PowerKey 4/94

Inhalt

Preface/Vorstand

Editorial 4

NiCE Internals

Veranstaltungskalender 5

In eigener Sache 11

Aktuell

NEXTSTEP Release 3.3 Upgrade 6

Enterprise Objects Framework ausgeliefert . . 10

NEXTSTEP 3.3 Developer 20

Know-how

Abhören erlaubt! 12

Software

Oberon V4 – 2. Teil 16

Hardware

Pentium FPU-Bug 21

TeX

TeX-Kurs, Teil 7 (Ende) 22

Editorial

Liebe Leserin, lieber Leser

Ein ereignisreiches Jahr geht zu Ende. Gerade aus der Sicht von NeXT ist 1994 eine Menge passiert: Im März kündigten NeXT und DEC ihre PDOs (Portable Distributed Objects) für OSF/1 an, im Juni wurde die vorläufige Spezifikation von OpenStep veröffentlicht und im gleichen Monat noch die Portierung von OpenStep auf DEC Alpha-Plattformen unter OSF/1 angekündigt. Im August wurde NEXTSTEP 3.2 für HP-PA sowie PDO 2.0 für HP, SunOS und Solaris ausgeliefert, im September erschien die endgültige Spezifikation von OpenStep. Ende Oktober begann NeXT mit der Auslieferung von EOF (Enterprise Objects Framework) sowie NEXTSTEP 3.3, von dem bis Ende Jahr auch die Version für Sun's SPARC verfügbar sein sollte. Glaubt man dem Fahrplan von NeXT, so sollte im ersten Quartal 1995 die OpenStep-Version für Solaris und Mitte Jahr die Version 4.0 von NEXTSTEP folgen.
Trotz dieser umfangreichen Liste von Ankündigungen und tatsächlich eingehaltenen Auslieferungsterminen darf man nicht vergessen, dass NeXT immer noch mit den gleichen Problemen wie seit jeher zu kämpfen hat. Grösste Sorge ist die relativ kleine Anzahl von Usern, die heute NEXTSTEP einsetzen. Was NeXT seit fünf Jahren an (Betriebs-) Systemen verkauft hat ist gerademal soviel wie Mac jeden Monat und Microsoft jede Woche umsetzen!

Auch innerhalb der NiCE hat das letzte Jahr einige Änderungen gebracht – eine davon haltet Ihr jetzt gerade in den Händen. Das neue Redaktor/Verleger-Gespann des ***PowerKey*** hat wie versprochen vier weitere Ausgaben produziert und (mehr oder weniger) pünktlich in Eurem Briefkasten abgeliefert. Unser Dank geht an all jene, die aktiv dazu beigetragen haben, indem Sie uns eigene Beiträge zur Verfügung gestellt haben, oder einfach nur ein Feedback auf unser Tun gaben.

Ich bin nach wie vor der Meinung, dass vier Ausgaben pro Jahr ausreichend sind, und das nicht nur, weil für mehr ***PowerKey***'s ganz einfach die Artikel fehlen. Mit diesem Entscheid mussten wir aber auf den Anspruch verzichten, dass das ***PowerKey*** eine aktuelle Mitgliederzeitung sein soll. Aktuell in dem Sinn, dass wir kurzfristig – d.h. innerhalb eines Monats – über bevorstehende oder vergangene Ereignisse berichten können. Um diese kurzfristige Information weiterhin anbieten zu können, wurde eigens eine NiCE-Postkarte geschaffen, die, wie der Name schon sagt, in Postkartengrösse auf dringliche Dinge aufmerksam machen soll.

Auch im nächsten Jahr wird es wieder vier Ausgabe des ***PowerKey*** geben, vorausgesetzt es finden sich genügend Autorinnen und Autoren, die Ihr Werk in einem ***PowerKey*** veröffentlicht sehen wollen – am Einsatz von Redaktor und Verleger soll es sicher nicht scheitern!

Nun aber zum Inhalt: In der aktuellen Ausgabe von ***PowerKey*** erscheinen Berichte über den Upgrade der User-Version 3.3 von NEXTSTEP, über den Stand der Developer Version 3.3, zum Thema abhörsichere Verschlüsselungsverfahren sowie zur Version 4 von NeXT-Oberon. Teil 7 des TeX-Kurses bildet den Abschluss dieser Serie und gleichzeitig dieser Ausgabe. Daneben wie immer aktuelle News rund um NeXT und die NiCE.

Zum Schluss möchte ich im Namen des NiCE-Vorstands allen Mitgliedern und Ihren Angehörigen schöne Feiertage und einen guten Rutsch ins neue Jahr wünschen. Euch allen viel Vergnügen mit dieser Ausgabe des ***PowerKey***.

Dominik Moser, Redaktor

Veranstaltungen

talk & copy

Wann: jeden zweiten und vierten Dienstag im Monat
Beginn 19.00 Uhr

Wo: Informatik-Gebäude (IFW) der ETH
Raum IFW A44
Adresse: Haldeneggsteig 4/Weinbergstr.
Tram 6/7/10/15, Haltestelle Haldenegg

Themen: Hier haben Mitglieder die Möglichkeit, Fragen zu stellen, Erfahrungen auszutauschen sowie die neuste Software aus unserem grossen Archiv zu kopieren. Bring doch einfach Deine Disketten oder besser Deine Festplatte mit! Die NiCE besitzt (fast) alle dazu erforderlichen SCSI-Kabel und -Terminatoren.

NiCE-Meetings

Wann: Meetings werden speziell angekündigt
Beginn 19.00 Uhr (bis ca. 21.30)

Wo: ETH Zürich, Hauptgebäude
Raum siehe Ankündigung
Adresse: Rämistr. 101
Tram 6/9/10, Haltestelle ETH/Uni'spital

Themen: Werden nach Möglichkeit im ***PowerKey*** bekanntgegeben.

Achtung!

An der letzten Vorstandssitzung wurde beschlossen, die NiCE-Meetings vorläufig aus dem Veranstaltungskalender zu streichen. Neu findet nun zweimal pro Monat ein *talk&copy* in IFW-Gebäude der ETH statt. Meetings oder andere ausserplanmässige Anlässe werden in Zukunft speziell angekündigt werden, entweder an dieser Stelle im ***PowerKey*** oder mittels einer Postkarte.

NiCE-Agenda 1994/95

27. Dezember: Fällt wegen Weihnachtsferien aus!

10. Januar: talk & copy

24. Januar: **NiCE-GV**

14. Februar: talk & copy

28. Februar: talk & copy

14. März: talk & copy

15. März: Redaktionsschluss ***PowerKey***

28. März: talk & copy

FTP Server

Auf dem NiCE ftp-Server befinden sich die neusten Applikationen, Tools und Daten direkt ab dem grossen ftp-Archiv in München. Zugriff für Mitglieder entweder anonym über Internet oder persönlich während jeder *talk&copy* Veranstaltung.

Neue Fax-Nummer

Im Rahmen der Zuteilung von neuen Telefonnummern für die ETH Zürich bekam auch unser NiCE-Fax eine neue Rufnummer. Der Abdruck auf dem Umschlag des ***PowerKey*** ist also nicht mehr aktuell!

Alte Nummer: 01/261 53 89
Neue Nummer: **01/632 12 20**

NEXTSTEP Release 3.3 Upgrade

Ende Oktober wurde über das Usenet die Ankündigung von NeXT verbreitet, dass das Upgrade für die NEXTSTEP User-Version 3.3 ab sofort bestellt werden kann. Gleichzeitig wurde als NeXTAnswer Nr. 1706 eine offizielle Beschreibung des neuen Release veröffentlicht, die hier im Originalwortlaut wiedergegeben werden soll.

NEXTSTEP Release 3.3 combines the robustness of mainframe operating systems with the flexibility of PCs, making it the best environment for enterprise-wide deployment of object-oriented custom applications.

NEXTSTEP™ is the first and only UNIX-based operating environment to integrate leading edge object-oriented technology, industry standard networking and connectivity, and an award winning graphical user interface. With the addition of NEXTSTEP Developer, users can rapidly develop client/server custom applications that can be seamlessly integrated with shrink-wrapped software to create comprehensive business solutions.

NEXTSTEP Release 3.3 incorporates enhancements based on feedback from customers with large-scale NEXTSTEP environments. Release 3.3 addresses many critical deployment issues including scalability, robustness, interoperability, and ease of use. By incorporating core technologies from Mach 3.0 and BSD 4.4 networking, Release 3.3 operates more reliably and scales better to networks with tens of thousands of users.

NeXTmail™ has been significantly improved in Release 3.3 to give users greater flexibility in how electronic mail is accessed, managed, and stored. In addition, MIME support enables NEXTSTEP users to send and receive multimedia messages with users of other popular e-mail applications through the same intuitive interface.

To make NEXTSTEP run on a wider selection of PC configurations, Release 3.3 includes support for PCI and ISA Plug & Play, PCMCIA, Advanced Power Management, 8 bit color, and many new peripherals.

HIGHLIGHTS

The NEXTSTEP User Experience

NEXTSTEP is the only object-oriented operating system available today. NEXTSTEP delivers a consistent graphical user interface across wide-area networks such as NFS, NetWare®, and AppleTalk. NEXTSTEP unifies the desktop by providing interoperability between custom solutions developed for NEXTSTEP and applications written for the MS-Windows, X/Motif, and mainframe environments.

ROBUSTNESS

The smooth operation of a company depends on the uninterrupted service of its mission-critical applications. If a financial trading company cannot make trades for fifteen minutes due to a system down-time, it could lose hundreds of thousands in revenue. NEXTSTEP Release 3.3 is designed to service these mission-critical applications 24 hours a day, seven days a week. Hundreds of carefully chosen focused enhancements and improvements have been added throughout the system to increase the overall robustness of the operating system while ensuring backward compatibility. This strategy enables NeXT to introduce new technologies to the NEXTSTEP community while providing a seamless and painless upgrade path.

SCALABILITY

Today's enterprise-wide deployment means tens of thousands of users on the same network across multiple routers and bridges. NEXTSTEP Release 3.3 includes critical technologies to enable large-scale networks and reduce support costs by allowing enterprise-wide deployment with a much lower ratio of system administrators to users. For example, with the new UserManager, system administrators can now install and maintain large numbers of network user accounts quickly. With the enhancements in NetInfo™, NEXTSTEP can now support tens of thousands of network nodes. With

NEXTSTEP Release 3.3, companies can grow their installation exponentially with ease without the need to increase system administration resources at the same pace. Release 3.3's system administration tools allows new system administrators to be productive with minimum training, and empowers experienced system administrators to maintain a large enterprise-wide network with greater ease.

