Oberon News

INSTITUTE FOR COMPUTER SYSTEMS   THE OBERON USER GROUP

Number 3, December 1994

Contents

Editorial

JOHANNES L. MARAIS

Everything that happens three times becomes tradition and will repeat itself again. Consequently this third issue of the newsletter predicts a bright Oberon future — and if we look at the number of contributions in this issue in contrast to the previous newsletters, an ever growing one too. We would like to thank all the authors who took time to write a contribution.

I would like to point out two "must read" articles in this issue (with no offense to the other authors intended). The first is a report on *Joint Modular Languages* conference held in Germany by Peter Schulthess. If you are interested in Oberon, the conference proceedings are definitely something to have. The other article that promises to revolutionize the Oberon world is titled *The Oberon Module Interchange* by Michael Franz. Happy reading!

The Joint Modular Languages Conference

PETER SCHULTHESS, Department of Distributed Systems, Ulm.

From the 28th to the 30th September the Joint Modular Languages Conference 1994 was organized at the University of Ulm, Germany. The tradition of the conference goes back to earlier conferences on Modula-2 in 1989, 90, 91, 92. The traditional theme of the conference was extended to provide a comprehensive perspective of the current programming language scene. The scope of the selected papers included contributions on the use of Oberon, Modula-2, Modula-3, C++, Ada, Eiffel, Beta, and other languages. However, following the tradition and the subtitle of the conference ("Modula-2, Oberon and Friends") many of the contributions centered on the Oberon Language. Commercial products of relevance to the conference topic were presented at a product exhibition during the event. Compilers and programming environments for Unix, MS-DOS, Macintosh and DEC-Stations attracted considerable interest by the visitors. Tutorials preceding the conference offered the opportunity to the participants to familiarize themselves with C++ or Oberon. Tutorials and conference were attended by approximately 170 participants.

The keynote presentation entitled "A Brief History of Modula and Lilith" was given by Prof. Niklaus Wirth. The gradual progress of modular languages and the concurrent design of the Lilith computer was described. Object-oriented paradigms were later added to the original module concepts and led to the design of the Oberon Language and Oberon Systems. Professor Wirth's fascinating tour through the history of modular languages and into the imminent future of object-oriented languages has indeed set the stage for a large number of interesting presentations during the conference.

Important topics of the conference sessions were Object-Oriented Development Tools and Techniques, Language Design & Implementation Projects, Realtime Programming and Large Software Systems. Particular attention was paid to different concepts of inheritance. The question of single versus multiple inheritance was extensively discussed and several innovative schemes for restricted and partial inheritance were put forward. Attention appears to have shifted from the Modula-2 language to Oberon and a large number of implementations on different hardware and operating system platforms were reported. Successful usage of Modula-2 and Oberon in real-time programming projects was reported but it is highly desirable that future conferences should solicit even more actively contributions from "real programming language users".

A significant group of programmers has adopted the language as a tool to write typesafe and efficient application programs within existing operating systems, but they refuse to work with the Oberon system interface. Typically, however, it was recognized that the Gadgets system now provides an alternative user interface to the earlier rather frugal Oberon user interface. Gadgets was demonstrated to provide unrivalled flexibility in extensible graphical user interfaces. In addition to Oberon implementations on existing Unix, MS-DOS, VMS, or Macintosh operating systems, native Oberon systems are emerging which are directly based on a relevant hardware platform. Experimental distributed Oberon systems might soon provide a mode of lean distributed processing which will fit seamlessly into existing network environments.

Two panel discussions were held — the first one on Modula-2 standardisation and the second on standardisation of Oberon. The history of Modula-2 standardisation was reported and lessons were learned how not to standardize a language. The activity of standardisation must be kept separate from the actual design process and members of a standards committee should refrain from redesigning and extending the language. The issue whether an ISO standard for Oberon would be needed was not resolved. Oberon documentation from ETH was recommended as a de facto standards document.

The JMLC 94 conference has clearly demonstrated that language design is still an active area of research and it is expected that the next Joint Modular Languages Conference will again offer reports on scientific progress and case studies on object-oriented and modular language usage in business and industrial applications. This conference is likely to be organized in spring 1996 at the University of Linz in Austria. The proceedings of the JMLC 94 conference can be ordered from Universitaetsverlag Ulm GmbH, P.O.Box 4204, D-89032 Ulm for DM 90 plus mailing.

Oberon Day '94

Erich Oswald, The Oberon User Group.

More than two hundred participants attended the Oberon Day '94 on 14th September in the Auditorium Maximum at ETH Zurich. After the big success last year, the Oberon User Group in cooperation with the Institute for Computer Systems had decided to organize a second edition of the event.

This year's theme was titled "New Ways in Computer Science Education" and was specifically addressed to teachers of computer science. The organizers were very pleased with the high number of attendees which suggested the theme had appealed to a lot of other people as well.

Talk topics included introductions to Oberon programming and Oberon concepts, case studies of Oberon usage for teaching and Oberon product announcements. Among the speakers were several persons well known to the Oberon community like Prof. J. Gutknecht, Dr. M. Reiser, and Dr. J. Templ.

The Oberon User Group is planning to organise further events in the future which will be announced in the Oberon News.

The Oberon Module Interchange (OMI)

Michael Franz, Institute for Computer Systems.

