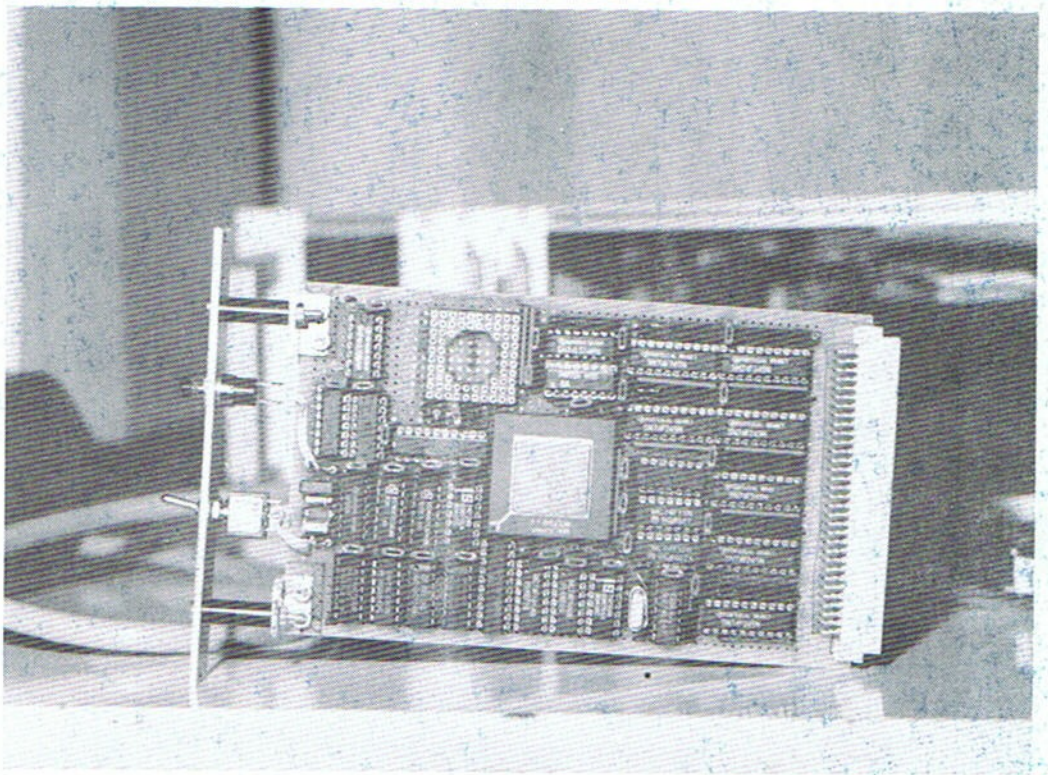
De µP Kenner

Vijftiende jaargang nr. 2
April 1991
71



In dit nummer o.a.:

Virusbestrijding
KGN-68k project
PATH voor DOS65
Computerkretologie
PC-serie: Super VGA en EISA

Inhoudsopgave

De μP Kenner

Nummer 71, april 1991
Verschijnt 5 maal per jaar
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:

Gert van Opbroek
Bram de Bruine
Antoine Megens
Nico de Vries
Joost Voorhaar

Eindredactie:

Gert van Opbroek

Vormgeving:

Joost Voorhaar
Nico de Vries

Redactieadres:

Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

De μP Kenner nummer 72 verschijnt op 15 juni 1991.

Kopijsluitingsdatum voor nummer 72 is vastgesteld op 1 juni 1991.

Algemeen

Vereniging

Talen/Software

Systemen

Hardware

Datacommunicatie

De μP Kenner is het huisorgaan van de KIM gebruikersclub Nederland en wordt bij verschijnen gratis toegezonden aan alle leden van de club. De μP Kenner verschijnt vijf maal per jaar, in principe op de derde zaterdag van de maanden februari, april, augustus, oktober en december.

Kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze kopij kan op papier, maar liever in machine-leesbare vorm opgestuurd worden aan het redactieadres. Kopij kan ook op het Bulletin Board van de vereniging gepost worden in de redactie area.
Nadere informatie kan bij het redactieadres of via het bulletin board opgevraagd worden.

Redactioneel

Voor veel mensen is april/mei zo'n beetje het einde van het computerseizoen. Het weer geeft dan weer aanleiding tot allerlei buitenactiviteiten en het is dus logisch dat de computer-hobby op een wat lager pitje gezet wordt. Dit geldt helaas ook bijna altijd voor de clubbijeenkomst in mei. Meestal is dit de bijeenkomst die door de minste mensen bezocht wordt. Ik vindt dat heel jammer. In de eerste plaats is het natuurlijk jammer dat je een bijeenkomst organiseert die maar door een paar mensen bezocht wordt. Verder is het ook voor de vrijwilligers die de ruimte beheren jammer dat ze hun vrije zaterdag opofferen voor maar een handjevol KGN-leden. Kortom komt in grote getale en laat u niet kisten door het (voor de computerhobby) slechte weer met veel zon en zomerse temperaturen.

Voor een aantal mensen breekt dit jaar in mei niet een computervakantie aan. Dit betreft onder andere de redactie. We gaan namelijk dit jaar weer een zesde nummer van de μP Kenner uitbrengen; sterker nog, we gaan eind juni een 68k en Minix-special uitbrengen. Dit nummer zal dus als zwaartepunt de processoren uit de Motorola 68000-lijn en het operating systeem Minix hebben. U kunt deze special ook zien als het begin van de verspreiding van kennis uit het KGN-68k project. Mocht u zelf ook bijdragen voor het blad hebben over 680x0, Minix, of Unix, dan ontvang ik dat graag uiterlijk eind mei van u. Dit brengt me meteen bij de tweede groep mensen die in de zomer doorwerken: De leden van de projectgroep KGN-68k. Deze mensen doen er alles aan om in november met de presentatie van het project te kunnen komen. U kunt, zoals gewoonlijk, in dit blad hun rapportage lezen.

Verder is er een tweede projectgroep van start gegaan. Eén enkele vraag van Antoine Megens heeft heel wat in gang gezet. Het ziet er echt naar uit dat ook DOS-65 nieuw leven ingeblazen zal gaan worden. Er heeft zich rond onze nieuwe software-coördinator voor DOS-65 (Frank Bens) een groep mensen verzameld die met DOS-65 versie 3 aan de slag gaan. Ik neem aan dat we ook van deze groep nog veel zullen horen. Verder gaf ik al even aan dat we een nieuwe DOS-65 coördinator hebben. Frank Bens, die zelf nog zeer actief met DOS-65 is, bood aan de coördinatie van DOS-65 op zich te nemen. Binnen het bestuur hebben we dit aanbod in dank aanvaard en Frank heeft alle DOS-65 spullen die we binnen de club hadden gekregen. Wat hij echter zeker niet gekregen zal hebben, zijn alle spullen die niet officieel bij de redactie, het bulletin board of de vorige software coördinator zijn ingeleverd. Vandaar ook zijn oproep in dit blad om gezamelijk de software-bibliotheek voor DOS-65 bij te gaan werken zodat de nieuwste versies van de programma's altijd via de club beschikbaar zijn. Als deze bibliotheek weer bijgewerkt is, zullen er lijsten verspreid gaan worden met een overzicht van wat er in de bibliotheek zit. Leden van de KGN kunnen dan tegen vergoeding van kopieer- en verzendkosten (fl. 7,50 per floppy) over deze software beschikken. Voor manuals geldt in principe hetzelfde. De kosten van de manuals zullen afgeleid zijn van de kopieer- en verzendkosten. Voor de verdere gang van zaken m.b.t. bestellen en afrekenen zullen we de komende weken procedures opstellen. U hoort hier nog van.

Ik heb net al even aangegeven dat kopij voor de komende 68k/Minix special van harte welkom is. Hetzelfde geldt echter ook voor de komende nummers van de μP Kenner. We kunnen namelijk zo langzamerhand wel weer wat kopij gebruiken. Met name hardware-onderwerpen zouden zeer goed van pas komen. Ook programma's in welke taal dan ook zijn van harte welkom. Uiteraard moeten programma's niet te lang zijn; we willen namelijk niet dat het grootste deel van het blad gevuld wordt met slechts één programmalisting. Als u een programma instuurt, doe er dan ook een kleine beschrijving bij. Als voorbeeld kunt u eens kijken naar het PATH-programma van Henk Speksnijder die in deze μP Kenner staat. Een dergelijke beschrijving + programma-listing kan door de redactie zeer goed verwerkt worden. Verder willen we de bijdragen zoveel mogelijk in magnetische vorm hebben. We kunnen zo langzamerhand alles wel verwerken wat gangbaar is, als we maar weten wat het is. Kortom als u een floppy opstuurt, schrijf er dan even bij in welk formaat en voor welk systeem het bedoeld is. Het gemakkelijkste te verwerken voor de redactie is MS-DOS formaat 5.25" (360 kb) en 3.5" (720 kb). Als u tekeningen of schema's inlevert, dan mag dat op papier en anders bij voorkeur in het zogenaamde HPGL-formaat. Vooral bij OrCad komt het wel eens voor dat we een schema krijgen en dat we de bijbehorende bibliotheken niet hebben. Als u een tekening in wilt leveren en u kunt geen HPGL-formaat maken, neem dan even contact op met de redactie; we kunnen dan bekijken wat de beste manier is om de tekening in te leveren.

Tenslotte wens ik u uiteraard weer veel plezier bij uw computer-hobby en tot ziens op de bijeenkomst in Almere of tot de volgende uitgave van de μP Kenner.

Gert van Opbroek

Uitnodiging voor de clubbijeenkomst

Datum: Zaterdag 25 mei 1991

Attentie

In verband met Pinksteren is de datum van de bijeenkomst verplaatst naar de vierde zaterdag in mei.

Locatie: Buurtcentrum "De Inloop"
's-Hertogenboschplein 8
1324 WB Almere-Stad
Telefoon 03240-33737
Entree: fl. 10,--
Thema: Operating systems: OS-9/68k

Routebeschrijving

Per auto

Vanaf de A6 neemt u de afslag Almere-Stad. Direct daarna afslag Gooise kant (Paralelweg A6). Dan linksaf de Havendreef, rechts aanhouden, de Stedendreef op. Doorrijden tot het benzinepompstation Esso, dan rechtsaf de Rotterdamweg op. Deze weg volgen tot aan het water (voor de busbaan) daar rechtsaf. Doorrijden tot het winkelcentrum met het buurtcentrum De Inloop.

Per openbaar vervoer:

Met de trein rijdt u naar het station Almere-Muziekwijk. Vervolgens kunt u met de stadsbus (lijn 12) tot voor De Inloop vervoerd worden.

Programma

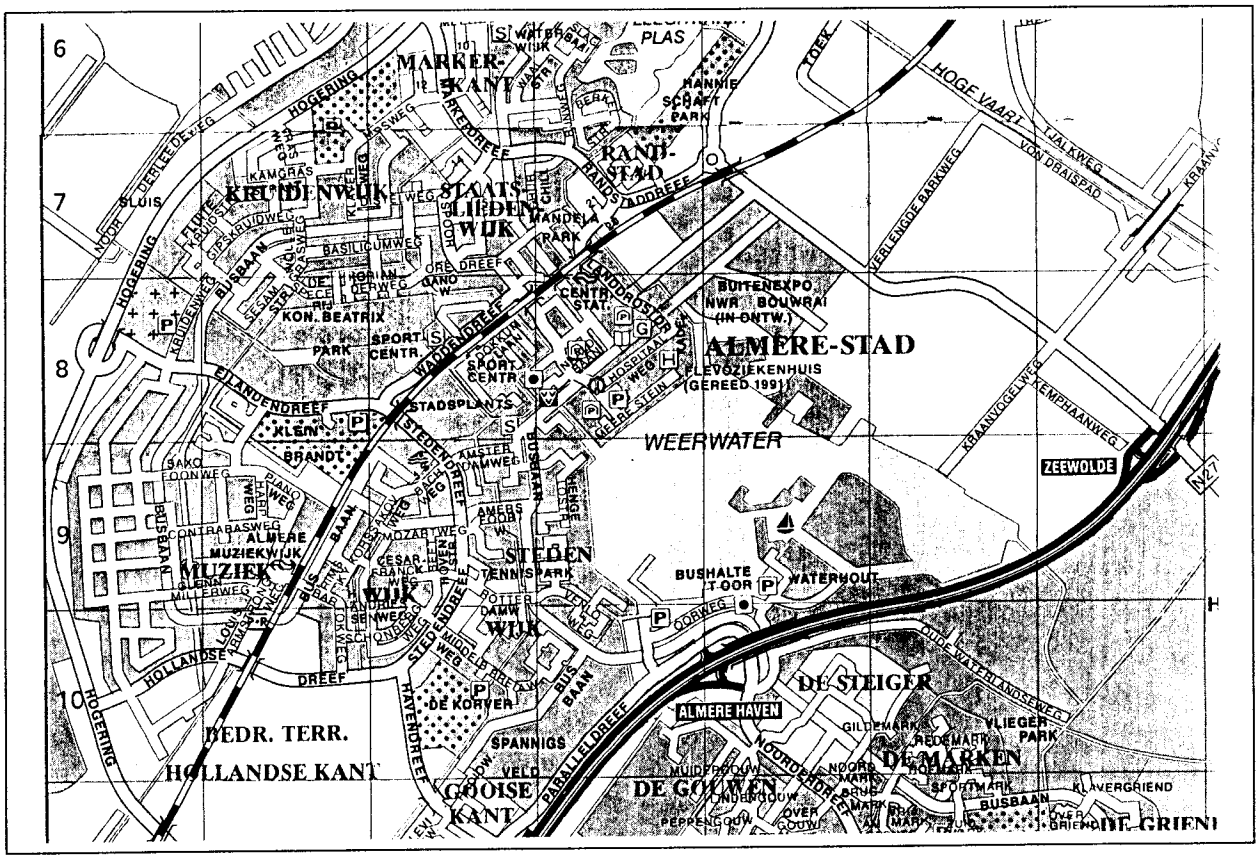
10:00	Zaal open
10:30	Opening van de clubbijeenkomst
10:45	Voordracht door Geert Stappers (Snijder Micro Systems) over het operating system OS-9/68k
11:30	Verslaggeving lopende projecten
12:00	Forum en markt
12:30	Lunch; consumpties tegen betaling

Na de lunch het informele gedeelte waarin er uitgebreid gelegenheid is met mede clubleden van gedachten te wisselen over hardware software etc. en om Public Domain software uit te wisselen. Op deze bijeenkomst zullen systemen draaiend onder OS9-/68k, MINIX, MS-DOS en uiteraard DOS-65 te bewonderen zijn.

17:00 Sluiting

Let op

Het is ten strengste verboden illegale kopieën van software te verspreiden. Aan personen die deze regel overtreden zal de verdere toegang tot de bijeenkomst ontzegd worden. Breng verder alleen software mee die u legaal in uw bezit heeft. Het bestuur aanvaardt geen enkele aansprakelijkheid voor de gevolgen van het in bezit hebben van illegale software.



Het standaardoperating system UNIX (Deel 4)

Vorige keer hebben we een eerste blik geslagen op de shell. Vandaag ga ik dieper in op de, in vergelijking met MS-DOS, oneindig krachtig lijkende mogelijkheden van de UNIX shell.

Control flow

De shell biedt een aantal mogelijkheden voor het afwijken van de "normale" top-down command sequence. We beginnen bij het begin: de for-do lus. Er zijn twee verschillende mogelijkheden voor het opzetten van een for-do lus. De eerste maakt gebruik van de parameter list. Stel dat we in een password file willen zoeken naar namen. Met de grep-utility kunnen we maar op één expressie tegelijkertijd zoeken, dus moet grep voor iedere naam opnieuw gestart worden. We besluiten een eenvoudig shellscript te maken met de naam "zoek". Het shellscript ziet er als volgt uit:

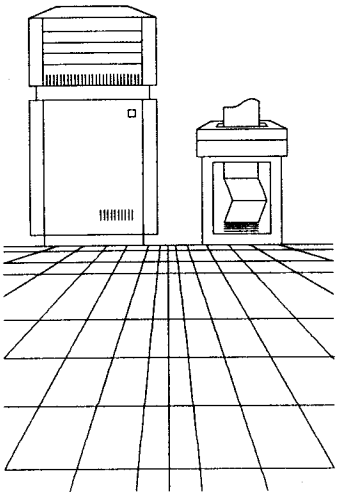
```
for i
do
        grep "^$i:" /etc/passwd
done
```

Ons simpele shellscript neemt voor i de opgegeven parameters. Voor iedere parameter wordt nu één maal grep opgestart. De expressie interpretor van grep verwerkt zogenaamde "regular expressions". In een regular expression geeft de "^" het begin aan van een regel; grep zoekt dus naar de naam van de persoon aan het begin van de regel en gevolgd door een dubbele punt. Immers: de password file van UNIX bestaat uit regels van de vorm: <naam>:<passwd>:<uid>:<gid>:<commen t>:<homedir>:<shell>.

De tweede vorm van de for-do lus zal duidelijker zijn (persoonlijk vond ik de voorgaande fo-do constructie nogal inconsequent overkomen). De algemene vorm van de for-do lus heeft de vorm:

for *name* **in** *w1 w2 ...*
do *command-list*
done

Als het gedeelte "in ..." mist, dan wordt dus uitgegaan van de parameters zoals die aan het shellscript opgegeven zijn. Staat er wel een "in ..." gedeelte, dan wordt voor de shellvariabele "name" de afzonderlijke elementen zoals genoemd achter "in" genomen. Bijvoorbeeld:



```
for i in *
do
        echo -n $i
        wc $i
done
```

Dit shellscript neemt voor i alle files in de current directory (de wildcard specificatie asterix!) en drukt voor iedere file de naam af, gevolgd door het aantal characters, woorden en regels in de betreffende file.

Intermezzo

Voorwaardelijke statements zijn ook mogelijk in shellscripts. Om een voorwaardelijk statement te kunnen bouwen is uiteraard een voorwaarde nodig. Die voorwaarde zou in een shellscript bijvoorbeeld kunnen zijn: "als het een directory is". Alvorens nader in te gaan op de werkelijke voorwaardelijke statements wilde ik het eerst gaan hebben over een aantal standaard UNIX-utilities die slechts een booleaanse waarde teruggeven.

Dé utilities die het meest gebruikt worden als voorwaarde zijn grep en test. Grep heeft een speciaal vlaggetje ("-s") dat aangeeft dat het resultaat van grep niet de regel(s) is/zijn die gevonden worden, maar de boolean waarde "TRUE" als er inderdaad een regel is gevonden die aan de voorwaarde voldoet en "FALSE" als er geen regel(s) gevonden zijn die aan de voorwaarde voldoen.

De test-utility staat toe files en/of strings te bekijken en te beoordelen. Aan de hand van verschillende vlaggetjes verandert de werking van test. De meest gebruikelijke vlaggetjes worden hieronder weergegeven:

test *s*	TRUE als het argument "s" niet de nul-string is
test *-f file*	TRUE als "file" bestaat
test *-r file*	TRUE als de gebruiker read-access op "file" heeft
test *-w file*	TRUE als de gebruiker write-access op "file" heeft
test *-d file*	TRUE als "file" een directory is

Daarnaast kunnen de meeste versies van test ook nog vergelijkingen maken tussen strings, en kunnen zelfs rekenkundige expressies aan! Het uitgebreid toelichten van de test-utility valt een beetje buiten het bestek van dit artikel. Voor de exacte syntaxis en de mogelijkheden van test verwijs ik dan ook naar de manuals en de verschillende UNIX-boeken.

Control flow, gecontinueerd

Het meest elementaire voorwaardelijke statement is uiteraard het if-statement. Het if-statement heeft in UNIX de algemene vorm:

```
if command-list
then command-list
else command-list
fi
```

Het else-gedeelte is hierin optioneel. De statements achter "**then**" worden alleen uitgevoerd als de command-list achter "**if**" TRUE oplevert. Is dat niet het geval, dan wordt de command-list achter "**else**" uitgevoerd, mist aanwezig. Een voorbeeld:

```
if test -f a.out
then    echo -n 'a.out already exists. Overwrite [y/n] ? '
        read yesno
else    yesno=y
fi
```

Het kan voorkomen dat een aantal if-statements in elkaars "else" gedeelte voorkomen. Zoals bijvoorbeeld in:

```
if test $# != 1
then echo 'Usage: blabla'
else
        if test -f $1
        then cc $1
        else    if test -d $1
                then echo $1' is een directory!'
                fi
        fi
fi
```

Het veelvuldig voorkomen van "else if" kan ingekort worden tot één commando: elif. Bovenstaande voorbeeld wordt dan:

```
if test $# != 1
then echo 'Usage: blabla'
elif    test -f $1
        then cc $1
        elif test -d $1
        then echo $1' is een directory!'
fi
```

Hele lange if-then-else statements die steeds testen of een variabele een bepaalde waarde heeft, en al naar gelang de waarde een andere actie moeten ondernemen zijn ook niet handig. UNIX heeft daarvoor de beschikking over een case-statement. Veel voorkomend is bijvoorbeeld het testen van shellparameters. Van een shutdown-script is de syntax bijvoorbeeld: "shutdown [-now]". De vierkante haken geven aan dat het deel "-now" optioneel is. Het stukje code om de syntax van shutdown te testen ziet er als volgt uit:

```
case $# in
   0) ;;  # Do nothing. User just types "shutdown"!
   1) if test $1 != -now
      then echo 'Usage: shutdown [-now]'; exit 1
      fi;;
   *) echo 'Usage: shutdown [-now]'; exit 1;;
esac
```

De algemene vorm van het case statement is:

```
case varname in
   pattern) command list ;;
   ...
esac
```

Daar de asterix als wildcard character aan alle strings voldoet, kan dit gebruikt worden als "default" optie.