PERFORMANCE
In a large-scale deployment with tens of thousands of users, productivity gains from overall increased system responsiveness and better tools means thousands of dollars in savings. By taking advantage of BSD 4.4 networking, NEXTSTEP Release 3.3 delivers better wide-area-network performance while technologies from Mach 3.0 enables NEXTSTEP to task switch more efficiently. By streamlining and accelerating frequently performed tasks such as finding, sorting, and addressing in NeXTmail, adding or deleting a large number of users in UserManager, bulk upgrade in Upgrader, and on-the-fly NetInfo backup & restore in NetInfoManager, NEXTSTEP Release 3.3 provide tools that enable users to work faster by working smarter.

OPEN STANDARDS
With Release 3.3, NeXT continues its commitment and willingness to adopt industry standard technologies to ensure interoperability, flexibility, and a high degree of collaboration between technologies from different vendors. MIME support enables NEXTSTEP users to easily exchange files and images with widely used PC and UNIX mail applications. The support of PCI and ISA Plug & Play greatly reduces the complexity of PC hardware configuration. PCMCIA, 8 bit color, and Advanced Power Management enables users to operate NEXTSTEP on standard PC notebooks.

EASE OF USE
Just as performance gains translate into productivity gains, tools that enable new users to learn an operating environment quickly are equally important. In addition, support costs can be significantly lower if users are equipped to help themselves in times of trouble. Release 3.3 includes a new Quick Start Guide that enables new users to learn all the basics in 30 minutes; this tool greatly accelerates the learning curve of NEXTSTEP. Once these users are familiar with the basics of NEXTSTEP, they can rely on the new Documentation Roadmap to provide pointers to all the documentation including comprehensive on-line help for most of the major applications that are bundled with NEXTSTEP. To empower those who are already familiar with the system, a new Power Tips manual provides many tips and shortcuts. These tools keep support and training costs low as the number of users and network nodes increases.

NEW FEATURES AND BENEFITS

Mail
MIME Support
As a widely supported standard, MIME support enables NEXTSTEP users to exchange mutimedia e-mail with other non-NeXT MIME users. Through the same intuitive NeXTmail interface, users can transparently read and send messages containing file attachments, rich text, images and sound objects.

Mail Size Management
Information such as the number of read and deleted messages, size of each message, and the number of new messages is now easily accessible. Users can also quickly locate large e-mail messages through the feature "sorting by size". These features greatly simplify the task of overall mailbox size management.

Selective & Summary Printing
Users can now print out the summary of an entire mailbox or a group of selected messages. This feature enables users to manage their e-mail messages off-line when necessary.

Enhanced Searching
In addition to searching by subject and sender, users can now locate mail by searching through the message body. A new "Focus" feature makes it more efficient to navigate and view only the selected messages. After several iterations of "Find" and "Focus," users can quickly filter all non-relevant messages.

Save & Restore Drafts
Similar to a word processing document, partially composed unsent messages can be saved and brought back with the subject and address fields intact. Users can now begin composing multiple messages and use this feature to preserve work-in-progress from accidental loss – or to conveniently come back to it at a later time.

Outgoing Message Archive
In addition to automatic archival of all outgoing messages, users who are concerned with disk space can selectively archive specific messages.

In-line Address Completion
By typing the first few letters of an address in the "To" field and pressing the Esc key, NeXTmail will automatically scroll through the possible options from the list of public and private users and groups. This feature reduces the number of keystrokes and the frequency of access to the Address panel.

Hierarchical Mailboxes
Mailboxes in Release 3.3 can now be organized into a tree-like hierarchy. Messages stored in a meaningful structure of mailboxes gain relevance through their implicit association with folder names. Retrieval of messages can also be faster by simply navigating through the hierarchy.

User Defined Fields
Custom fields can now be added to a message header to label outgoing messages. For instance, a user can label a message with a field called "priority" and set it to "high" or "low". This allows recipients to quickly identify important messages.

New Productivity tools
The mail icon now displays the number of unread messages in the Active mailbox. In addition, users can manage the incoming mail workload by individually marking messages in any mailbox as read or unread.

Networking
NetInfo
The only proven enterprise-wide network administration database. NetInfo now supports 10,000+ nodes across wide-area networks which enables corporations to grow their NEXTSTEP networks globally without sacrificing data integrity or network performance. Other new features include integrated DNS support and improved diagnostic messages.

UserManager
The tool for managing user accounts and user groups has been updated to support large corporate networks. With the new bulk add-and-delete feature, system administrators can manage thousands of user accounts using custom templates and domain-based settings. Support for international user accounts is also provided through templates. In addition, UserManager can be customized to perform specific task through APIs. This feature allows system administrators to add or override default functionalities to tailor UserManager to the needs of their sites.

NetInfoManager
The graphical tool for accessing and manipulating NetInfo databases has been dramatically improved. Administrative tasks are simpler, allowing the system-administrator to dynamically restart NetInfo services, save and restore NetInfo databases, or join NetInfo hierarchies. The user interface has been improved to ensure ease-of-use throughout the application. Searching in NetInfo data is more efficient with Multi-domain and regular-expression search. Remote Server Management allows the system-administrator to comprehensively manage resources from any part of the network.

SimpleNetworkStarter
The simple but powerful tool for setting up small networks of NEXTSTEP computers has been extended to include NetInfo Configuration across SNMP configurations and subnets. All transactions are fully logged. The undo feature allows dynamic configuration changes.

Networking Commands
NetInfo data can be accessed and manipulated via a variety of new and extended set of UNIX utilities. Niutil's new options provide a full set of data-management features, including resync, rename, and statistics options. Nigrep, nifind, and nireport complement the existing

NetInfo utilities to provide full search and report capabilities in a multi-domain environment.

BSD 4.4 Networking
New gateway congestion and header prediction algorithms improve overall network performance, especially in large networks that include bridges and routers. In addition, IP Multi-Cast support has been added to provide the framework for multimedia networking.

OS
MACH 3.0
Core components of MACH 3.0 have been integrated into Release 3.3 to improve the robustness and performance of the operating system, especially under heavy load.

Intel Enablement
PCMCIA
NEXTSTEP users can now take advantage of PCMCIA modems, SCSI adaptors, LAN cards, and wireless networking cards both for portable and desktop systems.

ISA Plug & Play
On a system with the ISA Plug & Play BIOS, NEXTSTEP automatically configures ISA Plug & Play cards bypassing the need to deal with jumpers, switches, and configuration utilities.

PCI Plug & Play
Similar to ISA Plug & Play, NEXTSTEP automatically configures PCI cards.

Advance Power Management
By taking advantage of power saving features on systems that support the APM BIOS, NEXTSTEP now operates efficiently on portable systems for an extended amount of time.

720K Floppy
In addition to 1.44 megabyte floppy support, NEXTSTEP Release 3.3 also supports 720k.

More Drivers
NEXTSTEP Release 3.3 comes bundled with over 50 drivers to support a wide variety of peripheral cards and systems.

Installations & Upgrades
Network Install
User can now install NEXTSTEP through a network NFS server without requiring a CD-ROM drive. System Administrators can also preconfigure a template server so that all systems built from the server inherit the same configurations. This greatly reduces the overhead of installing NEXTSTEP at large sites.

Network Bulk Upgrade
The Upgrader application now enables system administrators to pre-configure “template servers” and propagate upgrades from these servers to designated NEXTSTEP systems. In effect, this allows system administrators to “push” upgrades automatically onto systems on the network without the need for any user intervention.

Learning Tools
NEXTSTEP Quick Start: By stepping through this small manual, users can quickly learn all the basics of NEXTSTEP and be productive within 30 minutes.

Power Tips & Documentation Roadmap
The power user’s guide contains a comprehensive list of short cuts and tips for the Workspace and bundled applications. A new on-line & hardcopy roadmap helps novice users to easily and quickly navigate through existing documentation.

Internationalization
Global Software
NEXTSTEP can be customized at setup or installation time to provide global settings for keyboards, paper sizes, time zones, etc. This allows users to tailor their environment to specific local needs. Key globalization features include updates to Preferences.app and Configure.app.

International Installation
Users can now select which language needs to be installed at the "package" level. Instead of installing a specific language for all of NEXTSTEP, users have a choice of installing specific languages for each Installer package similar to Installer's support for different platforms. Users with limited disk space now have greater flexibility in the installation of packages. This feature applies to Upgrader, CD Installation (e.g., BuildDisk), as well as the Installer.

NEXTIME is NeXT's entry into time-based digital media for corporate customers interested in video applications. In its first release, NEXTIME provides NEXTSTEP users the ability to play movie files. NEXTIME supports the industry-standard QuickTime file format and Cinepak compression scheme. Future releases will add support for recording movies using video capture cards, as well as additional file formats and compression schemes.

NEXTSTEP upgrades are scheduled to ship mid-December 1994.

Bearbeitung: (dm)

Anm. der Redaktion:
Preislich wurde das Upgrade wie folgt festgelegt:

NEXTSTEP User 3.3 Upgrade	(Motorola)	$199
NEXTSTEP User 3.3 Upgrade	(Intel)	$199

In der Schweiz kann das Upgrade über die Uptime Object Factory bezogen werden. Zur Zeit der Auslieferung des ***PowerKey*** ist die Weihnachtsaktion der Uptime bereits abgelaufen; das Upgrade kostet jetzt Fr. 420.- und wird ohne NEXTTIME ausgeliefert.

Kontaktadresse:
• *Uptime Object Factory Inc*, Technopark Zürich,
Telefon 01 445 16 99, Fax 01 445 16 98

Enterprise Objects Framework ausgeliefert

Wie NeXT Computer, Inc. am 26. Oktober in Redwood City, Kalifornien bekanntgab, ist das Enterprise Objects Framework (EOF) ab sofort verfügbar.