"That's a program for the Macintosh, it will not run on my PC" — It has been fifteen years since the arrival of the personal computer, and by now even technical laymen understand that there are different kinds of computer, which require different kinds of software. Although this diversity adds huge costs to software development and distribution, it is generally accepted as an unalterable fact of life.

The OMI project attempts nothing less than to change this long-established view. It introduces the concept of a "portable object file" that can be used on more than one kind of target machine. The mechanism behind OMI is completely transparent to the user; there are no converter programs to be run or installation procedures to be executed before a portable object file can be used on another machine participating in OMI. From the user's viewpoint, the only difference between "portable" object files and "regular" ones is that the former are usable on a wider range of machines than the latter.

Well Then, How Does It Work? It should of course be obvious that "portable object files" cannot contain "real" object code. Instead, they contain an *abstract program description* using a clever encoding that later permits the fast generation of appropriate native code for the eventual target machine, which in OMI happens during *loading*. Since OMI is integrated into the Oberon system, in which modules can be loaded dynamically at any time during a computing session, this means that OMI can incrementally compile parts of a software system while other parts are already running. For example, when the user (or a program) invokes a command from a "portable" module that has not yet been loaded, OMI kicks in, compiles the module on-the-fly, executes the newly created body of the module, and returns to the command handler that will then jump to the new command.

On-the-fly code generation at load time would be useless if it were too slow, thereby inconveniencing interactive users, or if it produced bad code. Luckily, it seems that both of these problems have been solved in OMI. In its first implementation, OMI is able to compile-and-load "portable object files" in about the same time that would be required for reading equivalent "regular" object files from disk. The reason for this is that "portable object files" are much smaller than their native counterparts, while program loading time is dominated by I/O overhead. Since processor performance is rising more quickly than storage speeds, it is even highly probable that on-the-fly code generation will outperform loading of "traditional" object files in the long run.

Sounds Good In Theory. What About Practice? OMI is an ongoing research project at ETH's Institute for Computer Systems, building on the dissertation work of the author. At present, OMI is available only for MC680x0-based Macintoshes, but several additional platforms will be supported in the near future.

There are several reasons for releasing OMI already at this time, even before there are further implementations that would make it more useful. First of all, it is hoped that the existing implementation will convince even hard-line sceptics that the ideas behind OMI are sound. Secondly, OMI should provide the outside world with an idea of where future versions of the Oberon system are headed. Thirdly, the existence of OMI is likely to benefit the cause of spreading Oberon, so that publicising it early on might prevent some not-so-easily-reversable decisions in favor of other systems. And finally, it is impossible to debug a complex system without any user feedback. Hopefully, many people will experiment with OMI, comment on it and provide error reports, so that OMI will reach "industrial strength" by the time further implementations are ready.

Most importantly, there will be an implementation of OMI for the Apple PowerMacintosh in early 1995, which will form the core of a new MacOberon release that will run *in native mode* both on MC680x0 processors and on PowerPC processors. Hence, as far as users are concerned, ETH will continue to supply only a single MacOberon system, but in the future, this system will provide native performance on both of Apple's hardware platforms. This may not sound so extraordinary, as there are already some commercial applications for the Apple Macintosh that run in native mode on either kind of Macintosh, but behind the scenes, the OMI approach is truly revolutionary.

Up until now, the problem of allowing programs to run on more than one kind of machine has been solved by what is called *fat binaries*. This means that there are actually several executable versions of a program within a single "application program file", one for each type of target processor, and the system chooses at startup which part of the file is used. Of course, using "fat binaries" also means that programs grow to several times their original size. For example, applications that run on both kinds of Macintosh are usually even more than twice as big as their identical MC680x0-only counterparts, owing to the fact that the PowerPC has a lower code density than the MC680x0. Using OMI technology, the future MacOberon will be able to provide *slim binaries*. Readers who download the present release of OMI from our server and study it will find that OMI's "portable object files" are actually quite a lot smaller than the corresponding MC680x0 object files. Nevertheless, they are capable of replacing the native object files for both kinds of Macintosh systems simultaneously. Since most of the Oberon system can be represented as portable code, the future MacOberon will not grow larger than the present one, although Oberon's core modules and OMI itself will need to be present in "fat binary" format. Eventually, OMI-interchangeability will be extended to further Oberon implementations. In fact, some projects are already under way. Programmers will then be able to release their modules for use by the whole

OMI community, but without having to publish the source code. This will bring us one step closer to the old dream of "software components" that might yet revolutionize the discipline of software engineering.

To Probe Further OMI is now an optional part of ETH's MacOberon distribution. Optional means that, for the time being, modules are provided both in native and in OMI formats. Users on sufficiently powerful (i.e., MC68040) machines should not notice any difference when using the OMI-encoded modules, except for much smaller object files. On older Macintoshes, the performance balance between CPU and I/O is less to the advantage of on-the-fly code generation, and users of OMI will experience longer loading times (but no change in performance).

The technical details behind what has now become OMI are described in the author's doctoral dissertation. It is available via World-Wide Web from `http://www.inf.ethz.ch-/department/publications/diss.html`.

The Hybrid Project

Pieter Muller, Johan de Villiers, de Villiers de Wet, Jaco Geldenhuys, University of Stellenbosch, South Africa.

The Hybrid group at the Computer Science department of the University of Stellenbosch is primarily interested in reactive systems — protocols, operating systems and dedicated control systems. We have developed a kernel to support distributed client-server applications.