Voorwaardelijke lussen kent UNIX ook. Maast de for-do lus kent de shell ook het while-statement. De algemene vorm van dit statement is:

```
while command-list
do command-list
done
```

Zolang de command-list achter while de waarde TRUE oplevert wordt de command list tussen "do" en "done" uitgevoerd. Er is ook een versie van de while-list waarbij het eerste commando juist FALSE moet opleveren. In plaats van "while" wordt dan "until" gebruikt. Voor pascal-programmeurs levert dit nogal eens verwarring op, omdat de UNTIL-lus in pascal impliceert dat de command-list van een UNTIL-lus tenminste één maal uitgevoerd wordt. In UNIX is niet alleen de syntaxis afwijkend, de betekenis is ook geheel anders!

Here-documents

Als men een file heeft die automatisch gegenereerd wordt, maar waar nogal wat extra rommel in zit die het ding onbruikbaar maken, moet in de meeste operating systems eerst met de hand of met een speciaal programma bijgewerkt worden. Meestal resulteert dit in het maken van een superdeluxe macro die heel wat zweetdruppeltjes kost. Om vervolgens nog steeds met de hand de file in een editor in te lezen en dan handmatig de macro op te starten. UNIX heeft een heel elegante oplossing voor dit probleem: het here-document. Met een zogenaamd here-document kan een gedeelte van de scriptfile gebruikt worden als standaard invoer voor een bepaald programma. Stel dat men een rij met namen en telefoonnummers heeft waarvan men automatisch met behulp van een modem wil kunnen bellen. We gaan er even vanuit dat het modem zich op tty1 bevindt. Het "belscript" zou er als volgt uit kunnen zien:

```
for i
do number=`grep $i << !
Joost:30142
Harry:75262
Joke:62721
!`
if test -z "$nummer"
then echo $i': niet gevonden!'
else    echo ATDT`echo $nummer | sed 's/.*://
        '"',;' > /dev/tty1
        echo 'Pak de telefoon op a.u.b.!'
        sleep 2
        echo ATH > /dev/tty1
        echo -n 'Press <ENTER> to continue...'
        read junk
fi
```

Er komen wat meer nieuwe dingen bij dan eigenlijk de bedoeling was, maar ja. Het shellscript begint met een bekende: voor alle parameters wordt de hele zaak één keer uitgevoerd. Vervolgens wordt de variabele "nummer" met het resultaat van grep gevuld. Die omgekeerde apostrophe (de `) geeft aan dat het volgende geintepreteerd moet worden als een commando. Het resultaat van dat commando wordt dan in de variabele gestopt. Het commando moet ook weer afgesloten worden met een `.

Het gedeelte tussen de twee kleiner-dan tekens en het uitroepteken en het uitroepteken dat een paar regels naar beneden staat wordt nu niet als commando opgevat, maar als data. De constructie met twee kleiner-dan tekens heet een "here-document". Achter de kleiner-dan tekens volgt een string waarmee later aangegeven zal worden dat het here-document is beëindigd. Op deze manier laat zich ook een (regel-) editor zoals "ed" besturen. In ed heeft het commando om een string te vervangen door een andere de vorm "(1,$)s/string1/string2/g". De editor kan vanuit een shellscript gestuurd worden door middel van een here-document:

```
ed file << %
(1,$)s/$1/$2/g
w
q
%
```

Terug naar ons bel-script. De test kijkt of de lengte van de opgegeven string gelijk is aan 0. Als dat zo is, dan is blijkbaar geen persoon met de juiste naam gevonden en dus... een foutmelding. In het andere geval moet het telefoonnummer geïsoleerd worden en gebeld. Dat gebeurt met behulp van een echo naar het modem in de vorm "ATDTnummer,;". Het nummer wordt geïsoleerd door de stream-editor. Alweer zo'n typische UNIX-utility. Editeren op een stream! Ideaal, maar wel een beetje verwarrend zo nu en dan. De gebruiker krijgt na het bellen 2 seconden de tijd om te telefoon op te nemen zodat het modem weer opgehangen kan worden.

Al met al zijn we inmiddels al vrij diep aan het graven gegaan in UNIX. Tijd om eens te kijken naar de AutoExec.Bat van UNIX: .profile. Als in de homedirectory van een user een file staat met de naam ".profile" wordt aangenomen dat dit een shellscript is en de commando's die er in staan worden netjes uitgevoerd. Normaliter worden er in de .profile een aantal variabelen gezet die later van belang zijn. Helaas, als een shellscript afgelopen is, worden de daarin aangemaakte variabelen weggegooid en oude waarden van vóór het shellscript worden weer hersteld. Om nu toch variabelen te kunnen zetten, is er een speciaal commando: export. Export wordt gevolgd door de namen van de variabelen die geëxporteerd moeten worden naar de shell. Bijvoorbeeld: "export HOME MAIL" zorgt ervoor dat nadat het shellscript afgelopen is de waarden $HOME en $MAIL gelijk zijn aan die, die in het script gezet zijn.

Volgende keer meer over UNIX voor super-users. Ik ga dan dieper in op beveiliging, onderhoud en het trimmen van een UNIX systeem.

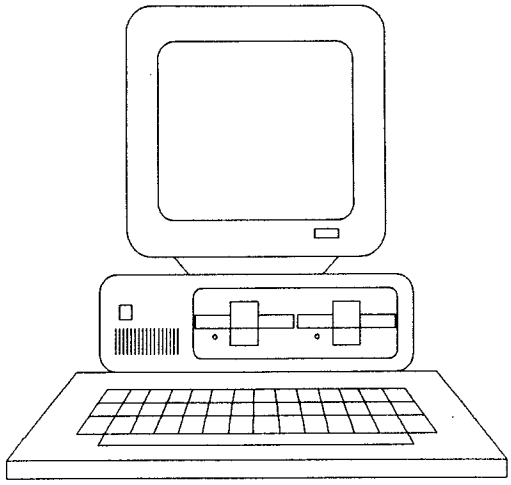
Literatuur:

1: S.R. Bourne, An Introduction To The UNIX Shell, Bell Laboratories

Joost Voorhaar

De IBM-PC en z'n klonen (Deel 13)

Zou dertien toch het veelgevreesde ongeluksgetal zijn? Voor deze artikelenreeks in ieder geval wel. Uw nijvere intyperd (want schrijven doen tegenwoordig niet meer) is namelijk zo ongeveer door zijn kennis van PC's heen. Niet dat alles nu al voor het voetlicht is geweest. Want met DOS is bijvoorbeeld nog niets gedaan in deze reeks, behalve dan dat het geboot is met INT 19h. Kun je heel wat over vertellen. Doe ik niet. Ruud Uphoff wel. Zie elders in dit nummer. Er zijn nog twee dingen die ik nog kan behandelen. Dat zijn de VGA-kaart en de EISA-bus. En dat doen we dus deze keer. Voordat we aan de VGA-kaart gaan snuffelen eerst een kleine uitstapje naar super-EGA kaarten. Want daar zit het begin van de geschiedenis van de huidige VGA-kaarten.



EGA met kunstjes

De volhouders onder de lezers van deze serie hebben in deel 5 (dat was oktober 1989) al kennis gemaakt met de EGA-kaart. Toch even een opfrissertje. Na de CGA- en MDA-kaarten, die allebei werden bediend door het systeem-BIOS op het moederbord zelf, kwam IBM bij de introductie van de PC/AT ook met een nieuwe displaykaart: de EGA. Met die kaart werden de mogelijkheden en ook de kwaliteit van de displays op PC's verder uitgebreid.

In kleur hadden we nu dezelfde tekstkwaliteit als in monochrome: tekens van 14 scanlijnen hoog. De EGA-kaart had ook meer videoRAM. Dat uitte zich zowel in tekstmode als in grafisch bedrijf in meer mogelijke kleuren. Een EGA-kaart kan een bepaalde modes 64 verschillende kleuren genereren. IBM zorgde ervoor dat alle oude zaken gewoon bleven werken, zodat de reeds bestaande MDA en CGA monitoren gewoon op de EGA-kaart konden worden aangesloten. Wilde je het onderste uit de kan, dan moest er een echte EGA kleurenmonitor aan te pas komen.

De hogere resolutie in tekst mode en ook de nieuwe grafische mode 640x350 vergden een nieuwe timing en ook een nieuwe clockfrequentie voor de pixelclock. Op een EGA-kaart is die frequentie dus om te schakelen in software. Een ieder die eens de 6845 heeft bestudeerd en begrepen, weet, dat je uitgaande van de registers een groot aantal beeldformaten kunt programmeren. Zeker als je ook de pixelclock kun veranderen. En dat is wat een EGA-kaart dan ook doet als je van CGA naar EGA mode overgaat.

Al deze pret werd bestuurd door een eigen video-BIOS, dat op de EGA-kaart zelf was ondergebracht op adres C0000h. Dat BIOS wordt gevonden en geactiveerd door de ROMscan van het systeem-BIOS.

Met die programmeerbare registers in de CRTC op de EGA-kaart, samen met nog meer pixelclockfrequenties en dat op de kaart aanwezige BIOS kun je nog meer doen. Hogere resoluties maken bijvoorbeeld. Of andere beeldformaten. Of Hercules graphics toevoegen. Zo geschiedde dan ook. Twee bedrijven zijn hierin toonaangevend geweest: ATI (in Canada) en Tseng Laboratories (in Taiwan). ATI kwam als een van de eersten met de EGA Wonder kaart op de markt. Extra's: Hercules mode, 132x43 en 132x25 tekstmodes, en 640x480 graphics in kleur. Netjes ondersteund in het BIOS, met voor iedere afwijkende mode keurig een modenummer.

Intussen had NEC een bijzonder soort monitor uitgevonden: de multiSync. Dat was een bijzonder geval: als er een willekeurige lijnfrequentie tussen 15 en 37 kHz en een rasterfrequentie tussen 50 en 65 Hz instopt, levert de monitor keurig een plaatje. De concurrentie volgde. De EGA-bouwers zaten eveneens niet stil. Een wildgroei aan modes, speciaal grafisch ontstond, want plotseling was de monitor niet langer de beperkende factor in het geheel. Er verschenen kaarten met hogere resoluties: grafisch 768x512, 800x600 en zelfs 1024x768 is vertoond. Prachtig voor grafische paketten als Windows, AutoCAD en DTP pret. Uiteraard had iedere fabrikant een set modenummers, die soms uiterst ongelukkig gekozen waren. In de praktijk bleef het gebruik beperkt tot de normale EGAmodes, meestal door een gebrek aan drivers voor de paketten.

Tseng Labs onderkende dit probleem al in een vroeg stadium, en heeft lange tijd geijverd voor genormaliseerde modenummers. In de EGAtijd bleken ze de beroemde schreeuwer in de woestijn. Later kwam

het toch nog goed. Maar eerst was IBM weer aan zet.

De VGA-kaart

IBM dacht de truc met de PC/AT nog wel een keer te kunnen herhalen. Dus verscheen er een nieuwe reeks PC's, PS/2 geheten. Met een nieuw operating systeem, OS/2 gedoopt. Die PC's wilden we niet: ze waren niet compatibel met de bestaande machines dankzij een maffe bus, die IBM Micro Channel had gedoopt. Bovendien waren klonen goedkoper en sneller. Het operating system OS/2 hoefden we ook niet. Het was niet af. Het wemelde van de bugs. Het was traag. De beloofde grafische interface, de Presentation Manager, ontbrak bij de eerste release. Het vrat geheugen: je moest minstens 2 Mbyte RAM hebben. Einde OS/2. De geschiedenis herhaalde zich wel met de displayadapter.

De PS/2 machines hadden net als bij de PC/AT een nieuwe displayadapter in zich: de VGA. Dat betekent Video Graphics Array. Er waren, net als bij de EGA-kaart een aantal nieuwe zaken te melden. Als eerste en belangrijkste feit: analoge sturing van de monitor. Dat wil zeggen: de drie basiskleuren in de monitor R, G en B kunnen elke intensiteit tussen nul en maximaal aannemen. Dus niet alleen uit, maximaal en intensified zoals bij de andere kaarten. Dat houdt in dat de VGA-kaart op een kleurenbuis in principe iedere willekeurige kleur kan laten zien. Er is nog wel een beperking aan die kleuren: het aantal kleuren dat tegelijk in 1 beeld getoond kan worden is helaas beperkt tot 256. Maar die 256 kleuren kunnen gekozen worden uit een totaal van 2^{18} verschillende kleuren. Voor diegenen die geen kleuren wilden of nodig hadden verscheen er een monochrome monitor. Analoog. Kan dus 256 grijstinten laten zien. Lijn en raster frequentie identiek aan de kleurenbroeder. 1 display adapter dus voor beide markten.

Er was natuurlijk nog meer. Nieuwe grafische modes bijvoorbeeld. 640x480 vond IBM wel genoeg. Wel voor het eerst een verhouding tussen hoogte en breedte van 3:4. Een verhouding die vrij veel voorkomt in de praktijk. Wat ook nieuw was: meer videoRAM, nu tot 512k byte aan toe. Dan kun je ook bij 640x480 nog 256 verschillende kleuren laten zien. Tekstmodes vertoonden nu karakters van 16 scanlijnen hoog. In het VGA-ROM dat net als bij de EGA-kaart op adres C0000h begint zitten verder ook 8 en 14 punts karaktergeneratoren. De VGA-kaart emuleert namelijk op verzoek ook de EGA en CGA modes op een VGA-monitor. Met de nadruk op emuleert: de lijn- en rasterfrequenties blijven op de VGA waarden van 31,5 kHz en 60Hz. Meer hoeft ook niet, want de oude TTL-monitoren kunnen niet op een IBM VGA-kaart worden aangesloten.

Analoog en digitaal

De VGA-kaart werkt intern nog steeds digitaal. Er zit gewoon video RAM op, dat ook op de gebruikelijke wijze wordt gebruikt: in tekstbedrijf als karakter/attribuut RAM, in grafisch bedrijf stellen 2 tot 8 bits de kleur van 1 pixel voor. De monitor is echter analoog. De omzetting wordt gedaan door en speciale chip: de video DAC. Nu kunt je op de 8 bits die nog steeds maximaal parallel uit het RAM komen een 8 bits DAC aansluiten, waarvan de uitgang naar de monitor gaat, maar het kan mooier. De architectuur van de videoDAC is namelijk complexer.

In de video DAC zitten namelijk drie DACs, voor iedere kleur (R, G en B) eentje. Iedere DAC heeft een resolutie van 6 bits, dat wil zeggen dat iedere primaire kleur 64 helderheidsniveaus kan aannemen. Dat levert dus theoretisch 64x64x64 = 262144 verschillende kleuren op. Om die 6 bit voor de drie DACs te maken, heb je dus 18 bits totaal nodig. Er komen er echter maar 8 uit het videoRAM. Hier wordt dus een truc speciaal (mayonnaise, ketchup en uitjes) uitgehaald.

Hier wordt dus een truc speciaal (mayonnaise, ketchup en uitjes) uitgehaald.

Tussen de inkomende 8 bit data, en de drie DACs wordt een supersnel RAM geschakeld dat de naam paletRAM draagt. De adres lijnen van dat RAM hangen aan de datalijnen van het videoRAM. Daar zijn er dus 8 van. Het RAM is 18 bits breed voor iedere DAC 6 bits. Aan het RAM zit verder een interface, zodat de CPU erin kan schrijven. De CPU ziet echter 3 maal 256 locaties van 8 bit op de bus, waarvan de hoogste twee bits altijd nul zijn. Al die locaties worden niet lineair geadresseerd, dat zou teveel geheugenruimte kosten. Er is maar 1 locatie, waarin sequentieel 3x256 = 768 bytes geschreven moeten worden. Tellers in de videoDAC verzorgen dan de adressering.

De oplettende lezer heeft het nu natuurlijk al door: van ieder van de 256 kleuren die tegelijkertijd getoond kunnen worden, kan met 6 bits nauwkeurigheid ingesteld worden hoeveel rood, groen en blauw die kleur bevat. Er kunnen door middel van het palet 64x64x64 = 262144 verschillende kleuren gemaakt worden. Door de 8 bit videoRAM databreedte kunnen er daarvan 256 getoond worden.

De op de IBM-kaart aanwezige chip werd ontwikkeld door de pionier op videoRAMgebied, Brooktree. Inmos volgde snel met een second source. Inmiddels zijn ook een aantal chipsboeren uit Taiwan met deze chip op de markt, waaronder UMC.

Super VGA

Bekijk je de VGA-kaart op zijn merites, dan blijkt dat IBM weer wat hiaten heeft laten vallen. Vanuit het standpunt van IBM gezien is er sprake van vooruitgang en compatibiliteit. Worden de uitgebreide EGA-kaarten echter in het plaatje betrokken, dat blijken deze meestal hogere resoluties te bieden. En zo geschiedde het wederom. De VGA-chipset werd door verschillende fabrikanten uitgebreid en in 1 enkele superchip gestampt. Die enkele chip zorgde voor lage prijzen. Dankzij het bestaan van de multisync monitoren schroomde men niet om allerlei resoluties in te bouwen: in de EGAtijd was geleerd hoe dat moest. In no-time verschenen er zogenaamde Super-VGA-kaarten op de markt. De hogere resoluties beperkten zich meestal tot 800x600 en 1024x768. Afhankelijk van de hoeveelheid RAM op de VGA-kaart kunnen dergelijke kaarten 16 tot 256 kleuren laten zien bij 800x600, en 4 tot 16 bij 1024x768. Standaard heeft iedere VGA-kaart net als bij IBM 256 kbyte RAM. De messte Super-VGA's hebben 512 kbyte RAM, sommige zelfs 1 Mbyte. Een aantal kaarten ondersteunen 1024x768 alleen bij 512 kbyte RAM.

Nu men toch aan het chips bakken was werd er nog meer uitgebreid.

Nu men toch aan het chips bakken was werd er nog meer uitgebreid. De meeste VGAchips konden natuurlijk CGA, EGA, MDA en Hercules modes emuleren, want dat had IBM ook. Men ging echter verder: deze modes kunnen meestal ook gegeneerd worden, zelfs compleet met TTL-uitgang voor de oudere monitoren. Dat levert trouwens meteen VGA met TTL-sturing op.

Net als bij EGA zijn bij VGA twee firma's actief geweest met wat tegenwoordig heet extended VGA-modes: alweer Tseng Labs en Video7. Tseng Labs komt de eer toe de meest wilde VGA chip te hebben gebakken van iedereen: zij ondersteunden als eerste 1 Mbyte video RAM en kunnen reeds 800x600 in 4 kleuren laten zien bij slechts 256 kbyte videoRAM. Dat was niet alles: de Tseng ET3000 VGAchip ken ook nog hardware windowing en zoomen met factoren 2, 3, 4, 5, 6 en 7.

Het aloude probleem van de wildgroei in modenummers leek weer even de kop op te steken, maar dit is in de praktijk meegevallen. Ten eerste heeft bijna iedereen voor modenummers gekozen tussen 50h en 60h. Daar was IBM nog lang niet aan toe, dus dat leverde geen conflicten op. Tseng Labs ging een andere weg: zij bleven opwaarts compatibel met hun EGA-kaarten. En ze deden nog wat. Tseng stelde een lijst van modes (zowel bestaande als nog te verschijnen modes zoals 1280x1024) en scanfrequenties samen en droeg deze voor ter normalisatie. Een aantal fabrikanten houdt zich inmiddels aan de lijst.

Niet allemaal. Tseng breidde namelijk de IBM definitie voor een modenummer uit: het geheugengebruik in de mode moet ook hetzelfde zijn. En in 800x600 zijn er heel wat verschillende mappings in gebruik op de diverse kaarten. Het feit dat de ene kaart alleen 800x600 in 16 kleuren kan bij 512 kbyte, en een andere al 800x600 bij 4 kleuren presteert bij 256k en tot 256 kleuren komt in dezelfde mode bij 512k zegt eigenlijk al genoeg.

8514/A

Er verscheen nog een tweede videokaart van IBM. Eentje die heel wat in zijn mars had en eigenlijk bedoeld was voor de high-end grafische markt. De resolutie bedraagt 1024x768. De scanning is echter non-interlaced. Dat wil zeggen dat per raster alle 1024 (zichtbare) beeldlijnen worden geschreven. De meeste Super VGA-kaarten geven namelijk een interlaced plaatje bij 1024x768. Die schrijven eerst alle even lijnen, en het volgende raster alle oneven lijnen daar precies tussen in. De interlaced heeft een voordeel en een nadeel. Het voordeel is, dat de pixelclock de helft is. En dus ook de maximale snelheid van de videoDAC. Het nadeel is dat het beeld de neiging heeft te trillen of te flikkeren. Iedereen kent dat: de TV is ook interlaced.

De 8514/A kaart had dus bestaansrecht. Er moest een speciale monitor aan, wiens IBM-nummer de kaart zijn naam heeft gegeven. Een aantal Super VGA-kaarten beheersen, soms na modificatie ook de 8514/A mode. Vaak is wel een multi-sync monitor van een betere klasse nodig, want de lijnfrequentie is dan opgelopen naar 48 kHz, en de raster frequentie naar 70 Hz. Maar mooie plaatjes geeft het wel.

Overigens weten veel multi-sync monitoren ook geen raad met 1024x768 interlaced. De zeer bijzondere NEC MultiSync 3D met microprocessor gestuurde automatische beeldcentrering en -maataanpassing beheerste deze mode als eerste. De meeste mensen geven echter de voorkeur aan de noninterlaced 800x600 mode vanwege de flikkering bij interlaced.

Ondergetekende ook, want dit blad wordt in 800x600 opgemaakt.