EOF wurde an der diesjährigen NEXTSTEP Expo als zukunftsweisende Technologie vorgestellt, die die Vorteile objektorientierter Applikationsentwicklung mit der Zuverlässigkeit und Vielseitigkeit von relationalen Datenbanken verbinden soll. Entwickler können zukünftig sogenannte *enterprise buiness objetcs* erstellten und wiederverwenden, oder wie NeXT es ausdrückte: Daten in Information transformieren. Nach Aussagen von Steven Jobs verbindet EOF Geschäftsdaten elegant und nahtlos mit der Geschäftspolitik, die den Daten erst ihre Bedeutung geben. Wesentlicher Vorteil neben einer Wiederverwendbarkeit der *enterprise objects*, ist eine Zeit- und Kostenreduktion bei der Entwicklung und im Unterhalt von Applikationen.
EOF besteht aus drei Modulen: der *Enterprise Object Modeler* wird von Entwicklern dazu benutzt EO's zu erstellen. Das *Framework* stellt die Laufzeitumgebung von EOF dar. Sie wird für alle Systeme benötigt auf denen EOF später laufen soll. Derzeit ist nur das Modul für NEXTSTEP erhältlich, aber schon Anfang des nächsten Jahres soll die PDO-Version auf den Markt kommen. Damit wird die Entwicklung von Enterprise Objects auf Servern unter HP-UX, SUN OS, Solaris und Digital OSF/1 möglich werden. Der dritte Teil, der *Adaptor Layer*, macht die Applikation durch Verwendung einer transparenten Kommunikationsschnittstelle zwischen den EO's und dem Adaptor Layer datenbankunabhängig. Ausgeliefert wird EOF derzeit mit Adaptern für Oracle und Sybase – weitere Produkte von NeXT oder von Drittanbietern sind in der Zukunft zu erwarten.
Kostenpunkt für Enterprise Objects Framework: 299 Dollar – in der Schweiz ca. 700 Franken.

(dm)

NiCE Internals

Liebe Leserin, lieber Leser

Dies ist nun schon die vierte ***PowerKey***-Ausgabe, die ich verlegen darf. Ja, Ihr habt richtig gelesen, darf. Es ist zwar eine gewisse zeitliche Belastung und sie kommt meistens zum falschen Zeitpunkt. So geschehen bei den Ausgaben 1 und 2 (defekter Bildschirm – Dank an den Bildschirmdoktor –, Umzug, Probleme mit den Adressetiketten etc.). Entschuldigt bitte die daraus entstandenen Verspätungen.

Doch schon bevor ich dieses Amt annahm freute ich mich auf jeden neuen ***PowerKey***. Als vorläufig noch «nur Anwender» ist für mich die Mischung aus News und Praxistips sehr wertvoll. Dass ich nun durch meine Verlegertätigkeit zum Weiterbestehen dieses Magazins beitragen kann, ist also auch ein bisschen Selbstzweck.
Als nächstes möchte ich mal mit der Programmentwicklung beginnen.

Ich hoffe, dass Ihr auch so viel Freude am Lesen des ***PowerKey*** habt und vielleicht auch ein wenig zur Attraktivität dieses Magazins beiträgt.

Herzlich, Euer neuer Verleger
Peter Burgdorfer

Neues Vorstandsmitglied

Anlässlich der letzten Vorstandssitzung Anfang November wurde beschlossen, Felix Rauch, den Einkäufer der NiCE, in den Vorstand aufzunehmen. Felix ist seit Anfang dieses Jahres offizieller Verantwortlicher für Einkäufe innerhalb der NiCE, die Bestätigung seiner Wahl erfolgt am 24. Januar 1995, anlässlich der NiCE-GV. Unter seiner Leitung wurde bereits die Apple CD-ROM Bestellaktion durchgeführt.

Bei der neusten Aktion der NiCE, der Bestellung der sogenannten «Fatted Calf CD», sieht es nun allerdings nicht sehr erfreulich aus. Wie ihr durch unsere NiCE-Info Postkarte erfahren habt, konnten die CD's noch nicht bestellt werden, da die amerikanische Firma (Ensulting Technologies), welche die CD's verkauft, nicht zu erreichen war!
In der Zwischenzeit sind leider keine Neuigkeiten mehr eingetroffen. Natürlich werden wir euch auch weiterhin innerhalb des ***PowerKey*** bzw. mittels einer NiCE-Info Postkarte über den Fortbestand der Aktion informieren. Falls jemand nicht mehr länger warten möchte, so kann er/sie sich bei Felix Rauch melden. Der eingezahlte Betrag wird dann umgehend zurückerstattet.

(Felix Rauch/dm)

Korrigenda

In der letzten Ausgabe des ***PowerKey*** ist uns leider ein kleiner Fehler unterlaufen. Im Artikel «*Verbindung mit dem Internet via SLIP und EUnet*» von Neil Franklin (Ausgabe 3/94 ab Seite 10) wurde der Filenamen der SLIP-Dateien falsch angegeben:

Falsch:
```
~ftp/pub/Unix/communication/SLIP_920904-A.*
```
Richtig:
```
~ftp/pub/Unix/communication/SLIP.920904-A.*
```

Sorry!

NiCE-Markt

Zu verkaufen:

- WordPerfect 1.0.1 for NEXTSTEP Fr. 300.-
- NEXTSTEP/Intel 3.1 Developer Fr. 500.-

Martin Beerli
Büchnerstr. 19
8006 Zürich

Tel.: 01/362 95 72 (tagsüber)

Abhören erlaubt!

Während sich Fachleute in den USA immer noch heftig darüber streiten, ob die dereinst vielleicht staatlich vorgeschriebene Clipper-Chiffrierung genügend sicher sei, gibt es erste Anzeichen für neuartige Chiffriermethoden, bei denen die kritische Phase des Schlüsselaustauschs komplett entfallen soll.

Der Einsatz moderner Chiffrierverfahren führt dazu, dass die Justizbehörden die Möglichkeit der elektronischen Überwachung verlieren. Es resultiert ein Interessenskonflikt zwischen Justiz und privater Freiheit. Die Clinton-Administration hat im April 1993 mit der Clipper-Initiative hitzige Diskusionen darüber ausgelöst, ob die Art der Datenverschlüsselung staatlich vorgeschrieben werden soll.
Neue Verschlüsselungsmethoden scheinen nun diese Diskussion ad absurdum zu führen, denn sie behaupten von sich nichts geringeres, als dass man Nachrichten so verschlüsseln kann, dass sie jeglichem Lauschangriff widerstehen – egal, welche Mittel dem Gegener zur Verfügung stehen.

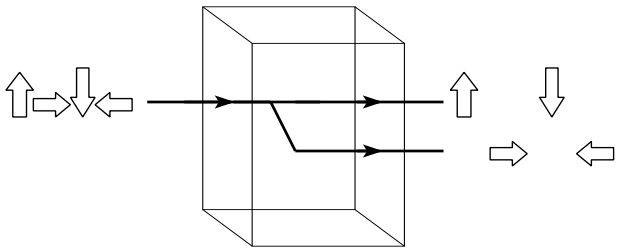
Fachleute wissen schon lange, dass perfekte Geheimhaltung im mathematischen Sinne möglich ist. Der amerikanische Informationstheoretiker Claude Shannon hat 1949 bewiesen, dass eine Nachricht, die mit einem gleichlangen Schlüssel codiert ist, ohne diesen Schlüssel niemals entziffert werden kann [1]. Das theoretisch perfekte System hat aber einen grossen Hacken: Wenn Sender und Empfänger der Botschaft den Schlüssel untereinander austauschen müssen, besteht das Risiko, dass er unterwegs in falsche Hände gerät. Schliesslich können Datenleitungen angezapft, Briefe geöffnet, Meldeläufer abgefangen und Brieftauben im Rahmen einer Reform abgeschafft werden. Dieses Risiko lässt sich nur minimieren, wenn der Schlüssel persönlich oder durch einen zuverlässigen Kurier übergeben wird. Da es zudem für jede Botschaft einen neuen Schlüssel braucht, könnte sich der Absender den Umweg über den Schlüssel sparen und die Nachricht gleich selber übergeben.

Genau an diesem Punkt setzen einige neue Ansätze für ein abhörsicheres Verschlüsselungssystem an: Die Abhörsicherheit soll schon während der kritischen Schlüsselaustauschphase gewährleistet werden. Zwei dieser Arbeiten möchte ich an dieser Stelle kurz vorstellen. Während der erste Ansatz auf purer Mathematik beruht, verwendet der zweite Ansatz Erkenntnisse aus der Quantenmechanik. (Keine Angst es wird schon nicht zu wissenschaftlich ausfallen!) Beiden gemeinsam ist, dass sie in einer ersten Phase einen für beide Parteien eindeutigen, mathematisch zufälligen Schlüssel erzeugen, der gleichen Länge wie die zu sendende Botschaft hat. Ist dieser Schlüssel erst einmal erzeugt, kann in einer zweiten Phase auf bereits bekannte Verschlüsselungstechniken, wie z.B. die Vernam-Codierung [2], zurückgegriffen werden. Nach Shannon ist diese Codierung perfekt.

Ein fundamental anderes Prinzip

Ein auf Quantenmechanik basierendes System arbeitet typischerweise mit Lichtquanten (Photonen) und nutzt das sog. Heissenbergsche Unbestimmtheitsprinzip. Aus dieser grundlegenden Aussage der Quantenmechanik folgt, dass jede Beobachtung eines physikalischen Systems dessen Zustand stört, wodurch ein Teil der Information über seinen vorherigen Zustand verlorengeht. Dazu später mehr.

Ein Quantenübertragungskanal enthält auf der Senderseite typischerweise ein Polarisationsfilter, das Photonen definierter Schwingungsrichtung ausstrahlt, und auf der Empfängerseite eine Apparatur, welche die Polarisation der ankommenden Photonen messen kann. Dazu verwendet man einen doppelbrechenden Kristall (Prisma), der die einfallenden Photonen, ohne sie zu absorbieren, je nach ihrer Polarisation in eine von zwei Richtungen ablenkt. Wenn ein Photon auf das Prisma auftrifft, durchdringt es den Kristall entweder ungebrochen und hat beim Austritt eine zu dessen optischen Achse senkrechte Polarisation, oder es wird abgelenkt und ist dann parallel zur optischen Achse polarisiert. Ist es beim Auftreffen bereits in einer der genannten Richtungen polarisiert, behält es diese bei und wird dementsprechend abgelenkt oder auch nicht.



«Gerade» Photonen durch ein Prisma

Liegt die Polarisation jedoch dazwischen, folgt es mit gewissen Wahrscheinlichkeiten entweder dem geraden oder dem gebrochenen Weg. In jedem Fall büsst es seine ursprüngliche Polarisation ein und nimmt die dem Weg entsprechende an. Am wenigsten vorhersagbar ist das Verhalten eines Photons, dessen Schwingungsrichtung halbwegs zwischen den beiden Hauptrichtungen liegt, also bei 45 oder 135 Grad.

Angenommen Bob[1] erfährt vorab, dass das nächste Photon, das er empfangen wird, entweder senkrecht (90 Grad) oder waagerecht (0 Grad) relativ zu einer gewissen Bezugsrichtung polarisiert ist; solche Photonen wollen wir im folgenden «gerade» nennen. Welcher der beiden Fälle vorliegt, kann Bob dann leicht ermitteln, indem er das Photon auf ein senkrecht orientiertes Prisma auftreffen lässt und in beide möglichen Strahlungswege je einen Detektor für den Nachweis einzelner Photonen stellt. Mit einem solchen Gerät kann Bob allerdings keine «schrägen» Photonen unterscheiden – solche, deren Polarisationswinkel 45 oder 135 Grad beträgt; dies ist nur möglich, wenn er das Nachweisgerät um 45 Grad dreht. Dann aber kann es keine Information über gerade Photonen liefern.