The Hybrid kernel is small, stable, efficient and flexible. It supports multiple virtual machines, which are protected address spaces containing several light weight processes. A virtual machine supports the non-privileged instructions of the underlying machine (Intel 386), plus three additional instructions for interprocess communication. These instructions are implemented as traps to the kernel and provide synchronous rendez-vous message passing. The kernel runs on a bare 386 in 32-bit protected mode. Messages to VMs on remote machines are transparently delivered by the kernel via Ethernet. This facilitates the construction of flexible distributed systems.

The kernel is in everyday use in a number of factories where it supports a production management system. The system consists of a number of microprocessor-based computing nodes interconnected via Ethernet and running the Hybrid kernel. It communicates to industry standard platforms such as Novell, Unix and Windows via a TCP/IP server running in a VM.

An experimental distributed computing environment based on the kernel has been developed. This system supports teams of virtual machines working together to solve a single problem. An interactive factory scheduling system is under development to enable factory operators to manipulate the order in which jobs are selected to optimize throughput or other important parameters.

The kernel consists of about 16,000 lines of Modula-2 code (including the Ethernet drivers, IP and VMTP protocols). A further 170,000 lines of device drivers, servers and application programs have been developed. All the code was cross-developed on a Unix system using the MCS compiler. The system is maintained by graduate students and a staff member, who is also responsible for the factory applications.

Oberon on the Hybrid Kernel During 1993, Oberon System 3 was ported to run in a VM on the Hybrid kernel. Hybrid Oberon can be placed in full control by running it as a single VM on the kernel, or it can co-exist with the existing Hybrid window server. The Gadgets system and several applications from ETH were also recompiled to run on Hybrid Oberon.

The facilities provided by Hybrid make it possible to do background processing in a separate VM (possibly on another machine), thereby achieving true multitasking. The Oberon System was extended to support the transparent invocation of remote services.

At the implementation level, multiple light weight processes were used to marry the polling loop of Oberon with the synchronous message passing supported by Hybrid. In order to improve efficiency in a multitasking system, the Oberon loop was altered to block when it is idle.

The starting point of the porting exercise was the DOS Oberon System, which meant that the compiler could be used unchanged. However, the inner core of DOS Oberon was restructured by removing the Loader module and statically linking the Kernel, Modules, FileDir and Files modules. A full-featured static linker was developed for this purpose.

We have experimented with several file systems for Oberon. Our first file system uses a Unix machine to store the files. The Files module forwards requests to a Unix process via the TCP/IP server. This file system allows us to use our existing diskless PCs and also allows file sharing with the Unix system. Unfortunately it is somewhat inefficient. Our most efficient file system is the one from Project Oberon, with some caching added. However, this file system can not be used on diskless

PCs. A central network file system based on the Project Oberon code is under development as a student project.

Hybrid: Switching to Oberon The success of the Hybrid Oberon port prompted us to switch to the Oberon language for all our programming. During 1994 the Hybrid kernel was re-implemented in the Oberon language. The basic design has stayed the same and the new kernel (now called Gneiss) can execute VMs written for the old kernel, without recompilation.

Implementation of the Gneiss kernel is sufficiently advanced that it supports the Hybrid Oberon system using the Project Oberon file system on a local hard disk. The new kernel is smaller and faster than the Modula-2 based kernel, even though the compilers generate code of comparable efficiency. This can be attributed to several factors. The new kernel was implemented by a single person over a period of a few months, whereas several people have worked on the old kernel over a period of four years. The experience gained in the implementation of the old kernel helped to make the new one more efficient. New techniques, e.g. continuations, were used in the implementation of the new kernel.

During a visit to ETH in October the Gneiss kernel author used code from the kernel and the Hybrid Oberon system to create a prototype of an Oberon system running natively on a PC. Work is continuing to develop device drivers to be used for Gneiss and native Oberon.

Using Oberon for Embedded Systems The current academic focus of the Hybrid group is the development of reliable embedded system software. Testing is not a sufficiently powerful method to develop highly reliable systems. For such systems it is necessary to build a validation model of the design that can be proved to conform to the intentions of the designer.

We have developed a validation tool — a model checker — that can be used to detect errors in system designs. A modelling language was developed to allow high-level models of reactive systems to be expressed concisely. The model checker checks the system model against system specifications written in a formal logic syntax. A model stepper tool is used to gain insight into errors found by the model checker. A code generator is being developed to translate a correct model into an executable code skeleton which captures the control-flow of the application. These tools form a verification workbench running on Oberon.

Although the model checker allows one to generate correct control-flow skeletons, the Oberon code that is hand-written to flesh out the skeletons may also contain errors. In embedded systems, errors may also be caused by the peripheral hardware. To find these kinds of errors, there is no replacement for testing an embedded system in-place.

For this purpose, a remote debugger for embedded applications is under development. The first prototype supports post-mortem analysis, as well as interactive debugging via breakpoints and stepping. Oberon modules are linked into an executable image which is loaded into a VM on any Hybrid machine. The debugger communicates with the kernel on the target machine to monitor and control the debugged application.

Source and assembly level debugging is supported. Code breakpoints can be set and machine instructions can be stepped one at a time. A small modification was made to the Oberon compiler to generate a mapping between Oberon statements and machine addresses.

