienhaaras-
sa liftasivat kaulakkain Mäkinen ja Reima, joka kovaäänisesti ilmoitti lähtevänsä mihin vain, missä ei tarvitse kinata tosikko-
Tasanen kanssa tietokoneasioista. Paikalle pysähtyikin musta letukka täynnä rock-juhlille mat-
kaavia nuoria. Mäkisen estelyistä huolimatta Reima tunkeutui takapenkin ahtauteen vetäen vastuullisen myyntineuvottelijan mukanaan. Nuoret hieman protestoivat ylilihavia yritysjohtajia, mutta pian matka jatkui kohti Ranta-Rockia. Mittariautossaan piilossa Tapsa ja Leena naureskelivat Reiman tahdonlujuutta, ja jatkoivat kohti erä-
mökkiiä.

Kotvan kuluttua porhalsi vastaan Ritu treenatulla askeleella ja jatkoi täysillä tulijoihin reagoimatta kaupungin suuntaan.

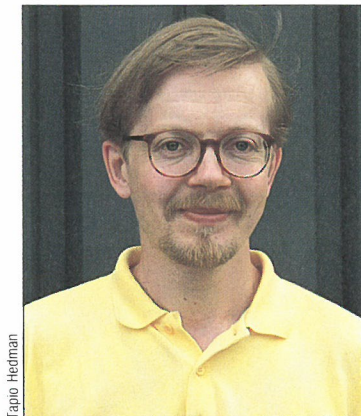
– Mikä se oli, ihmetteli Leena.

– Yhtiömme sihteeri siinä valmistautuu huomista rankkaa työpäivää varten. Lyödäänkö vettä, että kohta puuskuttaa Ketola perässä.

Lähiverkolla varmaan lentokonehuoltoon

Finnairilla on työnteos-
sa perinteisesti sovellet-
tu ryhmätyöajatus. Nykyään töitä tehdään paikassa kuin paikassa suurelta osin henkilökohtaisella tietokoneella, PC:llä. Näin ollen työryhmien muodostamiseksi täytyy PC:t voida kytkeä toisiinsa. Tällöin ryhmän sisäisessä viestinnässä pystytään käyttämään hyväksi yhteisiä tiedostoja ja oheislaitteita. Finnairilla hankitaan nyt kokemuksia todellisesta työryhmätyöskentelystä.

"Lentokoneet huolletaan ilmailuviranomaisten määrittelemien vaatimusten mukaisesti. TokenNetiä käytetään meillä Finnairilla noiden vaatimusten sanelemien huoltotöiden ohjeistuksen laadintaan", sanoo atk-projektipäällikkö Pekka Rossi.



Tapio Hedman

Miksi ylipäätään on tarpeellista työskennellä yhdessä toisten kanssa? Eikö pätevä työntekijä saa hyvää jälkeä aikaan yksinäänkin? Kyllä varmasti saa, mutta tulokset ovat lähes poikkeuksetta parempia ryhmän yhteistyöllä saavutettuina. Tässä pätee synergian laki: kokonaisuus on enemmän kuin osiensä summa.

Nokia Datan Tuotekehitysyksikön projektipäällikkö Heikki Seppälän mukaan 70 prosenttia työssä tarvittavasta tiedosta löytyy työryhmän sisältä. Vain 30 prosenttia työlle merkityksellisestä tiedosta sijaitsee muualla organisaatiossa. "Verkko-työskentelyn ajatus on se, että käytetään lähellä olevaa tietoa hyväksi. Työryhmiä voi luonnollisesti olla useita, jotka sitten ovat yhteydessä ylempään

tason järjestelmiin."

Seppälä kertoo, että "Finnairilla on sisäistetty ajatus, jonka mukaan tiedon on oltava siellä missä sitä tarvitaan." Hän uskookin 1990-luvun tietotekniikan kehittyvän tähän suuntaan. "Todelliset hajautetut järjestelmät ovat tulevaisuudessa sellaisia, joissa sovelluksetkin on hajautettu", Seppälä povaa.

TokenNet Finnairin ratkaisu

Finnair Oy:ssä työryhmäajatus on toteutettu Nokia Datan TokenNet-lähiverkkoratkaisuna. TokenNet on Nokia Datan omaa tuotantoa ja se sopii yhteen teollisuusstandardin mukaisen tokenring-verkon kanssa. Finnair-toimituksessa varsinainen fyysinen kaapelointi on toteutettu tavanomaisella säteittäisellä tokenring-kaapeloinnilla.

Verkko on nyt liitetty 12 työasemaa. Näistä neljä on varustettu värinäytöllä ja kahdeksan positiivinäytöllä. Verkossa toimii yksi varsinainen kymmenen työaseman ryhmä, kaksi muuta työasemaa toimivat toistaiseksi itsenäisesti. Kaikki työasemat sijaitsevat samassa rakennuksessa, tosin kahdessa kerroksessa.

TokenNet-verkkoa käytetään Finnairilla sekä tavallisessa toimistoautomaatiossa että atk-kehitysympäristönä. Kaikkien verkon jäsenten käytössä olevaan toimistoautomaatiopakettiin kuuluvat muun muassa arkisto ja kalenteri. Kehitysympäristö tarkoittaa sitä, että Finnair rakentaa varsinaiset toimialakohittaiset sovellukset itse.

Lentokoneiden huoltotyön ohjeistus

Mihin tarkoitukseen Finnairin verkko on hankittu? Yhtiön atk-projektipäällikkö Pekka Rossin mukaan TokenNetiin liitetyissä työasemissa tehdään pääasiassa lentokoneiden huoltotyön ohjeistusta. "Lentokoneiden huolto perustuu huoltojärjestelmään, joka määrittää ilmailuviranomaisten hyväksymien huoltovaatimuksina. Vaatimukset muun muassa kertovat, missä ajanjaksossa mikin huoltotoimenpide on tehtävä. Ohjeistus yksilöi, miten työ käytännössä hallitsa tehdään", selvittää Ros-

si. Lyhyet ja ytimekkäät vaatimukset siis konkretisoidaan käytännön työohjeiksi.

"TokenNet-hanke liittyy osana suurempaan projektiin, jossa tarkoitus on siirtää huoltojen pakettivalvonnasta jokaisen koneen jokaisen huoltovaatimuksen valvontaan. Mittasuhdetta kuvastaa se, että yhteen huoltopakettiin saattaa kuulua toista tuhatta vaatimusta", kuvailee Rossi. "Ohjeistus tallennetaan suorkoneessa olevaan tietokantaan. Varsinaiset ohjeet laaditaan ja ylläpidetään verkon työasemissa. Nyt niissä hoidetaan jo tekstinkäsittely. Standardien vakiinnuttua työasemissa on tarkoitus hoitaa myös kuvankäsittely."

Suorkoneen tietokanta sisältää huomattavan määrän tietoa esimerkiksi huoltovaateista, materiaaleista ja korvaavuuksista. "Huolto-ohjeen tekosovelluksessa määritellään lähtötietojen perusteella, miten jokin toimenpide suoritetaan. Ohje sisältää työn tekstikuvauksen, kertoo materiaalitarpeet ja luettelee tarvittavat työkalut", Rossi tämentää.

Sovelluksen jako APPC-tuotteella

Finnairille toimitettua TokenNet-järjestelmää voidaan pitää uuden kehityssuunnan ensiaskeleena. Jo muutamia vuosia PC:itä on käytetty sekä henkilökohtaisina tietokoneina että keskustietokoneiden päätteinä. Nyt toimitetussa järjestelmässä käytetään hyväksi Nokia Datan APPC-tuotetta (Advanced Programme to Programme Communication), joka mahdollistaa sovelluksen jakamisen keskustietokoneen ja PC-työaseman kesken. Näin saadaan molempien ympäristöjen parhaat puolet esiin. Esimerkiksi operatiivisen, koko yritystä palvelevan sovelluksen pääosa voi sijaita keskustietokoneessa, kun vaikkapa grafiikkaa hyödyntävä käyttöliittymä on hajautettu työasemiin.

Nokia Datan ratkaisussa suurin osa APPC-ohjelmistosta sijaitsee erillisessä SNA-siltayksikössä. Tämä vähentää keskustietokoneen kuormaa. Kun vain osa ohjelmistosta on työasemissa, varsinaiselle työasemasovellukselle jää enemmän muistitilaa.



Hajauta ja hallitse

Mahdollisuus saada suora liittymä tokenring-ympäristöstä suorkoneeseen oli yksi TokenNetin valintaa puoltava tekijä Finnairilla. TokenNetin APPC-ominaisuus oli Pekka Rossin mukaan kuitenkin yksi painavimmista syistä, joka tuki Nokia Datan valintaa yhteistyökumppaniksi. "APPC on teollisuusstandardiympäristössä ainoa protokolla, jolla saadaan hajautetusti tehtyä luotettavia ja riittävän selkeärakenteisia tapahtumankäsittelysovelluksia."

"Projektimme on hajautetun tietojenkäsittelyn pilottihanke, jossa käytetään työaseman tehoa isomman sovelluksen yhteydessä. Suurin osa sovelluksesta sijaitsee työasemassa, sillä koko sovelluksen logiikka on siellä. APPC kuitenkin vähentää työasematietokoneen kuormaa eikä yksittäinen työasema kuormitu liikaa", kertoo Rossi.

"Meillä hajautettu tietojenkäsittely tarkoittaa sitä, että pyrimme vähentämään keskuskoneen CPU-kuormaa. Toisaalta ajatusena on tehdä mahdolliseksi miellyttävämmät käyttöliittymät", selostaa Finnairin atk-projektipäällikkö. "Hajautus äärimmilleen vietynä ei meillä toden-

näköisesti kuitenkaan tule kysymykseen, sillä ohjeistuksen täytyttyä käytännössä olla saatavilla 24 tuntia vuorokaudessa."