Der quantenmechanischen Unschärferelation zufolge ist diese Einschränkung grundlegender Art; sie gilt also nicht nur für die beschriebene Messanordnung, sondern für alle nur denkbaren Detektoren. Gerade und schräge Polarisierung bilden ein Paar zueinander komplementärer Eigenschaften: Die Messung der einen Eigenschaft zerstört die Information über die andere!

1 Aus historischen Gründen heissen die Kommunikationspartner bei einem Chiffrierungsprozess Alice und Bob.

Auf dieser Erkenntnis beruht das hier vorgestellte Verfahren von Charles H. Bennet und Gilles Brassard [3]. Alice und Bob können einen geheimen, zufallsbestimmten Schlüssel vereinbaren, der ihnen später zur Codierung von Nachrichten dient. Das Verfahren verwendet einen Quantenkanal, über den Alice und Bob polarisierte Photonen übertragen, und einen öffentlichen Kanal, über den sie auf herkömmliche Weise kommunizieren. Eine Lauscherin – sie heisst traditionell Eve – kann die Polarisationsrichtung der Photonen im Quantenkanal zwar zu bestimmen versuchen, würde jedoch unvermeintlich diesen Zustand ändern. Ausserdem kann Eve alle über den öffentlichen Kanal übertragenen Nachrichten mithören, diese aber weder stören noch verändern.

Schlüsselvergabe

Alice und Bob vergleichen die durch den Quantenkanal gesendeten Daten über den öffentlichen Kanal, um sich gegen einen möglichen Lauschangriff von Eve abzusichern. Wenn dieser – durch den Vergleich erwiesen – nicht stattgefunden hat, können sie aus ihren Daten einen Schlüssel ermitteln, der garantiert für beide Seiten identisch, vom Zufall bestimmt und geheim ist – unabhängig von Eves technischen Möglichkeiten und der ihr zur Verfügung stehenden Rechnerleistung.

Das Verfahren funktioniert folgendermassen: Zunächst erzeugt Alice eine Serie von Photonen, deren Polarisation in ungeordneter Folge 0, 45, 90 oder 135 Grad betragen, und sendet sie an Bob. Beim Empfang jedes Photons entscheidet Bob nach dem Zufallsprinzip, ob er dessen gerade oder schräge Polarisation misst. Als nächstes gibt Bob über den öffentlichen Kanal für jedes Photon bekannt, welche Art von Messung er vorgenommen hat (gerade oder schräg), nicht aber das Messergebnis (0 oder 90 bzw. 45 oder 135 Grad). Alice teilt ihm dann ebenfalls über den öffentlichen Kanal für jedes Photon mit, ob er die richtige Art von Messung gewählt hat. Alice und Bob verwerfen dann alle Fälle, in denen Bob die falsche Messung gemacht hat. Wenn niemand den Quantenkanal abgehört hat, kennen nur Alice und Bob die verbleibenden Polarisationen – und niemand sonst.

Alice und Bob überprüfen nun, ob sie abgehört worden sind, indem sie beispielsweise öffentlich einen beliebigen Teil ihrer Polarisationsdaten miteinander vergleichen. Ergibt sich, dass jemand gelauscht hat, verwerfen sie den gesamten Datensatz und beginnen mit einer neuen Photonenfolge. Im anderen Fall können sie davon ausgehen, dass die übrigen, noch nicht öffentlich genannten Polarisationen ihr gemeinsames Geheimnis sind. Sie machen daraus eine Bitfolge, indem sie beispielsweise waagerechte und 135-Grad-Photonen in binäre Nullen, vertikale und 45-Grad-Photonen in binäre Einsen umsetzen.

Wegen der Unschärferelation kann Eve, die Lauscherin, ein Photon nicht sowohl auf gerade als auch auf schräge Polarisation untersuchen. Wenn sie bei einem Photon die falsche Messung vornimmt, hat sie die Information über die ursprünglich von Alice gewählte Polarisation zerstört. Es nützt also nichts, wenn sie Bob ein dem Ergebnis ihrer Messung entsprechendes Photon weiterschickt: Bei Bob kommt dann ein Viertel der abgehörten Photonen fehlerhaft an.

Der Satellit als Würfel

Der zweite Ansatz benutzt ein ähnliches Verfahren zur Bestimmung eines gemeinsamen Schlüssels, geht dabei aber nicht von einem Quantenkanal aus, sondern von einem Satellitensystem, das Zufallszahlen aussendet. Die Idee stammt von Ueli M. Maurer, Informatik-Professor an der ETHZ. Maurers Satellitenlösung wirft die Frage auf, ob Eve, unsere Lauscherin, die die Satellitensignale mithört, die Schlüssel nicht nachbauen kann. Wenn die Anlage geschickt genug konstruiert ist, so der Erfinder, ist dies nicht nur unwahrscheinlich, sondern sogar beweisbar unmöglich. Der Trick besteht darin, den Sender im Satelliten so schwach einzustellen, dass auf der Erde kein fehlerfreier Empfang möglich ist. Mit anderen Worten: Alice, Bob und Eve verwenden zwar die gleiche Quelle, um sich einen bestimmten Schlüssel zu basteln, aber jeder von ihnen empfängt aus technischen Gründen ein anderes Signal. Damit die beiden Kommunikationspartner Alice und Bob trotzdem auf einen gemeinsamen Schlüssel kommen, sind wie bei der Methode mit dem Quantenkanal einige Verarbeitungsschritte nötig. Zunächst unterteilen Alice und Bob ihre empfangenen Signale in Gruppen und eruieren gegenseitig, bei welchen dieser Gruppen sie Übereinstimmungen haben. Alle anderen Gruppen werden aus den weiteren Betrachtungen ausgeschlossen. Auf diese Weise arbeiten sich Alice und Bob schrittweise vor. Sie erhalten so immer kürzere Zahlenfolgen, die sich immer ähnlicher werden. Am Schluss stimmen die von Alice und Bob getrennt errechneten Resultate exakt überein und können somit als echt geheimer Schlüssel dienen.

Beispiel

Nehmen wir an, der Satellit sende die folgenden Zahlen:

```
1001 1100 1011 1101 1001 0101 1110 1011
0000 0010 1101 1100
```

(Inder Praxis würde die Zahlenfolge natürlich nicht abreissen; Alice und Bob würden ihre Länge so wählen, dass sie daraus einen Schlüssel extrahieren könnten, der gleich lang ist wie die geheime Nachricht.)

Nehmen wir weiter an, Alice empfange vom Satelliten die Zahlenfolge:

```
1000 1100 1111 0101 1000 0101 1110 1001
0010 0010 1101 0100
```

Alice teilt die empfangene Folge in Zweierpakete und erreichnet von jedem Paket den XOR-Wert. Daraus ergibt sich eine neue zweite Folge, die pro ursprünglich empfangenen Zweierblock nur noch eine Ziffer enthält:

```
1 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 0 1
0 1 1 0
```

Dieses Resultat schickt Alice über einen öffentlichen Kanal an Bob. Nehmen wir weiter an, Bob habe gleichzeitig mit Alice vom Satelliten folgende Zahlenfolge empfangen:

```
1001 1000 1011 0100 0101 0001 1110 0011
1000 0010 1101 1000
```

Auch Bob teilt die empfangene Zahlenfolge in Zweierpakete und errechnet von jedem Paket den XOR-Wert. Das ergibt bei Bob als zweite Folge:

```
1 1 1 0 1 0 1 0 1 1 0 1 0 1 0 0 1 0 0 1
0 1 1 0
```

Bob vergleicht seine zweite Folge mit jener, die er von Alice empfangen hat und meldet ihr umgehend, wo Übereinstimmung herrscht. Alice und Bob betrachten jetzt nur noch die Zweierblöcke ihrer ursprünglich empfangenen Folge, bei denen ihre zweiten Folgen Übereinstimmung zeigen. Von jedem dieser Zweierblöcke nehmen sie nur das erste Bit, das sie vom Satelliten empfangen haben.
Aus der Folge, die Alice vom Satelliten empfangen hat, nimmt sie also lediglich die Zweierblöcke

```
10 00 11 01 10 01 11 10 00 10 11 01 01
00
```

und von diesen das erste Bit, was eine dritte Folge ergibt:

```
1 0 1 0 1 0 1 1 0 1 1 0 0 0
```

Bob geht gleichermassen vor. Aus seiner Zahlenfolge entstehen die Zweierblöcke

```
10 00 11 01 01 01 11 10 00 10 11 01 10
00
```

Nimmt er von diesen das erste Bit, ergibt sich als dritte Folge:

```
1 0 1 0 0 0 1 1 0 1 1 0 1 0
```

Man sieht: Die dritten Folgen stimmen bei Alice und Bob schon recht stark überein. Diskrepanzen zeigen sich nur noch an zwei Stellen.

Mit Iteration zum Ziel

Da auch die dritten Folgen von Alice und Bob noch nicht übereinstimmen, iterieren die Kommunikationspartner das Vefahren. Alice teilt ihre dritte Folge wieder in Zweiterpakete auf und errechnet von jedem Paket den XOR-Wert. Die vierte Folge lautet dann bei Alice:

```
1 1 1 0 1 1 0
```

Alice schickt ihre vierte Folge über den öffentlichen Kanal an Bob, während Bob seine dritte Folge ebenfalls in Zweierpakete aufteilt und seine vierte Folge berechnet:

```
1 1 0 0 1 1 1
```

Bob vergleicht sodann seine vierte Folge mit jener von Alice und teilt dieser die Stellen mit, an denen Übereinstimmung herrscht. Für die fünfte Folge nehmen beide Kommunikationspartner das erste Bit jener Zweierblöcke, bei denen Übereinstimmung herrscht. So ergibt sich für Alice

```
1 1 1 0 1
```

Bob erhält das gleiche Resultat. Damit sind die beiden fast am Ziel: Sie haben jetzt eine übereinstimmende Folge, über die alle anderen höchstens unvollständige Informationen besitzen.

Der Schlüssel ist wirklich geheim

Bleibt nachzuweisen, dass Lauscherin Eve die gemeinsame Folge von Alice und Bob unter keinen Umständen eruieren kann. Selbst wenn Eve das Satellitensignal besser (mit weniger Fehlern) als Alice und Bob empfängt, wird die Fehlerrate ihrer Folge mit jeder Iteration statistisch zunehmen und schliesslich 50% betragen – Eve könnte ebensogut versuchen, die Folge mit Münzwürfen zu eruieren.