Andere bussen

Terloops werd al even opgemerkt aan het begin van dit verhaal, dat IBM met de introductie van de PS/2 machines ook maar even een nieuwe bus introduceerde: Micro Channel. Niemand wilde die bus hebben want de oude kaarten pasten er niet in. De reden voor Micro Channel was erachter voor de hand liggend: de 32-bit 80386 processor. Als die full speed moet blijven, moet je 32-bits breed bij het geheugen kunnen. En dat kon niet bij de AT-bus, die bij 16-bit was blijven steken.

Er was nog een tweede reden voor Micro Channel. IBM had er schoon genoeg van wel hoge ontwikkelkosten te hebben, maar vervolgens grote inkomsten te missen aan die na-apers uit het Verre Oosten. Dat moest maar eens veranderen. Zo geschiedde. Als je een machine met Micro Channel wilt maken moet je rechten betalen aan meneer IBM. En die waren niet mals, die rechten. Een reactie uit de markt bleef natuurlijk niet uit. Een aantal belangrijke fabrikanten van krachtige PC's waaronder HP, Compaq en Epson staken de koppen eens bij elkaar en verzonnen een eigen 32-bits bus voor de 80386 en de 80486. De AT-bus van IBM was de industriestandaard geworden en ging inmiddels door het leven als ISA-bus (Industry Standard Architecture bus). HP en consorten noemden hun bus de EISA-bus. Gewoon Enhanced ervoor dus.

De EISA-bus speelt handig in op de marktbehoeften. De bus is 32-bit breed (Zowel adres als data) en is zowaar compatibel met de oude XT- en AT-kaarten. Dat is gerealiseerd door een connector te verzinnen die in twee etages is opgebouwd. De bovenste etage is precies gelijk aan de oude AT-connector met 62 plus 36 contacten. Daarin passen de oude kaarten. De onderste laag heeft net zoveel contacten, die echter precies tussen de contacten van de bovenste laag in zitten. Door een nok in de connector kan een gewone AT-kaart niet doordringen tot de onderste rij contacten. Echte 32-bit EISA kaarten hebben twee rijen contactvingers en een uitsparing, en kunnen helemaal in de busconnector gestoken worden. De EISA-bus is dus zowel 32-bit, als compatibel met de ISA-bus. De connector is bedacht door de firma Burndy, samen met Amphenol.

Ook de elektrische specificaties van de EISA-bus werden nauwkeurig vastgelegd, om de compatibiliteit goed te kunnen garanderen. De EISAbus schrijft zelfs de timing voor van iedere cyclus. Een dergelijke specificatie ontbreekt voor de ISA-bus.

Micro Channel is niet echt doorgebroken. EISA ook (nog?) niet echt. Kennelijk is de behoefte aan een echte 32-bit bus niet zo groot. Temeer omdat er steeds meer RAM op de moederborden kan. De toekomst zal het leren.

De volgende keer...

die komt niet meer... Dit was wat mij betreft het laatste deel van deze reeks. Ik heb er met veel plezier aan geschreven (pardon, getypt). Ik hoop dat u, de lezer, er met plezier in gelezen heeft en wat meer idee heeft gekregen wat er zich in de kast van de privé-PC afspeelt.

De volgende keer...
die komt niet meer...
Dit was het laatste
deel van deze reeks.

Misschien heeft u wel eens gedacht: "waar haalt 'ie het allemaal vandaan?" Dat is niet zo moeilijk. Het is begonnen met het uit elkaar pulken van een verschrikkelijk slecht XT-BIOS uit Taiwan. Dat BIOS is nu onder andere het club-BIOS. Omstreeks dezelfde tijd kwam iemand aandragen met de Technical Reference Manual van de IBM PC. Die werd grondig versleten met het bestuderen. Want documentatie maken, dat kunnen ze IBM. Zelfs de listings van het BIOS staan erin, met commentaar!

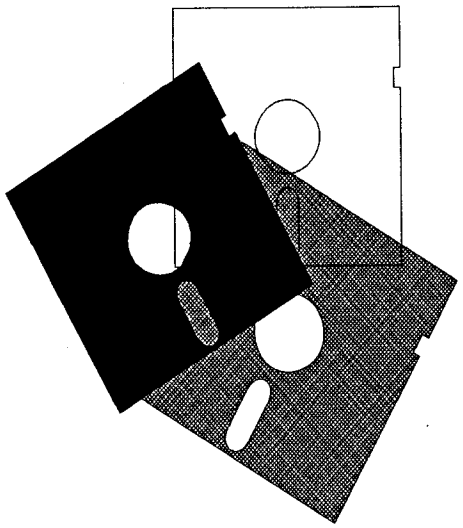
Bij een werkgever kwam ik de Technical Reference tegen van de IBM EGA kaart. En bij een bevriend bedrijf werd de Technical Reference van de PC/AT aangetroffen. Samen met het boek Advanced MS-DOS van Ray Duncan waren dat de bronnen voor deze reeks verhaaltjes. Plus wat praktijkervaring. En veel, heel veel tijdschriften lezen.

PC's zijn best leuk. Niet alleen om mee te werken, maar ook om aan te werken. Sommige dingen lijken stom opgelost. Zoals de maximale 640 kbyte RAM. Maar u weet inmiddels waarom en ook hoe het anders kan. En daar ging het om. Veel plezier bij het terugbladeren.

Nico de Vries

To Share Or Not To Share, That's The Question

Eens in de zoveel tijd steekt het verschijnsel "computervirus" weer de kop op. Geruchten doen argeloze gebruikers 's nachts wakkerliggen en bezorgen als virus-buster bekend staande freaks handen vol werk. Hoe reëel het besmettingsgevaar is, valt nog te bezien. Nemen we het bulletin board van de vereniging als uitgangspunt, dan valt het allemaal hard mee. Slechts één maal is er bij een grote beurt wat gevonden, maar dat is dan ook alles. Vanuit het je-weet-maar-nooit-standpunt gezien is het echter best aardig een anti-virus toolbox achter de hand te hebben. Vóór het mogelijk is een dergelijke gereedschapskist samen te stellen, is het noodzakelijk te weten wat een virus is, hoe je het kan ontdekken en vooral hoe je het moet bestrijden.



Virussen en ander ongedierte

Een virus is een stukje programmatuur dat zich in een programma of bootsector verstopt, zich vermenigvuldigt en eventueel informatie vernietigt of andere schade aan computersystemen toebrengt. Naast virussen zijn er ook bacteriën, "rabbits", logische (tijd-) bommen, Trojaanse paarden en wormen die ongeveer dezelfde eigenschappen bezitten. Kenmerkend voor virussen is echter de parasitaire voortplantingswijze van deze "levensvorm". Ze hebben andere programma's of bootsectors nodig om zich te kunnen verspreiden, dit in tegenstelling tot de andere genoemde vormen.

Ook de virussen zijn nog in verschillende categorieën op te delen. Zo zijn er virussen die zich in bootsectors nestelen, sommige tasten alleen SYS-files ("resident device drivers") aan of infecteren .EXE of .COM programma's. Er zijn virussen die in het geheugen blijven hangen en alles aantasten dat wordt opgestart en er zijn er die andere programma's op disk benaderen.

Om een virus te ontdekken en te bestrijden is een heel scala aan programmatuur en hardware beschikbaar. Het hardwaredeel valt buiten de doelstellingen van deze rubriek en wordt dientengevolge hier niet besproken. De software kan ook weer opgedeeld worden in een aantal verschillende categorieën. Er is software die op specifieke codes bestanden scant en er is software die probeert programmatuur te beschermen door de disks zoveel mogelijk te beschermen tegen verdachte diskactiviteiten (schrijven naar bootsectors, openen van executables en overlays in write- of appending mode...). De gereedschapskist kan het beste voorzien zijn van software die deze functies allemaal in zich bergt.

"Voorkomen is beter dan genezen", is de spreuk van menig huisarts. Dat gaat natuurlijk ook op voor computervirussen. Een virusdichte machine is een machine waar nooit nieuwe software op komt, en die niet voorzien is van allerlei remote links. Een eenzame machine dus, die eigenlijk nooit gebruikt zal worden. Infectiegevaar is te verkleinen door nieuwe software altijd even te scannen voor gebruik, geen postverbindingen open te laten staan en vooral... het op de juiste wijze draaien van backups. Dat wil zeggen: voor het draaien van een backup éérst checken op virussen, dan pas floppen steken! Een redelijke methode die ook ruimte op het backup medium bespaart bestaat uit het alléén backuppen van datafiles. Van de executables zal altijd nog wel ergens een origineel zijn!

De gereedschapskist

Voor de gereedschapskist neme men een nieuwe flop. Daarop wordt een schoon systeem gezet, bootable en wel. Deze disk moet ook voorzien zijn van alle programmatuur om een disk te kunnen installeren. Dus óók een (low-level-) formatteer programma en een restore utility. Verder natuurlijk de virusscanner met datafiles en eventuele beschermende programmatuur. Deze disk is natuurlijk altijd tegen schrijven beveiligd en voorzien van de nieuwste versies...

Het gereedschap

Een gereedschapskist zonder gereedschap is... leeg. Het gereedschap dat we hier willen bespreken bestaat uit een detector, een virusscanner, een virusvernietiger en een hoop documentatie. Laten we maar beginnen bij de detector. Vroeger nam men in de steenkolenmijnen altijd een kanarie mee naar beneden. Niet omdat het beestje zo leuk kon fluiten, maar omdat kanaries de eigenschap hebben veel ge-

voeliger te zijn voor giftige gassen dan mensen. Men gebruikt ze tegenwoordig trouwens ook op allerlei slagvelden waar een reële dreiging van gifgassen is. Analoog hieraan is het programma "canary" de computer-vogel. Het ding checkt zichzelf op alle mogelijk wijzen bij het opstarten en geeft een melding als hij zichzelf niet in orde bevindt. Persoonlijk vind ik het ding een beetje een mosterd-na-de-maaltijd geval; als de kanarie overlijdt is er in ieder geval iets niet in orde. Het programma vindt slechts een heel beperkte categorie virussen: alleen die, die zich in .EXE programma's nestelen. Het voordeel van canary is natuurlijk dat onbekende virussen en andere onfrisse zaken ook gedetecteerd worden, in tegenstelling tot scanners die op zoek gaan naar vaste codes.

De scanners zijn er in twee soorten: zij die een externe datafile nodig hebben (HtScan en TBScan) en zij die de specifieke codes intern in de executable opgenomen hebben. Bij de laatste categorie hoort onder andere de scanen clean programmatuur van McAfee thuis.

Dan scant HtScan werkelijk alle files en vindt dan ook prompt een virus...

Op het eerste gezicht lijken programma's die werken met externe datafiles een groot voordeel te hebben ten opzichte van programmatuur die werkt met interne tabellen. De nieuwste virussen kunnen makkelijker bijgewerkt worden met een losse lijst, die makkelijker en vooral goedkoper te verspreiden is. Doordat de lijst los verstrekt wordt is hij echter ook vatbaarder voor al dan niet opzettelijk toegebrachte storingen. Doordat de lijst in een door iedereen leesbare vorm verspreid wordt heeft de ingebouwde CRC-code zin. De lijst die bij de programma's HtScan en TBScan gebruikt wordt vindt zijn oorsprong bij de Nederlandse virus-goeroe Jan Terpstra. Deze houdt de lijst ook up-to-date en zorgt voor de primaire verspreiding van het ding.

HtScan is de meest "domme" virusscanner van de drie. Hij test alleen het geheugen, de bootsector en files met een .EXE of .COM extensie. Resident devicedrivers (.SYS) en overlays worden niet standaard door HtScan bekeken. Het is wel mogelijk alle files te doorzoeken op virussen door middel van de /A optie. Dan scant HtScan werkelijk alle files en vindt dan ook prompt een virus... in een pure ASCII-file! "Oops, foutje!", zullen we dan maar denken. Voor de beginner op computergebied is er echter geen enkele reden aan te nemen dat die beruchte virussen niet in .DOC files kunnen voorkomen, en dus... paniek in de tent!

TbScan is geheel geschreven in assembler. Daarnaast beweert de auteur dat alleen die delen van programma's gescand worden waar het virus zich op zou kunnen houden. Nogal gevaarlijk lijkt me, want het is zeer wel mogelijk een virus te bouwen dat zich ergens anders in een executable nestelt. De resultaten van deze techniek vinden we terug in de snelheid; hij is ruim twee maal zo snel als zijn grootste concurrent, de virusscanner van McAfee. Het is echter ook bij TbScan raadzaam een grote pot koffie bij de hand te houden.

We zullen ons verder bezighouden met de virustools van deze McAfee. Die tools bestaan uit een scanner (SCAN), een vernietiger (CLEAN) en een copieerprogramma dat meteen op virussen scant bij het copiëren van files (VCOPY). Daarnaast heeft McAfee ook een memory-resident programma geschreven dat virussen tegen moet houden: VSHIELD. Al deze programma's gaan vergezeld van een programmaatje waarmee gecheckt kan worden of ze onbeschadigd en ongewijzigd zijn en een lange lijst met alle bekende virussen erop. McAfee houdt er een speciale versienummering op na. De versie die wij ter test hadden liggen is bijvoorbeeld versie 6.9V75. Waarschijnlijk is het getal vóór de V het versienummer van de programmatuur, het nummer ná de V geeft de virus informatie update aan. Versie 6.9V75 kent maar liefst 222 verschillende virussen in 481 variaties. Komt hij ergens een virus tegen, dan kan hij de geïnfecteerde file zelf weggooien. Voor bootsectors en masterbootrecords gaat dat natuurlijk niet op. Soms is het weggooien van geïnfecteerde files ook niet voldoende, dan is een bewerking met CLEAN nog nodig. CLEAN heeft dezelfde versienummering als SCAN. CLEAN is ook in staat om de bekende virussen op te sporen. SCAN doet echter meer dan CLEAN in dit opzicht; het is dus niet voldoende er van uit te gaan dat CLEAN alle virussen zelf wel vindt. Zo kan SCAN .COM en .EXE files voorzien van een CRC-code waarmee veranderingen in de file gedetecteerd kunnen worden. Die veranderingen hoeven natuurlijk niet een virus te betekenen; er zijn nog steeds een hoop programma's die tijdens een eventuele installatie procedure aangepast worden. Daarnaast scant SCAN ook op virussen die op dat moment in het geheugen van de machine actief zijn.

Documentatie

De documentatie van McAfee is wat magertjes. Twaalf pagina's is genoeg om een (semi-) professional alles te vertellen wat hij weten wil, maar voor het onwetende slachtoffer mist er een hele hoop infor-

matie. Gelukkig is er een hoop informatie over virussen "los" verkrijgbaar. Ik verwijs hiervoor naar de files "VIRUSDOC.ZIP" en "VSUM9103.ZIP". De eerste file omvat een 26-pagina's tellende publicatie van het IBM Los Angelos Scientific Center onder de titel "Coping with Computer Virusses and Related Problems". Dit artikel bevat een beschouwing van het virus-verschijnsel en geeft wat tips over virussen in samenhang met MS/PC-DOS. De tweede file (de cijfers zullen wel op de maand/jaar combinatie van uitgifte slaan en dus kunnen die afwijken!) bevat een halve Megabyte aan omschrijvingen van virussen. Dat wil zeggen: van ieder bekend virus staat er precies in beschreven wat ze doen, hoe ze te herkennen en te vernietigen zijn en vaak ook nog wat informatie over de geschiedenis van het betreffende virus.

Tenslotte

Virussen kunnen heel venijnige en gemene beesten zijn. Gelukkig is het verschijnsel in de praktijk niet zo'n grote plaag als de nieuwsmedia ons willen doen geloven. Ze komen echt niet zomaar via het modem de computer binnenwandelen en ze kunnen niet tot leven gewekt worden vóórdat het geïnfecteerde programma opgestart wordt. In de praktijk betekent dat, dat nieuwe programmatuur even bekeken moet worden met behulp van een virusscanner voor de zaak gestart wordt. Security envelopes op gearchiveerde bestanden en eventueel toegevoegde CRC-checks kunnen het risico van besmetting nog kleiner maken. Het regelmatig maken van backups garandeert niemand een virusvrij systeem, maar kan wel een grote opluchting betekenen in geval van een catastrofe. Zet in zo'n geval de systeemtijd even terug naar de datum van de backup, restore de hele zaak en zoek met een virusscanner naar eventuele besmettingen. Denk er daarbij aan dat archieven uitgepakt moeten worden vóór het scannen! Geïnfecteerde bestanden kunnen dan weggegooid (wipedisk!) of gerepareerd worden met bijvoorbeeld CLEAN. Moge de anti-virusspuitbussen met u zijn!

Joost Voorhaar

Een beoordeling van de besproken software zult u deze keer niet aantreffen bij dit artikel. Virusscanners blinken nu eenmaal niet uit in documentatie, popup- en pulldown menus, helpschermen en muisondersteuning. De werking van de scanners en vernietigers kan ook maar ten dele getest worden; het test-systeem is gelukkig virusvrij.

Besproken produkten (tussen haakjes staat steeds de filename, gevolgd door de grootte van de file in bytes):

Scanners

McAfee's SCAN, versie 6.9V75 (SCANV75.ZIP, 75452)
Frans Veldman's TBScan, versie 1.8 (TBSCAN18.ZIP, 46731)
Harry Thijssen's HtScan, versie 1.13 (HTSCAN13.ZIP, 35241)

Detectors

McAfee's TSR detector VSHIELD, versie 6.9V75 (VSHIELD75.ZIP, 69872)
McAfee's kopieerprogramma VCOPY, versie 0.5V74 (VCOPY74.ZIP, 41947) Alarmerings programma Canary, versie 91e (CNY91E.ZIP, 82854)

Documentatie en datafiles

"Coping with Computer Virusses and Related Problems" (VIRUSDOC.ZIP, 27665)
"Virus Information Summary List" (VSUM9103.ZIP, 152366)
Datafile "VIRSCAN.DAT" voor TbScan en HtScan (VIRUSSIG.ZIP, 14505)

Deze files zijn beschikbaar op het bulletin board van de vereniging, "The Ultimate" in de filearea "Algemeen". Het alarmeringsprogramma "Canary" vindt u terug in de SDN-areas.

Het programmeren van de 8088 in de IBM (Deel 4)

Een ander soort programma: *.COM

Tot nu toe heeft u programmavoorbeelden gezien die uiteindelijk na assembleren een *.EXE file opleveren. Het voordeel van dat type programma is de onbeperkte omvang van de code. Als die meer dan 64Kb gaat worden, pakken we gewoon een volgend segment. Nadeel is dat we een stacksegment moeten declareren en dat zo'n leeg segment ook in de file op de schijf komt te staan. Het kan ook anders. We laten daartoe de 8088 degraderen tot een oude vertrouwde 8 bitter die slechts 64K kan adresseren. Dat wil zeggen: Het blijft een 16 bitter, maar alle code, data en stack staan in een segment. De eerste geheugenpagina van dat segment is weer gereserveerd voor het PSP en dus begint ons programma altijd op adres 100H.

Als algemene eisen voor een programma van een .COM structuur gelden:
- Code, Data en Stack staan in hetzelfde segment. (Zie het ASSUME statement)
- Het hoofdprogramma moet beginnen op offset 100H
- Er mogen geen instructies in voorkomen die naar een segment verwijzen.

Het programma dat we nu gaan bekijken gebruikt deze eenvoudige structuur.

Filters

Als we de listing van het programma TWOCOL bekijken, lijkt het alsof we met een buitengewoon zinloos geval te maken hebben. Als we het zouden executeren vanaf de DOS prompt, blijkt dat we alleen maar regels tekst kunnen intikken vanaf het toetsenbord en dat na elke tweede regel die we intikken, die twee laatste regels worden afgedrukt. Alleen is daarbij de CRLF aan het einde van de eerste regel van elk paartje weggehaald en is hij aangevuld met spaties tot kolom 40 alwaar de tweede regel eraan geplakt is. Parameters kent het programma niet. Er worden geen files geopend of gesloten. Vergelijken we nu de subroutine READBYT met die uit de vorige aflevering, dan zien we slechts twee kleine verschillen:
- De handle die wordt gebruikt om te lezen is niet ter beschikking gekomen door het openen van een file, maar we gebruiken handle #0. Het "standard input device", meestal het toetsenbord.
- Als het einde van de file wordt aangetroffen, wordt deze niet gesloten, zoals we in het programma FINDS deden.

Ook de routine die een foutmelding afdrukt is iets veranderd. Deze maakt uitsluitend gebruik van het beeldscherm. Daarvoor is door DOS de handle #2 gere-serveerd die niet kan worden omgeleid. (Althans niet door COMMAND.COM)

Het hoofdprogramma is een loop die telkens een buffer van ruim 80 tekens met spaties vult en dan twee regels van het toetsenbord leest. Als i.p.v. de eerste regel "einde file" wordt aangetroffen stopt het programma meteen. Dat zal gebeuren als u alleen een CTRL-Z intikt. Zoniet, dan wordt die regel gecopieerd naar het buffer met de naam CONCAT, dat we eerst met spaties hadden gevuld. Vervolgens wordt in kolom 39 netjes een vertikaal streepje gezet (Code 179) en wordt een tweede regel van het toetsenbord gelezen. Als dat een lege regel is, dan zou dat einde file kunnen betekenen, maar dat hoeft niet. Daarom plakken we hem vanaf positie 40 in het buffer CONCAT, achter de eerste regel en zetten er een CRLF achter. Dit laatste is nodig omdat we in de routine READLN de CRLF van elke regel eraf halen. De regel in CONCAT wordt nu afgedrukt op het beeldscherm door te schrijven naar de altijd geopende handle #1.