The user can view the stack (active procedures and local variables), global variables and registers at any stage. A useful feature of the debugger is that pointers can be dereferenced. Symbolic dereferencing cannot be done, but a hexadecimal dump of the memory is given, and if it is a record containing pointers, they are marked and can in turn be followed.

Currently work is being done to support the same debugging interface over a serial link, for debugging stand-alone embedded applications.

E-mail: `hybrid@cs.sun.ac.za`
Finger: `@hybrid.sun.ac.za`
Ftp: `ftp.sun.ac.za:pub/US/hybrid`

PC-Oberon: A Progress Report

F. Arickx, J. Broeckhove, T. Van den Eede, L. Vinck, Onderzoeksgroep Toegepaste Informatica, Universiteit Antwerpen (RUCA), Belgium.

As introduced at the JMLC conference, held last September at the University of Ulm (Germany), PC-Oberon represents a native port of the Oberon operating system to the PC compatible hardware platform, featuring i80386 processors or above. PC-Oberon provides a full 32-bit version of the Oberon OS, based on a custom 32-bit kernel, running in i80386 protected mode.

The whole porting project consists of several subtasks: (1) implementation of the (real mode) master bootrecord code, including management of the hard disk partitioning capabilities; (2) implementation of logical (partition based) bootrecord code, preparing the processor to run in protected mode, switching the processor in this protected mode, loading the 32-bit kernel and transferring control to the Oberon boot code; (3) implementation of a 32-bit reentrant kernel, providing a software interface to the hardware components; (4) implementation of 32-bit Oberon boot code transferring control to the proper Oberon operating system code.

The bootrecord codes, master as well as logical, were written, compiled and linked on a DOS system, using Borland C and Turbo assembler. The 32-bit kernel contains a functional interface for access to floppy and IDE hard disk, keyboard, serial and parallel ports, sound system, memory management, real-time clock, etc. This 32-bit code is written in C and assembler, also using Borland's 32-bit code generating compilers and linker. The kernel is functional and was low-level tested by specific routines for each hardware component; the test programs are temporarily included in the kernel and can be invoked at boot time. High-level testing occurs (will occur) from the Oberon system.

The Oberon interface to the kernel consists of a single software interrupt. One of the parameters is always, as we call it, the invoke number, identifying the desired kernel function. All required parameters for the kernel function, including the invoke number, are pushed on the stack before issuing the software interrupt. This allows for reentrancy of the kernel. For efficiency reasons, the kernel calls are coded using inline assembly, and are concentrated in a single Oberon module Invoke.Mod.

The Oberon system used as a starting point in this project is DOS-Oberon System 3 Version 1.5. This choice was mainly made because DOS-Oberon is based on a 32-bit i80386 code generating Oberon compiler. All real mode system calls to BIOS and DOS were removed from DOS-Oberon and replaced, where possible, by the 32-bit kernel calls. The original Ceres oriented filesystem modules were used to replace the DOS file interface.

The current status of the system is that PC-Oberon correctly accomplishes its boot sequence, and we are currently eliminating bugs from the kernel. We expect to have a fully operational version of PC-Oberon in the near future. This will however not be the end of this project. The current kernel will be extended to provide additional hardware support, in particular network functionality. Support within the kernel, for additional operating system features such as e.g. "active objects" will be considered.

Progress of the project may be monitored at any time through the World Wide Web at URL `http://www.ruca.ua.ac.be/Memex/Oberon`.

Dynamic Online Documents in Oberon

Ralph Sommerer, Institute for Computer Systems.

Dynamic online documents provide a unified abstract model for interactive information services. Their most important properties compared to "ordinary" documents are their non-locality (online documents may be distributed over several locations) and the lack of a static global state (online documents or parts of it may be computed at the time of their access i.e. "on the fly"). Two variants of such dynamic online documents have recently been integrated into the Oberon system.

TeleNews is an online document that presents itself as an electronic newspaper. Its content is generated, maintained and updated by a *Teletext* server (Teletext is a page-oriented information service that is broadcast together with the television video signal). TeleNews allows clients to interactively access and obtain news articles from the Teletext service in hypertext form. An *electronic TV guide* as part of the electronic newspaper allows excerpts of a television program list (program *projections*) to be constructed depending on, for example, a range of broadcast time or the type of the program. The last page of the newsletter shows a display snapshot of the TeleNews panel.

World-Wide Web (WWW) is a network information service that is structured as a global hypertext document whose pages contain reference links that allow to switch context to logically related items (further hypertext documents, but also images, video sequences and sound patterns). These items may be physically located at very different locations in the world. The definition of the World-Wide Web consists of a network protocol (HTTP, hypertext transfer protocol) and a markup notation for hypertext documents (HTML, hypertext markup language). World-Wide Web can be viewed as an online document whose parts are accessed via network. Its integration into the Oberon system bases on an almost complete implementation of the current HTML definition, including, since recently, also fill-out forms. On the last page of the newsletter you see a snapshot of the WWW panel.

Although both services have different semantics and accessing schemes, they are integrated into the Oberon system based on a unified user interface model which centralizes all service specific aspects within a single concept called *active hypertext link* (or *active link* for short).

News around Oberon System 3 and Gadgets

Johannes L. Marais, Institute for Computer Systems.

Oberon System 3 is an important research project at the Institute for Computer Systems to improve the Oberon system and its use. The system has turned out to be a successful platform for experimentation with new ideas in system design and user interfaces. It has spawned a number of research projects at the Institute for Computer Systems and other institutes at ETH.