Dominik Moser

Literaturhinweise:

[1] Shannon, Claude Elwood: «Communication Theory of Secrecy Systems», *Bell System Technical Journal,* 28/1949, 656–715

[2] Vernam, Gilbert S.: «Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communication», *Journal of American Institute of Electric Engineering,* XLV/1926, 109–128

[3] Bennett, Brassard: «Experimental Quantum Cryptography», *Journal of Cryptology,* 5/1992, 3–28

Oberon V4 – 2. Teil

Nachdem ich im letzten PowerKey auf den ersten Betarelease von Oberon für NEXTSTEP verwiesen habe, möchte ich in dieser Ausgabe zeigen, wie das System aufgebaut ist und funktioniert (womit wahrscheinlich die Hacker angesprochen sind).

Erscheinungsbild

In Bild 1 sieht man die typische Arbeitsumgebung von Oberon V4. Was als erstes auffällt sind die sogenannten «Tiled Windows» – viele Leute meinen sogar, das seien gar keine echten Windows (ich finde das ganz natürlich: Schliesslich gleichen sich die kommerziellen Fenstersysteme gegenseitig immer mehr an. Schon gewusst, dass *Windows95* einen Nextstep-Window-Close-Button hat?!). Nun hat aber das Tiling auch einige Vorteile: Es ist sehr einfach zu implementieren und spart dem Benutzer die Suche nach verdeckten Fenstern. Den Nachteil hab' ich schon angetönt: Es sieht leider nicht so poppig aus…

Neben dem obligaten *MineSweeper* und dem *Draw* haben es sicherlich die Texte in sich. Im *Welcome.Text* habe ich mit wenigen Kommandi einige Elemente eingesetzt, die wie Zeichen im Text mitschwimmen. Neben Tabellen, Grafiken, Uhr, Icons (das Männchen mit Hut ist der Screensaver!) und Datum sind beliebige Erweiterungen möglich. Erstaunlicherweise sind auch Standardoperationen wie Blocksatz und Einrücken über Elemente implementiert.

Permission to use, copy, modify and distribute this software and its

ParcElem im Blocksatz-Modus

Verschiedene Version dieser Paragraphenelemente können auch mit einem Namen versehen werden, um den Effekt von konsistenten Absatzformaten wie in kommerziellen Textverarbeitungsdinosauriern (z.B. Word, Frame, etc.) zu erreichen.

Paragraph

1. Introduction

Oberon-2 is a general-purpose language in the tradition of Oberon and Modula-2. Its most important features are block

StyleElems: Absatzformate mit demselben Namen bleiben im ganzen Dokument gleich, können aber an jeder Stelle geändert werden

Ein weiterer positiver Nebeneffekt dieser Architektur ist, dass Editoren (Grafik-, Text-) praktisch ohne Aufwand in Elemente verwandelt werden können, wobei zwischen Inplace-Editing und externem Editieren gewählt werden kann.

1.	BOOLEAN	the truth values TRUE and FALSE
2.	CHAR	the characters of the extended ASCII set (0X
3.	SHORTINT	the integers between MIN(SHORTINT) and M
4.	INTEGER	the integers between MIN(INTEGER) and MA
5.	LONGINT	the integers between MIN(LONGINT) and MA
6.	REAL	the real numbers between MIN(REAL) and M
7.	LONGREAL	the real numbers between MIN(LONGREAL)

Tabulatoren in ParcElems

Eine sehr praktische Anwendung von Textelementen sehen wir im kleine Sourcefenster *Hello.Mod*: Compilerfehler werden automatisch im Text als Elemente eingefügt. Damit der Quelltext nicht überladen wird, sind die ErrorElems zwischen Fehlercode und Text umschaltbar.

```
PROCEDURE GetDate*(year: INTEGER; VAR day, month: INTEG
(* Source: Knuth, The Art of Computer Programming, Vol. I,p.155,
   VAR a : INTEGER;
BEGIN
   IF year <= 1582 THEN HALT(99) END;
   g undeclared identifier := year MOD 19 + 1;
   c 0 := year DIV 100 + 1;
   x 0 := 3*c DIV 4 - 12; z 0 := (8*c+5) DIV 25 - 5;
   d 0 := 5*year DIV 4 - x 0 - 10;
   e 0 := (11*g+20+z 0 -x) MOD 30;
   IF ((e 0 =25) & (g>11 incompatible operands of dyadic operator ))
      OR (e=24 100 ) THEN INC(e 0 ) END;
   n 0 := 44-e; IF n 0 <21 THEN n 0 := n+30 END;
```

Beispiel für Fehler-Elemente: Zweimal derselbe Fehler, mit Text oder mit Code

Betriebssystem oder nicht?

Oberon in seiner Originalversion läuft als eigenständiges Betriebssystem und kontrolliert direkt die Hardware. Die meisten Portierungen auf Unix- oder PC-Systeme nutzen aber das bestehende Betriebssystem (in diesem Fall Mach/BSD). Was ist es jetzt genau? Läuft Oberon auf einem Emulator bzw. Simulator unter Unix? Eine weitere Frage bleibt unbeantwortet: Wie kann man ein System in Oberon schreiben, wenn das darunterliegende Unix keine Ahnung davon hat? Die Antwort ist einfach: Wie wenn wir auf nackter Hardware ein Betriebssystem booten, brauchen wir auch hier unseren Bootloader, um Oberon aus einem Unix-Prozess heraus zu starten.

Bootstrapping

Den Bootloader schreiben wir am besten in unserer vertrauten Unix-Umgebung, d.h. in Highlevel-Assembler (lies C). Damit wollen wir zwei Ziele erreichen: Eine Umgebung für Oberon schaffen (d.h. ein ausreichend grosses Stück Speicher – zwischen 2 und 4 MB – bereitstellen) und eine Schnittstelle zum Host-OS (I/O für Files, Screen, Events, …) erstellen. Der Algorithmus (in Pseudo Highlevel-Assembler) sieht etwa wie folgt aus:

```
1  typedef void (*Proc)();
2
3  FILE *fd;
4  int heapAdr;
5
6  static void dlsym(int handle,char *symbol,
   int *adr)
7  {
8  if (!rld_lookup(NULL, symbol_name, (unsigned
   long *)adr))
9       fprintf(stderr, "Oberon: Symbol not found:
        %s\n", symbol);
10 }
11
12 void Boot()
13 {
14      int fileHeapAdr, fileHeapSize, dlsymAdr;
15      Proc body;
16
17      fd = fopen("iOberon.Boot", "r"); // bootfilename
18      CopyBootfile(fd, heapAdr);
19      body = (Proc)(EntryPointInBootfile() +
        heapAdr - fileHeapAdr);
20      dlsymAdr = SearchDlsymAdrInBootfile();
21      heapAdr + dlsymAdr = (void *)dlsym;
22      fclose(fd);
23      (*body)();
24 }
25
26 void main(int argc, char *argv[])
27 {
28      heapSize = 0x400000;
29      heapAdr = (int) malloc(heapSize);
30      Boot();
31 }
```

Dieses Codestück ist nichts anderes als vereinfachter Bootloadercode. Das Original ist ca. 200 Zeilen lang.

Prozeduren wie *CopyBootfile*, *EntryPointInBootfile* und *SearchDlsymAdrInBootfile* sind einfache Iterationen die den Inhalt des Bootfiles kopieren bzw. nach Magics im Bootfile suchen. Jeder erkennt nun, dass Zeile 23 der heisse Punkt ist: Hier wird die erste Oberon-Prozedur gestartet. Damit das auch klappt, müssen die Module des inneren Kerns (siehe Modullisting) fertig reloziert, mit aufgelösten externen Referenzen (Imports) im Speicher stehen. Man sieht nun leicht, dass dazu eine weitere Transformation des Codes nötig ist. Wir benötigen also ein Programm, das die Module *Kernel*, *Console*, *Unix*, *Files* und *Modules* in ein Load-Image verwandelt und in einem File ablegt. Dieses Programm heisst Bootlinker und ist in Oberon geschrieben, weil wir es nicht beim Booten sondern vor dem Booten brauchen und somit auf einer anderen Maschine laufen lassen können. Warum nun aber gerade diese 5 Module? Sie stellen das Minimum dar, um weitere Module zu laden und somit das System hochzufahren. *Modules* (der dynamische Lader) importiert *Files* (lesen der Objectfiles), *Kernel* (Speicher allozieren) und *Files* wiederum braucht das Modul *Unix*, um die Files mit dem Unix-Systemcall *read* zu lesen. Das Modul *Console* ist optional eingebunden.

Outer Core

Sind die 5 Basismodule erfolgreich geladen wird der Oberon-Eventloop gestartet. Das ist genau so simpel wie es tönt, es genügt ein einziger Befehl:

```
ThisCommand(ThisModule("Oberon"),"Loop");
```

Im Hintergrund jedoch werden alle importierten Module von Oberon nachgeladen und initialisiert. Ebenso wird bei der Initialisierung von Oberon das Modul *System* geladen, welches das Logfenster und ein Systemtool