U zult misschien opmerken dat hier en daar instructies als REP MOVSB en REP STOSB voorkomen waar iets lijkt te ontbreken. Moest het ES register hier niet geïnitialiseerd worden op ES = DS? Ja dat moet, maar dat heeft DOS al voor ons gedaan. Bij de start van een *.COM programma hebben de segmentregisters alle-maal dezelfde waarde. Zolang we ES niet veranderen behoeven we ons daarover dus helemaal geen zorgen te maken.

Het doel van dit filter is van het simpele soort: Tikt u na het assembleren van het programma 2COL.COM maar vanaf de DOS prompt:

CDIR | 2COL | MORE

Misschien begrijpt u daaruit wat COMMAND.COM precies doet op het "pipe" tekentje "|". Het interne commando DIR wordt uitgevoerd, maar het |-symbool is voor COMMAND.COM de opdracht om even een tijdelijke file aan te maken op schijf, en daar de uitvoer van het commando DIR naartoe te sturen. COMMAND.COM sluit daartoe de handle #1 en opent een handle naar die tijdelijke file. Omdat DOS de gewoonte heeft om bij het openen van een file altijd het eerste het beste vrije nummer aan een handle toe te kennen, wordt dat dus handle #1. Na afloop van een dergelijk actie wordt

de handle door COMMAND.COM weer gesloten en opnieuw geopend naar het CON: device.

Vervolgens zal DOS ons programma 2COL uit gaan voeren. Aangezien zowel voor als na 2COL een pipe-symbool staat, zal COMMAND.COM nu ook de handle #0 voor het toetsenbord sluiten en deze openen voor invoer vanuit de laatste tijdelijke file. Voor de uitvoer gebeurt hetzelfde naar een nieuwe tijdelijke file. U kunt het hele proces nadoen met afzonderlijke commando's:

```
C>DIR >TEMP.000
C>2COL <TEMP.000 >TEMP.001
C>DEL TEMP.000
C>MORE <TEMP.001
C>DEL TEMP.001
```

Waaruit valt te leren dat commando's in een pipe u een hoop werk uit handen kunnen nemen en u weet nu ook hoe eenvoudig deze "filters" te schrijven zijn.

Volgende keer iets heel anders: De IN en OUT instructies van de 8088.

Ruud Uphoff

Fig. 1: sourcetekst van 2COL

```
                NAME    TWOCOL_ASM
                ;

OFS             EQU     OFFSET

JMPS            MACRO TARGET            ;JMP SHORT  is zo lang he...
                JMP     SHORT TARGET
                ENDM

BUFSIZE         EQU     4096            ;Omvang van het buffer.

                ;------ ALLEEN EEN CODESEGMENT ----------------------------

CODE            SEGMENT

                ASSUME                  CS:CODE, DS:CODE, SS:CODE

                ORG     100H            ;Vereist voor een .COM programma

START:          JMP     TWOCOL

                ;Teken uit de file lezen

READBYT         PROC    NEAR
                MOV     BX,BUFPTR       ;Buffer ptr in BX halen
                CMP     BX,BUFEND       ;Einde buffer?
                JB      PICKBYT
                XOR     BX,BX           ;Dan buffer eerst vullen
                MOV     CX,BUFSIZE      ;met maximaal BUFSIZE bytes
                MOV     DX,OFS BUFFER
                MOV     AH,3FH
                INT     21H
                JC      ERREXIT         ;Jammer van die leesfout..
                XOR     BX,BX           ;Na lezen blok pointer op 0 zetten
                MOV     BUFPTR,BX
                MOV     BUFEND,AX       ;en aantal bytes in buffer opslaan
                OR      AX,AX           ;Niets gelezen?
                JZ      ENDFILE         ;Oeps! Dat is einde file.
```

```
PICKBYT:    MOV     AL,BUFFER[BX]  ;Teken uit buffer in AL halen
            INC     BUFPTR         ;en pointer er voorbij zetten
            CMP     AL,1AH         ;Geen CTRL-Z
            JNE     MORE           ;dan zijn we klaar
ENDFILE:    MOV     AL,1AH         ;Vlag voor einde file zetten
            MOV     EOF,AL
            CMP     AL,AL          ;en ZF = 1 laten zien
MORE:       RET                    ;ZF = 1 voor einde file
READBYT     ENDP

            ;Regel in buffer lezen

READLN      PROC    NEAR
            MOV     DI,OFS LINE    ;ES:DI naar LINE buffer zetten
            CLD
            MOV     LINEL,0        ;Begin met lengte 0 voor LINE
MORESTR:    CALL    READBYT        ;Byte uit de file lezen
            JZ      LINEDN         ;Einde file? Dus ook einde regel
            STOSB
            INC     LINEL
            CMP     AL,0AH         ;LF?
            JNE     MORESTR        ;Nee, dus nog even door gaan
            SUB     LINEL,2        ;CRLF gooien we weg
LINEDN:     RET
READLN      ENDP

        ;Afbreken met foumelding

ERREXIT     PROC    NEAR
            PUSH    AX             ;Foutcode bewaren
            MOV     DX,OFS ERRMSG + 1      ;DS:DX op de foutmelding richten
            MOV     CL,ERRMSG      ;en lengte in CX halen
            XOR     CH,CH
            MOV     BX,2           ;Handle van STDERR (error device)
            MOV     AH,40H
            INT     21H
            POP     AX             ;Foutcode terug in AL
            MOV     AH,4CH         ;en achterlaten als "Error level"
            INT     21H            ;Terug naar de de DOS promt dus.
ERREXIT     ENDP

            ;Regel in het uitvoerbuffer zetten
            ;DI wijst naar dat buffer in kolom 0 of 40

COPSTR      PROC    NEAR
            MOV     SI,OFS LINE    ;DS:SI naar de string in LINE zetten
            MOV     CL,LINEL       ;en de lengte in CX
            XOR     CH,CH
            JCXZ    ECOP           ;Lege string? Dan klaar
            CLD
            REP     MOVSB          ;Anders: Moven met die handel
ECOP:       RET
COPSTR      ENDP
```

```
                ;Hoofdprogramma

TWOCOL      PROC    NEAR
MORELINES:  MOV     AL,20H          ;Buffer wissen met spaties
            MOV     DI,OFS CONCAT
            MOV     CX,85
            CLD
            REP     STOSB
            CALL    READLN          ;Regel van standaard invoer lezen
            CMP     EOF,0           ;Niets meer?
            JA      DONE            ;dan einde programma
            MOV     DI,OFS CONCAT   ;Ingelezen regel naar uit voerbuffer
            CALL    COPSTR
            MOV     CONCAT+39,179   ;en de vertikale lijn tekenen
            CALL    READLN          ;volgende regel lezen
            MOV     DI,OFS CONCAT+40        ;en nu in kolom 40 opslaan
            CALL    COPSTR
            MOV     AX,0A0DH        ;CRLF er achter zetten
            STOSW
            MOV     DX,OFS CONCAT           ;DS:DX naar uitvoerbuffer
            MOV     CX,DI           ;Offset naar einde string
            SUB     CX,DX           ;minus offset naar begin is lengte
            MOV     BX,1            ;Handle 1 is standaard uitvoer
            MOV     AH,40H
            INT     21H
            JC      ERREXIT         ;en altij jammer van zo'n schrijffout
            CMP     EOF,0           ;Was het einde file ?
            JZ      MORELINES       ;Nee. Ok dan volgende 2 regels
DONE:       MOV     AX,4C00H        ;en anders terug naar DOS
            INT     21H
TWOCOL      ENDP

        ;De data staat nu in het codesegment

BUFFER      DB      BUFSIZE DUP (?) ;Het file buffer
BUFPTR      DW      0               ;De leesindex in dat buffer
BUFEND      DW      0               ;en de positie van het laatste byte
LINEL       DB      0               ;Deze string is de regel die we..
LINE        DB      80 DUP (?)      ;.. aan het verwerken zijn.
CONCAT      DB      160 DUP (?)
EOF         DB      0               ;Intieel 0 dus niet einde file
ERRMSG      DB      ENDMSG-ERRMSG-1
            DB      'Lees/Schrijf fout.',13,10
ENDMSG      EQU     $

CODE        ENDS

            END     START
```

Voortgang KGN-68k projekt

Inleiding

Het mooie van voortgang houdt in dat er wat te berichten is. Op het hardware gebied is het meeste gebeurd, maar eerst wat algemene opmerkingen.

Algemeen

De werkgroep heeft de goedkeuring van het bestuur om door te gaan met het projekt. Er worden handelingen verricht zo als het de bedoeling is van de KGN. Voor het laten maken van de multilayer print, met de bijbehorende kosten, wordt later in het jaar besluit genomen. Op dat moment zijn er meer details bekend over het systeem en is er ook meer bekend over de vraag naar het processorbord.

Bij het projekt komen meer dingen kijken dan alleen technische zaken. Zo zijn we bezig met de markt te verkennen en sponsors te benaderen. Sponsors moet men niet lezen als geldschieters, maar bedrijven/personen die ons steunen met informatie, testfaciliteiten en materiaal voor prototypes. Wat we ook zoeken is rekentijd. Rekentijd voor simulatie van het schema, ontwerp van de print en voor het (cross)compileren van de 68k versie van Minix. Denk niet meteen: "Niets voor mij (mijn bedrijf), allemaal veel te groot!" Compileren op een Atari met harddisk onder Minix doet het ook. Het staat toch prachtig dat er straks in de documentatie staat: "Dit projekt werd mogelijk gemaakt door ...".

De werkgroep heeft namelijk dringend een Atari ST met harddisk nodig om vanuit de Minix voor de Atari een Minix voor de KGN-68k te maken. Uiteraard moet die Atari dan wel (ook) onder Minix draaien. Kortom, kun je een tijdje (pak hem beet tot het einde van 1991) je Atari met ST missen, stel hem dan ter beschikking aan de projectgroep (of wordt zelf lid van de projectgroep).

Verder zijn alle overige ideeën, opmerkingen en suggesties ten allen tijden welkom. Hiervoor kunt je uiteraard contact opnemen met mij, of met de redacteur die toevallig ook secretaris van de projectgroep is.

Hardware

Dat er een MC68030 processor op het bord komt, is bekend. Wat wel verteld kan worden is dat het prototype, in het begin stadium, met een 68EC030 zal draaien. EC betekent in dit geval Embedded Control en het is een '030 zonder Memory Managing Unit. Tijdens de eerste testen sneuvelt er dan eventueel een minder dure chip en de MMU zit zeker niet in de weg.

De soorten RAM varieerden onbehoorlijk, wel Statische RAM, geen SRAM, wel SRAM, Dynamische RAM 16 & 32 bits breed, DRAM alleen 32 bits breed, DRAM 16 & 32 bits breed. Het is ZOWEL SRAM als DRAM geworden, het SRAM wordt gebruikt om interrupt vectoren en config parameters in te zetten. De interrupt vectoren worden tijdens de interrupt acknowlege cycle uitgelezen als de interruptbron zelf geen interrupt vector kan geven. Dit is noodzakelijk bij merendeel van de devices die in de standaard MS-DOS machine zitten. Om system configuratie parameters te bewaren is het nog wel nodig om voor een battery backup te zorgen, maar die krijgen kado bij de Real Time Clock. Dit is een IC voet met chip waar een clock en een lithium batterij in zit. De breedte van het echte werkgeheugen is toch weer instelbaar. Je draait de eerste tijd met een kwart of de helft van de ram en geldige software licentie en later koop je de rest van de SIMMs er bij, die dan al weer goedkoper zijn.

De schetsen die in het begin van het jaar gemaakt zijn gemaakt, zijn ondertussen verwerkt in een schematekenpakket. Veel witte plekken zitten er niet meer in, wat wel nog 'wit' is, is de inhoud van de blokjes programmeerbare logica. Het ontwerpen houdt nu vooral het invullen van deze PLD's in. Deze bouwstenen vervangen nogal wat "glue". De interruptlogica bestaat bijna in zijn geheel uit Programable Logic Devices. Ook de DRAM Control logic is voornamelijk PLD. De chips zelf zijn de 16V8 en 22V10. De laatste is een bekende, die door veel verschillende fabrikanten wordt gemaakt. De eerste is ontworpen de Lattice semiconductor corporation, die in Nederland vertegenwoordigd wordt door o.a. Alcom Electronics. De GALs, zoals Lattice zijn chips noemt, zullen door de KGN geprogrammeerd worden. Daar hoef je je geen zorgen over te maken, als je te minste lid bent.

De eerste grote tegenvaller zijn we ook al tegengekomen. Het ontwerp van de AT staat het wel toe om de bus over te laten nemen door een andere busmaster dan de 80286 processor of de 8237 DMA controller, maar niet dat deze gebruik maakt van devices waarvan de DMA via de 8237 loopt. Nu wordt er getracht om gebruik te maken van DMA controller op het AT-Motherboard. Dit houdt in dat de 8237 bij het geheugen van de insteekkaart MOET kunnen, met als nadeel dat de 8237 maar een 64kbyte blok tegelijk kan verplaatsen, en de 80286 dan ook impliciet bij het werkgeheugen van de '030 KAN. Sterker nog, op dat moment kun je twee(meerdere) KGN-68k boards op de bus steken die bij elkaar in het geheugen kunnen lezen/schrijven!

De timer van de DUART, de seriële communicatie chip, zou gebruikt worden voor het genereren van de systemtick. Als de printruimte het toelaat komt er een andere chip bij die het ook een timer heeft. Deze chip heeft nog meer voordelen dan een extra Interrupt nivo zoals een bi-directionele centronics poort. Bovendien kan men dan met de DUART met exotische baudrates, zoals bijvoorbeeld MIDI, werken.

Software

Het software gedeelte komt nu ook op gang. Diverse leden van de werkgroep zijn zich aan het verdiepen in specifieke Minix eigenschappen. Als je zelf al pitfalls van Minix tegen bent gekomen of als je een gedegen kennis van Minix hebt, zouden we dat natuurlijk graag weten. We kunnen namelijk in de projectgroep nog wel wat specifieke Minix kennis gebruiken.

Behalve met de voorbereiding van het porteren van een kompleet operating system, zijn we ook bezig met het schrijven van een bootstrap die ook als monitor/debugger kan dienen. Eén van de mogelijkheden van de monitor is het kunnen downloaden van Motorola S-records. Op deze manier kan men stukken software in RAM zetten en vervolgens testen. S-records zullen wel bekend zijn, zie anders De 6502 kenner, nummer 56. Dit is trouwens weer een voorbeeld van kennis die al jaren in onze club zit!

Afsluiting

Tot zover voor dit nummer van de μP Kenner. Het laatste nieuws is ten allen tijden te vragen aan de leden van de werkgroep. Daar kun ook je terecht met je inbreng en opmerkingen.

Geert Stappers

Nu beschikbaar voor assembler-programmeurs: EDFIX Versie 3.00

EDFIX vormt een geïntegreerde omgeving voor het programmeren in machinetaal op de IBM PC en compatibelen. Het bestaat uit een editor die lijkt op de editor uit omgevingen als Borlands Turbo Pascal en Turbo C. EDFIX is bedoeld voor gebruik met de Microsoft Macro Assembler (MASM) versie 4.0 of met Borlands Turbo Assembler (TASM) versie 1.0. Met EDFIX kan gemakkelijk een source gecreëerd, geassembleerd, gelinkt en getest worden. EDFIX vangt de output van de assembler af en zal in het geval van fouten meteen de sourceregel met de fout in de editor tonen. Project management, zoals o.a. in Turbo Pascal behoort ook tot de mogelijkheden. Een context-gevoelige HELP is beschikbaar onder de F1-toets. Deze HELP bevat onder andere alle assembler instructies en de meeste MASM sleutelwoorden. De editor is van het "WordStar-compatibele" type, zoals ook de editors onder Turbo Pascal en Turbo C.

Opm.: MASM en TASM zijn produkten met copyright. MASM is verkrijgbaar via Microsoft Inc, TASM is verkrijgbaar via Borland International. Beide pakketten maken geen deel uit van het EDFIX pakket.

Systeem configuratie

EDFIX is ontworpen voor gebruik op IBM PC/XT/AT en compatibelen. Als EDFIX niet op uw systeem werkt, is niet echt compatibel, ondanks het feit dat uw dealer beweerde van wel! De DOS versie moet 3.20 of hoger zijn. Bij gebruik van een EGA of VGA kaart wordt DOS 4.0x aanbevolen. EDFIX is te groot om van floppy disk te draaien: u heeft minimaal een 10 Mbyte harde schijf nodig.

Bestellen

Leden van de KIM Gebruikersclub Nederland kunnen EDFIX bestellen door *f* 7,50 over te maken op giro 3757649 t.n.v. de KIM Gebruikersclub Nederland. S.v.p. even aangeven om welk programma het gaat. Indien u een 3.5 inch diskette wenst, dan ook dit s.v.p. even vermelden.

Ruud Uphoff

Nogmaals: Packers

Eindelijk eens objectief vergelijkingsmateriaal over dit soort utilities. Alleen vindt ik het erg jammer dat je iets heel essentieels over PAK-2.51 helemaal buiten beschouwing hebt gelaten. Dit programma kan namelijk ook .ARC .ZIP en .ZOO archives inpakken en uitpakken (Helaas geen LHARC) Het grote voordeel daarvan is, dat je van die afschuwelijke ongestandaardiseerde puinhoop op al die BBS'n af bent.

Ruud Uphoff

Sja, daar heb je zeker een punt dat aan mijn aandacht ontglipt is. Op het bulletin board van de vereniging staat alles zo veel mogelijk in ZIP-formaat. Uitzonderingen hierop vormen de files in de SDN- en MINIX areas (respectievelijk PAK en LHArc 1 formaat). Persoonlijk gebruik ik PAK bijna niet. Het grootste nadeel van de ingebouwde formaten van PAK vind ik dat ik toch naar PkZip moet grijpen als er morgen een nieuwe zipper- versie uitkomt. Daarbij is de snelheid van PKZIP gewoon wat beter dan van PAK.

Wil je niet steeds geconfronteerd worden met een stapel losse programma's omdat je steeds ruzie hebt met de parameters tijdens het uitpakken, dan heb ik twee verschillende oplossingen. De eerste is het gebruik van SPAZ. SPAZ is een utility die zelf uitzoekt welke packer gebruikt is en aan de hand daarvan de juiste (un-) packer opstart met de juiste parameters. Het nadeel van SPAZ is vrijwel hetzelfde als dat van PAK, nieuwe packers herkent het ding niet. De tweede oplossing is alleen voor gebruikers van 4DOS geschikt. Maak een handvol environment variabelen als volgt:

set .LZH = c:\utils\lharc.exe e
set .PAK = c:\utils\pak.exe e
set .ARC = c:\utils\pkunpak.exe
set .ZIP = c:\utils\pkunzip.exe
set .ARJ = c:\utils\arj.exe e

Voor het uitpakken van de file "handig.lzh" kun je stomweg "handig" intikken. 4DOS expandeert dit automatisch naar "c:\utils\lharc.exe e handig.lzh".

Het artikel is helaas al weer enigszins verouderd. Van LHArc is namelijk een nieuwe versie uit (2.11 en zelfs 2.12 is al gesignaleerd) en de packer "ARJ" is in opkomst. Beide programma's bereiken een superieure compressiefactor (scheelt zo'n 4 tot 5 procent met ZIP!). ARJ is echter vreselijk traag, terwijl LHArc nu een redelijk goede snelheid haalt.

Joost Voorhaar

PATH: het path-mechanisme in DOS-65 gemaakt

Bij sommige computers is het niet nodig dat alle commando's in dezelfde directory zitten. Het is dan bijvoorbeeld mogelijk een lijstje van verschillende directory's op te geven. Als daarna een commando wordt ingetypt, dan zoekt het systeem volgens dat lijstje het commando op. Bij DOS-65 kan een commando in een andere directory wel uitgevoerd worden maar dan moet u bijv. 1:d/DEMO intypen. Daarin is DEMO dan een commando op directory d/ van drive 1. Vooral tijdens het maken van een nieuwe utility is het heel handig als DOS-65 ook het path mechanisme kent. Men dan ook niet langer verplicht om alle utilities in 0: te zetten.

De werking van DOS-65:

a) DOS-65 zoekt z'n commando's eerst op in een lijst, daarin staan bijv. DIR, LOAD en SAVE.
b) als het commando daar niet gevonden is, dan volgt een sprong naar $C832 en wordt het commando op de systeem-drive gezocht. Wordt het commando daar niet gevonden, dan volgt een foutmelding m.b.v. jmp $C847.
c) Als het gevonden is dan wordt het commando uitgevoerd m.b.v. jmp $C849.

De wijziging is niet zo erg moeilijk

Het zoeken van a) blijft bestaan maar als het commando niet gevonden is, volgt een sprong naar een klein extra programma dat staat op $BF00 en in het geheugen blijft staan. Dit extra programma zoekt niet alleen in de systeem-drive, maar ook in andere directory's. Daartoe wordt een lijstje bijgehouden dat direct achter het programma staat: pathlijst. Is het zoeken niet succesvol, dan volgt exact dezelfde sprong als bij DOS-65. Als het zoeken wel succesvol is dan wordt het betreffende commando uitgevoerd, ook weer door te vervolgen op dezelfde plaats waar DOS-65 dat zou doen.

PATH

Om te kunnen instellen welke drive/dir in de lijst staat, is een utility gemaakt waarbij een beetje gekeken is naar ASN. Deze utility test eerst of het path er al is, door te kijken welk adres op orgjmpa staat. Indien het nog niet geïnstalleerd is, dan wordt dat alsnog gedaan. Daarbij wordt wel bewaakt of u de juiste versie van DOS-65 heeft.
Als er geen opties en geen drive/dir zijn opgegeven bij PATH dan volgt een display v.d. waarden die nu in de pathlijst staan. Is er wel drive/dir is opgegeven dan worden die toegevoegd aan de pathlijst. De pathlijst mag echter niet langer dan 8 worden. Dat is voor de veiligheid want met een floppy disk zou het zoeken anders wel erg lang gaan duren.