Oberon System 3 and Gadgets have experienced a remarkable dynamic development. Positive developments are continually being incorporated into the system so that it is consistently improving in quality and functionality. Currently we are entering another consolidation phase based on the experiences we made in the last two years. This is mainly to rectify two things. First, we found that some configurability we built in from the start was seldomly used, and can now be replaced with new and unified concepts that we developed later on. Secondly, from experience gained from projects outside the institute, some gadgets, notably the panels, have been redesigned for increased extensibility. Perhaps the most important, the Oberon System 3 documentation project with electronic books is gaining momentum. To allay the fears of those thinking we are changing everything again, I can assure users that we are very concerned about compatibility.

In addition to preparing the release 1.6 of Oberon System 3, we are also looking towards the future — we hope to deliver these ideas in future versions. The two research directions involve connecting Oberon to the world of electronic services, and to extend the Oberon System with concurrency in the form of "active objects". The first of which is progressing well, as you will see on the display snapshot at the end of the newsletter. If all goes well you may expect release 1.6 of Oberon System 3 around the beginning of next year.

TCP/IP for MacOberon and PowerMac Oberon

Daniel Scherer, TIK, ETH Zurich.

Implementations of TCP/IP are now available both for MacOberon and for PowerMac Oberon; they also work with both Oberon V4 and Oberon System 3. TCP/IP stands for Transmission Control Protocol/Internet Protocol and is a widely used standard for connecting computers over any distance and running many different operating systems. Therefore it is now possible to do worldwide communication from a Macintosh using an Oberon System.

Our implementations of TCP map the Oberon TCP procedures to calls of Apple's MacTCP driver. While at first sight this might seem trivial, it was rather tricky as MacTCP does not provide all services in the way required by the Oberon definition and also due to large numbers of parameters in MacTCP calls which required the consultation of the original paper defining TCP. Particular care has also been taken to ensure that TCP does not block an application if e.g. a communication partner does not respond. Therefore, most MacTCP calls are executed asynchronously and resources used by a MacTCP call are not released until its asynchronous completion. Furthermore, our implementations also provide finalization of connections (as specified by the definition) which signifies that TCP connections no longer used are automatically closed. A possible scenario in a TCP application is that a user closes a window displaying a TCP connection without first disconnecting. If the user thereby loses all references to that connection, TCP automatically closes it.

All the asynchronous operations are hidden by the simple synchronous interface. It provides ways to establish connections for both clients and servers including translation of host names to IP addresses as well as synchronous read and write procedures for various data types. These services may be used by application modules to implement specific communication protocols, e.g. for file transfer based on TCP. A TCP application normally also contains some asynchronous parts, e.g. by using an Oberon task to poll for the availability of data before using a blocking read operation.

At TIK, we plan to use TCP for communication in a distributed environment to be implemented on Oberon Systems running on (Power-) Macintosh computers. So far, we have implemented some basic application modules using TCP such as file transfer and message exchange among users. Our TCP implementations have been used successfully at TIK for several months already and currently contain no known bugs. They are available through anonymous ftp from `rudolf.ethz.ch`, directory `/pub/Oberon`, and feedback to `scherer@tik.ee.ethz.ch` is appreciated.

DART-Oberon

Libero Nigro, DEIS, Universita' della Calabria, Italy and Brian Kirk, Robinson Associates, UK.

DART — Distributed Architecture for Real Time — represents an ongoing project aimed at supporting the development of Real Time systems using Oberon-2 or C++ as the implementation language.

DART is centred on the concept of light-weight operating software: its mechanisms are not extensions of built-in OS mechanisms, rather they refer to active objects, i.e. instances of abstract data types. Active objects are modeled as finite state machines which communicate one with another by asynchronous message passing. Messages are transparently captured and managed by a scheduler object which imposes a dispatching policy which can be tuned to the application needs. In an object, message reception is implicit. The arrival of a message triggers a state transition and then the execution of an atomic action. Action execution extends the thread of control of the scheduler. The runtime model is non-preemptive. Action granularity can be fine-grained in order to guarantee real time constraints to be met. An overall system is organised as a collection of subsystems. Each subsystem corresponds to an application domain, and thus to a particular set of aspects of the system. Objects belonging to different subsystems are allowed to communicate using a system-wide non-blocking send, which relies on a heterogeneous message format. As a matter of simplification, a subsystem can admit a special object named a coordinator which is the target of inter-subsystem communications and delegates sub-tasks to hidden objects of the subsystem. DART programming in the small is conveniently supported by Oberon-2 or C++. The project addresses interoperability: a major goal is supporting mixed platforms and implementations. Subsystems programmed in C++ can interact with Oberon-2 subsystems and vice versa. A prototype implementation of DART has been achieved on a PC network under Novell provision of AT&T TLI (Transport Layer Interface). DOS and Windows platforms can be used in combination. User interface issues can be dealt with on a Windows platform where the Robinson's Oberon-2/386 compiler and POW! environment are used. On a DOS platform the Extacy Oberon package is used. More information on DART can be found in JMLC '94 proceedings. For more details contact Robinson Associates, Painswick, UK, email `robinsons@cix.compulink.co.uk` or L Nigro, DEIS, Universita' della Calabria, Italy, email `nigro@ccusc1.unical.it`

Logic Magicians' Oberon

Taylor Hutt, Logic Magician, USA.