öffnet. Zu diesem Zeitpunkt ist das Betriebssystem voll funktionsfähig und verarbeitet alle Mouse- und Keyboard-Events. In der Modulliste sehen wir die bis jetzt geladenen Module unterhalb von System.

```
System.ShowModules
IconElems       codesize = 5071       refcnt = 0
StampElems      codesize = 3667       refcnt = 0
TableElems      codesize = 15640      refcnt = 0
ClockElems      codesize = 3506       refcnt = 0
ErrorElems      codesize = 4215       refcnt = 0
iCompiler       codesize = 3188       refcnt = 0
iOPV            codesize = 10545      refcnt = 1
iOPC            codesize = 39069      refcnt = 2
iOPL            codesize = 37554      refcnt = 3
iOPO            codesize = 9826       refcnt = 4
iOPP            codesize = 17948      refcnt = 1
iOPB            codesize = 26988      refcnt = 2
iOPT            codesize = 11994      refcnt = 7
iOPS            codesize = 5415       refcnt = 8
iOPM            codesize = 3058       refcnt = 9
Kepler6         codesize = 6706       refcnt = 0
Kepler5         codesize = 817        refcnt = 0
Kepler4         codesize = 3249       refcnt = 0
Kepler1         codesize = 6541       refcnt = 0
Kepler8         codesize = 7408       refcnt = 0
Kepler2         codesize = 1521       refcnt = 1
Kepler9         codesize = 5644       refcnt = 0
Math            codesize = 380        refcnt = 7
Kepler          codesize = 6815       refcnt = 0
In              codesize = 1083       refcnt = 3
KeplerFrames    codesize = 16085      refcnt = 8
KeplerGraphs    codesize = 9219       refcnt = 9
Types           codesize = 433        refcnt = 1
KeplerPorts     codesize = 11251      refcnt = 10
Display1        codesize = 1645       refcnt = 2
Rectangles      codesize = 3533       refcnt = 0
Draw            codesize = 5843       refcnt = 0
GraphicFrames   codesize = 9997       refcnt = 2
Graphics        codesize = 11112      refcnt = 3
MineSweeper     codesize = 16303      refcnt = 0
XE              codesize = 22616      refcnt = 0
FoldElems       codesize = 7458       refcnt = 1
LineElems       codesize = 2777       refcnt = 0
Edit            codesize = 12753      refcnt = 0
PopupElems      codesize = 8061       refcnt = 0
System          codesize = 15625      refcnt = 0
ParcElems       codesize = 14565      refcnt = 1
TextPrinter     codesize = 12272      refcnt = 12
Printer         codesize = 20080      refcnt = 14
TextFrames      codesize = 31435      refcnt = 20
MenuViewers     codesize = 5004       refcnt = 13
Oberon          codesize = 7083       refcnt = 30
Texts           codesize = 17296      refcnt = 40
Reals           codesize = 1296       refcnt = 1
Viewers         codesize = 3844       refcnt = 31
Fonts           codesize = 2726       refcnt = 44
FontsType3      codesize = 11055      refcnt = 1
Display         codesize = 7317       refcnt = 48
Input           codesize = 793        refcnt = 18
DPS             codesize = 2776       refcnt = 8
Kernel          codesize = 7482       refcnt = 19
Unix            codesize = 4040       refcnt = 10
Console         codesize = 811        refcnt = 9
Files           codesize = 10573      refcnt = 38
Modules         codesize = 8924       refcnt = 13
```

i386 Compiler

Oberon Outer Core (System ohne Kern)

Oberon Inner Core (Kernel, Bootfile)

Die Liste der geladenen Module. Sie entspricht dem Systemzustand des Screenshots in Bild 1

Abgesehen von Anpassungen bei *Display*, *Input* und im Traphandler entspricht der Code des Outer Core demjenigen, der im Standardoberonsystem verwendet wird (veröffentlicht in Wirth's Buch *Project Oberon* oder zu beziehen via: *ftp:/neptune.ethz.ch//pub/Oberon/Sources/ProjectOberon.V4.tar.Z*).

Benutzermodule und Objekte

Wie es von hier nun weitergeht, hängt vom Anwender ab. Beim Anklicken von Kommandi der Form

```
Module.Procedure
```

wird *Module* dynamisch nachgeladen (falls noch nicht im Speicher). Dynamische Objekte die innerhalb von Modulen mit `NEW` instanziert wurden, werden vom Garbage Collector verwaltet und nach Gebrauch automatisch wieder eingesammelt.

Zurück zur Frage: Ist es nun ein Betriebssystem? Ein Emulator? Ein Simulator? So etwas wie SoftPC? Ein Betriebssystem zweifelsfrei, schliesslich läuft (fast) derselbe Code auf nackter Hardware und die Objectfiles enthalten Maschinencode für den Prozessor. Die Unterschiede liegen darin, dass beim Ansatz des eingebetteten Systems Devicedriver durch abstrakte Hüllen ersetzt wurden, die mit dem Host-OS verhandeln, um denselben Effekt zu erreichen.

George Fankhauser
<george@ifor.math.ethz.ch>

Anm. der Redaktion:
Der aktuelle Release von Oberon V4 für NEXTSTEP befindet sich auf dem Fileserver der NiCE im Verzeichnis *~ftp/NiCE* und zwar für

i386: Neue Version mit vielen Goodies
m68k: Beta (keine Änderungen)

Software

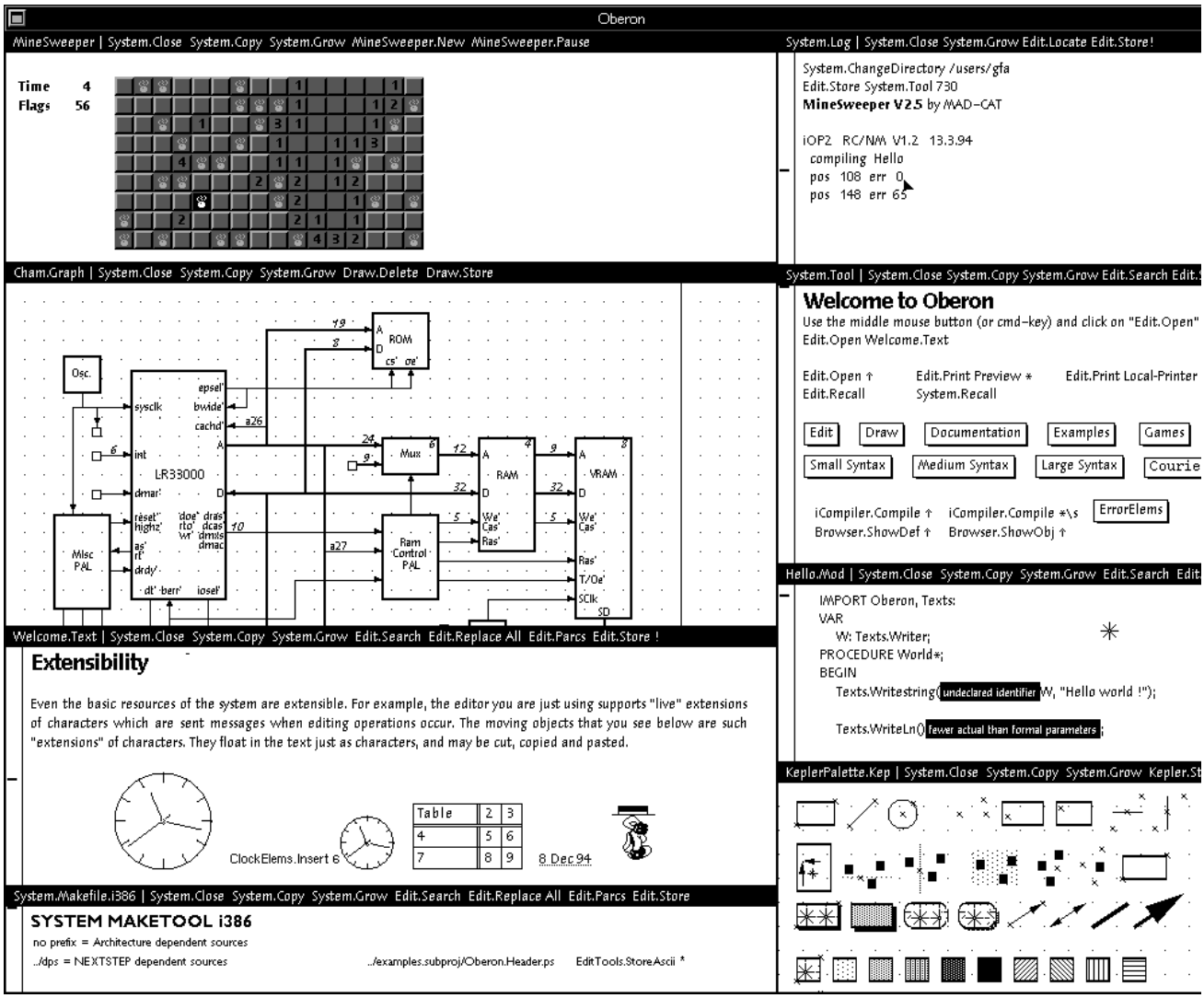


Bild 1: Oberon-Screenshot

NEXTSTEP 3.3 Developer

Wie NeXT Computer Inc. am 7. Dezember 94 per Usenet bekanntgab, begann die Auslieferung des Upgrades auf NEXTSTEP Release 3.3 ohne Verzögerungen. Gleichzeitig wurde mitgeteilt, dass sich die Developer Version des Release 3.3 jetzt im Beta-Stadium befindet.

Anm. der Redaktion: Da die Mitteilung von NeXT genau zum Zeitpunkt des Redaktionsschlusses dieser Ausgabe erschien, wird sie (mangels Übersetzungszeit) im Originalwortlaut wiedergegeben.

NeXT SHIPS RELEASE 3.3 FOR INTEL AND ANNOUNCES BETA FOR NEXTSTEP DEVELOPER 3.3

REDWOOD CITY, Calif.-December 7, 1994 – NeXT Computer, Inc. today announced that it is now shipping a new version of its object-oriented operating system, NEXTSTEP Release 3.3 for Intel and Motorola processors. The company also announced that it is in beta with NEXTSTEP Developer Release 3.3.

"The NEXTSTEP products that we are shipping today are a result of customer feedback and our goal in 1995 is to continue to offer NEXTSTEP/OpenStep customers even greater development and deployment choices to solve their toughest business problems," said Steven P. Jobs, Chairman and CEO of NeXT Computer, Inc. "As a result, NEXTSTEP Release 3.3 for PA-RISC and SPARC processors will be available in mid-1995."

NEXTSTEP Developer Release 3.3 Enters Beta
NEXTSTEP Developer Release 3.3, now in beta, is optimized to enable developers to build enterprise-wide, client/server production applications for Intel, Motorola, PA-RISC and SPARC architectures. The product also includes a new Foundation Kit and increased C++ support. Specifically, NEXTSTEP Developer Release 3.3 simplifies application development by allowing developers to create multi-architecture executables which run on all four architectures supported by NeXT.

For example, the compiler allows developers to create executables for Intel, Motorola, Sparc and PA-RISC processors from any of these machines. The result: applications can be installed on a network, yet run on any of the four architectures.

NEXTSTEP Developer Release 3.3 also includes a C++ compiler which allows developers to create C++ objects, as well as Objective C objects. The improved compiler enhances C++ support and includes multiple inheritance and templates. NEXTSTEP Developer 3.3 continues to also support Objective C++, NeXT's integrated Objective C & C++ compiler.

As part of NEXTSTEP Developer 3.3, NeXT will include it's Foundation Kit which provides significant leverage to developers by greatly simplifying the task of programming basic functions. Foundation provides the building blocks upon which OpenStep and NeXT's Enterprise Objects Framework is built. Foundation provides unicode capable string objects, collection objects such as arrays and dictionaries, notification and archiving objects as well as objects for interacting with the file system and other OS features such as threads and processes. Additionally, Foundation provides a powerful object allocation strategy which simplifies many APIs. This allocation strategy is used throughout OpenStep and Enterprise Object Framework.