Wordt als optie -R gegeven, dan wordt alleen de laatste van de pathlijst verwijderd. De rest v.d. pathlijst blijft dus bestaan. Hierbij is ook toegestaan in te geven PATH -R 1a. In dit geval wordt de laatste v.d. pathlijst verwijderd en daarna 1:A/ aan de pathlijst toegevoegd. Ook wordt ervoor gewaakt dat er geen lege lijst ontstaat want dan zouden er helemaal geen commando's meer uitgevoerd kunnen worden. Daarom wordt de eerste in de lijst alleen overschreven door de combinatie: PATH -c 1b [1c]. Daarbij wordt dus de gehele pathlijst uitgewist en dan worden 1:B/ [en evt. 1:C/] erin gezet. Dus PATH -c mag wel, maar dan wordt de lijst niet helemaal uitgewist. Het resultaat van een commando op de pathlijst is gemakkelijk zichtbaar te maken door de -? optie.

Assembleren van path.mac:

De variabele selectproc bepaalt voor welke processor er geassembleerd wordt:

```
selectproc   equ   nmos    standaard NMOS 6502
selectproc   equ   cmos    standaard CMOS 6502
selectproc   equ   rcmos   Rockwell CMOS 6502
```

Enige belangrijke variabelen in PATH:

```
opt       equ  $FE     vlag geset door het resident
                       programma
freemem   equ  $BF00   vrij gedeelte in DOS-65
                       (max 1 pagina )
csufl     equ  $ABD0   vlag voor commando
sdrive    equ  $ABD1   system drive
orgjmpa   equ  $C66B   de plaats van het originele
                       jmp adres.
orgjmpd   equ  $C832   de bestemming v.h originele
                       jmp adres.
comerr    equ  $C847   branch in DOS-65 als
                       commando niet gevonden is.
comfnd    equ  $C849   vervolg in dos65 als het
                       commando gevonden is.
```

Ter informatie:

De getallen die DOS-65 gebruikt voor de specificatie van de drive en de directory zijn tussen 0 en $1F. Enige voorbeelden:

```
$00 = 0:
$04 = 0:a/
$08 = 0:b/
$0C = 0:c/
$10 = 0:d/
$01 = 1:
$05 = 1:a/
$09 = 1:b/
$0D = 1:c/
$1D = 1:g/
```

H.Speksnijder

```
;file path.mac
;                                       19900815
;                                       Henk Speksnijder
;                                       Capelle a/d IJssel
;
;assembleer voor rcmos, cmos of nmos processor.
;default is RCMOS.
;
freemem     equ     $BF00       een pagina vrij geheugen in Dos65.
;                               PAS OP: er wordt een klein programma
;                               resident geinstalleerd op dit adres.
;                               De rest van deze utility is alleen nodig
;                               bij het installeren of wijzigen v.h. path.
;
;----------CONSTANTEN----------------------------------------
;
CR          equ     $0D         carriage return
space       equ     $20         space
;
;----------dos IO routines-----------------------------------
;
inch        equ     $C020       dos input character
prch        equ     $C023       dos output print character
crlf        equ     $C02F       print cr + lf
prhx        equ     $C038       print een byte hex
prtx        equ     $C03b       print text tot een nul
loupch      equ     $C041       accu lower to upper
sopt1       equ     $C068       get option
spar1       equ     $C06B       get parameter
open        equ     $D03F       open a file, AY wijst naar filenaam
close       equ     $D048       close a file, X is filenummer
ermes1      equ     $D0B7       error messages
;
;----------RCMOS, CMOS, NMOS---------------------------------
;
rcmos       equ     2           Rockwell cmos
cmos        equ     1           standard cmos
nmos        equ     0           gewone nmos
;
selectproc  equ     rcmos
;
            if      selectproc==rcmos
            opt     rc02
            elseif  selectproc==cmos
            opt     c02
            elseif  selectproc==nmos

ldai        macro   ad          vervang de cmos: lda [ad]
            ldy     #0
            lda     [ad],y
            endm

stai        macro   ad          vervang de cmos: sta [ad]
            ldy     #0
            sta     [ad],y
            endm
```

```
bra             macro   ad              vervang de cmos: bra ad
                jmp     ad
                endm

inca            macro                   vervang de cmos inc accu
                clc
                adc     #1
                endm

deca            macro                   vervang de cmos dec accu
                sec
                sbc     #1
                endm

                endif

                if      selectproc! = nmos

ldai            macro   ad              gebruik echte cmos instructie
                lda     [ad]
                endm

stai            macro   ad              gebruik echte cmos instructie
                lda     [ad]
                endm

                endif
;
;----------algemene MACRO'S----------
;
cpxa            macro   op              vergelijk woord in XA met operand
                if      !#,op
                cpx     op+1
                bne     200.f
                cmp     op
200
                else
                cpx     op>>8
                bne     200.f
                cmp     op&255
200
                endif
                endm

ldxa            macro   op              load met A = low byte en X = high byte
                if      !#,op
                lda     op
                ldx     op+1
                else
                lda     op&255
                ldx     op>>8
                endif
                endm

stxa            macro   ad
                sta     ad
                stx     ad+1
```

```
                endm

;------------------Dos 65 intern-----------------------------------------
;
                ;page zero
optmask         equ     $A0             byte om de optie's te bewaren
                ;enkele variablen gebruikt in Dos65.
t1              equ     $F0             temp (pointer) voor deze utility
t2              equ     $F2             temp (input byte) voor deze utility
opt             equ     $FE             flag gezet door residente prog. t.b.v. dos65.
csufl           equ     $ABD0           flag voor commando
sdrive          equ     $ABD1           system drive
orgjmpa         equ     $C66B           de plaats van het originele jmp adres.
orgjmpd         equ     $C832           de bestemming v.h originele jmp adres.
comerr          equ     $C847           branch in dos65 als commando niet gevonden is.
comfnd          equ     $C849           vervolg in dos65 als het commando gevonden is.

;---------------Opties-------------------------------------------------
;
erase           equ     $80             -E : Verwijdert het path en het residente programma
;                                       en herstelt Dos65 in originele staat.
clear           equ     $40             -C : Maakt de gehele path lijst schoon.
remove          equ     $20             -R : Verwijdert alleen de laatste van de path lijst.
show            equ     $10             -? : Laat actuele pathlijst zien op display.
help            equ     $08             -H : Laat help-text van PATH zien.
;
;-------------MAIN--hoofdprogramma------------------------------
;
                org     $A000
begin           jmp     pastart
                fcc     $C8,$C5,$CC,$D0  support Dos65: HELP
seehelp         jsr     prtx
                fcc     '\rDefines the command-path'
                fcc     '\rDefault is the system drive'
                fcc     '\rSyntax : PATH [-ECAR?H] [drive:directory] etc.'
                fcc     '\roptions :'
                fcc     '\r   -E erase entire path'
                fcc     '\r   -C clear the pathlist'
                fcc     '\r   -R remove last entry from the pathlist'
                fcc     '\r   -? show actual pathlist'
                fcc     '\r   -H Show this comment'
                fcc     '\rNo abbreviation.    max. 8 drive/dir specs'
                fcc     '\rthe characters / , : @ are optional'
                fcc     '\rExamples:'
                fcc     '\rPATH 0:@/ 0:d/    drive 0 main dir. and dir d/'
                fcc     '\rPATH 0a 0d 1a     drive 0 dir. a/ + d/ + drive 1 dir. a/'
                fcc     '\rPATH              show the current situation'
                fcc     '\rPATH -? 1a        add 1:a/ to the pathlist and print it'
                fcc     '\rPATH -R           remove last entry from path'
                fcc     '\r'
                fcc     '\rPATH -C 1:a       clear pathlist and then put 1:a/ in it'
                fcc     '\rremark: only this will change the first entry in pathlist'
                fcc     '\r',0
                rts
;
;BEGIN VAN PROGRAMMA
;
```

```
pastart     jsr     sopt1           sopt1 = scan opties
            fcc     'ECR?H',0
            bcc     1.f
            jmp     ermes1          error met optie's
1           sty     t1              save command buffer pointer
            sta     t1+1
            stx     optmask         save opties masker
            txa
            bit     #help
            beq     2.f
            jmp     seehelp         show help-text
            ;
2           ldxa    orgjmpa
            cpxa    #path           test of path al geinitialiseerd is
            beq     p2
            ;
            cpxa    #orgjmpd        test of adresses goed zijn
            beq     p1
            jmp     error1          onjuiste dos65 versie
            ;
p1          ldxa    #path           initializeer path
            stxa    orgjmpa         dos zal jmp uitvoeren naar path routine
            lda     sdrive
            sta     pathlis
            lda     #$FF            clear path list,
            sta     pathlis+1       only the first entry is there now.
            ;
p2          lda     optmask
            bit     #erase          indien nodig erase het path
            beq     1.f
            jmp     stop
1           bit     #clear          indien nodig clear het path
            beq     2.f
            lda     #$FF
            sta     pathlis         (sentinel op eerste positie )
            bra     10.f
2           bit     #remove         indien nodig verwijder laatste v.d. lijst
            beq     10.f
            jsr     findend
            cpx     #1              (er moet er 1 in de lijst blijven )
            beq     10.f
            dex
            lda     #$FF            (nieuwe sentinel )
            sta     pathlis,x
            ;
10          ldy     #0              scan i.v.m. empty string
            jsr     skipspa
            bne     p3
            jsr     tespath
            jmp     seepath
            ;
p3          jsr     loupch          scan voor drive:dir specificatie
            cmp     #'0'
            bcc     31.f            branch if accu<'0'
            cmp     #'4'            drive nummer
            bcc     32.f            branch if accu<'4'
31          jsr     error2
```

```
        jmp     p9
32      eor     #'0'
        sta     t2
        jsr     next
        beq     p4
        cmp     #':'
        bne     33.f
        jsr     next
        beq     p4
33      jsr     loupch          directory
        eor     #'@'
        cmp     #8
        bcc     35.f            branch if accu<8
        jsr     error3
        jmp     p9
35      asla
        asla
        ora     t2
        sta     t2
        jsr     next
        beq     p4
        cmp     #'/'
        bne     36.f
        jsr     next
        beq     p4
36      jsr     error2
        jmp     p9
        ;
p4      ldx     #0              zoek of deze drive/dir al in de lijst stond
1       lda     pathlis,x
        cmp     #$FF            ( sentinel )
        beq     p5
        cmp     t2
        beq     2.f
        inx
        bpl     1.b
        bra     p5
2       jsr     seedrdi         display drive/directory
        jsr     error4          "drive/dir already in the list"
        bra     p6
        ;
p5      cpx     #8
        bcc     1.f             branch if x<8
        jsr     error5          path is vol
        jmp     p9
1       lda     t2
        sta     pathlis,x       nieuwe entry in pathlist
        inx
        lda     #$FF
        sta     pathlis,x       (nieuwe sentinel )
        ;
p6      jsr     skipspa
        beq     p9              indien command line niet leeg dan continue
        jmp     p3
p9      jsr     tespath
        lda     optmask         indien nodig display actuele pathlijst
        bit     #show
```

```
            beq     9.f
            jsr     seepath
9           rts

tespath     lda     #$FF            Voor de veiligheid, want het Dos65 system
            cmp     pathlis         kan niet werken zonder drive.
            bne     9.f
            sta     pathlis+1
            lda     sdrive
            sta     pathlis
9           rts

stop        ldxa    #orgjmpd        herstel oude jump adres in dos65
            stxa    orgjmpa
            jsr     prtx
            fcc     'Path erased\r\r',0
            rts
error1      jsr     prtx
            fcc     '\rIncorrect Dos65 version',0
            rts
error2      jsr     prtx
            fcc     'Usage: PATH drive:directory\r',0
            rts
error3      jsr     prtx
            fcc     'Incorrect drive and/or directory assignment\r',0
            rts
error4      jsr     prtx
            fcc     'already in the path list\r',0
            rts
error5      jsr     prtx
            fcc     'path is full\r',0
            rts

seepath     jsr     prtx
            fcc     'Path ',0
            ldx     #0
10          jsr     seedrdi         display deze drive/directory
            inx
            lda     pathlis,x
            cmp     #$FF
            bne     10.b
            jsr     crlf
            rts

            ;display drive/directory
            ; het X-reg. geeft aan welk nummer van de pathlijst.
seedrdi     lda     pathlis,x
            and     #3              neem drive en zet om naar character
            ora     #'0'
            jsr     prch
            lda     #':'
            jsr     prch
            lda     pathlis,x
            and     #$1C            neem dir. en zet om naar character
            beq     11.f
            lsra
            lsra
```

```
                eor     #'@'
                jsr     prch
                lda     #'/'
                jsr     prch
11              lda     #space
                jsr     prch
                lda     #space
                jsr     prch
                rts

1               iny
skipspa         lda     [t1],y
                beq     9.f
                cmp     #space
                beq     1.b
                cmp     #CR
9               rts

next            iny
                lda     [t1],y
                beq     9.f
                cmp     #CR
                beq     9.f
                cmp     #space
9               rts

findend         ldx     #0              vind het einde van de pathlijst
1               lda     pathlis,x
                cmp     #$FF
                beq     2.f
                inx
                bpl     1.b
2               rts

;---------- PATH----------------------------------------------
;PAS OP: hier komt dan het programma dat resident in het geheugen blijft.

                org     freemem         vrij beschikbare gebiedje in dos65
path            sta     asave
                sty     ysave
                lda     #0
                sta     pacount         tel hoe ver in de pathlijst
                ;
1               ldx     pacount
                lda     pathlis,x       instellen systeem drive
                cmp     #$FF
                beq     6.f             indien $FF dan einde v.d. lijst
                sta     sdrive
                inc     pacount
                bmi     6.f
                lda     asave
                ldy     ysave           oude waarden van a en y terugzetten
                ;
                ldx     #$81            -\
                stx     opt              \
                stx     csufl             \__dit deel is exact als in dos65
                ldx     #$88              /
```

```
                jsr     open            __/
                bcc     8.f             carry = 0 = file found
                cmp     #$13            accu = $13 = file not found
                beq     1.b             herhaal tot het gevonden is.
                bra     7.f             ;opm.: als de accu< >$13 dan is er een file
                                        ;      gevonden waar iets mee fout is.

6               lda     #$13            error nummer 13 = file niet gevonden
7               jsr     restore
                sec
                jmp     comerr          commando niet gevonden

8               clc
                jsr     restore
                jmp     comfnd          commando gevonden

restore         ldy     pathlis         ( niet de ACCU veranderen )
                sty     sdrive          oude systeem drive terugzetten
                ldy     #0
                sty     csufl           clear csufl
                rts

asave           fcc     0               bewaren accu en Y-reg
ysave           fcc     0
pacount         fcc     0               tel hoe ver in de pathlijst

                ;pathlijst: max. acht drive/dir specs.
pathlis         equ     $
                ;opm.: gebruik geen "pathlis fcc 0"
                ;want dan wordt de lijst iedere keer overschreven
                ;als PATH uitgevoerd wordt.

;ter informatie:
;               $00 = drive 0 hoofddir
;               $01 = drive 1 hoofddir.
;               $04 = 0:a/
;               $08 = 0:b/
;               $05 = 1:a/
;               $09 = 1:b/
;               $0C = 0:c/
;               $10 = 0:d/
;               $0D = 1:c/
;               $1D = 1:g/
;               $06 = 0:c/

                end     pastart
```

Computalk

Vorige week ving ik in de trein een flard op van het volgende gesprek:

"Ik ben dus vorige week op cursus geweest hè... enne... nu kan ik alles met WeurdPeurfekt!"
- "Ja, ik ben ook al eens op zo'n kursus geweest, maar ik weet nog steeds niet precies hoe dat met die ef negen toets nou allemaal werkt... "
"Nou, daarmee kun je het einde van een veld aangeven hè"
- "Ja, dat zei die kursus lijder ook, maar ik begrijp er nog niks van."

Op dit punt aangekomen stopten we op het station van Deventer en ik moest overstappen op de trein naar Enschede. Het gebeurt wel vaker dat ik dit soort gesprekken opvang, en ik moet dan heimelijk altijd een beetje glimlachen. Niet omdat het voor mij wellicht zo eenvoudige stof is, alsof je aan een garagehouder gaat uitleggen hoe je moet schakelen, maar vooral om de toon van dergelijke gesprekken. Al snel ontstaat er een duidelijke "leerling-meester" situatie. De één verafgoodt de ander hoewel die cycloopje speelt in het land der blinden.

Deze cycloop der computerwetenschappen zat waarschijnlijk diezelfde avond nog naar een groen of geel oplichtend schermpje te staren en zich af te vragen wat al die rare termen nu eigenlijk écht betekenen. Hij heeft zich door mooie folders laten verlokken tot het opofferen van een vakantie zodat 'ie een nu een fraaie PC-privé computer thuis heeft staan. Vraag hem niet waarom, maar het leek zo mooi om een eigen computer te hebben...

Rare prietpraat eigenlijk, dat computertaaltje. Onze buren hebben allebei zo hun eigen uitdrukkingen voor de verschillende onderdelen en toepassingen. In Nederland schijnen we weer te moeten vervallen in een soort veredeld steenkolen-Engels. Maar zelfs in die Engelstalige therminologie zitten opmerkelijke verschillen. Waar de hele wereld over "hard disks" spreekt, heeft Big Blue IBM het nog altijd over "fixed disks". Hoe ze daar verwisselbare harddisks noemen weet ik eigenlijk niet...

De Duitsers weten trouwens ook wel weg met vreemde benamingen. De manier waarop informatie fysiek op een schijf wordt gezet heet bij hen een "Aufzeichnungsformat". Op "die Scheibe" kan een bestand staan... uhhh "ein Datei".

Jawel, ook in Nederland hebben we zo onze vertalingen. Het was een verademing een bekend clublid eens een computer te horen verkopen. Consequent gebruikte hij puur Nederlands taalgebruik, en dan nog wel zodanig dat het allemaal heel natuurlijk klonk ook. Geen rare half-Engelse benamingen als "diskette" maar consequent "schijfje". Onze zuiderburen heb ik die dingen overigens wel eens horen betitelen als "flodderschijf"... Gaan we wat dieper in de computer, dan staken zo'n beetje alle Nederlandse vertalingen. Een "stapelwijzer" is vrijwel overal een stackpointer en een connector sluit beter aan bij het ABCN (Algemeen Beschaafd Computer Nederlands) dan een aansluitpunt.

Ook softwarebedrijven die, al dan niet op maat gemaakte, software vertalen in het Nederlands hebben nogal eens de neiging tot in het krampachtige te blijven vertalen. dBase III heeft een wel heel speciale vertaling van de originele zinsnede "10 records left" gevonden: "10 velden links" heet het in de Nederlandse vertaling.

Een "stapelwijzer" is vrijwel overal een stackpointer

Tijden lang heb ik WordPerfect gezien als de "perfecte" vertaling. Dat dat ook wel tegenvalt blijkt eigenlijk wel heel erg snel; men spreekt bij WordPerfect wel netjes over "bestanden", maar als ze afgedrukt moeten worden dan heet dat opeens "printen" in plaats van "afdrukken". Papierladers heten er "sheetfeeders" en ze hebben het ook over een "disk" in plaats van over een schijf. De ene keer hebben ze het over een "index", de volgende keer weer eens over een "directory".

De verwarring groeit. Een "slot" is voor de Engelsen een gleuf of een gat, maar voor ons is het iets waarmee je iets stevig kan afsluiten tegen inbraak en vandalisme. Veel computers zijn dus voorzien van een Nederlands slot en een aantal Engelse slots. Geen sloten natuurlijk, want dan loop je het risico dat er water doorheen stroomt...

Nee, ik houd het maar op een steenkolen Engels. Of heet zoiets "sicillum Amerikaans"? Nou ja, "what's in a name?" zei Shakespeare al. Voorlopig heeft de computer hier nog steeds "computer", soms ook "rekendoor" en een enkele keer gewoon "dat kreng". Mijn keyboard blijft een keyboard en die drie-en-een-half-inch-schijfjes, sja, die blijven gewoon "flop" heten...

Driek

Computerkretologie

Naar aanleiding van een discussie tijdens de laatste bijeenkomst, wil ik in dit artikeltje een poging doen een aantal termen uit te leggen. De discussie ontstond over de plannen voor DOS-65 versie 3. Deze versie van DOS-65 ondersteunt multi-tasking en men vroeg zich af wat multi-tasking is en wat je er mee kunt doen.

In dit artikel zal ik, behalve het uitleggen van de termen, ook een poging doen alles in een historisch perspectief te plaatsen.

De computer-oudheid

In de oertijd van de computers (zo rond 1960) was een computer een ding dat ergens in een grote zaal stond en die alleen door "deskundigen" bediend kon worden. Om een dergelijke computer te laten draaien moest eerst het programma geladen worden. Dit gebeurde meestal door het inlezen van een hoeveelheid ponskaarten (van die dingen met gaatjes, zoals tot voor enkele jaren terug door de giro nog gebruikt werd). Als het programma ingelezen was, werd het gestart door het inbrengen van het startadres en het drukken op een RUN-knop. Nadat het programma afgelopen was, werd het resultaat afgedrukt op een printer of op ponskaarten.