When compared to other languages and operating systems on a PC, using Oberon is undoubtedly a pleasurable experience for you. Yet, despite the advantages which Oberon brings into the realm of programming, you still want more. You secretly desire a single system which runs under DOS, Windows, & OS/2 using a 2 or 3 button mouse and places no special constraints on your system configuration. Just in case someone happens to be listening to your secret fantasies, you quickly add that you want to have an Oberon-2 compiler, source code for most of the system and Postscript documentation for undocumented parts of Oberon. FREE.

Perhaps someone has been listening to your thoughts, because the Logic Magicians' Oberon System V2, complete with garbage collection, is now available as freeware. Sporting an Oberon-2 compiler which allows 96Kb of code and 128Kb of constants/data per module, this system relies on the industry standard DPMI (DOS Protected Mode Interface) specification to provide a 32bit, protected mode, flat memory model environment. Special care has been taken during the porting of this system to limit the impact on your system configuration; to that extent, we have been extremely successful: there are no special requirements for CONFIG.SYS or AUTOEXEC.BAT and both 2 and 3 button Microsoft compatible mice are supported. A standard VGA is required for 640x480x16, but an ET4000 chipset is supported at 1024x768x256 (more display drivers are planned).

Not surprisingly, several advantages are realized because a DPMI server has been used as the base for the DOS extender. First of all, the system is portable to any DPMI v0.90+ server. Secondly, the amount of heap space is only limited by the DPMI server — the minimum is 2Mb and the maximum is 32Mb. Thirdly, and most importantly, differences in low-level hardware are masked by the DPMI server, thus allowing greater compatibility with various machines.

In an effort to increase the usage and understanding of Oberon, virtually all the source code to the operating system is available for this package. Further, Postscript documentation has been made available for many previously undocumented aspects of the V2 Oberon system, plus a complete description on how to write display drivers. Standard printing of Oberon source and documents can be achieved if a Postscript or HP laser printer is connected to the parallel port. An auto-import feature has been designed to make it easy to include new software into an installed Oberon system. It works transparently and does not require intervention on the part of the user.

You may get this Oberon implementation from `ftp.clark.net:/pub/thutt/distrib/V2`. See the `00index` file for instructions on which parts to download. If you do not have access to ftp, you may get the most recent copy on disk by sending a check or money order for US $3 to

Taylor Hutt
3428 Moultree Place
Baltimore, MD 21236-3110 USA

Integrating Multimedia on the workstation Ceres-2

PETER RYSER, Institute for Computer Systems.

The workstation Ceres was designed and constructed in 1985. Its simple and efficient hardware interface gives a wide range of opportunities for hardware extensions. In 1993 we built an Ethernet card that allows for direct access to the services of the Internet, and in the spring of 1994, in his diploma thesis, a student built an audio- and video board (AVB). The AVB allows to capture video from a video source such as a camera or a VCR. The quality of the captured video data ranges from 8 bit grey scale to 24 bit RGB. The audio channel samples and plays stereo data up to 48k samples at 16 bits per sample. The acquisition of audio and video data is interrupt driven and is handled completely in the background. Watching TV at a resolution of 320x400 pixel and a rate of 25 frames/s (the PAL frame rate) on the monitor of the Ceres workstation was a first step of integrating multimedia into Oberon. A more sophisticated solution uses a client/server approach. The server, a Ceres workstation equipped with an AVB and an Ethernet card, broadcasts the captured audio and video data streams over the local subnet. The clients listen on a specific port, get the incoming data, display the video data on the screen and simultaneously play the audio data through an audio device. The interface to the user on the client side is made through an extension of Oberon V4's text elements. These multimedia elements (MMElems) have the same properties as all the other text elements. Therefore, one can have one or several TV screens floating in a text. All MMElems are updated simultaneously at the arriving of a new frame. "Video on demand", a catchword not thought of when the Ceres workstation was designed ten years ago, becomes possible on this platform because of simple but powerful hard- and software. For more details contact `ryser@inf.ethz.ch`.

Oberon Tutorial

MICHAEL FRANZ, Institute for Computer Systems.

ETH is pleased to announce another three-day intensive tutorial on the programming language Oberon. The tutorial will take place in Zurich from Wednesday, 5th April 1995 to Friday, 7th April 1995. The tutorial language is German. A detailed description (in German) follows below; for further information please contact:

Madeleine Bernard
Kurswesen
Departement Informatik
ETH Zurich
CH-8092 Zurich
Switzerland

Moderne Programmierparadigmen — vom "Structured Programming" über den objektorientierten Ansatz zum "Extensible Programming". N. Wirth, M. Franz (Kursleitung), M. Brandis, S. Ludwig, J. Supcik

Die Entwicklung der Programmiersprachen ist geprägt von der Entstehung immer mächtigerer Abstraktionsmechanismen. Was einst bescheiden mit der Einführung mnemonischer Codes (für die Operationen) und Variablennamen (für die Operanden) begann, hat in der Zwischenzeit wiederholt vollständig neue Ansätze hervorgebracht, die nicht nur die eigentliche Programmierung, sondern auch die Methodik des Software-Design grundlegend verändert haben.