Verkko on turvallinen

"Työryhmäverkossa on lukuisia etuja tietojenkäsittelyn keskitettyyn ratkaisuun verrattuna", vakuuttaa Tiimityöryhmäverkkosen tuotepäällikkö Olli Nurminen Nokia Datan Tuotemarkkinoinnista. "Ensinnäkin standardius takaa avoimuuden. Näin ollen ohjelmistohankinta on helppoa ja järjestelmää on yksinkertaista laajentaa. Toinen etu on edullisuus, sillä sekä prosessori että massamuistikapasiteetti ovat halvempia verrattuna keskitettyyn ratkaisuun. Lisäksi työasemaa voidaan käyttää henkilökohtaisena mikrona. Kokonaisuus on varsin ergonominen, sillä käytössä voidaan hyödyntää esimerkiksi hiirtä, grafiikkaa ja käyttöliittymää."

Tietosuoja tai -turvallisuus on päivän sana. Miten tämä asia liittyy Finnairin TokenNetiin? "Tietosuojassa tärkeintä meille on se, että tallennetut ja sovelusten tietokannassa käytettävissä olevat tiedot ovat oikeita. Kuka tahansa ei saa päästä niitä muuttamaan", kertoo Pekka Rossi. "Hakkeriongelma ei mei-

tä suoranaisesti koske, sillä verkkohan on suljettu", Rossi jatkaa. Työryhmäverkot ovat tietosuojan osalta varsin turvallisia - tiedothan ovat vain siellä missä niitä käytetään.

Katse horisonttiin

Rossi korostaa puheessaan verkon liitettävyyttä, APPC-ominaisuutta sekä hajautettua tietojenkäsittelyä. "Verkko, joka mahdollistaa helpon liitettävyyden, tuo sovelluksiin melkoisesti uusia ulottuvuuksia. Sovelluksia on mahdollista modifioida ja tehdä niistä osittain hajautettuja - jopa niinkin, että ne käyttävät osittain paikallista tietokantaa ja osittain suurkonetietoa. Tavoitteenamme on, että paikallisesti voidaan säilyttää työryhmälle tarpeellista tietoa, jota puolestaan voidaan soveltaa yhdessä yleisesti käytössä olevan tiedon kanssa", hän maalailee.

"Lähtökohtamme on ollut, että standardiratkaisulla tehdään mahdollisimman luotettavia hajautettuja sovelluksia", tiivistää Pekka Rossi. Finnairin pilottihanke on tarkoitus saada kulu- van vuoden loppuun mennessä tuotantokäyttöön. Kokemukset viitoittavat tietä hajautetun tietojenkäsittelyn laajentamiselle.

Ilmailussa turvallisuus ja varmuus ovat kaksi avainsanaa. Kun TokenNet-järjestelmä on Finnairilla lopullisesti valmis, niin lentokoneiden huoltotöiden suorittajat saavat suoraan tulostimelta yksityiskohtaisen kuvauksen siitä, miten mikin yksittäinen huoltotoimenpide on tehtävä.

Muisti pettää, EMS apuun

Kun mikron muisti ei riitä, tarvitaan lisämuistia. Nokia toimittaa Intelin Above Board PS/286 -kortteja, jotka toimivat LIM/EMS 4.0 -määrittelyn mukaisesti ja joiden muistityyppi on määriteltävissä.

MS-DOS-käyttöjärjestelmän ja teollisuusstandardin mukaisten mikrojen suosio on tuottanut runsaasti tähän ympäristöön ohjelmistoja, joista monet ovat nykyisin varsin laajoja. Tämän kehityksen seurauksena MS-DOS:n 640 kilotavun muisti ei enää aina riitä.

DOS:n 640 kilotavuun rajoituvaa muistialuetta sanotaan conventional-tyyppiseksi. Muistipulaan on kehitetty avuksi erilaisia lisämuisteja kuten extended-, EMS-, EEMS- ja LIM/EMS-muisteja. Näistä ei kuitenkaan ole apua joka hätään. Ohjelmiston on nimenomaan oltava tehty siten, että se pystyy hyödyntämään ao. lisämuistia.

Yleisimmin käytetyiksi expanded-muistin määrittelyiksi muodostuivat LIM/EMS (Lotus/Intel/Microsoft Expanded Memory Specification) versio 3.2 ja EEMS. Lotus, Intel ja Microsoft kehittivät LIM/EMS-määrittelyn. EEMS on AST:n, Quadramin ja Ashton-Taten muistimäärittely. Yhdenmukaisen määrittelyn aikaansaamiseksi yhdistettiin molemmat mainitut määrittelyt uudeksi määrittelyksi, joka tunnetaan nimellä LIM/EMS versio 4.0. Uusi versio sisältää myös uusia ominaisuuksia edeltäjiinsä verrattuna.

Lisämuisti ohjelmiston mukaan

Ennen lisämuistin hankintaa selvitetään, pystyykö ohjelmisto hyödyntämään extended- vai expanded-tyyppistä lisämuistia. Tämän tiedon löytää ohjelmiston dokumentoinnista, esimerkiksi käsikirjoissa ja esitteissä on yleensä mainittu hyödyntäkö tai vaatiiko kyseinen ohjelmisto lisämuistia. Myös lisämuistin tyyppi on yleensä kerrottu.

Välttämättä ei ole samantekevää mitä muistityyppejä käytetään, vaikka ohjelmisto voisi hyödyntää sekä extended- että expanded-muistia. Sovellukseen sisältyvän ohjelmiston avulla voidaan ehkä käyttää extended-muistia expanded-muistin tapaan, mutta "aito" expanded-muisti olisi tehokkaampi. Sovelluksen joidenkin ominaisuuksien käyttö saattaa myös edellyttää aitoa expanded-muistia. Käytettävä muistityyppi voi siis muuttaa ohjelmiston käyttöominaisuuksia. Expanded-muistista ei ole apua, jos laitteistossa on käytettävissä vähemmän conventional-tyyppistä muistia kuin pelkästään sovelluksen käynnistys vaatii.

Esimerkiksi Xenix, OS/2,

Oracle, Alsys ADA ja Autocad hyödyntävät extended-muistia. Expanded-muistia hyödyntävät muun muassa Lotus 1-2-3 versio 2.0 tai uudempi, Symphony versio 1.2 tai uudempi, Windows 2, Excel ja Paradox.

Intel Above Board PS/286

Nokian expanded-muisti on Intelin Above Board PS/286, joka perustuu LIM/EMS 4.0 -tasoon. Muistikortti on 16-bittinen ja se tarjoaa 2 megatavua muistia, yhden sarja- ja rinnakkaisportin. Kortin toimitukseen sisältyy lisäksi tarvittava asennus- ja testi-ohjelmisto sekä käsikirja.

Muistikortilla oleva muisti voidaan määritellä conventional-, extended-, expanded-tyyppiseksi tai osaksi edellä mainittuja. Kortti toimitetaan 80286-suorittimeen pohjautuviin MikroMikko 3 ja MikroMikko 3TT -laitteistoihin. Kortin voi tilata laitteiston yhteydessä, jolloin kortti asennetaan jo tehtaalla tai erillisenä lisäkorttina.

Kortti konfiguroidaan ja portit asetetaan kortin mukana toimitettavan asennusohjelmiston avulla. Näin ollen on helppoa muuttaa kortin määrittelyjä





myöhemmin eikä korttia tarvitse poistaa laitteistosta esimerkiksi DIP-kytkinten asetusten muuttamisen ajaksi.

Muistin määrittely

Nokia toimittaa Intel 'Above Board' -muistikortin kokonaan expanded-tyyppisenä muistina. Muisti määritellään asennusohjelmiston avulla osittain tai kokonaan conventional-, extended- tai expanded-tyyppiseksi. Kortti ei korvaa valikoimassa jo aiemmin olleita extended-tyypisiä 1, 2 ja 3 megatavun lisämuistikortteja. Viimeksi mainittuja extended-muistikortteja ei voi sellaisenaan käyttää expanded-tyyppisenä muistina.

Conventional-tyyppisestä muistista ei ole hyötyä, jos MikroMikossa jo on 640 kilotavua tai enemmän työmuistia. Extended-tyyppistä muistia voi lisätä asentamalla puhtaasti extended-tyyppinen lisämuistikortti, joita Nokian valikoimassa on 1, 2 tai 3 megatavua. Vaihtoehtoisesti voi määrittellä Intel Above -kortilla olevan muistin osaksi tai kokonaan extended-tyyppiseksi.

Sarja- ja rinnakkaisporttien asetukset

Sarja- ja rinnakkaisporttien osoitteet sekä keskeytystasot asetetaan laitteistokokoonpanon mukaisesti. Asennusohjelma määrittelee keskeytystason (IRQ5 tai IRQ7) rinnakkaisportille, eikä määriteltyä tasoa voi muuttaa. Kortilla olevat lisäportit voivat joissakin tilanteissa korvata MikroMikko 3TT -laitteistossa käytettävän sarja- ja rinnakkaisliitännän lisäkortin. Muistikortin sarjaportti ei sovelu modeemikäyttöön.

Tarkista ennen asennusta

Ennen asennusohjelmiston suoritusta on syytä selvittää laitteistoon asennettujen muiden lisäkorttien ja porttien asetukset kuten osoitteet, keskeytystasot ja systeemimuistialueen käyttö. Samojen asetusarvojen esiintymistä useammalla kortilla voisi verrata tilanteeseen, jossa monella talolla on sama osoite. On siinä Mikro-postinkantajalla miettimistä. Useimmille asetuksille löytyy vaihtoehtoja, joilla voidaan välttää päällekkäisiä arvo-

ja. Ellei käytettävissä olevista vaihtoehtoista ole apua, jokin ominaisuus esimerkiksi rinnakkaisportti on poistettava käytöstä.

Systeemimuistialueeksi nimitetään muistialuetta, joka ulottuu 640 kilotavusta 1 megatavuun. Tämä muistialue on varattu erikoiskäyttöön. Intelin Above Board PS/286 varaa systeemi-muistialueesta 64 kilotavun alueen, jonka alkuosoite oletusarvoisesti on D0000 (hexa). Alkuosoite näkyy arvona D000 EMM-määrittelyssä, jonka asennusohjelmisto automaattisesti lisää CONFIG.SYS-tiedostoon. Systeemimuistialueelle löytyy muitakin käyttäjiä kuten VGA tai EGA. Myöskään systeemi-muistialueen käytössä ei saa esiintyä päällekkäisyyttä, mutta tämän seikan selvittäminen saat- taan kuitenkin olla vaikeampaa kuin I/O-osoitteiden ja keskeytystasojen selvittäminen.