Mensen die binnen de club de KIM (ja hoor, de club is genoemd naar een echte computer!) of de Junior meegemaakt hebben, kennen de manier van werken waarschijnlijk nog wel. Het programma werd ingelezen van een cassettebandje of via het toetsenbordje in machinecode ingetikt. Vervolgens werd in de zogenaamde Program Counter het startadres gezet en werd op de knop GO gedrukt. Als alles goed ging, dan kwam het resultaat op het display of op een ander uitvoerapparaat.

Bij de oude computers kreeg men op een gegeven moment ook de beschikking over allerlei hulpprogramma's. Dit waren bijvoorbeeld assemblers en zelfs echte compilers voor talen als Fortran, Algol en Cobol. Deze hulpprogramma's werden eerst op ponskaart verkocht, later ook op ponsband en magneetband. Het principe blijft echter hetzelfde. Om een programma te schrijven, gebeurde het volgende. De programmeur schreef zijn programma op zogenaamde coderingsformulieren waarvan in figuur 1 een voorbeeld afgedrukt is. Dit formulier ging ver-

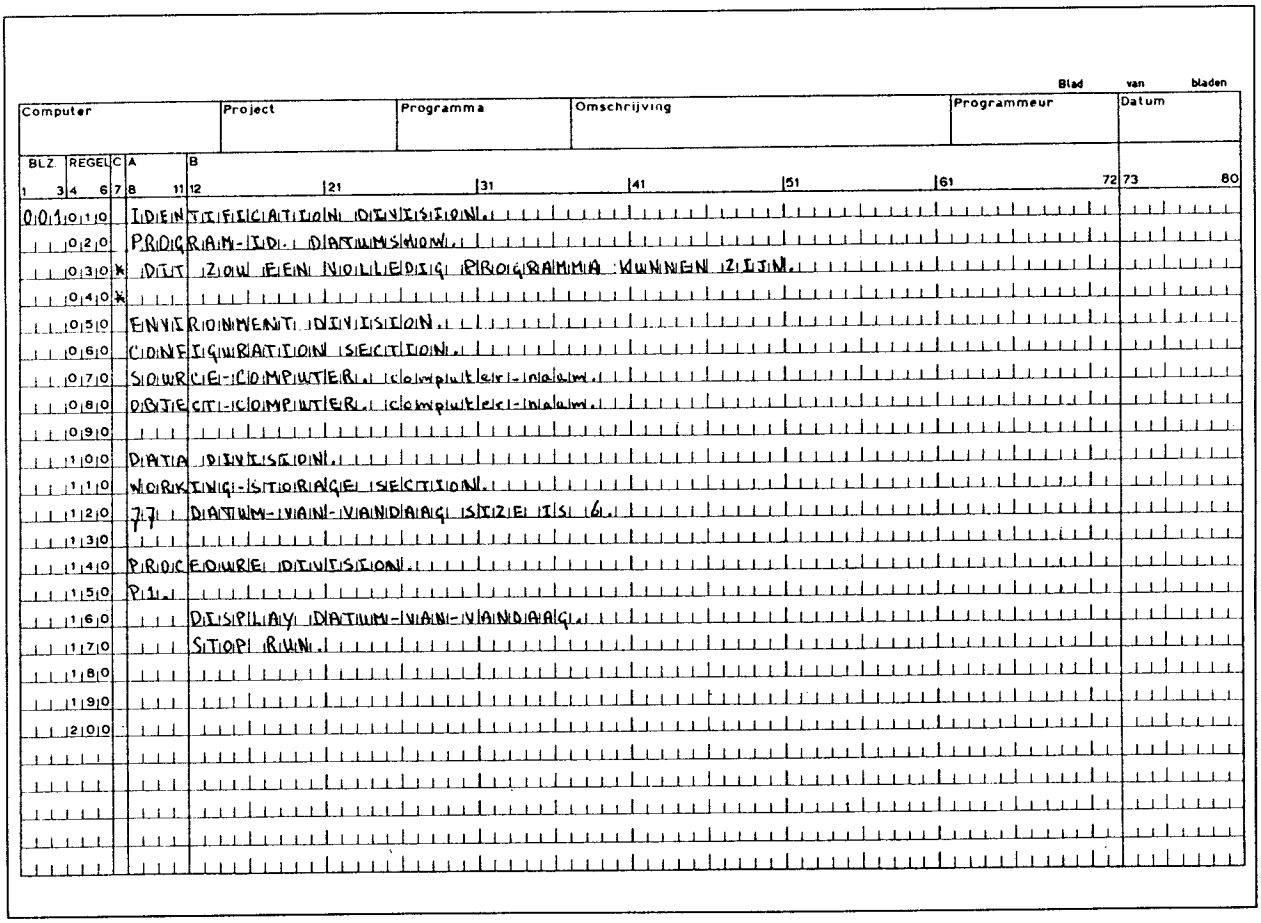


Fig. 1: voorbeeld van een COBOL coderingsformulier

volgens naar de ponskamer waar ponstypistes de informatie op het formulier overtypten op de ponskaarten. Vervolgens kreeg de programmeur een stapeltje ponskaarten terug.

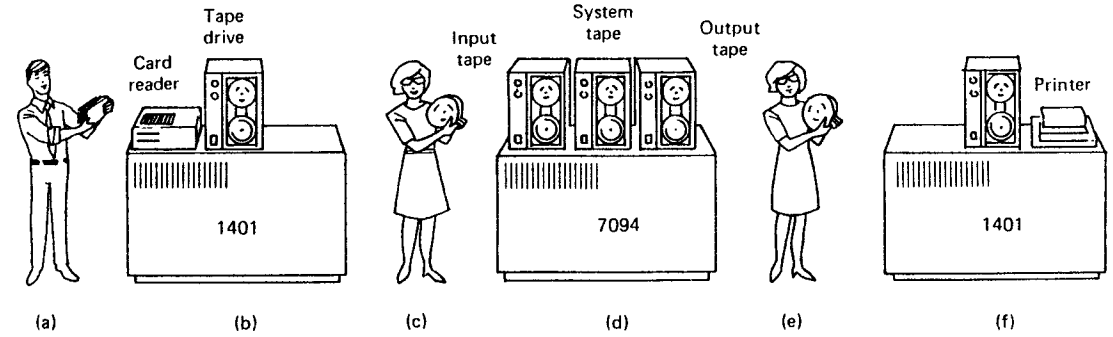
Nadat het programma "geponst" was, kon het ingelezen worden in de computer. De programmeur leverde zijn ponskaarten, vergezeld van een werkbriefje, in bij de balie van het computercentrum. Als het programma aan de beurt was, moest eerst de compiler geladen worden. Deze werd ingelezen vanaf ponskaarten of magneetband. Nadat de compiler gestart was, konden de ponskaarten ingelezen worden. De compiler ging vervolgens het programma onderzoeken op tikfouten, verkeerde opdrachten en dat soort zaken. Meestal werd er meteen een listing afgedrukt met daarbij alle fouten die door de compiler gevonden waren. Zaten er fouten in het programma, dan was de "JOB" klaar en kon de programmeur de fouten aanpassen. Voor de gewijzigde regels werden nieuwe ponskaarten gemaakt en het job werd opnieuw aangeboden.

Zeg dat het programma nu wel goed was, dan gaf de computer na de eerste fase (PASS 1) aan dat de tweede fase (PASS 2) kon beginnen. De ponskaarten moesten dan opnieuw worden ingelezen waarna de computer, in opdracht van de compiler, een nieuwe stapel ponskaarten of een magneetband produceerde met de zogenaamde object, het vertaalde programma. Soms was dit programma kant en klaar om gedraaid te worden, maar meestal moest het programma nog een bewerking ondergaan waarin allerlei subroutines aan het programma toegevoegd werden. Hiervoor was er een apart programma, de zogenaamde linker, die de object en de zogenaamde bibliotheken inlas en vervolgens de executable, het programma in uitvoerbare vorm, produceerde. Ook deze executable stond op ponskaart of magneetband.

Nadat alle voorgaande fasen goed doorlopen waren, kon het programma ingelezen en opgestart worden. Had het programma invoer nodig, dan werd deze invoer ook in de vorm van ponskaarten aangeleverd waarna het programma zijn resultaat op een printer uitprintte. Liep het programma ergens vast, dan werd meestal een zogenaamde "DUMP" geproduceerd waarin een overzicht stond van de inhoud van het geheugen van de computer. In enkele gevallen kon de programmeur daaruit zijn fout afleiden en herstellen.

Uit het voorgaande komen een aantal punten naar voren die voor het vervolg van dit artikel van belang zijn:

1 Een computer werkte slechts aan één klus tegelijk voor slechts één persoon.
2 De computer werd voor het uitvoeren van een programma steeds opnieuw met een programma geladen. Dit laden ging door middel van het inlezen van ponskaart, magneetband etc.
3 De computer voerde gewoon het programma uit dat als eerste ingelezen werd. Er was geen mogelijkheid de computer te vertellen welk programma hij uit moest voeren.
4 De computer had geen mogelijkheid om een archief van programma's of gegevens bij te houden die hij zelf beheerde.
5 Invoer werd regel voor regel ingelezen op het moment dat het programma het nodig had. Uitvoer werd direct op de printer uitgeprint.



An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

Fig. 2

6 Om een computer te laten werken waren er een groot aantal mensen nodig: programmeurs, ponstypistes, operators enz.
7 De tijd die verliep tussen het inleveren van de job en het moment waarop je het resultaat terugkreeg varieerde sterk.

Hoewel de KIM en de Junior iets later ontstaan zijn, werd de bovengenoemde manier van werken ook bij deze systemen gebruikt. Bij dit type computers werden de programma's ingelezen vanaf cassette en de resultaten werden uitgevoerd op een beeldscherm of iets dergelijks. De invoer kon wel vanaf een toetsenbord ingetikt worden.

Uit het bovenstaande komt duidelijk naar voren dat, om een computer te laten werken er een groot aantal handelingen door mensen uitgevoerd moesten worden. De mensen moesten ponskaarten inlezen, magneetbanden opspannen enzovoorts. Verder kon een men pas beginnen aan een volgende job als de vorige compleet afgewerkt was en de resultaten uitgeprint waren. Met name het inlezen vanaf ponskaart koste vrij veel tijd. Al spoedig ging men er daarom toe over het inlezen vanaf ponskaart op een aparte machine te doen die vervolgens magneetbanden aanmaakte waarop de jobs kan en klaar achter elkaar stonden. De computer kon dan gewoon vooraan beginnen en de jobs achter elkaar afwerken. De resultaten werden ook op magneetband geschreven die vervolgens op een aparte afdruk-machine uitgeprint werden. In figuur 2 is een dergelijke situatie getekend.

Het operating systeem

In de loop van de jaren zestig, werden de computers uitgerust met schijven waarop de informatie geschreven kon worden. Het inlezen van de jobs gebeurde nu niet meer op een aparte inleesmachine maar op de computer zelf die vervolgens de informatie op een schijf wegschreef. Ook het afdrukken van de resultaten ging nu op een andere manier. De resultaten werden ook op schijf weggeschreven. Voor het printen werden vervolgens de gegevens van schijf gehaald en uitgeprint. De computer deed dus drie dingen min of meer gelijktijdig: Het inlezen van nieuwe jobs, het verwerken van de jobs en het uitvoeren van de resultaten. Voor elk van deze taken had de computer aparte onderdelen die min of meer onafhankelijk van elkaar konden werken. De drie machines uit figuur 2 werden als het ware samengevoegd tot één machine waarbij de magneetbanden werden vervangen door één of meer schijven. Uiteraard moesten natuurlijk ook alle gegevens op de schijven netjes beheerd worden zodat de computer de jobs weer terug kon vinden. Deze manier van werken had nog een groot ander voordeel. Om een compiler te laden, hoefde men nu geen aparte magneetband meer in te lezen, de compiler stond gewoon op één van de schijven en de computer was in staat om hem op te zoeken en in te lezen. Hetzelfde gold ook voor de bibliotheken met standaard (sub-) routines.

Dit is eigenlijk het begin van een operating systeem. Omdat we hier te maken hebben met schijven, wordt dit ook wel een DOS of Disk Operating System genoemd. Andere namen voor het operating systeem zijn "bedrijfssysteem" of "besturingssysteem". Het operating systeem was in staat jobs te verwerken, gegevens en programma's van schijf te halen en

op te starten en resultaten op schijf weg te schrijven. Verder kon hij kopieren van ponskaart en magneetband naar schijf en van schijf naar printers, magneetband en eventueel ponskaart.

Interactieve systemen

In de loop van de jaren zeventig, kwamen er steeds meer systemen waarop men interactief kon werken. Dit in tegenstelling tot de batch-systemen zoals die in de vorige paragraaf beschreven zijn. Bij de batch-systemen wordt de complete job inclusief alle invoer voor de computer klaar gezet, bij de interactieve systemen wordt er met een persoon (de gebruiker) een dialoog gevoerd. De computer vraagt aan de gebruiker om invoer en verwerkt deze. Dit geldt niet alleen voor de invoer voor programma's maar ook voor uit uitvoeren van de programma's zelf. De gebruiker kan de computer opdracht geven een bepaald programma (bijvoorbeeld een tekstverwerker) op te starten waarna het programma in dialoog met de gebruiker zijn werk doet. Uiteraard moet het programma dan wel voor de computer beschikbaar zijn d.w.z. dat het programma op één van de schijven van de computer moet staan.

In vergelijking met de batch-systemen heeft een interactief systeem natuurlijk enorme voordelen. De programmeur hoeft nu zijn wijzigingen niet meer door de ponsafdeling op ponskaarten te laten zetten, hij kan meteen de tikfoutjes in zijn programma verbeteren. Verder hoeft de ponstypiste de gegevens niet meer op ponskaart te zetten, ze kan ze meteen intikken in de computer. Het is zelfs zover gekomen dat er helemaal geen ponskaarten en bijbehorende apparatuur meer bestaan. De gegevens (en de programma's) worden rechtstreeks in een computer ingetikt.

De meeste grotere computers kennen nog wel een batch-verwerking waarbij een job zonder tussenkomst van de gebruiker uitgevoerd wordt. In een aantal gevallen is dat handig. Denk bijvoorbeeld aan het genereren van betalingsopdrachten voor de sala-

rissen. Deze job moet één keer per maand lopen en de gegevens zijn elke maand vrijwel gelijk. Met een dialoogprogramma worden de wijzigingen in de loop van de maand ingebracht waarna het salarisprogramma zelf aan het einde van de maand in batch loopt.

Uiteraard is de afloop van een interactief programma vaak compleet anders dan die van een batchprogramma. Bij een interactief programma brengt de gebruiker de invoer tijdens het draaien van het programma in. Dit kan op diverse manieren: door middel van een vraag- en antwoordspel, invulformulieren, door het aanwijzen met een muis of joystick, het geven van commando's enz. Bij een batchprogramma wordt meestal een file geopend waarna de gegevens regel voor regel ingelezen worden.

Als voorbeeld een invulformulier uit bijvoorbeeld een databasepakket. De computer toont het formulier waarna de bedieningsman de velden één voor één in moet vullen. De computer controleert vervolgens de gegevens en geeft eventueel een foutmelding of brengt de gegevens aan in de database.

Alle computer-systemen die wij tegenwoordig kennen zijn interactieve systemen: DOS-65, MS-DOS, VAX/VMS, IBM AS/400, UNIX enz. enz. Hierbij zorgt het operating systeem ervoor dat alles goed afloopt.

Nemen we nu de huidige versie van DOS-65 of van MS-DOS als voorbeeld, dan valt op dat de computer slechts voor één gebruiker gelijktijdig werkt. De computer is zoals dat heet SINGLE USER". Verder draait er op de computer altijd maar één programma gelijktijdig. Dit wil zeggen dat het systeem "SINGLE TASKING" is; het programma kan slechts één taak gelijktijdig uitvoeren. Zowel DOS-65 als MS-DOS kunnen naast die ene taak die ze uitvoeren nog wel een zogenaamde "PRINTER SPOOLER" draaien zodat het uitprinten van gegevens wel parallel afloopt met uitvoeren van een programma, maar daarmee is de koek ook op.

Real Time

Alvorens in te gaan op het tegenovergestelde van single user en single tasking wil ik het eerst even hebben over de term "REAL TIME". De "RESPONSTIJD" van een systeem is de tijd die verstrijkt tussen moment waarop een opdracht aan de computer gegeven wordt en het moment waarop de computer antwoordt. Deze tijd kan, afhankelijk van de opdracht, kort zijn of zeer lang. In een aantal gevallen is het van groot belang nauwkeurig te weten hoe lang de responstijd van een systeem is. Denk bijvoorbeeld aan de computer van een elektriciteitscentrale die de stroomvoorziening bewaakt. Op het moment dat de spanning dreigt weg te vallen, moet de computer liefst binnen een fractie van een seconde reageren en een extra generator inschakelen. Als de tijd die de computer nodig heeft om te reageren op een gebeurtenis nauwkeurig bekend is en ook altijd vrijwel gelijk is, dan spreken we over een "REAL TIME" systeem. Je weet zeker dat de com-

Invoeren klantgegevens **03-04-991 20:30**

Klantnummer ..... : 125-4

Naam klant ..... : _____

Adres ..... : _____

Postcode + woonplaats ..... : _____

Landcode ..... : _____

Telefoonnummer ..... : _____ - _____

Kortingspercentage ..... : _____ %

Datum laatste order ..... : _____ - _____ - _____

Fig. 3: voorbeeld van een invulformulier

puter altijd binnen een fractie van een seconde reageert en niet zo nu en dan tien minuten nodig heeft om de klus waar hij mee bezig is af te maken. Real Time systemen komen met name voor bij de procesbesturing, zoals bij de bovengenoemde elektriciteitscentrale. Verder zijn ook de meeste grafische spelletjes tegenwoordig Real Time. Als we een spelletje Flight Simulator spelen, dan moet het vliegtuig natuurlijk wel continu doorvliegen en ook meteen reageren op de commando's van de piloot.

Multi Tasking

DOS-65 versie 3 ondersteunt Multi Tasking, evenals MINIX en nog een groot aantal andere besturingssystemen. Wat houdt dit in? Welnu, bij interactieve systemen is het meestal zo dat de computer een groot deel van de tijd niet aan het programma (bijvoorbeeld de tekstverwerker) kan werken omdat de gebruiker langzaam tikt of nadenkt over wat hij de computer voor opdracht kan geven. Het zou dan natuurlijk mooi zijn als de computer intussen iets anders zou kunnen doen; een andere taak uitvoeren. Het is tenslotte zonde een dure computer gedurende 90 % van de tijd alleen maar te laten wachten op invoer van de gebruiker. Binnen een Multi Tasking operating systeem kan de computer aan meerdere taken tegelijk werken. Zo kan de man aan het toetsenbord bijvoorbeeld een nieuw programma intikken terwijl de computer ondertussen de vorige versie van het programma aan het compileren is en ook nog bewaakt dat de koffie niet te heet wordt. De aandacht van de computer wordt dus als het ware verdeeld over meerdere taken.

Gert, het is bedtijd !

Uiteraard moet de aandacht van de computer natuurlijk zo verdeeld worden dat iedereen zo goed mogelijk tevreden is. De man die met de tekstverwerker werkt zal het niet prettig vinden als de computer tussen de "e" en de "r" die hij intikt eerst een half uur gaat compileren. Hiervoor moet het operating systeem een strategie hebben die er voor zorgt dat met name de interactieve gebruiker geen hinder ondervindt van de rest van de taken.

Afhankelijk van de omstandigheden kan men op diverse manieren met een Multi Tasking systeem omgaan. Het club-bulletinboard bedient momenteel twee telefoonlijnen. Verder kan de Sysop gelijktijdig ook nog wat met het systeem doen. Hier draait het systeem dus minimaal drie taken; twee voor de twee telefoonlijnen en één voor de sysop. Een andere mogelijkheid hebben we bij de Amiga. Hier kun je aan een taak een zogenaamd window toekennen waarin de in- en uitvoer van de taak geschreven wordt. Door nu van window te wisselen, kan de gebruiker bijvoorbeeld even iets in het systeem opzoeken of een programma draaien en later weer verder gaan met bijvoorbeeld de tekstverwerker. Verder kun je natuurlijk ook een wekkertaak laten lopen die netjes om 22:30 waarschuwt met "Gert, het is bedtijd" of eenvoudigweg alle processen afbreekt en middels een relais de spanning uitschakelt. Met behulp van een Multi Tasking operating systeem kan een computer eventueel ook aan meerdere batchjobs, al dan niet in combinatie met een dialoogtaak werken.

Multi User

Als een computer al Multi Tasking is, dan is het nog maar een kleine moeite om het systeem Multi User te maken. Bij Multi User bedient de computer meerdere gebruikers gelijktijdig. Dat betekent dus dat er niet één bedieningsman aan een toetsenbord plus beeldscherm zit te werken, nee dat kunnen er wel enkele honderden zijn. Die gebruikers hebben in dat geval meestal een zogenaamde (beeldscherm-) terminal op hun bureau staan. Dit is eigenlijk een toetsenbord en beeldscherm op afstand. Tegenwoordig zie je echter ook vaak dat er in plaats van terminals gebruik wordt gemaakt van een PC of ander type kleine computer waarop dan een terminalprogramma (bijvoorbeeld Kermit of Telix) draait. De PC is dan meestal via een seriële RS-232 kabel met de computer verbonden.

Afsluiting

In de komende maanden zullen we binnen de club nog uitgebreid te maken krijgen met Multi Tasking en Multi User. Van de lopende projecten heeft DOS-65 versie 3 Multi Tasking mogelijkheden. MINIX heeft zowel Multi Tasking als Multi User mogelijkheden. Aangezien beide projecten in de μP Kenner verder begeleid worden, komen we op deze materie in de vorm van artikelen nog terug.