Mit der Schaffung der Programmiersprachen Pascal, Modula-2 und Oberon durch Professor Niklaus Wirth ist die ETH Zürich an diesem Entwicklungsprozess seit langem massgeblich beteiligt. Jede dieser Sprachen repräsentiert einen Meilenstein in der Entstehungsgeschichte der Programmierparadigmen: Pascal steht für "Structured Programming", Modula-2 für "Modular Programming", und Oberon schliesslich für objektorientiertes und "Extensible Programming", wobei jede dieser Sprachen die Konzepte ihrer jeweiligen Vorgänger natürlich einschliesst.

Der Kurs vermittelt diejenigen Programmierparadigmen, die erst nach der Definition von Pascal vor mehr als 25 Jahren entstanden sind, und die heute in Oberon wiederzufinden sind. Er folgt dabei in drei Schritten der Evolution der Programmiersprachen, beginnend am ersten Kurstag mit den Konzepten der strukturierten und modularen Programmierung. Der zweite Kurstag beschäftigt sich mit den Ideen der objektorientierten Programmierung, während der dritte Tag schliesslich den erweiterbaren Programmsystemen gewidmet ist.

Ein wichtiger Bestandteil des Kurses sind betreute Übungen, in denen die Teilnehmer die Möglichkeit haben, die gelernten Konzepte in kleinen Gruppen am Computer zu erproben. Als Basis für die Übungen wird die Pascal-Nachfolgesprache Oberon verwendet, weshalb eine gewisse Vertrautheit mit Pascal oder einer Pascal-ähnlichen Sprache Voraussetzung für den Besuch dieses Kurses ist.

Power Gadgets Available

Andreas Wuertz, TIK, ETH.

By the time you read this, there should be a beta release of Oberon System 3 for Power Macintosh available on `neptune.inf.ethz.ch`. Look in the directory `/pub/Oberon/System3/POWERMAC`.

Some features are:

- Up to 8 bit colour support on screen and pictures.
- Colour/grayscale printing on all newer QuickDraw and PostScript printer drivers.
- Display3 and Printer3 optimised for Macintosh clipping architecture
- Full Oberon-2 Compiler (same as Power V4, except it allows up to 127 imports)
- Hierarchical file system (same as Power V4)

There are still a few known bugs. The colour model is not very stable yet. Specially when editing pictures with Paint, the background color may unexpectedly change. Although the system is based on PowerOberon 1.0 and MacOberon 4.1xx, it is still in beta state and not tested for stability, so be careful!

Please send questions, comments and bug reports to:

Andreas Wuertz
TIK, ETH Zürich
Gloriastr. 35
CH-8092 Zürich
Switzerland
email `wuertz@tik.ee.ethz.ch`

Action-Oberon

Eric Hedman, Åbo Akademi University, Finland.

The department of computer science at Åbo Akademi University has an active programming methodology research group. One of the groups research topics is *action systems*. An action system is a model for specifying and reasoning about parallel programs, based on Dijkstra's guarded command language. Lately, action systems have been extended with modular constructs in [1]. *Action-Oberon* is an extension of both the Oberon system and the Oberon language that incorporates action systems in Oberon according to these principles.

In an action system the individual actions are atomic, but may be interleaved arbitrarily, or executed in parallel if they do not share any common variables. As such, the Oberon Loop can be considered an action system, and is thus a good starting point for a scheduler of actions. The extended Oberon Loop provides support for interacting with the scheduling of actions. The actions, which by themselves are guarded commands, are introduced at the module level of Action-Oberon, making modules *active* at load-time. Parallel composition of action systems is then modelled by two active modules loaded into memory at the same time. While preserving the original semantics of the Oberon Loop, this approach extends it with the parallel behaviour of action systems.

At the language level only one extension is really needed, namely the concept of guarded commands. Action-Oberon distinguishes between two types of guarded commands, actions and guarded procedures. In the action system model, guarded procedures can be used for modelling most synchronisation and communication mechanisms.

Action-Oberon also includes an interactive environment for monitoring the execution of action systems. One of the goals of Action-Oberon is to provide a specification environment where fairly abstract action systems can be executed and monitored. This has to some extent defined the terms of the design and outweighed optimal performance in the implementation.

Action-Oberon has been implemented in SPARC Oberon and will be released to the public in the (hopefully) near future. The release will be announced in the Usenet newsgroup `comp.lang.oberon`. For more information about the implementation contact `Eric.Hedman@abo.fi` and for information about the Programming Methodology Group at Åbo Akademi and its publications (including [1] as TR A-154, 1994) consult our WWW server at `http://www.abo.fi/~mbutler/pmg/`.

References

[1] R. J. R. Back and K. Sere. From Action Systems to Modular Systems. In Naftalin, Denvir and Bertran, editors, *Proceedings of FME '94*. Springer-Verlag, 873, 1994.

ONTIME

Karl Rege, Institute for Computer Systems.