Kortin tehokas käyttö vaatii jonkin verran perehtymistä kortin toimintaan, korttia hyödyntävään ohjelmistoon sekä käyttöympäristöön, mutta kortti tarjoaa vastapalveluna kehittyneiden ohjelmistojen käyttömahdollisuuden DOS-ympäristössä.

MikroPOS valvoo kaupan kannattavuutta



Lähiverkkoon kytetyt mikrotietokoneet ja kassapäätteet muodostavat MikroPOS-kassajärjestelmän. Myymälän tulos ja kassavirta saadaan sillä tosiaikaiseen seurantaan.

Viivakoodien käyttöönotto tavaroiden merkinnässä on tuonut tietotekniikan vähittäiskauppaan. Tietokoneiden ohjaama kassajärjestelmä nostaa koko myymälän toiminnan tehokkuutta, kun kassalinjan toiminta nopeutuu ja hintahallinta helpottuu. Samalla tuottavuus on tarkassa seurannassa.

MikroPOS-kassajärjestelmä on ensisijaisesti tarkoitettu pienen tai keskikokoiseen päivittäistavarakauppaan. Järjestelmä toimitetaan aina kokonaisuutena, johon kuuluvat laitteisto, sovellusohjelmisto, käyttökoulutus ja järjestelmän asennukseen sekä huoltoon liittyvät palvelut.

Oma tosiaikainen käyttöjärjestelmä

Vähittäiskaupan sovellusalue eroaa melkoisesti totutusta mikrotietokoneiden käyttöympäristöstä. Myymälän tietojenkäsittely on luonteeltaan hajautettua, sillä tapahtumia syntyy jatkuvasti useassa kassapisteessä. Pienheikössäkin myymälässä on val-

tavasti tapahtumia, koska jokaisen myydyn tuotteen muutosvaihtus koko myymälän tilanteeseen pitää erikseen rekisteröidä. Lisäksi kassan myyntitoiminoissa edellytetään nopeita vasteaikoja, alle 0,5 sekuntia kuittiriviä kohti. Myös kassojen hinta-, tarjous- ja tuotetietoja on voitava muuttaa nopeasti ja helposti.

Kassajärjestelmä vaatii toimiaukseen tehokkaan tosiaikaisen usean käyttäjän käyttöjärjestelmän. Mikrojen tyypilliset käyttöjärjestelmät, kuten MS-DOS palvelevat vain yhtä käyttäjää. Nykyiset verkkokäyttöjärjestelmätkin, esimerkiksi MS-NET, tarjoavat vain tiedonsiirtoyhteyden ja resurssien jaon verkon eri laitteiden välillä.

Koska valmista ympäristöä ei järjestelmää suunniteltaessa ollut saatavana, MikroPOS-kassajärjestelmän ohjelmisto on rakennettu Nokian oman POS-OS käyttöjärjestelmän päälle. Se on tosiaikainen käyttöjärjestelmä, joka sovellusten näkökulmasta muistuttaa erittäin paljon MS-

DOS-ympäristöä. Näin päästään kassajärjestelmän oman ohjelmiston lisäksi hyödyntämään useimpia mikrokäyttöön tarkoitettuja valmisohjelmistoja ja ohjelmistotyökaluja.

Tyhvät kassapäätteet älykkäässä verkossa

MikroPOS-kassajärjestelmä perustuu väylärakenteiseen verkkoon, jossa on vähintään kaksi MPS 4 -keskusyksikköä ja tavallisesti kolmesta kymmeneen CRS 300 -kassapäättettä. Perusjärjestelmän mikroja kutsutaan pääkoneeksi ja kassaohjaimiksi. Pääkoneeseen on kytketty näyttöpääte ja hyllynreunaetiketti sekä raporttikirjoittimet.

Kassapäätteet eivät koskaan voi toimia itsenäisesti, vaan niiden toiminnasta huolehtii aina kassaohjaimessa toimiva myyntiohjelmisto. Kassapäätteet ovat tavallaan pelkkiä syöttö- ja tulostuslaitteita, jotka nopean paikallisväylän kautta ovat yhteydessä muuhun järjestelmään.

Näin saavutetaan hyvä tehokkuus, kun koko muuttuva tietosisältö voidaan keskittää kassaohjaimeen. Kullakin kassalla on välitön pääsy järjestelmän yhteiseen tietokantaan. Tarvittava kaapelointi on minimaalinen, kun kaikki laitteet on liitetty suoraan yhteiseen koaksiaalikaapeliin.

Kassajärjestelmän ohjelmisto

Koska järjestelmän tietokoneiden tehtävät ovat hyvin erilaisia, niissä toimivat ohjelmat ovat pitkälle erikoistuneita. Näin suorituskky pysyy korkeana, ohjelmat ovat selkeän modulaarisia ja kapasiteetti kasvatettavissa mikrojen määrää lisäämällä.

Kassaohjaimen ohjelmaa kutsutaan myyntisovellukseksi. Sen tehtävänä on vastata yksinomaan kassojen ohjauksesta, myyntitiedon keräilystä ja talletuksesta.

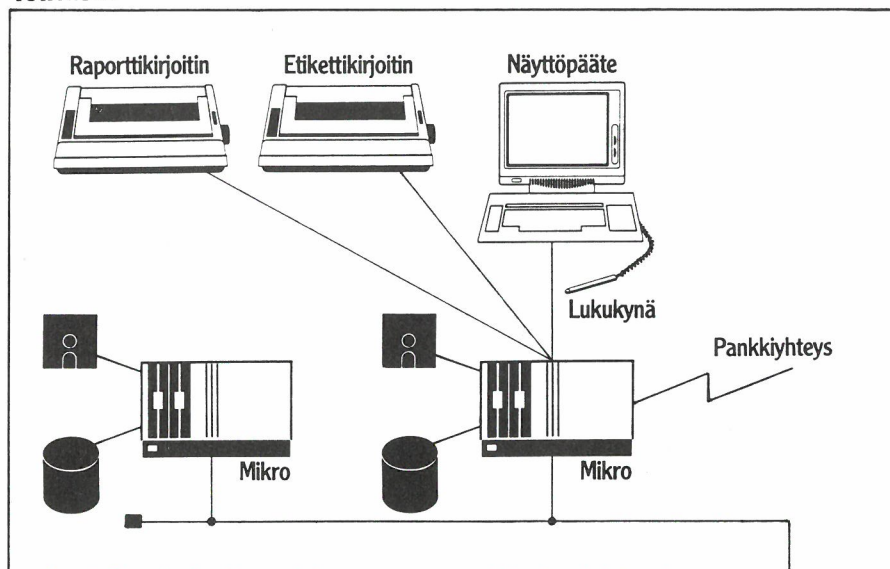
Järjestelmän pääkoneella suoritetaan tietokannan ylläpito, raportointi ja runsaasti laskentaa vaativat eräajotoiminnot. Ohjelmisto koostuu MS-DOS-ympäristössä toimivista sovellusohjelmista, jotka ladataan muistiin sitä mukaan kuin niitä tarvitaan. Pääkone voi siten huolehtia hyvinkin erilaisista tehtävistä päivän aikana. Siihen voidaan myös liittää modeemi ja muodostaa yksinkertainen tiedonsiirtoyhteys pankkeihin, korttiyhtiöihin tai kaupan keskusliikkeisiin.

MikroPOS-järjestelmän sovellusohjelmistot huolehtivat kaupan myyntitietojen yhdistelystä ja kertyneen tuoton laskennasta. Tiedot ovat luettavissa joko näyttöpäätteeltä tai raporteista. Näyttöpäätetyöskentelyä helpottavat muun muassa kenttäkohtaiset opasteet. Käteisen lisäksi ovat mahdollisia muut yleisimmät maksutavat. Ominaisuuksiin kuuluu esimerkiksi monipuolinen tili- ja maksukorttimyynti. Ohjelmistot on kehitetty verkko-ympäristöön MPSFORMS-näytönkäsitteily- ja MPSINDEX-tiedostonhallintatyökalujen avulla. Varusohjelmisto sisältää sanomanvälitysmekanismin, jolla eri koneissa sijaitsevat sovellukset voivat siirtää tietoa toisilleen synkronoidusti.

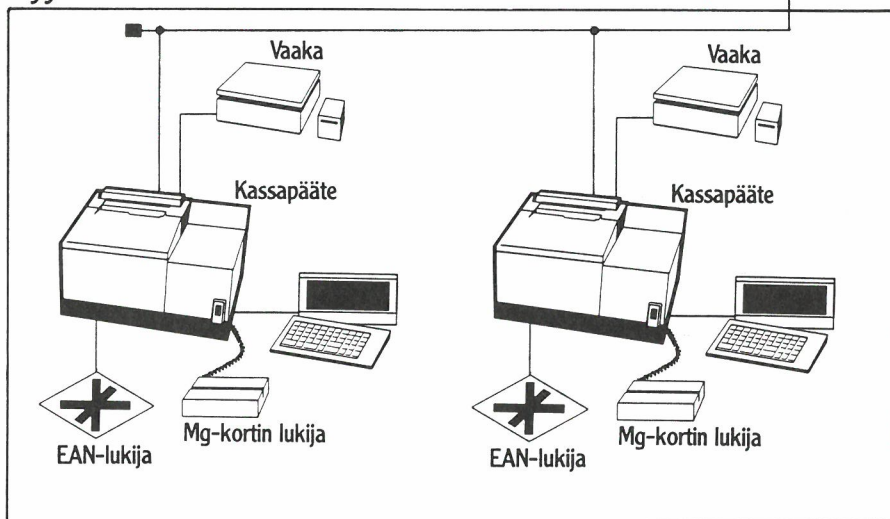
Toiminnan jatkuvuus tärkeintä

Myymäläjärjestelmien kaikkein tärkein vaatimus on toiminnan ehdoton keskeytymättömyys. Jotta taloudellisilta tappioilta vältytään, minkään yksittäisen osan vikaantuminen ei saa aiheuttaa merkittävää häiriötä myynnissä. Jatkuvuuden takaa-

Toimisto



Myymälä



miseksi onkin kaikki myyntitoiminnassa tarvittavat laiteresurssit vähintään kahdennettu.