Op zeer korte termijn (25 mei) houdt Geert Stappers een voordracht over OS-9. Dit operating systeem heeft zowel Real Time als Multi Tasking en Multi User. Op de bijeenkomst zullen deze facetten door Geert in zijn lezing en op de aanwezige OS-9 systemen gedemonstreerd worden. Verder zal er op de bijeenkomst minimaal één MINIX systeem aanwezig zijn; helaas nog draaiend op een PC of iets dergelijks en nog niet op de KGN-68k.

Gert van Opbroek

Methoden en technieken voor datacommunicatie (Deel 7)

Inleiding

De vorige keer hebben we ons bezig gehouden met enkele protocollen voor file-transfer. In deze aflevering wil ik eens een heel ander onderwerp uit de datacom-wereld pakken, namelijk de netwerken. Strikt genomen hebben netwerken evenveel met datacommunicatie te maken als een auto met wielen te maken heeft: zoals een auto wielen gebruikt om te kunnen rijden, zo maken netwerken gebruik van datacommunicatie. U begrijpt het waarschijnlijk al, een netwerk gaat heel veel verder dan alleen maar datacommunicatie.

Waarom netwerken?

In een netwerk worden meerdere computers en/of randapparaten met elkaar verbonden op een zodanige wijze dat vanuit elk knooppunt elk ander knooppunt bereikt kan worden. Het hele computernet kan dan opgevat worden als één heel groot systeem waarin computers, schijven en randapparaten gemeenschappelijk bezit zijn. Zo kan de dure laserprinter die aangesloten is op de computer van afdeling XXX opdracht krijgen van de computer op afdeling YYY om files die op de harde schijf van de computer van afdeling ZZZ staan uit te printen. Op deze manier kan men een randapparaat (bijv. laserprinter) economischer gebruiken en hoeft men ook voor het uitwisselen en kopieren van informatie niet met floppies heen en weer te lopen; je kunt vanuit je eigen computer op eenvoudige wijze de informatie op een andere computer bereiken.

Een ander voordeel van een netwerk is dat de computer-capaciteit zelf beter benut kan worden. Omvangrijke berekeningen hoef je niet op je eigen kleine 8088 PC op 4.77 MHz uit te voeren, die laat je uitvoeren op de 80386 PC op 33 MHz met 80387 coprocessor van je baas die de machine toch alleen maar uit status-overwegingen op zijn bureau gekregen heeft. Je kunt intussen je eigen PC gebruiken voor een spelletje space-invaders of flight simulator terwijl de computer van je baas het werk voor je doet. Nadat de 80386 zijn klus geklaard heeft en het spelletje ten einde is, haal je via het netwerk de resultaten van de berekeningen op die je lokaal met de tekstverwerker verder bewerkt. Tenslotte laat je het rapport uitprinten op de super-de-luxe Laserprinter van de afdeling verkoop.

Dat laat je uitvoeren op de 80386 PC op 33 MHz met 80387 coprocessor van je baas die de machine toch alleen maar uit status-overwegingen op zijn bureau gekregen heeft.

Een netwerk waarin de taken over de systemen verdeeld kunnen worden heeft ook als voordeel dat de totale computer-capaciteit (en dus de investering) minder groot behoeft te zijn. Meestal worden namelijk in een organisatie niet alle computers gelijktijdig even zwaar belast. Zo is er in Amerika een computernetwerk waarop zowel computers aan de oostkust als computers aan de westkust aangesloten zijn. Nu is het uiteraard zo dat het meeste werk tussen 9 en 17 uur lokale tijd gebeurt. Zou je alle programma's lokaal draaien, dan betekent dit dat de computer alleen tussen 9 en 17 uur optimaal benut wordt; buiten deze periode is er nog nauwelijks werk voor het systeem. De investering wordt dus maar voor pakweg 30% gebruikt. Nu wil het geval dat als het aan de oostkust 9 uur is, het aan de westkust nog pas 6 uur is. Als de computer aan de oostkust het drukker krijgt, is het op de computer aan de westkust nog enkele uren rustig. Hetzelfde is het geval als het in het westen 17 uur is, dan is het in het oosten al 20 uur. Door nu via een netwerk het werk te spreiden over meerdere computers in het land, kunnen programma's (jobs) die om 9 uur opgestart worden in het oosten, draaien op de rustige computer in het westen. Op deze manier gebruik je het computersysteem vanaf 9 uur in het westen tot 17 uur in het oosten. Dit betekent dat je effectief het computersysteem 11 uur benut. Ideaal zou natuurlijk zijn, als je computers over de hele wereld in een netwerk met elkaar zou kunnen verbinden. Op dat moment is er altijd wel een computer te vinden in een gebied waar het rustig is. Een ander voordeel van een dergelijk netwerk, is het feit dat je toch door kunt werken als één van de computers niet beschikbaar is. Als een computer in onderhoud is, doe je je werk gewoon op een andere computer in het netwerk. Was er geen netwerk, dan konden alle gebruikers van de computer die niet beschikbaar was, meteen een vrije dag nemen.

Het mooiste is natuurlijk als het operating systeem zelf uitzoekt op welke computer een job het beste kan draaien. Voorwaarde is dan natuurlijk wel dat alle benodigde gegevens op het moment van draaien ook daadwerkelijk op die computer aanwezig zijn. In de praktijk zijn er inderdaad al enkele operating systemen beschikbaar die hiertoe in staat zijn. Bij mijn weten kan dat echter alleen nog maar als de compu-

ters niet meer dan enkele meters van elkaar verwijderd zijn. In een dergelijk systeem (ook wel een cluster genoemd) worden de schijven namelijk verbonden met alle computers in het netwerk zodat elke computer de schijven als "zijn" schijven ziet en de informatie dus altijd op elke computer aanwezig is.

In het bovenstaande verhaal staat de hardware centraal binnen het netwerk. Met behulp van het netwerk wordt een klus (rekenopdracht, printopdracht) daar neergelegd waar hij het beste past. Een ander, veel voorkomend, fenomeen is het geval waarin de data (gegevens) centraal staat in het netwerk.

Neem bijvoorbeeld een groot bedrijf in elektronische onderdelen. Het bedrijf heeft drie winkels en doet in postorders. Voor die postorders hebben ze een aparte afdeling. Omdat het een groot bedrijf is, heeft het bedrijf verder een afdeling inkoop, een afdeling klantenservice, een afdeling inkomend goed, een financiële afdeling en een afdeling verzending. De voorraad aan onderdelen wordt in een database vastgelegd. In deze database staat voor elk onderdeel precies hoeveel er in voorraad is, hoeveel in bestelling en welke klanten het onderdeel besteld hebben. Verder staat aangegeven waar het onderdeel te vinden is, zowel voor het centrale magazijn als voor de drie winkels (winkel 2, stelling 14, plank 7, bakje 3, vakje D; dus locatie W2-14-07-03-D). Uiteraard moeten alle bovengenoemde afdelingen deze informatie kunnen benaderen: in de winkel vraagt een klant om een 6502 en men moet kunnen nagaan of er in een stoffig hoekje nog zo'n ouderwetse (grapje!) 6502 ligt en zo ja, in welk stoffig hoekje ze hem kunnen vinden. De afdeling klantenservice moet voor een bestuurslid van de KGN die telefonisch vraagt of het bedrijf vijftig 68030's in 25 MHz uitvoering kan leveren het bestand kunnen benaderen om op te vragen hoeveel er nog in voorraad zijn en hoelang de levertijd eventueel is. De afdeling postorders moet de binnengekomen orders verwerken en ook daarvoor is het nodig na te kunnen gaan hoeveel van een bepaald artikel in voorraad is. Kortom alle afdelingen moeten gebruik kunnen maken van de bovengenoemde database.

Men moet kunnen nagaan of er in een stoffig hoekje nog zo'n ouderwetse (grapje!) 6502 ligt .

Nu kun je natuurlijk elke afdeling een eigen computer geven met elk hun eigen kopie van de database maar dat heeft hele grote nadelen. Stel er is van de 6502 nog net één exemplaar aanwezig in winkel 1. De handelaar verkoopt deze processor aan een klant die voor het eerst sinds jaren hierom komt vragen. Uiteraard verwerkt de bediende de voorraadmutatie in zijn kopie van de database. Gelijktijdig komt er bij de afdeling postorders een fax binnen waarmee ook een 6502 besteld wordt. De persoon die deze order afhandelt kijkt in zijn kopie van de database en ziet dat er inderdaad nog één is, waarna hij opdracht geeft aan verzending de bestelling klaar te maken en de financiële afdeling opdracht geeft de factuur aan te maken. De 6502 is echter al weg doch vanwege het feit dat iedereen met een kopie van de database werkt is hij intussen wel tweemaal verkocht. De klant van de postorder zal waarschijnlijk wel een factuur, maar niet de 6502 krijgen. Hij belt met klanten-service die ondertussen een bijgewerkte kopie van de database gekregen heeft. Klanten-service kan dan zien dat de 6502 twee maal verkocht is en zal zeggen dat het een foutje van de computer is en vertellen dat ze een zogenaamde credit-nota zullen sturen en de order opnieuw in behandeling zullen nemen met een levertijd van 8 weken.

Uit het bovenstaande verhaal blijkt wel dat het niet verstandig is met kopieën van de database te werken. Elke afdeling moet toegang hebben tot dezelfde database en elke wijziging in de database moet ook meteen bij elke afdeling bekend zijn. Ook in dat geval biedt een netwerk uitkomst. De database wordt centraal op slechts één plaats bewaard en alle afdelingen hebben via het netwerk toegang tot deze ene database. Op deze manier krijgt elke afdeling altijd de laatste stand en weet men precies wanneer een bepaald artikel uitverkocht is. Deze ene database zal meestal op de schijven van één computer in het netwerk aanwezig zijn. Dit is dan de zogenaamde file-server; een computer die alleen het beheer van de database doet en verder niets anders. Eventueel kunnen meerdere computers gezamelijk de database beheren. Men spreekt dan over een zogenaamde gedistribueerde database. In het bovenstaande voorbeeld houdt dan de computer van elke winkel zijn eigen voorraad bij en een aparte file-server die van het centrale magazijn. Als de bestanden van de winkels en het centrale magazijn toch samen één database vormen, dan spreken we van een gedistribueerde database.

Met moderne netwerk- en database-software worden alle toegangen tot de al dan niet gedistribueerde centrale database zodanig geregeld dat de persoon die met de gegevens werkt hier niets van merkt. Hooguit aan het feit dat er geen lampje op de lokale harde schijf gaat branden kan afgeleid worden dat de gegevens "ergens anders" vandaan komen. Uiter-

aard worden de mutaties ook meteen in deze centrale database aangebracht waarbij voor het geval dat twee afdelingen gelijktijdig vechten om een 6502 er voor één van de twee een foutmelding gegeven moet worden. Ook dat regelt de database-software. In het moderne computer-jargon wil dit alles zeggen dat het netwerk en de database transparant zijn voor de gebruiker. De gebruiker merkt eigenlijk geen enkel verschil tussen het werken met een lokale database op zijn eigen computer en de centrale database op de file-server.

Local of Wide of Global, wat wilt u?

Heel veel mensen zullen de kreet Local Area Network of LAN we eens gehoord hebben. Minder bekend is de kreet Wide Area Network of WAN. Welnu, een Local Area Network, de naam zegt het al, is een netwerk dat een beperkt gebied bestrijkt. Dit kan bijvoorbeeld een fabrieksterrein zijn of een kantoorgebouw of zelfs het huis van een KGN-lid. Binnen een LAN worden apparaten verbonden die hooguit enkele honderden meters van elkaar af staan.

Wordt het grondgebied dat het netwerk bestrijkt groter, dan komen we in het gebied van de WAN's; de Wide Area Networks of ook wel Long Haul Network genaamd. Dit zijn netwerken die een veel groter gebied bestrijken; denk bijvoorbeeld aan een computernetwerk dat alle geldautomaten met elkaar verbindt. Eventueel kan men zelfs zover gaan dat het netwerk de hele wereld bestrijkt waardoor we zouden kunnen gaan praten over een GAN; een zogenaamd Global Area Network. Een dergelijk groot netwerk wordt meestal opgebouwd uit WAN's die met elkaar verbonden zijn.

Het is wel aardig de opdeling in computernetwerken te gaan vergelijken met de opdeling in het openbare telefoonnet uit aflevering 2 in deze serie. Een Local Area Network kun je vergelijken met een lokaal telefoonnet. Een Wide Area Network kun je vergelijken met een nationaal telefoonnet en door de nationale telefoonnetten met elkaar te verbinden, krijg je een groot internationaal netwerk.

Uiteraard is de problematiek die speelt in een LAN anders dan de problematiek die speelt in een WAN. In een LAN worden de verbindingen bijna uitsluitend door middel van kabels van koperdraad of glasvezel gelegd, in een WAN kan men ook gebruik maken van (radio) straalverbindingen en zelfs satellieten. Ook het aantal stations dat aangesloten is op het netwerk kan in een WAN veel groter zijn. In een LAN hebben we te maken met enkele stations tot hooguit een paar honderd, in een WAN kan dat oplopen tot enkele duizenden. Verder speelt in een LAN de snelheid waarmee de informatie zich door de verbindingen voortplant nauwelijks een rol terwijl men bij satelliet-verbindingen toch rekening moet houden met een vertraging van ongeveer een halve seconde (let bijvoorbeeld maar eens op het NOS Journaal als men vanuit Hilversum een vraag stelt aan iemand in Washington).

Om het allemaal niet te ingewikkeld te maken en omdat het aanleggen van een netwerk naar alle leden van de KGN toch niet aan de orde is, zal ik mij in dit artikel beperken tot de LAN's.

Technische uitvoering

1: De netwerk-topologie en de adressering

Netwerken zijn er in verschillende uitvoeringen. Eén van de eerste criteria is de topologie of anders gezegd de plattegrond van het netwerk. In figuur 1 zijn enkele voorbeelden van netwerk-topologieën getekend.

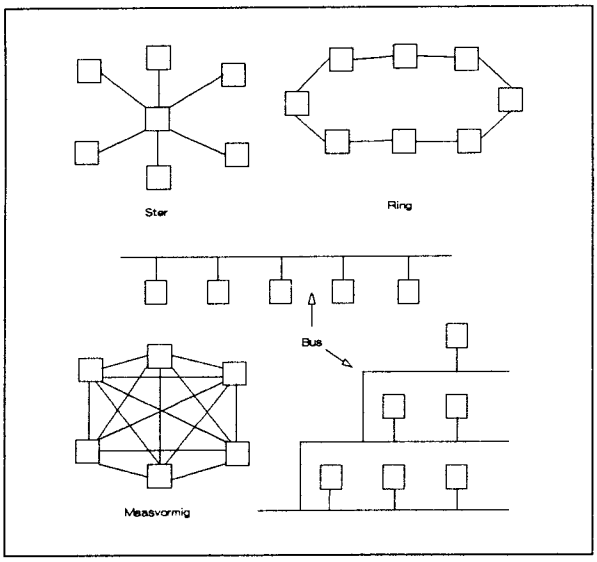


Fig. 1: enkele netwerktopologieën

Uiteraard heeft elke plattegrond zijn eigen specifieke voor- en nadelen. Zo zijn in een maasvormig-netwerk de bekabelingskosten vrij hoog. Het is echter zeer eenvoudig de informatie op de juiste plaats te krijgen, de afzender heeft een rechtstreekse verbinding met de ontvanger. Een ster-netwerk heeft als nadeel dat het netwerk niet meer gebruikt kan worden als het centrale knooppunt niet werkt. Het busnetwerk heeft als voordeel dat de bekabelingskosten relatief laag zijn. Nadeel is echter dat er een aantal maatregelen genomen moeten worden voor het op een ongestoorde wijze overdragen van informatie. De oudste vormen van netwerken hadden een ringstructuur. Elk knooppunt in een dergelijke structuur kan berichten ontvangen van zijn beide buren en ook

doorsturen naar zijn twee buren. Het grote voordeel van een dergelijk netwerk is, naast de relatief lage bekabelingskosten het feit dat het netwerk nog compleet blijft functioneren als één van de verbindingen verbroken wordt; je stuurt een bericht naar je linker buurman dan gewoon rechtsom.

Uiteraard is de wereld niet zo eenvoudig als in de plaatjes in figuur 1. Computer-netwerken bestaan meestal uit een mengsel van verschillende plattegronden. Ook worden vaak meerdere netwerken gekoppeld tot één groter netwerk.

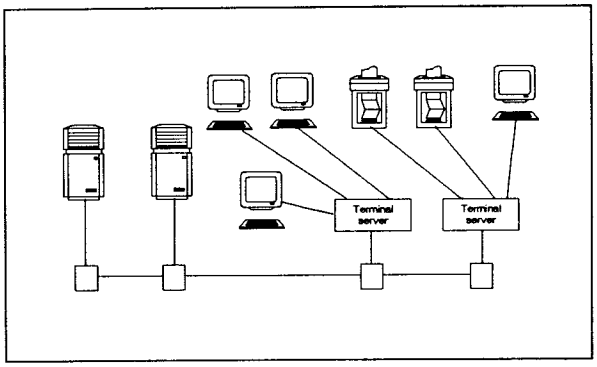


Fig. 2: praktrijkvoorbeeld van een computernetwerk

In figuur 2 is een praktijkvoorbeeld van een computernetwerk (LAN) getekend. Dit netwerk is afgeleid van een installatie met VAX-computers van de firma Digital Equipment Corporation of kortweg DEC. In dit netwerk zien we twee VAX-computers die aangesloten zijn op een zogenaamde Ethernet-kabel. Aan deze Ethernet-kabel zijn twee zogenaamde DEC-servers aangesloten waarop een aantal printers of beeldschermen (terminals) aangesloten kunnen worden. De aansluitingen van de printers en terminals aan de DEC-servers gaat door middel van een standaard RS-232 of RS-423 (vrijwel gelijk aan RS-232) verbinding. De communicatie tussen de computers onderling en tussen de computers en de DEC-servers gaat door middel van DECnet; een pakket voor communicatie via een Ethernet kabel.

In dit netwerk hebben de beide VAX-computers en de DEC-servers een eigen adres. Dit is de zogenaamde node-naam of knooppunt-naam. Een dergelijke naam dient uniek te zijn in het LAN. De DEC-servers hebben meerdere aansluitingen, de zogenaamde poorten. Voor elke server kan met voor elke poort een aparte naam instellen. Deze naam dient uniek te zijn binnen de server. Op deze manier kan men terminals en printers benaderen door hun combinatie van knooppuntnaam en poortnaam bijvoorbeeld "Gebouw_B/Printer 3". De VAX-computers krijgen ook elk hun eigen naam bijvoorbeeld "Castor" en "Pollux", naar de beroemde tweeling uit de oudheid. Om nu informatie op één van de twee computers te benaderen geeft men de volgende combinatie van gegevens: Knooppunt, device (schijf), directory-pad, filenaam. Bijvoorbeeld:

Castor::$DISK1:[verkoop.dbase]artikelbestand.idx

Uiteraard is het zo dat je niet altijd het hele pad op behoeft te geven. Ben reeds ingelogd op Castor en sta je op de directory [verkoop.dbase] op $DISK1, dan hoef je alleen de filenaam op te geven. In andere netwerken gaat dit allemaal op ongeveer dezelfde manier waarbij uiteraard de notatie zal verschillen.

2: Het medium

Nadat de plattegrond van het netwerk bepaald is, moet beslist worden met wat de diverse verbindingen gemaakt worden. In een LAN zijn daarvoor een aantal mogelijkheden. In de eerste plaats zijn er de zogenaamde Twisted Pairs. Dit zijn twee geïsoleerde koperdraadjes die om elkaar heen gedraaid zijn. De meest bekende toepassing van twisted pairs is het telefoonnet dat tot voor kort compleet uitgevoerd met dergelijke verbindingen. Een tweede mogelijkheid is de zogenaamde coax-kabel die we ook kennen als antennekabel voor de televisie. Een plaatje van een stuk coax-kabel is afgebeeld in figuur 3.

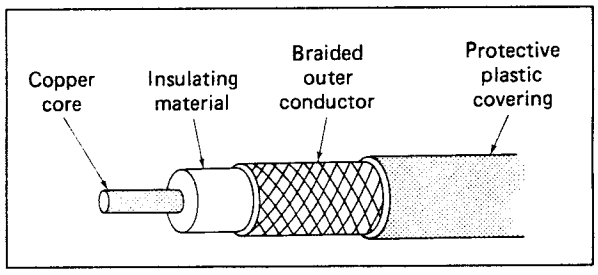


Fig. 3: doorsnede van een coaxkabel

Een derde mogelijkheid, die tegenwoordig steeds meer gebruikt wordt, is de zogenaamde glasvezelkabel. Hierin wordt de informatie niet door middel van een elektrisch signaal overgebracht maar door middel van een lichtsignaal. Dit lichtsignaal wordt als het ware "gevangen" in een zeer dun draadje (0.00001 mm tot 0.0003 mm) van glas of soms van kunstof waardoor men kilometers verder, aan de andere kant van de vezel, het licht weer op kan vangen. Een voorbeeld van een lichtstraal die gevangen is in een glasvezel is getekend in figuur 4.

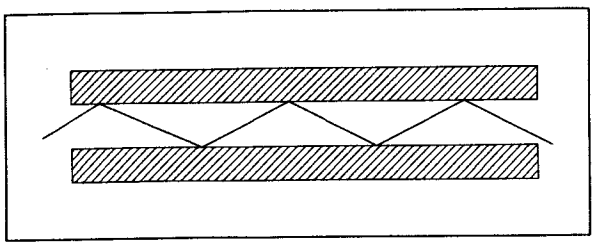


Fig. 4: lichtstraal gevangen in een glasvezel

Via het medium worden de bits overgestuurd. Dit kan in principe op twee manieren. De eerste manier is gebruik te maken van een zogenaamde Baseband. Hierbij worden de bits min of meer rechtstreeks aan het medium aangeboden. Geen spanning of licht betekent dan bijvoorbeeld een 0, licht of een spanning van 5 volt betekent dan een één.