Pricing and availability
NEXTSTEP Release 3.3 for Intel and Motorola processors is available now for $799 with upgrades priced at $199. NEXTSTEP Release 3.3 for PA-RISC and SPARC processors will be available in mid-1995 for the same price. NEXTSTEP Developer Release 3.3 for Intel, Motorola, PA-RISC and SPARC will be available in mid-1995. Pricing will be announced at that time.

Bearbeitung: (dm)

Pentium FPU-Bug

In den letzten Wochen wurde die Usenet-Group comp.sys.intel mit Meldungen geradezu überflutet, wonach sich in der FPU (floating point unit) des Pentium ein Designfehler befinden soll. Gleitkommadivisionen (FDIV) mit doppelter Genauigkeit können unter gewissen Umständen falsche Resultate zur Folge haben.

Die ganze Aufregung begann mit einem Artikel in der Zeitschrift *EE Times* (Ausgabe 822 vom 7. November). Darin war zu lesen, dass die Intel Corp. zur Korrektur von «Anomalien» die zu falschen Rechenresultaten geführt hatten, die FPU des Pentium abändern musste. Der Fehler sei bereits Mitte Jahr erkannt und behoben worden.

Die Öffentlichkeit erfuhr davon vorerst nichts, denn nicht Intel informierte über den Bug, sondern Thomas Nicely, ein Professor für Mathematik am Lynchburg College in Virginia. Nach seinen Angaben erhält man für gewisse Gleitkommadivisionen fehlerhafte Resultate. Eines seiner Beispiele bedient sich der Zahl

```
p = 824633702441
```

Mit exakter Rechnung würde

```
q = 1 - (1/p)*p
```

Null ergeben. Mit Gleitkommaoperationen müsste *q* zumindest in der Grössenordnung der Maschinengenauigkeit (eps) liegen. Auf den meisten Rechnern erhält man deshalb auch nicht Null sondern

```
q = eps/2 = 2^(-53) ≈ 1.11e-16
```

Ein Pentiumsystem hingegen liefert als Ergebnis

```
q = 2^(-28) ≈ 3.72e-09
```

Das ist nicht einmal einfache Genauigkeit. Alle Ziffern der Division *1/p* ab der achten signifikanten Stelle sind falsch.

Dramatischer sieht das folgende Beispiel aus:

```
x = 4195835
y = 3145727
z = x - (x/y)*y
```

Wiederum würde exakte Rechnung Null ergeben. Auf den meisten Maschinen erhält man sogar Null, eingeschlossen auf solchen, die Intel's 386'er oder 486'er benutzen. Der Pentium hingegen behauptet, dass

```
z = 256
```

ist. Der relative Fehler *z/x* ist ungefähr 6.1e-5.

Noch ist nicht klar, welche Operanden auf dem Pentium zu Rechnerfehlern führen. Der Fehler liegt ganz klar im Chip selber; er tritt bei allen Frequenzen auf. Bis heute sind noch keine Meldungen von Pentium-Prozessoren bekannt, bei denen besagter Fehler nicht auftritt. Angesichts der gut 2 Millionen Stück, die Intel verkauft hat, wird es wohl noch ein Weilchen dauern, bis die ersten korrigierten Exemplare im Verkauf sind.

Keep your desktop tidy !

Getreu dem Motto *«Wer den Schaden hat braucht für den Spott nicht zu sorgen»*, kursierten kurze Zeit später die ersten Intel-Jokes auf dem Internet. Zum Abschluss deshalb der «neue» Firmenslogan von Intel:

Intel inside – but can it divide?

(dm)

Was sie schon immer über TeX wissen wollten – Teil 7

In der letzten Folge des TeX-Kurses geht es darum, mit TeX etwas ausgefallenere Dinge zu tun als «einfach nur Text» zu setzen. Dabei wird man sich zwangsläufig ein wenig mit der internen Arbeitsweise von TeX auseinandersetzen müssen.

Es ist nichts sonderlich Ausgefallenes was ich als Beispiel für diese Folge ausgewählt habe, aber es dient dem Zweck, einen kleinen Einblick in die Arbeitsweise von TeX zu geben. Das gestellte Problem ist das folgende: Innerhalb einer Absatzbreite sollen drei Grafiken in Form eines Dreiecks

Grafik1 *Grafik2*
Grafik3

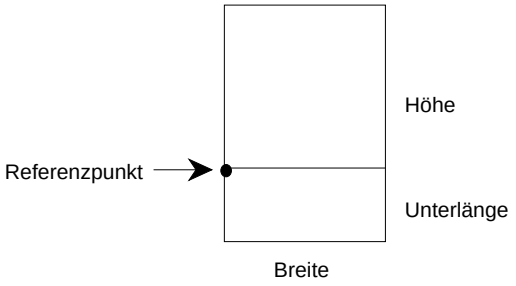
gesetzt werden. Um hierbei die richtigen Boxoperationen herauszufinden, muss man sich mit der Funktionsweise von TeX auseinandersetzen.

Die wichtigsten TeX-Interna

Obwohl in den ersten Folgen dieses Kurses schon auf die Zeichenkodierung von TeX eingegangen wurde, möchte ich hier der Vollständigkeit halber noch einmal kurz die wichtigsten Punkte wiederholen. Bei der Bearbeitung eines *.tex*-Files durch TeX wird jedes Eingabezeichen gelesen und bekommt intern einen Zahlenkode zugeordnet. Neben dem *Kodierungswert* für die eingelesenen Zeichen kennt TeX zusätzlich einen sog. *Bedeutungswert*, d.h. jedem Zeichen wird nicht nur ein Kodierungswert 0…256, sondern auch ein Bedeutungswert (*Kategoriekode*) 0…15 zugewiesen. Enthält der Eingabetext ein Befehlswort, so wird dieses nicht weiter aufgespalten, sondern als eigenständiges *Atom* angesehen, das keinen Kategoriekode sondern lediglich eine interne TeX-Kennung besitzt. Der Eingabetext wird also in einem ersten Schritt in einen Tokenstrom verwandelt, bestehend aus Zeichentoken und Befehlstoken. Bei allen weiteren Bearbeitungsschritten ist der Begriff der *Box* von zentraler Bedeutung.

Alles ist Box

Knuth vergleicht TeX mit einem lebenden Organismus, der einen Mund und einen Magen besitzt. Die Daten, mit denen der Magen von TeX operiert, heissen Boxen. Eine Box ist ein Rechteck, dem drei charakteristische Werte zugewiesen werden:



Der Referenzpunkt einer Box ist stets der Schnittpunkt der Grundlinie mit dem linken Rand der Box. Die drei charakteristischen Werte *Breite* (width), *Höhe* (height) und *Tiefe* (depth) – im dt. Sprachraum oft *Unterlänge* genannt –, können positive oder negative Masse annehmen.

Die einfachste Box schliesst den einzelnen Zeichentoken ein. Der Buchstabe ’e’ und der Buchstabe ’h’ besitzen beispielsweise verschiedene Höhen, aber gleiche Unterlängen (bei beiden mit Wert 0). Ein Buchstabe mit positiver Unterlänge ist ’g’. Hier stimmt die Höhe mit der von ’e’ überein. Hier einige Buchstaben mit ihren dazugehörigen Boxen:

$abcghi$ 

Die charakteristischen Boxwerte eines Fonts entnimmt TeX den zugehörigen *.tfm* Files. Des weiteren benötigt das Ausgabeprogramm die Information, wie die Rechtecke mit Farbe zu füllen sind. Diese Information enthalten die Dateien mit der Endung *.pk*.

Die einzelnen Zeichenboxen bilden, neben- oder übereinandergestellt, grössere umschliessende Boxen. Werden Boxen nebeneinander angeordnet, so werden sie so ausgerichtet, dass ihre Grundlinie eine gemeinsame horizontale Linie bildet. Bei übereinander gestellten Boxen stehen deren Referenzpunkte senkrecht übereinander, so dass sie durch eine vertikale Linie verbunden werden können.

Ob Boxen horizontal oder vertikal angeordnet werden, hängt vom Bearbeitungsmodus ab, in dem sich TeX gerade befindet. Im horizontalen Bearbeitungsmodus (horizontal mode) werden benachbarte Boxen nebeneinander angeordnet, im vertikalen Bearbeitungsmodus (vertical mode) dagegen übereinander. Normalerweise erfolgt die Umschaltung zwischen diesen Modi durch TeX, ohne dass sich der Anwender hierüber Gedanken machen muss. Mit den Befehlen `\hbox` oder `\vbox` kann der Anwender bei Bedarf umschliessende horizontale oder vertikale Boxen zur Aufnahme von elementaren Boxen einrichten. Hierzu stehen u.a. folgende Syntaxformen bereit:

`\hbox{`*hor_text*`}`
`\vbox{`*vert_text*`}`
`\hbox to` *mass* `{`*hor_text*`}`
`\vbox to` *mass* `{`*vert_text*`}`

Hierin steht *hor_text* für beliebige horizontale Textstrukturen, d.h. für normalen Text, der mit Befehlen vermischt sein darf, die den horizontalen Bearbeitungsmodus nicht verlassen. Umgekehrt steht *vert_text* für solche Textstrukturen, die übereinander angeordnet werden. Diese bestehen im allgemeinen aus horizontalen Teilstrukturen, wie zum Beispiel mehrere horizontale Boxen, die innerhalb der `\vbox` übereinander positioniert werden.

Bei den Boxbefehlen in der ersten Syntaxform bestimmt der eingeschlossene Text die horizontale Ausdehnung einer `\hbox` bzw. die vertikale Ausdehnung einer `\vbox`. Die Eingabe

```
\hbox{Dies ist eine typische Textzeile}
```

baut sich aus den Zeichenboxen strukturmässig so auf:

Die dünne Umrahmung stellt die erzeugte `\hbox` als Ganzes dar, die sich aus den etwas stärker gezeichneten Zeichenboxen aufbaut. Die Eingabe für eine `\vbox` in Form mehrerer horizontaler Boxen

```
\vbox{
  \hbox{Zwei Zeilen mit normalem Text}
  \hbox{erzeugen eine vertikale Box}
}
```

führt zu folgendem Ergebnis:

Bei der zweiten Syntaxform der TeX-Boxstrukturen wird die Breite für eine `\hbox` bzw. die Höhe für eine `\vbox` fest vorgegeben. Die Angabe

```
\hbox to 60mm {Zu wenig Text f\"ur
  60mm}\par
\hbox to 30mm {Zu viel Text f\"ur
  30mm}
```

richtet im ersten Fall eine `\hbox` mit der Breite 60 *mm* ein, in der der übergebene Text mit entsprechenden Wortabständen angeordnet wird. Als Folge der grossen Wortabstände erscheint während der Bearbeitung eine `"Underfull \hbox …"`-Warnung. Mit dem zweiten Befehl wird eine `\hbox` der festen Breite von 30 *mm* eingerichtet, für die der übergebene Text zu lang ist. Er ragt über den rechten Rand der Box hinaus, mit dem Ergebnis einer `"Overfull \hbox …"`-Warnung. Das Bearbeitungsergebnis selbst erscheint als

Zu wenig Text für 60mm
Zu viel Text für 30mm

Die Ausführungen zu den horizontalen Boxen vorgegebener Breite lassen sich sinngemäss auf vertikale Boxen vorgegebener Vertikalabmessung übertragen.