Laitteiston kriittisin osa on kassaohjain, jonka häiriö saattaa vaikuttaa useaan kassapisteseen. Tällaisessa tilanteessa voidaan pääkone parissa minuutissa vaihtaa ohjelmallisesti kassaohjaimeksi. Kassapäätteiden toiminnot siirtyvät silloin sille. Mitään erityisiä kytkentätoimenpiteitä tai kaapeloinnin muuttamista ei tarvita.

Samaten on kahdennettu kaikki tärkeä järjestelmässä ylläpidettävä muuttuva tieto. Minikään yksittäisen muistilaitteen vikaantuminen ei voi aiheuttaa tallettuneen tiedon häviämistä.

Virtakatkosten varalta on yleensä pääkone, kassaohjain ja yksi kassa kytketty UPS-varavoi- malaitteeseen. Tällöin voidaan myymälässä katkoshetkellä olevat asiakkaat rahastaa ja sitten vasta pysäyttää tietokonejärjestelmä odottamaan sähkön palautumista.

Lisäominaisuuksia vaativallekin käyttäjälle

Väylärakenteisen kassajärjestelmän ominaisuudet ja koko ovat tarvittaessa laajennettavissa. Suuremmissa yli 10 kassapisteen kokoonpanoissa käytetään suorituskyyvyn takaamiseksi pääkoneen ohella kahta kassaohjainta. Näyttöpäätteellä varustetun työasemakoneen lisääminen mahdollistaa tietokannan ylläpitotoiminnot useammassa työpisteessä. Tarvittaessa saa samaan kaapeliin liitettyksi erillisen tietoliikenneohjaimen useille BSC3780- tai X.25-yhteyksille ja toimistokäyttöön tarkoitetun MikroMikko 3:n.

Kuhunkin kassapäätteeseen voidaan kytkeä viivakoodinlukija (lukukynä tai scanner), kassa-vaaka, magneettikortinlukija ja ylimääräinen asiakasnäyttö.

Ohjelmallisesti voidaan peruskassajärjestelmää laajentaa maksukorttien käsittely-, pankkiyhteys ja vaakaverkko-ohjelmistoilla. ■

Learning Ada

Part 1

Now we start learning Ada. In the next issues of Softari we will learn more.

```
with TEXT_IO;  
procedure HELLO is  
begin  
  TEXT_IO.PUT_LINE ("Hello world!");  
end HELLO;
```

Figure 1: Hello world program

Ada is a high-level language for system and application programming. It has been standardized by the International Organisation for standardization (ISO) and, today, compilers exist for all widely used computers, including micros. Ada is designed to support the programming of large systems that have a long lifespan. The readability and separate compilation facility of Ada program units promote program maintainability and the construction of reusable software components.

A very simple Ada program is shown in figure 1. This program is often to be found at the beginning of programming language text books.

Figure 1: Hello world program. Although simple, it illustrates some of the basic properties of the Ada language. Firstly, we can see that Ada is a high-level programming language that uses mnemonic names in its operations and identifiers. Here the reserved words are written in lower case letters, and identifiers in upper case letters.

This example also tells us something about the basic structure of an Ada program. The main program HELLO, uses the predefined input/output package TEXT-IO, which provides the operations for simple text I/O. Dependency is indicated by adding a "with clause" at the beginning of the main procedure. So, our program consists not only of the procedure HELLO, but also of the unit TEXT-IO (and all the units it possibly uses). TEXT-IO has earlier been compiled into the program library; and during the compilation of HELLO the compiler checks that the TEXT-IO operations are used correctly. Once the compilation has ended successfully, the procedure HELLO is also added into the library.

Overview of the language

To give an overall picture of Ada, here is a brief description of the main features of the language:

The Ada **package** implements a module structure with a clear interface to other parts of a program. A package contains logically related entities. For example, Ada input/output is implemented using packages. The **separate compilation** of packages with program library management enables several programmers to co-operate on a large program. Separate compilation also reduces compiling, and enables construction of software component libraries. Software components are often **generic** packages or subprograms, that can be regarded as templates with which program units can be parameterized. A data type, for instance, could be such a parameter.

Programmers may define their own data types in Ada. Ada supports **strong data typing**, which means that every variable and constant has a fixed data type that is determined during compilation. This eliminates run time errors caused by type conflicts. In Ada, other run time errors are called **exceptions**. A program can handle the exceptions, thus providing run time error recovery.

Ada **subprograms** and **statements** are very similar to those in Pascal. In addition to sequential operations, Ada supports parallel processing that is implemented through **tasks**. Finally, there are several **low-level features** available for system programmers: insertion of machine instructions, bit-level packing of data, and so on.

Package structure

The **package** structure can be said to be the most important Ada concept. A package is a collection of entities: variables, constants, data types subprograms and packages.

An example of a package could be one containing all the system's basic data types and constants. This package is among the first units to be compiled, and the other parts of the

program can use the data types and constants by including the package name in their "with clauses".

A more advanced package may implement an **abstract data type**. This kind of a package gives us a data type and its operations. As an example, consider the package in figures 2 a) and 2 b). In its **specification** the package declares a data type **TREE_TYPE** and the basic operations needed to construct and traverse a binary tree.

```
package BINARY_TREE is
    -- Visible part
    type TREE_TYPE is limited private;
    subtype CONTENTS_TYPE is STRING (1..10);
    procedure CREATE (
        TREE : in out TREE_TYPE;
        CONTENTS : in CONTENTS_TYPE);
    -- Creates a new root with no subtrees.

    procedure MOVE_LEFT (
        FROM : in TREE_TYPE;
        TO : in TREE_TYPE);
    -- Moves a tree or a subtree to be the left subtree of
    -- a given node.

    procedure MOVE_RIGHT (
        FROM : in TREE_TYPE;
        TO : in TREE_TYPE);
    -- Moves a tree or a subtree to be the right subtree of
    -- a given node.

    function PARENT (TREE : TREE_TYPE) return TREE_TYPE;
    -- Returns a link to the father of a given tree (provided
    -- that the father is not the root of the tree in which
    -- case null is returned).

    function LEFT_CHILD (TREE : TREE_TYPE) return TREE_TYPE;
    -- Returns a link to the left subtree.

    function RIGHT_CHILD (TREE : TREE_TYPE) return TREE_TYPE;
    -- Returns a link to the right subtree.

    function GET (TREE : TREE_TYPE) return CONTENTS_TYPE;
    -- Returns contents of a given node.

    function EMPTY (TREE : TREE_TYPE) return BOOLEAN;
    -- Returns value TRUE if a given node does not exist.

    TREE_ERROR : exception;
    -- Raised in MOVE_LEFT and MOVE_RIGHT if the target
    -- subtree is already present.

private
    -- Private part
    type NODE_TYPE;
    type TREE_TYPE is access NODE_TYPE;
end BINARY_TREE;
```

Figure 2.a: Binary tree package – specification

```
package body BINARY_TREE is

    type NODE_TYPE is
    record
        CONTENTS : CONTENTS_TYPE;
        PARENT : TREE_TYPE;
        LEFT : TREE_TYPE;
        RIGHT : TREE_TYPE;
    end record;

    procedure CUT_SUBTREE (
        SUBTREE : in TREE_TYPE) is
    -- The link between the subtree and its father is cut.
    begin
        if SUBTREE.PARENT /= null then
            if SUBTREE.PARENT.LEFT = SUBTREE then
                SUBTREE.PARENT.LEFT := null;
            elsif SUBTREE.PARENT.RIGHT = SUBTREE then
                SUBTREE.PARENT.RIGHT := null;
            end if;
        end if;
    end CUT_SUBTREE;

    procedure CREATE (
        TREE : in out TREE_TYPE;
        CONTENTS : in CONTENTS_TYPE) is
    begin
        TREE := new NODE_TYPE' (CONTENTS, null, null, null);
    end CREATE;

    procedure MOVE_LEFT (
        FROM : in TREE_TYPE;
        TO : in TREE_TYPE) is
    begin
        if TO.LEFT /= null then
            raise TREE_ERROR;
        end if;
        CUT_SUBTREE (FROM);
        FROM.PARENT := TO;
        TO.LEFT := FROM;
    end MOVE_LEFT;

    procedure MOVE_RIGHT (
        FROM : in TREE_TYPE;
        TO : in TREE_TYPE) is
    begin
        if TO.RIGHT /= null then
            raise TREE_ERROR;
        end if;
        CUT_SUBTREE (FROM);
        FROM.PARENT := TO;
        TO.RIGHT := FROM;
    end MOVE_RIGHT;

    function PARENT (TREE : TREE_TYPE) return TREE_TYPE is
    begin
        return TREE.PARENT;
    end PARENT;

    function LEFT_CHILD (TREE : TREE_TYPE) return TREE_TYPE is
    begin
        return TREE.LEFT;
    end LEFT_CHILD;

    function RIGHT_CHILD (TREE : TREE_TYPE) return TREE_TYPE is
    begin
        return TREE.RIGHT;
    end RIGHT_CHILD;

    function GET (TREE : TREE_TYPE) return CONTENTS_TYPE is
    begin
        return TREE.CONTENTS;
    end GET;

    function EMPTY (TREE : TREE_TYPE) return BOOLEAN is
    begin
        return TREE = null;
    end EMPTY;

end BINARY_TREE;
```

Figure 2.b: Binary tree package – body

The package specification and body are program units that are separately compiled. The specification must be compiled before the body, but the two need not be given to the compiler as consecutive units; several specifications may be compiled before their bodies. The bodies are not required before the program is linked. Thus Ada facilitates programming with specifications.

Outside the package `BINARY-TREE` all that can be used of its resources is written in the **visible part** of the specification. Declarations in the

private part of the specification or in the package body are completely hidden from the user.

Figure 3 shows a main program declaring the variable `THE_TREE`, which is of the type `BINARY-TREE.TREE_TYPE`. The visible procedures and functions of `BINARY-TREE` can be applied to this variable; but there is no way of using the fact that a tree is represented as a collection of certain kinds of records. All details of the actual implementation of the tree operations are also encapsulated in the package body. Given that these are correctly implemented, the structure is always in a valid state. The principle of information hiding is thus nicely practised in Ada.