De tweede mogelijkheid is modulatie. Er wordt een signaal met een bepaalde frequentie (of kleur licht) op het medium gezet. Deze draaggolf wordt vervolgens in frequentie, amplitude of fase gemoduleerd aan de hand van de bits die overgestuurd moeten worden. Deze techniek is besproken in aflevering 3 van deze serie toen we het gehad hebben over de werking van modems. Een dergelijke vorm wordt Broadband genoemd omdat men in principe meerdere signalen, gemoduleerd op meerdere draaggolven gelijktijdig door het medium kan sturen.

Vanwege de veel hogere kosten, wordt er in LAN's vrijwel uitsluitend gebruik gemaakt van de Baseband. Wel worden de bits vaak gecodeerd door middel van een zogenaamde Manchester Encoding of Differential Manchester Encoding. Het nadeel van het zondermeer aanbieden van de bits in de vorm van een spanning is namelijk het ontbreken van een start-kenmerk voor een bit. Dit betekent dat er een zeer nauwkeurige gelijkloop tussen zender en ontvanger moet zijn want als de bittijd van de ontvanger 10% langer is dan die van de zender, dan denkt de ontvanger 9 bits ontvangen te hebben terwijl er 10 overgestuurd zijn. Dit kan opgelost worden door midden in een bit ook de spanning te wijzigen. Een 1 is dan een spanning van 5 volt gedurende de halve bittijd, gevolgd door een spanning van 0 volt gedurende de tweede helft. Bij een nul is dit dan uiteraard 0 volt gevolgd door 5 volt. Deze vorm van codering is de zogenaamde Manchester Encoding. Bij de Differential Manchester Encoding wordt dit principe ook gebruikt, echter betekent een overgang aan het begin van een bit dat het bit 0 is en geen overgang betekent een bit 1. Dus een 0 bestaat uit een spanningswijziging aan het begin van het bit en een spanningswijziging midden in het bit en een 1 heeft alleen een spanningswijziging in het midden van een bit. De beide vormen van Manchester Encoding hebben dus altijd een overgang midden in het bit. Deze overgang kan gebruikt worden om zender en ontvanger te synchroniseren.

De vormen van codering zijn in figuur 5 nog eens onder elkaar getekend.

3: Het oversturen van informatie

Over het medium wordt informatie uitgewisseld. Uiteraard wordt de informatie opgedeeld in zogenaamde packets die worden voorzien van de nodige informatie voor het corrigeren van fouten en het identificeren van afzender en geadresseerde. Het versturen van dergelijke packets over een netwerk kan op verschillende manieren die weer afhangen van de netwerk-topologie. Zo kan de informatie in een maasvormig netwerk op eenvoudige wijze over-

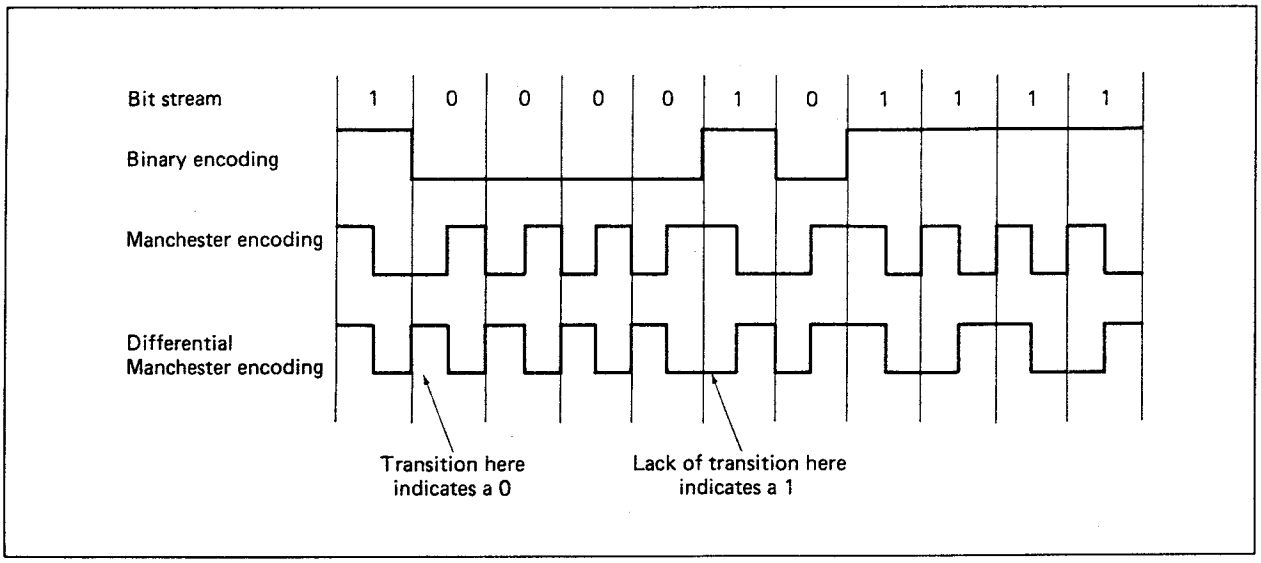


Fig. 5: voorbeelden van het coderen van bits

gebracht worden. Als een knooppunt (computer o.i.d.) een bericht naar een ander knooppunt wil sturen, dan hoeft hij alleen maar te kijken of de andere partij niet tegelijk een bericht naar hem wil sturen. Dit kan door middel van een eenvoudig master-slave protocol (zie aflevering 5 van deze serie) opgelost worden. In een stervormig netwerk kan men in principe hetzelfde doen. Het centrale knooppunt zorgt ervoor dat alle berichten die bij hem binnenkomen naar de juiste node doorgestuurd worden en ook in dit geval volstaat een eenvoudig master-slave protocol tussen het centrale knooppunt en de rest van de knooppunten. Er zijn echter ook stervormige netwerken waarin de het centrale knooppunt niets anders doet dan alle kabels met elkaar verbinden. In principe heb je dan de zogenaamde bus-structuur weer terug.

Bij een busnetwerk gaat het anders. Hier is het zo dat elk knooppunt alle berichten op het medium ziet. Als het bericht voor hem is, dan moet hij uiteraard op dit bericht gaan reageren, is het bericht niet voor hem, dan kan hij het gewoon negeren. Het probleem ontstaat echter als een knooppunt wil gaan zenden. Uiteraard mag een station pas gaan zenden als er geen ander bericht over de bus gestuurd wordt. Zou hij dat toch doen, dan komen de berichten door elkaar en ontstaat er een zogenaamde botsing of collission. Daar er dan van het signaal geen chocola meer te maken valt, moet deze situatie voorkomen of opgelost worden. Hiervoor zijn er een aantal technieken.

6 pagina's μP Kenner zijn lang niet genoeg om alle in's en out's van een netwerk te beschrijven.

De eerste techniek is het zogenaamde Multi Drop netwerk. In een dergelijk netwerk is er altijd een zogenaamde master. Dit knooppunt communiceert achtereenvolgens met elk knooppunt waarbij de master de berichten van dit knooppunt ophaalt en de nieuwe berichten naar het knooppunt toestuurt.

De tweede techniek is de CSMA/CD-techniek die onder andere toegepast wordt in een Ethernet netwerk. CSMA/CD staat voor Carrier Sense Multiple Access with Collision Detect. In deze techniek luistert een knooppunt die wil gaan zenden eerst of er al informatie over het netwerk overgestuurd wordt. Zolang er informatie uitgewisseld wordt, blijft het knooppunt wachten. Is het netwerk vrij, dan begint het station zelf te zenden. Mocht het nu voorkomen dat een ander station ook besloten heeft te gaan zenden, dan ontstaat er een collission. Doordat een zender ook tijdens het zenden naar het signaal op het netwerk blijft kijken, wordt deze botsing door beide partijen gedetecteerd. Beide stations houden acuut op met zenden en wachten vervolgens een random tijd voordat ze het opnieuw proberen. Vooral de random tijd is van belang want zouden de stations beide evenlang wachten, dan ontstaat er vanzelfsprekend weer een botsing.

De derde techniek is het zogenaamde Token Bus (voor bus-netwerken) en token Ring (voor ringvormige netwerken) mechanisme. Dit berust op het uitwisselen van een berichtje van het ene station naar het andere. Een station dat het token ontvangt, mag gedurende een bepaalde tijd berichten op het netwerk zetten. Hij voegt als het ware packets toe aan het toeken. Vervolgens stuurt hij het token door naar het volgende station. Heeft een station geen berichten te sturen, dan stuurt hij uiteraard meteen het token door naar het volgende station. Bij de Token Bus zal het station waarvoor het bericht is, dit meteen oppikken, bij een token ring kan een bericht alleen met het token mee naar het volgende station gestuurd worden. Het bericht gaat mee met het token totdat het bij de geadresseerde aangekomen is die het bericht vervangt door een mededeling dat hij het ontvangen heeft. Als deze bevestiging weer via de ring bij de afzender aangekomen is, dan is het kringetje zowel letterlijk als figuurlijk rond. In een token ring of token bus netwerk moet altijd bewaakt worden dat het token niet zoekraakt. Als een station dat in het bezit is van het token down gaat, dan is het token verloren en moet de communicatie opnieuw op gang gebracht worden. Mocht dit gebeuren, dan kan dat geconstateerd worden doordat een station niet binnen een bepaalde tijd het token krijgt of doordat er gedurende een bepaalde tijd geen activiteit op het netwerk is. In dat geval wordt er door het station dat dit detecteert (meestal de volgende in de rij) een nieuw token aangemaakt waarna het netwerk weer actief is. Uiteraard moeten vervolgens de berichten die tegelijk met het token ten onder gegaan zijn, weer aangemaakt worden.

Afsluiting

Uiteraard zijn zo'n 6 pagina's μP Kenner lang niet genoeg om alle in's en out's van een netwerk te beschrijven. In de literatuurlijst staat een boek van ruim 650 bladzijden waarvan dit artikel er misschien 50 samenvat. Er is dus nog veel meer over netwerken te vertellen. In de volgende aflevering wil ik daarom graag nog op het fenomeen netwerken terugkomen.

Vanwege het feit dat het Novell-netwerk vooral toegepast wordt in PC-omgevingen zou ik ook graag dit netwerk ook als voorbeeld gebruiken. Ik heb echter geen enkele documentatie van dit netwerk; ik weet niet meer dan dat het bestaat. Misschien dat één van de lezers met kan helpen aan een beknopte beschrijving van het Novell-netwerk? Met name een inleiding over wat je er mee kunt doen en hoe het werkt zou ik zeer op prijs stellen.

Literatuur

1: Andrew S. Tanenbaum: Computer Networks (Prentice-Hall)
2: H.M. Deitel: An introduction to Operating Systems (Addison Wesley)
3: JÜrgen Siebert: ARC vernetzt; mc 10/90, 11/90, 12/90
4: Nijkerk Elektronica: Documentatie over glasvezelkabels
5: PBNA: Poly automatiserings zakboekje (Koninklijke PBNA)
6: P.C. den Heijer/R. Tolsma: Datacommunicatie (Kluwer)

Gert van Opbroek

Een nieuwe DOS-65 coördinator

Mag ik me even voorstellen: Ik ben Frank Bens, 37 jaar, gehuwd en vader van twee kinderen. Voorts ben ik werkzaam bij de Koninklijke Marine én ben ik uw nieuwe DOS-65 coördinator.

Ik hoop dat ik bij het vervullen van deze taak op een ieders medewerking mag rekenen en dit geldt uiteraard ook vise-versa.

Even voor alle duidelijkheid volgt hier mijn volledige adres:

Frank Bens
DOS-65 coördinator
Tjalkstraat 25
1784 RX Den Helder
Tel: 02230-33379

Nu ik toch aan het schrijven ben, wil ik Ad Oerlemans als nieuw DOS-65 bezitter en -gebruiker verwelkomen en gelijk een opsomming geven van de in de bibliotheek aanwezige documentatie en software voor programmeertalen:

Manuals:	Prog. talen
- Basic V2.00/2.20	
- Basic V3.00	
- Small C	- Small C
	- Pascal
	- Forth

De volgende keer volgt meer informatie omtrent de rest van de beschikbare software en manuals. Tevens wil ik een beroep doen aan **ALLE** DOS-65 programmeurs: Houd de zelf geschreven programma's niet voor jezelf, stuur het in en laat daarmee een ander er ook van genieten. Als je modificaties/updates op reeds bestaande programmatuur hebt gemaakt, laat de DOS-65 coördinator het dan ook weten, zodat als mensen een bepaald programma willen hebben deze dan niet een programma krijgen met reviesiestand 0 terwijl b.v. rev. 100 al draaiend is. Dit zou zonde zijn met als gevolg dat verschillende mensen hetzelfde wiel uitvinden.
Ik hoop dat de toekomst veel plezierige DOS-65 uurtjes zullen opleveren en misschien tot ziens op één van de bijeenkomsten.

Frank Bens

Van de bestuurstafel

Bij het schrijven van de vorige "Van de bestuurstafel" viel de sneeuw, dit keer is het heel wat anders: buiten is het stralend weer. Er is meer anders dan de vorige keer. In het vorige nummer stond voor het eerst een stukje te lezen van een andere vereniging dan de onze: de NLMUG. Het lijkt er echter op, dat het tevens de laatste keer was. Het is een beetje stil geworden rond de NLMUG. En de berichten die doorsijpelen, zijn niet altijd even positief. De toekomst zal het leren. Feit is in ieder geval wel dat de voorzitter van de NLMUG, Fred van Kempen, een baan in het buitenland heeft geaccepteerd. De NLMUG zoekt dus in ieder geval een nieuwe voorzitter.

De laatste bijeenkomst in Geldrop ging zoals gebruikelijk eveneens vergezeld van stralend weer. In Geldrop hadden we trouwens een primeur, hetgeen diegene die gekomen waren misschien onmiddellijk zo ervaren hebben. We hadden een lezing gepland over netwerken. Typisch zo'n onderwerp waar iedereen zich wel wat bij kan voorstellen, maar waar de meesten niet echt het fijne van weten. In de bestuursvergadering bespreken we ook altijd wat we op een bijeenkomst gaan doen. In de vergadering voor Geldrop bleek dat we zeer waarschijnlijk een ervaren spreker voor het onderwerp hadden. Verder waren er tenminste drie uitwijkmogelijkheden. Komt wel goed, met die netwerken-spreekbeurt. Het werd woensdag. De spreker die we hadden gevraagd zegde af. Uitwijkmogelijkheid 1 aangesproken. Had zaterdag inmiddels andere verplichtingen. Het werd donderdag. UItwijkmogelijkheid 2 zag het uiteindelijk niet zo zitten. Het werd vrijdag. Uitwijkmogelijkheid 3 bleek ziek. Geen lezing, of toch? Ondergetekende had gezegd als er geen lezing zou worden gehouden hij in ieder geval een verhaaltje kon houden over de elektronica op een benzine station. En zo geschiedde. Dat was de primeur in Geldrop. Een lezing die geheel onvoorbereid gegevens moest worden, en die over een totaal ander onderwerp ging dan aangekondigd. Wist u overigens dat wanneer u de slang op een zelfbedieningstation uit de pomp oppakt dat er dan in totaal minstens 4 microprocessors aan de slag gaan? Doch wees gerust: in Almere doen we dat niet weer.

In Geldrop hebben we nog meer besproken: DOS65 versie 3. Er is duidelijk (weer) belangstelling voor, dus even een opfrissertje. Wat is DOS65 versie 3 nu precies? Als eerste: multi-tasking. Erg in de mode, zie bijvoorbeeld het artikel over computerkreten. Als tweede: de mogelijkheid om harde schijven te koppelen. Op dit moment zijn dat alleen nog maar SASI/SCSI drives, maar dat zou later nog best wel eens kunnen veranderen. Die verandering is overigens minder noodzakelijk dan vroeger, want SCSI-drives zijn in de PC-wereld al vrij normale en betaalbare dingen geworden. Hardwarematig vereist DOS65 meer geheugen, want iedere taak neemt wat geheugen in beslag. En dan kom je met 52kbyte RAM niet zover als daar de kern van het operating systeem ook nog in moet wonen. DAt heeft geleid tot een kaart die meeste DOS65 gebruikers wel kennen: de virtual disk kaart. Hij wordt zo genoemd, omdat in de huidige versie van DOS65 dat de enige zinvolle toepassing is.

Dit alles is niet nieuw. Ad Brouwer, de aanstichter van DOS65, Erwin Visschedijk, Adri Hankel (de twee grote promotors achter DOS65) en ondergetekende hebben al eens om de tafel gezeten hoe DOS65 versie 3 eruit moest gaan zien, en hoe we dat in hardware zouden gaan gieten. Want 1 van de dingen die toen al erg duidelijk was dat het bestaande ontwerp voor de virtual disk kaart niet zo goed te reproduceren bleek. Verder zijn er een paar moeilijke componenten in de kaart verwerkt (74S189's) en draait de kaart alleen op een 65C02 en op 1 MHz. Daarom is toen nagedacht over een ontwerp met SRAMs. Dat ontwerp is toen ook gemaakt. Dankzij het gebruik van 1 PAL (een 22V10) en acht 1 Mbit SRAMs blijft het aantal IC's op de kaart beperkt tot 17 en dat is lekker weinig voor een Eurokaartje met 1 Mbyte RAM erop. Een nadeel van SRAMs is zonder meer de prijs: DRAMs zijn stukken goedkoper. Voordelen zijn er ook: je hoeft ze niet te refreshen, zodat hoge snelheden (2 MHz bijvoorbeeld) en gestolen CPU-tijd geen problemen meer vormen. Er zijn nog meer voordelen. Batterij-backup kan eenvoudig worden toegevoegd. En als je 1 van de RAMs door een EPROM vervangt kun je zonder schijf booten als je daar het DOS in zet. En aan de prijs van de RAMs is ook al wat gedaan. Studie van een databoek heeft opgeleverd dat de wereld inmiddels is verblijd met zogenaamde pseudo-statische RAMs. Dat zijn DRAMs met een eigen refreshgenerator op de chip. Ze worden gemaakt met dezelfde pinning als echte SRAMs. De Japanners zijn inmiddels zo clever geworden in het maken van die dingen, dat je ze gewoon als SRAMs kunt behandelen. Activiteit op de adreslijnen is genoeg voor de refresh. En zo lang de processor loopt, zit dat wel goed.

Niet alleen buiten schijnt de zon. Voor DOS65 ook. U hoort er nog wel van denk ik. Van mij in ieder geval wel. Tot het volgende nummer.

De zeurvitter,

Nico de Vries

Informatie

De μP Kenner (De microprocessor Kenner) is een uitgave van de KIM gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht op 22 juni 1978 en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305. Het gironummer van de vereniging is 3757649.

De doelstellingen van de vereniging zijn sinds 1 januari 1989 als volgt geformuleerd:

- Het vergaren en verspreiden van kennis over componenten van microcomputers, de microcomputers zelf en de bijbehorende systeemsoftware.
- Het stimuleren en ondersteunen van het gebruik van micro-computers in de meer technische toepassingen.

Om deze doelstellingen zo goed mogelijk in te vullen, wordt onder andere 5 maal per jaar de μP Kenner uitgegeven. Verder worden er door het bestuur per jaar 5 landelijke bijeenkomsten georganiseerd, beheert het bestuur een Bulletin Board en wordt er een softwarebibliotheek en een technisch forum voor dediverse systemen in stand gehouden.

Landelijke bijeenkomsten:

Deze worden gehouden op bij voorkeur de derde zaterdag van de maanden januari, maart, mei, september en november. De exacte plaats en datum worden steeds in de μP Kenner bekend gemaakt in de rubriek Uitnodiging.

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld.
Het telefoonnummer is: 053-328506 of 053-303902 .

Software Bibliotheek en Technisch Forum:

Voor het beheer van de Software Bibliotheek en technischeondersteuning streeft het bestuur ernaar zgn. systeemcoordinators te benoemen. Van tijd tot tijd zal in de μP Kenner een overzicht gepubliceerd worden. Dit overzicht staat ook op het Bulletin Board.

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 99650
1000 NA Amsterdam

Het Bestuur:

Het bestuur van de vereniging wordt gevormd door een dagelijks bestuur bestaande uit een voorzitter, een secretaris en een penningmeester en een viertal gewone leden.

Nico de Vries (voorzitter)
Mari Andriessenrade 49
2907 MA Capelle a/d IJssel
Telefoon 010-4517154

Jacques H.G.M. Banser (penningmeester)
Haaksbergerstraat 199
7513 EM Enschede
Telefoon 053-324137

Jan D.J. Derksen(secretaris)
C.P. Soeteliefstraat 41
1785 CC Den Helder
Telefoon 02230-35002

Gert van Opbroek (redactie μP Kenner)
Bateweg 60
2481 AN Woubrugge
Telefoon 01729-8636

Ton Smits
De Meren 39
4731 WB Oudenbosch

Geert Stappers
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

Mick Agterberg
Davidvosstraat 29
1063 HV Amsterdam
Telefoon 020-131538

Ereleden:

Naast het bestuur zijn er een aantal ereleden, die zich in het verleden bijzonder verdienstelijk voor de club hebben gemaakt:

Erevoorzitter:
Siep de Vries

Ereleden:
Mevr. H. de Vries-van der Winden
Anton Müller
Rinus Vleesch Dubois