Fest, gedehnt und geleimt

Wenn TeX einen Text liest, werden die Boxen, die die Buchstaben darstellen, zu grösseren Boxen, den einzelnen Wörtern, zusammengefasst. Wird hierbei ein Abstand eingefügt, um beispielsweise ein Kerning auszuführen (diesen Abstand nennt man *kern*), handelt es sich dabei um einen starren Abstand, der vom Randausgleich und anderen Layouteingriffen unberücksichtigt bleibt! Solche Boxen erhalten als Referenzpunkt den Referenzpunkt der ersten Teilbox.

Zwischen den Wort-Boxen steht eine andere Form von Abständen, genannt *glue*. Das besondere daran sind die drei Bestimmungsstücke: Erstens die natürliche Grösse, die TeX zu erhalten trachtet, zweitens der *Schrumpfanteil*, um den verkürzt und drittens der *Dehnungsanteil*, um den verlängert werden darf. Diese Wörter-Boxen und die glue-Abstände dazwischen werden in eine lange Zeile eingetragen und durch ein `\par` in eine neue Box (den Absatz) gesetzt. Der Referenzpunkt eines Absatzes ist der Referenzpunkt der letzten Zeile, der wiederum mit dem Referenzpunkt des ersten Buchstaben dieser Zeile übereinstimmt.

Um einen Absatz als Rechteck ansehen zu können, muss man am Schluss der letzten Zeile einen beliebig dehnbaren Leerraum einfügen. Dieser ist als

```
\parfillskip=0pt plus 1fil
```

in plain-TeX (also im Makropaket *plain.tex*) vordefiniert, was bedeutet, dass seine natürliche Grösse 0 *pt* beträgt, seine Dehnbarkeit aber im Vergleich zu normalen (in physikalischen Dimensionen angegebenen) Abständen unendlich gross ist.

Kommt in einer Zeile ein `fil` vor, so wird die seit dem letzten `fil` oder dem Zeilenanfang angefangene Box entsprechend ihrer natürlichen Grösse gesetzt und von der weiteren Abstandsberechnung ausgeklammert.

Ist der Absatz beendet, so wird die Box, als die TeX ihn betrachtet, vertikal ausgerichtet. Der benutzte Abstand ist `\parskip`, ein glue mit natürlicher Grösse 0 *pt* und Dehnbarkeit 1 *pt*. Möchte man zwischen zwei Absätzen grundsätzlich einen grösseren Abstand als zwischen den Zeilen haben, so kann man diesen Parameter verändern.

Beginnt der nächste Absatz, so verlässt TeX zwischendurch den Vertikal-Modus des Absatzumbruches und befindet sich erneut im Horizontal-Modus, der beliebig von Mathematik- und Display-Modus unterbrochen werden darf. Daneben kann man auch explizit in einen inneren Vertikal-Modus `\vbox` oder in einen inneren Horizontal-Modus `\hbox` wechseln. Die inneren Modi unterscheiden sich nur in einigen wenigen Befehlsmöglichkeiten, die das System zur Fehlererkennung einbaut, von den entsprechenden globalen Modi.

Spitze nach unten

Mit Hilfe der inneren Modi lässt sich nun das Problem der drei Grafiken lösen. Die Grundidee ist, zunächst mittels `\par` in den globalen Vertikal-Modus zu wechseln und dann eine `\hbox` aufzumachen, in die die ersten beiden Grafiken gesetzt werden:

```
\par
\hbox to \hsize{
   Grafik1
   Grafik2
}
```

In dieses Gerüst fügt man nun dynamisches Material in Form von `fil` ein. Man will die beiden Grafiken jeweils zentriert in je einer Hälfte der Seite haben, fügt also je vor und nach der Grafik ein `fil` ein, die mittleren beiden können addiert werden.

```
\hbox to \hsize{
   \hfil
   Grafik1
   \hglue 0pt plus 2fil
   Grafik2
   \hfil
}
```

Die dritte Grafik wird in eine eigene `\hbox` der Form `{\hfil` *Grafik3* `\hfil}` gesetzt. Man geht nun daran, *Grafik1* und *Grafik2* zu spezifizieren. Es sind Boxen, deren Inhalte (Titelzeile, Leerraum und Fusszeile) zu-

einander vertikal ausgerichtet werden; wir müssen also eine \vbox dafür setzen. Die einzelnen Elemente der \vbox sind \hbox beliebigen Inhaltes:

```
\vbox to x true cm{
   \hbox{Kopfzeile}
   \vfil
   \hbox{Fusszeile}
}
```

Im Prinzip kann man mit dem Ergebnis schon zufrieden sein. Es fehlen nun mehr die Abstände zwischen den einzelnen \vbox und dem umgebenden Text. Damit erhalten wir die endgültige Lösung:

```
\midinsert
  \hbox to \hsize{
    \hfil
    \vbox to 4 true cm{
      \hbox{a)}
      \vfill
      \hbox{Abbildung 1}
    }
    \hglue 0pt plus 2fil
    \vbox to 4 true cm{
      \hbox{b)}
      \vfill
      \hbox{Abbildung 2}
    }
    \hfil
  }
  \vglue 1 true cm plus 0.5 true cm
  minus 0.5 true cm
  \hbox to \hsize{
    \hfil
    \vbox to 4 true cm{
      \hbox{c)}
      \vfill
      \hbox{Abbildung 3}
    }
    \hfil
  }
\endinsert
```

Das Ergebnis sieht (ohne Bilder) nicht sehr spektakulär aus, erfüllt aber alle gestellten Anforderungen. Aus diesem Grund verzichte ich hier auf eine Abbildung des entsprechenden Outputs.

Ausrichten und verschieben

Wie bereits erwähnt, stimmt der Referenzpunkt einer \vbox mit dem Referenzpunkt der untersten \hbox überein. Die ersten beiden Grafiken werden dementsprechend nach der untersten Fusszeile ausgerichtet. Möchte man hingegen zwei \vbox nach der ersten Zeile (was ja eine \hbox ist) ausrichten, so muss man den Befehl \vtop anstelle von \vbox verwenden. Daneben kann man noch ganze Boxen beliebig verschieben: \moveright und \moveleft verschiebt \vbox, \raise und \lower verschiebt \hbox. Das TeX-Logo wird zum Beispiel durch die Anweisung

```
T \kern -0.1667em \lower 0.5ex \hbox{E}
\kern -0.125em X
```

gesetzt, was der Reihe nach bedeutet: Setze ein 'T', rutsche um 0.1667 *em* nach links, setze ein um 0.5 *ex* nach unten verschobenes 'E', rutsche um 0.125 *em* nach links und setze ein 'X', wobei *em* und *ex* im Druckergewerbe übliche Abstände sind. Das ganze sieht bekannterweise folgendermassen aus:

$$T_{\mathrm{E}}X$$

Fazit

Damit sind wir am Ende dieses TeX-Kurses angelangt. In Zukunft werde ich in unregelmässigen Abständen in einer neuen TeX-Kolumne über Wissenswertes rund um TeX berichten. Seien das Neuigkeiten zu TeX oder LaTeX, interessante Erweiterungen und Tools, oder einfach nur konkrete Anwendungen.

Ich hoffe ich konnte mit diesem Kurs einen kleinen Einblick in die Vielseitigkeit von TeX vermitteln und vielleicht den einen oder anderen dazu ermuntern einmal mit TeX zu experimentieren.

Dominik Moser

Impressum

Herausgeber: NiCE – NeXT User Group

PowerKey ist das Magazin der NiCE und erscheint viermal jährlich. Ein Abonnement ist in der Mitgliedschaft bei der NiCE enthalten. ***PowerKey*** wird vollständig auf NeXT-Computern mit *WriteNow* erstellt.
Auflage: 230 Exemplare • Einzelverkaufspreis: Fr. 7.–

Redaktion:
Verantwortlicher Redaktor: Dominik Moser (dm)
Mitarbeiter dieser Ausgabe: George Fankhauser, Felix Rauch

"Wir bemühen uns, sowohl die männliche als auch die weibliche Schreibform zu verwenden. Wo wir dies zugunsten einer besseren Lesbarkeit nicht tun, beziehen sich sämtliche Aussagen auf Männer und Frauen."

Redaktionsadresse:
Dominik Moser, Dörflistrasse 41, 8942 Oberrieden
Anfragen und Inserate von Mitgliedern bitte nur schriftlich.
Adressänderungen bitte an den Aktuar bzw. Kassier.

Anzeigenpreise:
8 × 5 cm Fr. 60.–, 8 × 10 cm Fr. 100.–, 16 × 10 / 8 × 20 cm Fr. 175.–, 1 Seite A4 Fr. 300.–, Mengenrabatt bereits ab 2 Ausgaben!
Einmalige, nicht gewerbsmässige Inserate von Mitgliedern gratis.

NiCE – NeXT User Group

Vereinsadresse: NiCE – NeXT User Group
Rechenzentrum
ETH Zentrum
8092 Zürich

PC-Konto: 80-46102-5

Vorstand

Präsident: Neil Franklin
Morgenweg 8, 8404 Winterthur

Vizepräsident: Christian Limpach
Mainaustr. 44, 8008 Zürich

Aktuar: Albin Mächler
Jonas Furrer-Str. 97, 8400 Winterthur

Kassier: Werner Burri
Chisweg 17, 5313 Klingnau

Redaktor: Dominik Moser
Dörflistr. 41, 8942 Oberrieden

Verleger: Peter Burgdorfer
Illnauerstr. 42, 8307 Effretikon

Einkäufer: Felix Rauch
Im grünen Hof 88, 8133 Esslingen

Beisitzer: Adriano Gabaglio
Brunnmattstr. 22a, 6010 Kriens

Beisitzer: Patrik Lori
Schumacherweg 44, 8046 Zürich

e-mail Vorstand oder einzeln: Vorstand bzw. <Vorname>.<Name>@nice.ch

Mailbox: Tel. 01 251 20 02
`# call nice`
`nice login: mailbox`

***PowerKey 1/95* erscheint Ende März 1995**
Redaktions- und Anzeigenschluss: 15. März 1995