It is important to bear in mind that the `BINARY-TREE` package does not offer a single data structure but a data type – one that can be used to declare several separate binary tree variables.

The data type entitled `BINARY-TREE.TREE_TYPE` can be used in the same way as any other data type (e.g., `INTEGER`), enabling the `BINARY-TREE` package to be regarded as a language extension that provides a new abstraction **binary tree**. It is an abstraction, because we do not need to know details of the internal structure and operations. And because it is defined as a private type, it also completely hides the details from us. Strong data typing in Ada may be demonstrated using this example by pointing out that the package declaration in figure 2 a) does not allow us to insert an `INTEGER` value in a tree node. For instance, if we call the procedure `BINARY-TREE.CREATE` with the parameters `MY-TREE` and 28, all we get is an error message from the compiler because the types of formal and actual parameters do not match.

The Ada compiler provides for the same degree of checking between program units as within a unit. For the user, strong data typing means more checks during compilation but an absence of data type errors at run time. Note: by defining a generic unit, the tree package could be parameterized using the type of data in a tree node.

Program structure

The evolution of the structure of a program can be depicted as in figure 4.

An Ada program is composed of compilation units compiled in the program library. A compilation unit is a package or subprogram, or the body of a separately compiled program unit (such as `INSERT` and `DISPLAY` in figure 3). When we want to link a program object, we give the linker the name of the main subprogram. The linker searches the library for the main program and retrieves it, along with the compilation units that depend on it according to the **with clauses**. All these units are linked to form an executable program. Thus, an Ada program can be represented as a directed graph with compilation units as nodes and **with** relations as edges – see figure 5. For example, the `SCANNER` and `CONSTRUCT` packages both use the `TEXT-IO` package in that program.

In compliance with rules of Ada, if we make a change in the **specification** of a package (say, `CONSTRUCT`) and compile it, we also must compile all those units that use the recompiled unit (`PARSER` in our example) before we are able to link a new program object. This guarantees that we can never link a program containing mutually incompatible parts. The price to be paid for this is that occasionally we have to recompile considerable parts of our program. In practice, this is not a nuisance if we use proper tools, those that automatically determine where recompilations are required and perhaps generate batch files that can carry out the compilations. It must be stressed once more that the compilation of a package body does not usually lead to recompilation.

To reveal the dynamics of an Ada program, let us see what happens when the program is started. Before the execution of the main program can start, memory space is allocated for the data contained in all the packages. Many initializations must also be done: variables declared in packages can be initialized, and packages may have initialization statements that are executed.

```
with TEXT_IO, BINARY_TREE;

procedure MAKE_TREE is
  THE_TREE: BINARY_TREE.TREE_TYPE;
  subtype WORD is BINARY_TREE.CONTENTS_TYPE;
  A_WORD: WORD;
  A_LINE: STRING (1..80) := (others=>' ');
  LENGTH: NATURAL;

  procedure INSERT (
    A_WORD      : in WORD;
    IN_THE_TREE : in BINARY_TREE.TREE_TYPE) is separate;
  -- Finds a place for A_WORD in an ordered tree. The procedure
  -- is compiled later, and is not given here.

  procedure DISPLAY (
    THE_TREE : in BINARY_TREE.TREE_TYPE) is separate;
  -- Prints out the contents of a tree using inorder.
  -- The procedure is compiled later.

begin
  TEXT_IO.GET_LINE (A_LINE, LENGTH);
  A_WORD := A_LINE (1..A_WORD'LAST);
  -- The root node:
  BINARY_TREE.CREATE (THE_TREE, A_WORD);
  A_LINE := (others=>' ');
  TEXT_IO.GET_LINE (LENGTH);
  while LENGTH /= 0 loop -- Carriage return ends the loop
    A_WORD := A_LINE (1..A_WORD'LAST);
    INSERT (A_WORD, THE_TREE);
    A_LINE := (others=>' ');
    TEXT_IO.GET_LINE (A_LINE, LENGTH);
  end loop;
  DISPLAY (THE_TREE);
end MAKE_TREE;
```

Figure 3: Main program using the `BINARY_TREE` package

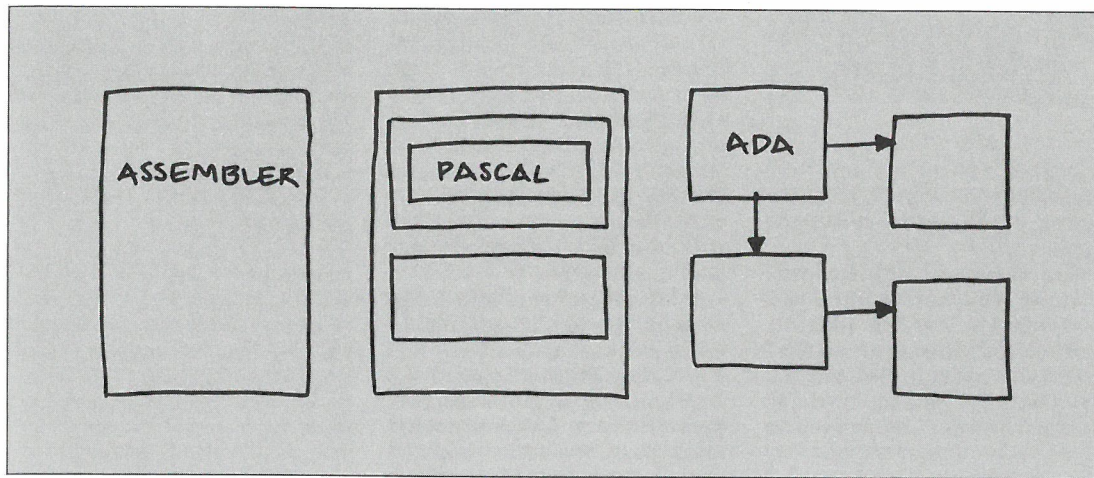


Figure 4: Evolution of program structure

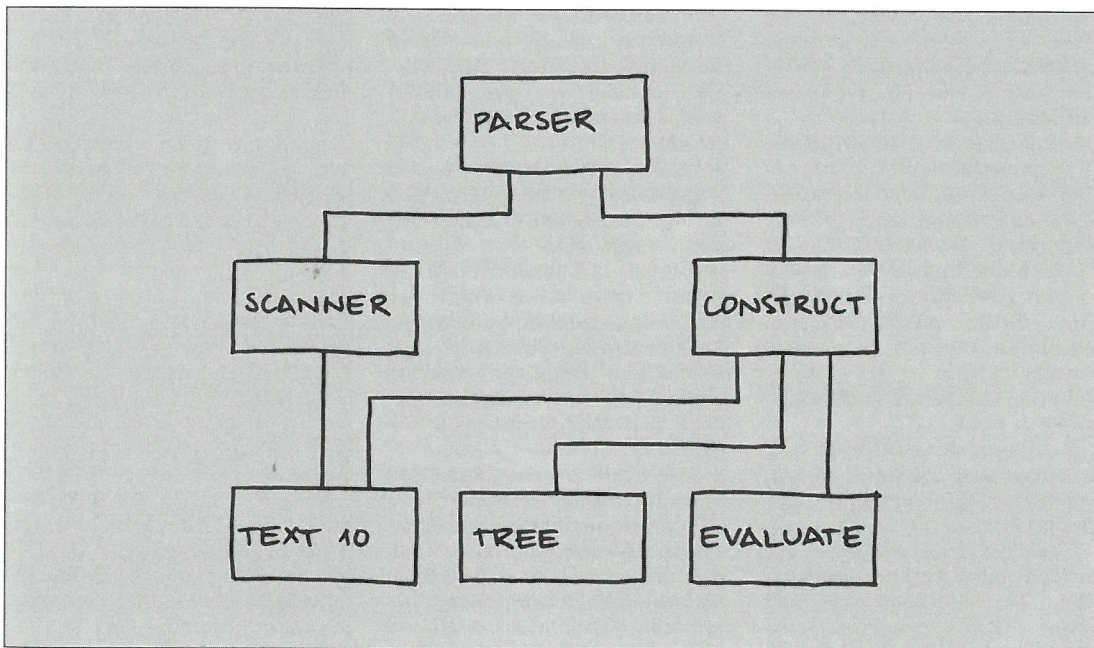


Figure 5: An Ada program as a tree

Once all this is completed, control passes to behind the **begin** in the main program and the program's statements are executed. From time to time, control is shifted to the subprograms in the package bodies. Execution of the program is completed when we reach the end of the main program.

Exercise

Modify the package specification in figure 2 a) to make it define a general tree type in which each node in a tree can have 0..n subtrees instead of 0..2 as in a binary tree. The solution will be published in the following issue of Softari. ■

References

- /1/ Reference Manual for the Ada programming Language, January 1983.
- /2/ Booch, G.: Software Engineering with Ada, Benjamin/Cummings 1986.
- /3/ Barnes, J.G.P.: Programming in Ada, Addison-Wesley, August 1983.

"Löylyä"

– Mekö! Mehän olemme niin mukavia ja luontevia, vastasi nopeasta hyökkäyksestä yllättynyt Tapsa.

– Jaa luontevia. Kuljette nenä pystyssä meidän tavallisten pääteikäyttäjien keskellä. Puhutte avaimista, fileistä ja jostakin helvetin sortista. Jos systeemi toimii väärin, se on aina käyttäjän toimintavirhe, ei koskaan teidän arvon Asiantuntijoiden syy.

– Mitäs töppäilette, ei me vaan, yritti Tapsa.

– Et ole tosissasi. Mitä ihmeellistä muuten niissä tietokoneissa oikein on? Olen katsellut sinuakin salaa, kun hivelet näyttöpäätettäsi, silittelet sitä kuin tyttöstävää, piirität sitä vuorokauden ympäri mistään työajoista välittämättä.

– Johtuisikohan se muun silitettävän puutteesta.

– Arvasin. Tyypillinen tietokonehullu sinä olet! Elämä juoksee ympärilläsi, ja sinä vain nuhjaat ohjelmiesi kimpussa. Oletko ikinä edes koskenut toista ihmistä? Koita tuosta, miltä käsivarren iho tuntuu, olet sen varmaan jo unohtanut.

– Tuntuu karvaisemmalta kuin päätteen pinta.

– Todella romanttista! Mikä niissä masiinoissa on niin ihanaa, neidän ovat niin rumiakin? Kerro minulle!

– Täydellisyys, tietokone ei tee virheitä eikä jankuta vastaan. Toimii juuri niin kuin vain minä pystyn sitä komentamaan. Kaikki on kiinni omasta älystäni, että saan jengat kohdalleen ohjelmakoodissa. Meillä menee hyvin. Kone ei nuri.

– Joo joo, pelkällä omalla älyllä. Viimeinen suomalainen käsityöammatti tämä teidän tulevaisuutteen ala on. Missään ei näe niin alkeellisia työtapoja, välineitä ja oloja kuin teillä. Yötä päivää rähjäätte, eikä valmista tule nimeksikään. Ihan tolkutonta porukkaa olette.

– Täytyy osaksi myöntää. En minä osaa kuitenkaan mitään muutaakaan hommaa. Ohjelman väentämisessä olen aika hyvä.

– Kehitä itseäsi. Opettele vaikka kirjoittamaan sillä päätteelläsi. Meidän talon atk-veijarit eivät ainakaan pistä hommistaan riviäkään paperille. Kaikki on vain omassa päässä. Kenellekään ei kerrota vahingossakaan, mitä tapahtuu. Palkkaansa sillä tavalla saa kyllä korotettua.

– Ei sitä tällä iällä pysty tapojaan muuttamaan.

– Pystyypäs, asiantuntija Saari-nen! Ja nyt aloitetaan kokeilemalla vaikka suutelemista!

– Uskaltaisikohan! Saari-nen alkoi hamuilla parempaa asentoa. Kesken kaiken kolisteli hölmistynyt Tasanen sisään hikisenä kantourakastaan. Aikansa kiroiltuaan muiden salaa katoamista päätti hankin tilata auton ja lähteä mukaan Erneston luo.

– Siinä oli kanssa yksi rotunne tyylipuhdas edustaja, totesi Leena oikoen samalla vaatteitaan. Tasanen luulee vielä, että nykyajan ihmistä voi perusteluita pompottaa ja asiakkaan määrätä ostamaan.

– Ottiko se niin koville, kun ei punatukkaista kirjanpitäjää huoltukaan matkaan mukaan, kiusasi Tasanen lisää.

– Otti, otti! Ymmärrettäväähän se on. Leena on ensiksikin naisen ja toiseksi osaa puhua sujuvasti englantia. Olisihan se ollut noloa, kun firman johto hänen nähtensä tilaa aina samaa ruokaa "This one" ja juomaksi "Same to me".

– Olisit silti voinut jättää sen härskin tempun tekemättä!

– Eihän se mikään tempu ollut. Pikkuisen vaan harmitti, kun paikallisten isäntien demon aikana koko ajan selkäni takana höpötettiin "kysy siltä että" ja "sano sille että". Sitten minä sain, kun kerran pyydettiin.

– Mitä sinä oikein laukaisit, innostui Mäkinenkin istumaan nousten.

– Sanoinpahan vaan sille hyvän näköiselle demotytölle, että takanasi seisova vaalea karju, atk-virtuosi Ketola Rauta-Yhtiöstä Suomesta haluaa kanssasi sänkyyn. Minä vain tulkkasin, kun pyydettiin.

– Ja isännät lopettivat koko touhun siihen paikkaan, muisteli Tasanen ääni vieläkin täristen. Eivät tulleet enää edes sovitulle

lounaalle. Hirveä moka. Onneksi raisut iltamenot paikkasivat teidän molempien tupeksinnat. Puhelimen pirinä keskeytti hyvin alkaneen löylytyksen. Tasanen palasi hetken kuluttua saunan puolelle kalpeana.

– Saatiinko kaikki potkut, ilveili Mäkinen.

– Melkein. Hallintojohtaja Reima Rauta-Yhtiöstä ilmoitti, että huomenna pitää järjestää neuvottelu ja saunailta. Muuten ei tule kuulemma kauppvoja, irvisti Tasanen.

– Huomenna! Vieläkin ollaan edellisestä sekaisin, murisi Saari-nen. Minä en tule mukaan!

– Tuletpas, tokaisi Tasanen, se kirjanpitäjäkin tulee! Nyt tämän palaverin luonne muuttuu. Kriitikki seis! Ryhdymme heti hiomaan huolella huomista iskua, me otamme sen kaupan. Nyt kaikki peliin, pojat!

– Niinkuin ei olisi jo, voivotteli Mäkinen ja taapersi seinä pidellen kylmään suihkuun. Saunailta firman vintillä sujui aluksi suunnitelmien mukaan. Paikalle taksisesti hälytetty sihteeri Ritu imi Ketolan huomion puoleensa täydellisesti. Asioiden käsittely eteni nopeasti, samoin tyhjenivät kaljakorit helteisenään vastapainoksi. Kun myyntitykki Mäkinen alkoi perinteiseen tapaan lausua omia loppusoinnuttomia runojaan mahtipontisesti painottaen, päätti Tasanen kätkeä oluet toimistohuoneeseen. Tällaisen kantokeikan aikana tekivät Ritu, Ketola, hallintojohtaja Reima ja Mäkinen harhautusliikkeen, tilasivat taksin ja lähtivät taksilla kohti Erneston mökkiä. Siellä ei juotava helposti loppunut. Väsynyt Saari-nen jäi kirjanpitäjä Leenan viereen sohvalle iltaa istumaan.

– Teitä tietokonemiehiä en ole koskaan oppinut ymmärtämään. Olen vuosia seurannut käyttäjän roolissa teidän touhujanne, oleteko te ihmisiä ollenkaan, aloitti Leena. ■

MikroMikko 3 TopTen

MikroMikko 3 TopTen perustuu nyt vuoden 1988 seitsemän ensimmäisen kuukauden myyntiluvuista laskettuihin suhteellisiin indekseihin.

TopTenin ehdoton ykkönen on Windows-ohjelmisto seurataan sitten yhden kuukauden tai koko alkuvuoden myyntiä. Yhtä varma veikkaus kakkossijan haltijaksi on WordPerfect-tekstinkäsittelyohjelmisto.

Pääte-emulaattorien kysyntä jatkuu vakaana.

Taulukkolaskenta on tuttu mikroyritysten käyttöalue. MultiPlanin nousu suosituimmaksi johtuu osaksi Nokia Datan koulumyynnistä. Se on perusohjelmisto, josta on hyvä aloittaa taulukkolaskennan opiskelu. Vaativimmista monipuolinen Excel on suosituin.

Designer-grafiikkaohjelman nousu listalle osoittaa, että mikroyritysten käyttö on laajenemassa myös kuvituksen puolelle. Julkaisuohjelmistoja listalla ei tosin vielä näy.

MikroMikko 3 TopTen

1. (1) Windows	116
2. (2) WordPerfect	41
3. (4) WS220-emulaattori	24
4. (-) MultiPlan	23
5. (6) WS7300-emulaattori	23
6. (3) Excel	22
7. (5) WS7800-emulaattori	13
8. (7) Lotus 1-2-3	11
9. (-) Symphony	10
10. (-) Designer	4

Suluissa sijoitus edellisessä Sof-tarissa 2/1988. Indeksien laskutapa on muuttunut. Nyt seurataan koko vuoden myyntiä yhden kuukauden myynnin sijasta.

ErgoWord

Nokia Data on tuonut markkinoille kehittämänsä suomenkielisen ErgoWord-ohjelmiston. Se on sekä tekstinkäsittelyohjelma että käyttöympäristö, joka voidaan ladata mikron käynnistykseen yhteydessä ja josta poistetaan vasta kun mikro suljetaan. Muut sovellukset voidaan käynnistää ErgoWordin ohjelmaliityntävalikosta.

Merkkipohjaisen ErgoWord-ohjelmiston suunnittelun lähtökohtana on ollut ergonomisuus. Ergonomisuus perustuu sekä selkeisiin valikoihin, tilannekohtaisiin ohjeteksteihin ja loogisiin näppäinkomentoihin että äidinkielen kieliseen kommunikointiin. ErgoWord on saatavissa myös ruotsin-, englannin- ja saksankielisenä. Ruotsinkielistä ErgoWordia on myyty jo noin 15 000 kappaletta.

ErgoWord perustuu asiakirjoihin ja niistä pidettäviin luetteloihin. Käsité asiakirja tarkoittaa ErgoWord-tekstien lisäksi muidenkin ohjelmien käyttämiä tiedostoja kuten Multiplan-taulukoita. Ohjelmiston luettelorakenne on kaksitasoinen. Pääluettelossa voi asiakirjojen lisäksi olla aliluetteloita, joihin on tallennettu aihepiiriltään samanlaiset asiakirjat. Asiakirjan ja aliluettelon tunnistamiseen on käytettävissä 50 merkkiä, 15 nimeä ja 35 avainsanaa. Lisäksi asiakirjaluettelosta näkyy suoraan asiakirjan sivumäärä ja päiväys, joka on käyttäjän vapaasti määriteltävissä.

ErgoWord on ainoa mikroissa toimiva tekstinkäsittelyohjelmisto, joka kykenee käsittelemään S2500-järjestelmän SAGA II -ohjelmistolla tuotettuja asiakirjoja.

ErgoWordiin saadaan lähiaikoina muunnosrutiinit asiakirjojen muuttamiseksi WordPerfect-muotoon ja takaisin. Näin voidaan ErgoWordilla tuotettujen asiakirjojen ulkoasua muuttaa vaikkapa Venturalla.

Lisätietoja:
Jari Petersen-Jessen,
p. (90) 5671.

Ohjelmistoja, mikroja ja tarvikkeita

Nokia Datalla on Mikromyymlä sekä Helsingissä että Tampereella. Helsingin myymälä sijaitsee Keskuskatu 7:ssä ja Tampereen myymälä Yliopistokatu 56:ssa. Myymälöistä saat kaikki Nokian hinnastossa olevat noin 100 valmisohjelmistoa joko heti mukaan tai noin kahden viikon toimitusajalla. Mikromyymlöiden hyllyistä löytyvät muun muassa Windows, WordPerfect, Excel ja MultiPlan.

Myymälöissä myydään kaikkia Nokian mikroja. Perusmallit ovat heti saatavissa. Muut mallit tilataan tehtaalta ja ne toimitetaan asiakkaalle sopimuksen mukaan. Myymälöissä myydään myös tietokonetarvikkeita, etupäässä värinauhoja ja levykkeitä.

Varsinainen Nokia Datan Tarvikemyynti sijaitsee Pitäjänmäellä, p. (90) 5671. Ohjelmistoja, MikroMikkoja ja tarvikkeita saat myös valtuutetuilta jälleenmyyjiltä kautta maan. Myymälät ottavat tilauksia vastaan myös numerossa (90) 180 71 ja (931) 509 111.

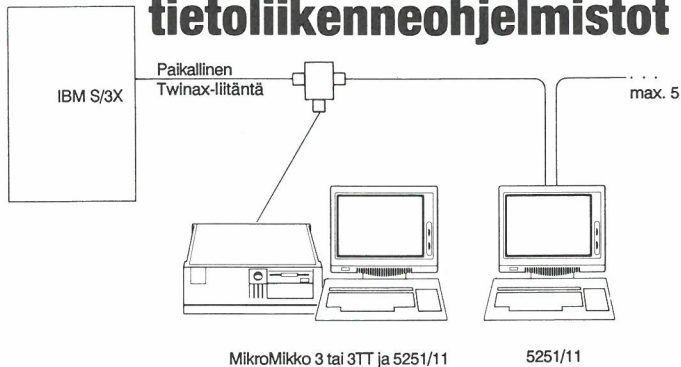
SCO ja AT&T

The Santa Cruz Operation, Inc. (SCO) ja AT&T ilmoittivat elokuun lopussa, että SCO on saanut lisenssin käyttää UNIX-tavaramerkkiä tulevaisuudessa julkaisemissaan 80386-pohjaisissa käyttöjärjestelmissä.

Seuraavan SCO:n julkistaman käyttöjärjestelmän nimi on SCO UNIX System V/38 versio 3.2. Käyttöjärjestelmä on testivalmis tämän vuoden lopulla ja myyntiin se tulee keväällä 1989. SCO UNIX System V/386 toimii kaikissa 386-pohjaisissa tietokoneissa, jotka on rakennettu teollisuusstandardin mukaiselle PC AT -arkkitehtuurille tai uudelle Micro Channel -arkkitehtuurille.

MikroMikkojen XENIX-käyttöjärjestelmän toimittaa SCO. MikroMikkoihin on kyseistä käyttöjärjestelmää noin puolesatoista vuodessa toimitettu jo noin 450 kappaletta. SCO UNIX System V/386 -käyttöjärjestelmän toimitukset alkavat Suomessa samaan aikaan maailmanlaajuisen jakelun kanssa.

5251/11- ja 5251/12- tietoliikenneohjelmistot



Saatavissa on nyt 5251/11- ja 5251/12-tietoliikenneohjelmistot. Kun MikroMikko 3 tai 3TT liitetään IBM System 34, 36 tai 38 -tietokoneeseen, ensiksi mainitulla ohjelmistolla MikroMikkoa voidaan käyttää kuten IBM 5251, 5291 tai 5292 -päättettä. Viimeksi mainitulla ohjelmistolla liitettäessä, MikroMikkoa voidaan taas käyttää niin kuin IBM 5251/12-päättettä. Mikron oheiskirjoittimia voidaan käyttää molemmissa tapauksissa kuten IBM 5219, 5224, 5225 tai 5256 -kirjoitinta.

5251/11 liitetään paikallisesti System 3X -koneeseen Twinax-kaapelilla ja 5251/12 liitetään synkronisten modeemien välityksellä. 5251/12 mahdollistaa myös suoran liittymän ilman modeemeja.

5251/11-toimitukseen kuuluu ohjelmiston lisäksi tietoliikennekortti ja T-liitin kortin liittämiseksi Twinax-kaapeliin. 5251/12 -ohjelmiston toimitukseen kuuluu myös tietoliikennekortti. Käsi- ja ohjekirjat seuraavat ohjelmiston mukana.

Molemmat ohjelmistot tukevat IBM PC Support ja FSU -ohjelmia. Lisäksi ne sisältävät oman tiedonsiirto-ohjelman. Niin sanotun hot-keyn avulla käyttäjä voi joustavasti siirtyä pääteistunnon MS-DOS-soveluksen välillä.

Lisätietoja:
Jouko Salo, p. (90) 5671.

Ascom IV

Ascom IV -tietoliikenneohjelmisto on tarkoitettu asynkroniseen tiedonsiirtoon. Kommunikointi tapahtuu MikroMikossa sarjaporttiin liitetyn modeemin tai korttimodeemin välityksellä. Myös suora yhteys toiseen tietokoneeseen on mahdollinen.

Tiedonsiirron yhteyskäytännöt ovat: ASCOM IV, BLOCK, BLOCKV, XMODEM, CRLF ja XONXOFF. Lisäksi tiedoston suora lähetys ja vastaanotto sekä merkkien ohjaus avattuun tiedostoon on mahdollista.

Ohjelmaa käytettäessä sekä lähtevien että vastaanotettujen merkkien muunnostaulukoita voi käyttäjä muokata vapaasti. Koska taulukot ovat 256-tavuisia, myös 8-bittisiä merkkejä voidaan muunnella tai poistaa. Valmiina ovat ASCII-EBCDIC ja EBCDIC ja EBCDIC-ASCII muunnostaulukot suurkoneyhteyksiä varten.

Ohjelmiston komentotiedostojen avulla voidaan monimutkaiset tiedonsiirtotehtävät käynnistää yhdellä komennolla. Tiedostot koostuvat ASCOM IV -komennosta. Ne voidaan luoda ohjelmiston omalla editorilla tai jollakin tekstinkäsittelyohjelmistolla.

Lisätietoja:
Jouko Salo, p. (90) 5671.

Graph Plus versio 1.1

Windows Graph 1.0 -grafiikkaohjelmasta on tullut uusi versio Graph Plus 1.1. Se sisältää uusia kaavioiden muotoja, piirto-ominaisuuksia, taustoja ja valmiita kuvia (General ClipArt). Se on yhteensopiva muun muassa Designer-, Draw- ja PageMaker-tiedostojen kanssa. Mukana toimittavan Excel-makron avulla saadaan DDE-linkki Excelistä Graph Plussaan. Lisäksi kirjoitinvalikoima on laajentunut.

Lisätietoja:
Jaakko Saarela, p. (90) 5671.

Development Kit versio 2.03

Microsoft Windows Development Kit versio 2.03 sisältää ohjelmankehitysohjelmia ja kirjastot Microsoft Windows 2 -ympäristössä toimivien ohjelmien tekoon. Uusia ominaisuuksia versiossa 2.03 ovat matematiikkasuorittimen tuki, dynaaminen tiedon linkitys eri sovellusten välillä (DDE) ja uudet grafiikkakuvot.

Software Development Kit sisältää RC-resurssikäntäjän ikonien, kursorien ja valintataulujen liittämiseksi ohjelmaan. Link4:n Windows ohjelmien linkitykseen. Symdeb-tutkintaohjelman ohjelmien tutkintaan ja Make-ylläpito-ohjelman ohjelmien kääntämisen sekä linkittämisen hallintaan. Käytettävissä on myös Iconedit-ikonieditori ikonien, osoittimien ja rastereiden luomiseen, Fontedit-fonttieditori uusien kirjaisimallien luomiseen ja muokkaukseen. Dialog valintataulujen muokkain ohjelman tarvitsemien valintataulujen tekoon ja Shaker-sekä Heapwalk-tutkintaohjelmat ohjelmien testaukseen.

Development Kit versio 2.03 tukee seuraavia kääntäjiä: Microsoft C v. 4.0, 5.0 ja 5.1, Microsoft Pascal 4.0 ja Masm 4.0.

Ohjelmistoon kuuluu useita esimerkkiohjelmia, joiden lähdekielinen ohjelmakoodi on mukana sekä hyödylliset SPY- ja SNAP-ohjelmat.

Lisätietoja:
Jari Luukkala, p. (90) 5671.

Excel verkossa

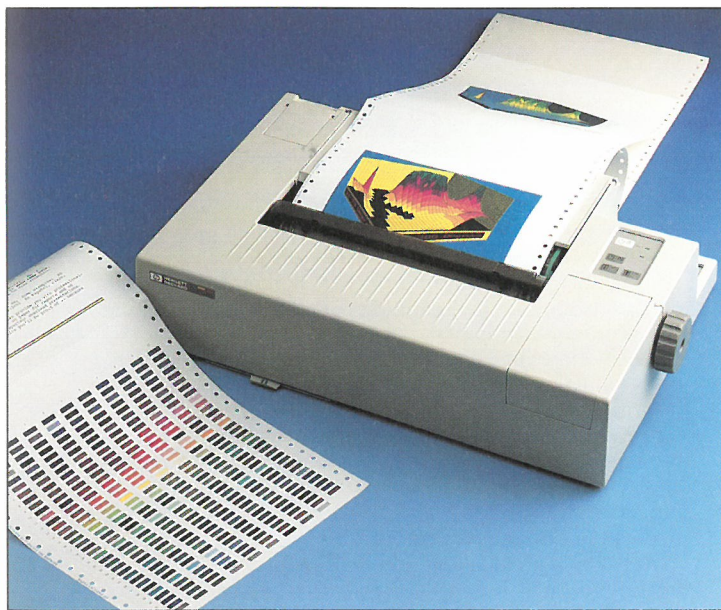
Excel-tilukkolaskentaohjelmistosta on nyt saatavissa verkko-versio 2.01.

Lisätietoja:
Sigfried Karlsson, p. (90) 5671.

Nokia F-1000 lasertulostin

Nokia F-1000 lasertulostin soveltuu tavanomaisen tulostuksen lisäksi julkaisu- ja grafiikkaohjelmien tulostukseen. Kirjasintyyppiä on vakiona HP-toimintatilassa 64 ja muissa tiloissa 34. Kirjoittimessa on kolme nk. dynaamista fonttia, joissa merkin koon ja muita parametrejä voi itse määritellä. IC-kortteja voi kirjoittimessa olla kaksi kerrallaan. Viivakoodityyppejä on 39.

Puskurimuistia on 512 kilotavua. Näin voidaan tulostaa noin kolmasosa sivua pistegrafiikkaa täydellä erotuskyvyllä. Erotuskyky on 300 pistettä tuumalla. Saatavissa on yhden megatavun laajennusmuisti, jolloin pistegrafiikkaa voi tulostaa sivullisen täydellä erotuskyvyllä.



PaintJet -väritulostin

PaintJet tulostaa tekstin lisäksi värigrafiikkaa joko piirtoheitin-kalvolle tai paperille. Erotuskyky on 180 pistettä tuumalla. Kirjasintyyppit ovat 12 pisteen Courier ja 12 sekä 8 pisteen Letter Gothic. Väreissä löytyy musta, punainen, keltainen, sininen, vihreä, cyan ja magenta.

ScanJet- kuvanlukija

ScanJet lukee ja tallentaa 300 pistettä tuumalla 20 sekunnissa. Suurin arkin koko lasilla on 216x297 millimetriä. Kuvat muunnetaan rasterigrafiikkatiedoiksi. Ennen tulostusta kuvaa voi korjata ja muunnella. ScanJetin mukana toimitetaan ohjelmisto, jota voi käyttää muun muassa PageMaker- ja Ventura-julkaisuohjelmistojen kanssa. Ohjelmiston avulla saa liitettyä julkaisuihin oikeankokoisia kaavioita ja kuvia oikeille paikoille.

SQLBaseen SQLWindows

SQLBase-tietokantajärjestelmään on nyt saatavissa SQLWindows-sovelluskehitin. SQLWindowsilla ohjelmia voi kehittää sovelluksia Microsoft Windows-ympäristöön ilman C-ohjelmointia tai Windowsin sovelluskehitysvälineitä.

SQLWindows-sovellukset toimivat kaikissa 80286- tai 80386-pohjaisissa mikroissa, joissa toimivat myös Windows/286 tai Windows/386-ohjelmat. Työasema voi olla liitettyä lähiverkkoon ja käyttää siinä olevan SQL-tietokantapalvelimen palveluja tai olla yksittäinen työasema, jossa on SQL-tietokanta tai ei.

SQLWindows sisältää graafisen lomakemuokkaimen, jolla tehdään lomakkeen layout. Siihen kuuluvat myös kentät, näppäimet, vieritysnäppäimet, valikot ja kuvat. SQLWindowsiin kuuluu korkeatasoinen sovelluslogiikan esittämiseen tarkoitettu muokkain. Koodia tehdään SQLWindowsin sisäisellä SAL-komentokielellä (SQLWindows Application Language). Siihen voidaan sisällyttää joitakin SQL-lauseita tietokantaan pääsemiseksi.

SQLWindows System Development Kit sisältää SQLWindowsin kehitys- ja ajonaikaisen version, SQR Report Writerin raporttien tekoon SQLTalkin vuorovaikutteiseen tietokannan käsittelyyn. Mukana on myös yhden käyttäjän SQLBase/16M - SQLBasen versio, joka toimii PC-DOS-järjestelmän 640 kilotavun muistin yläpuolella.

SQLWindows Application Design Module on saatavana myös erillisenä. SQLWindowsilla tehty sovellus voidaan kääntää ja "panna erillisjakeluun PC:lle saatavalla ajonaikaisella Runtime Execution moduulilla".

Lisätietoja:
Marko Nyroos, p. (90) 5671.

Englanninkielisen Gupta Technologies, Inc:n SQLBase- tai SQLWindows-esitteen voit tilata Softarin takasivulla olevalla palvelukortilla.

ColorPro-moni- väripiirturi

ColorPro-piirturilla tulostetaan piirtoheitin-kalvoille tai paperille monivärigrafiikkaa. Pienin osoitettava askelkoko on 0.025 millimetriä ja toistettavuus 0.1 millimetriä annetulla kynällä. Pyörötelineessä on kahdeksan kynää. Kynän ollessa ylhäällä nopeus on noin 52 senttiä sekunnissa ja alhaalla noin 40 senttiä sekunnissa.

00003 HELSINKI

VASTAUSLÄHETYS
Sopimus 00380/166
PL 780/10
Softari
Nokia Data

Nokia
maksaa
postitaksun

Softari 3/88
Palvelukortti

☐ Haluan esitteitä seuraavista
Nokian palveluista ja tuotteista.

☐ Pyydän, että minuun otetaan
yhteyttä.

puhelin: _____
Olen kiinnostunut _____

☐ Korjaan yllämainitut osoitetiedot.
☐ Poistan yllämainitut osoitetiedot
☐ Lisään allamainitut osoitetiedot
☐ Softarin ☐ NETin rekisteristä/rekisteriin

yritys _____
osasto _____
lähiosoite _____
postinro _____ postitoimip. _____
nimi _____

JOUKKOJULKAISU



8/00010

Softapäivitys

Tietotekniikka on tänään avaintekijä teknologia-, koulutus- ja työvoimapolitiisessa keskustelussa. Samoin kuin tieteenkin saavutukset ja oheistuotteet vaikuttavat jokapäiväiseen elämäämme, niin myös tietotekniikka on tullut, jollei kaikille kotiin asti, niin ainakin työelämään.

Mutta kuka puhuisi ja kertoisi meille tietotekniikasta meidän omalla kielellämme? En ainakaan minä. Minusta on hurjan kivaa pitää valta itselläni ja pistää tiedon sijasta vahinko kiertämään. Tai oli...

Ainakin niin kauan kuin kuvittelin, että tietämykseni ja viisauteni määrä oli suoraan verrannollinen puheessa käyttämäni koukeroiden määrään. Ainakin niin kauan kunnes ymmärsin, että tullakseni älykkäämmäksi tarvitsin seurakseni muitakin ihmisiä kuin meitä kapeakielisiä erikoissuperammattilaisia tai ihmisiä, jotka kunnioittivat meitä täysin tietämättä ja kysymättä itseltään miksi niin tekivät.

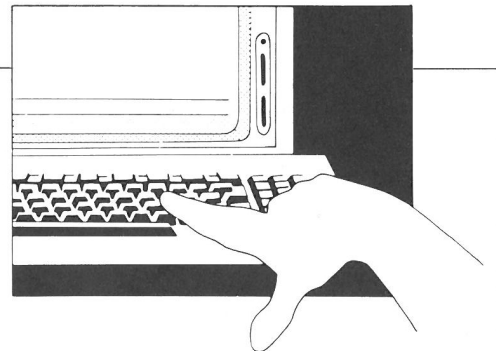
Oli hienoa puhua hienosti ja vaieta salaperäisesti. Mutta enemmän aivoja vaatii ilmaista itseään ymmärrettävästi.

Ymmärsin sen vasta, kun...

Olin vuosia ollut atk-alalla paiskien töitä yötä päivää burn outin rajoilla. Meni lujaa ja menestyin. Yhtäkkiä vaan huomasin kuinka tuttu toisensa jälkeen muuttui umpimieliseksi juntiksi, jonka kanssa oli mahdotonata keskustella mistään. Ainoat ymmärtäjät tuntuivat löytyvän naapuripäätteen ääreltä duunipaikalta.

Mutta kun vierustoverikin alkoi nostella liian tiuhaan tahtiin kulmakarvojaan minun puhuessani, niin pyysin vanhan hiekkalaatikokaverin kaljalle. Kale oli sähkömies. Vuosia sitten – pentuna – olimme ymmärtäneet toisiamme puolesta sanasta. Kommunikointivaikeudet olivat meille täyttä hebreaa. Jutteleminen auttaa, ajattelin – ja kerroin yksinkertaisista huolistani, jollaisten kanssa jokainen kamppailee silloin tällöin.

Minä kerroin, että...



Olin pitkät ajat prosessoinut muutamaa bittimäistä keisiä niin intensiivisesti, etten ollut saanut monena yönä huoltoajoproseduuria kunnolla käyntiin. Normaaliassa uni-sekvenssissä oli häiriöitä ja aamulla oli tosi lataamaton olo. Unet olivat olleet sentään niitä parhaimpia tapoja puhdistaa taustaprosessorista tarpeetonta roskaa.

Unettomuuden myötä oli tullut sellainen hantsi, että ellen pian saisi aivosoftaani händsheikkaamaan jonkun toisen prosessorin kanssa, keskusyksikköni sekoaisi täysin. Olin ajatellut kokeilla nauhojen ja talletusten purkamista vanhalle frendille, koska muut eivät junttiuttane enää minua ymmärtäneet.

Kerroin systeemit ystävälle softisti. Kerroin kuinka olin huomannut, että minulla olisi nyt softapäivitys paikallaan. Jutustelin siitä kuinka raskasta oli modifioida kokoajan uudelleen ja tallentaa vanha minä versionumeroineen enää minua tsekkausta varten.

Muistelin ääneen aikaa jolloin olin leikisti ja kontrolloidusti pyörittänyt toisen ihmisen softaa ja simuloinut joitain ajattelumalleja. Kerroin, että nyt kyseessä oli emulointi, koskapa olin tavannut sopivan tyypin siihen tarkoitukseen. Kyseessä oli ehkäpä rakastuminenkin. Ajoin nyt itselleni kovaa vauhtia uutta softaa.

Selitin ystävälleni tuskasta, joka syntyy, kun huomaa että seiskyt pros kapasiteetista menee bugien metsästyseen. Tiedäthän, softavirheiden...

Kaksi tuntia ja kuusi kaljaa meni ennenkuin frendi pääsi hämmennyneenä kommentoimaan. Kale lausui vakavasti, ettei ollut oikein saanut selvää mikä masiina minulla oli rikki. Mutta hän tulisi mielellään sitä kotiini katsomaan ja korjaamaan sen minkä tsägarin ammattitaidollaan pystyi.

Silloin ymmärsin itku kurkussa ja kaikkeni vuodattaneena, että ehkä ehkä ehkä minun kannattaisi muistella kuinka ihanaa oli olla ihan tavallinen juntti.

Softaaja