ONTIME is an acronym for an "**O**bject Orie**n**ted Framework Assisting **T**ime and **I**nformation **M**anagement of an **E**xecutive". It is the realization of an advanced prototype Personal Digital Assistant (PDA) based on Oberon System 3 and Gadgets (kernel) that co-design qualified the author of this article to exploit its features in an optimal way. We believe a PDA should not be merely a scaled down workstation system but a system provided with new qualities in terms of integration and ease of use. Consequently, the user interface follows mainly the direct manipulation user interaction paradigm. Non-modality and simplicity were properties inherited from the original Oberon system. To achieve a maximum level of integration, ONTIME is realized in an object-oriented manner, i.e. the system is composed of advanced and versatile objects. The essence of such a system reveals in these objects cooperative working "hand in hand". Specific functionalities of the ONTIME system include its powerful time management realized both as weekly and monthly diary. Various diary synchronizations are possible such as, for example, between different personal agendas to determine possible meeting times (also over a network). The integrated electronic mail system examines incoming mails and (optionally) inserts event announcements directly into the diary. To perform this task a regular grammar matcher is included. The grammar may be specified in an augmented EBNF notation. This grammar matcher is also applied to decompose arbitrarily formatted mail addresses (according to national uses) into their components to be inserted into a (standard) relational database. For the handling of text messages a textual data base (so called Archives) has been integrated into the system. Applying a full text search algorithm (using PAT-arrays) allows arbitrary queries to be performed in logarithmic time. Due to the homogeneous system architecture these Archives may be used for texts and objects. Furthermore, Archives serve as interfaces for remote access to such objects and hence allow to realize a system of distributed objects (comparable with DSOM). Links to documents may be specified including also external references, for example, to WWW-documents. For the spacial arrangement of the documents on the display a versatile non overlapping display space organization of these documents has been realized, that unifies the Oberon track model with a flexible application specific organization. Finally, there is support for ONTIME on pen based portable computers due to an integrated gesture and handwriting recognition [Xerox Unistrokes by Goldberg & Richardson, INTERCHI 93, Conference on Human Factors in Computer Systems].

Oberon V4 for the PowerMacintosh

H. Mössenböck, Johannes Kepler University, Linz.

The ETH Oberon System V4 has recently been ported to the PowerMacintosh where it runs in native mode on a PowerPC processor. It supports the hierarchical file system with user-definable search paths. Foreign language procedures from shared libraries (DLLs) can be called from within Oberon. Basic toolbox support is available; other toolbox interfaces can be implemented on demand. The compiler supports Oberon-2 and generates native PowerPC code. It compiles about 2000 lines per second on a 66 MHz PowerPC. The distribution contains also new tools and packages, such as a post mortem debugger, a package for building and using graphical user interfaces, a scanner/parser generator, and lots of new text elements. Most of this new software is available in source code. See also the articles about the debugger and the Dialogs package in this newsletter. PowerMac users with a one-button mouse can use a modified version of the TextFrames module which allows pointing, selecting and scrolling with the single mouse button (no modifier keys). This module also works with the familiar Macintosh scroll bars. The system and its documentation can be obtained via anonymous ftp from `oberon.ssw.uni-linz.ac.at`, `/pub/Oberon/PowerMac`. For further questions contact `moessenboeck@ssw.uni-linz.ac.at`.

Post Mortem Debugger for Oberon V4

M. Hof, Johannes Kepler University, Linz.

The original Oberon environment offered only limited means for inspecting run time information and determining the reason of traps. Only variables of basic types such as INTEGER or CHAR could be inspected. There is a new post mortem debugger which allows also the inspection of structured types such as records, arrays, and pointers. It supports views on local and global variables. Pointers can be followed in order to traverse complex data structures on the heap. Run time types are supported as well as open arrays. Additionally, the trap position in the source code can be viewed. The new debugger is available for the PowerMac and the Windows version of Oberon. It can be obtained via anonymous ftp as part of the respective system from `oberon.ssw.uni-linz.ac.at`, `/pub/Oberon/PowerMac` or `/pub/Oberon/Windows`. The debugger sources are included in the PowerMac version. Documentation is also included. For further questions contact `hof@ssw.uni-linz.ac.at`.

Oberon Dialogs: A Graphical User Interface for Oberon V4

Markus Knasmueller, Johannes Kepler University, Linz.

The Oberon System has a compact textual interface. This is convenient for professional programmers, but not always for end users who prefer a

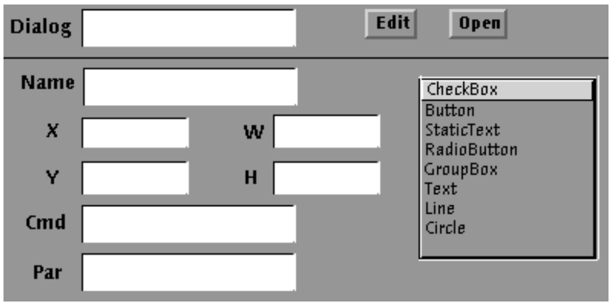

Figure 1: Dialog for creating new dialog panels

graphical interface. Therefore the *Oberon Dialogs* package was implemented at the University of Linz. This package allows a user to create and use dialog viewers with buttons, checkboxes, text fields and other user interface items. Dialogs fit smoothly into the Oberon system and should run under all (Oberon V4) platforms without any changes in the system. Existing tools can be augmented with a graphical user interface without having to be modified. Because of the object-oriented nature of Dialogs, new user interface items and new commands can be added by third party programmers.

A similar package for graphical user interfaces is the Gadgets system implemented for Oberon System 3. While the Gadgets system is more powerful (e.g. nested objects) it is also more complex and does not run under Oberon V4. The virtue of the Dialogs package is that it is extremly light-weight and smoothly fits into Oberon V4.

Working with dialogs is quite simple. There are commands to use, edit and print a dialog. Dialogs can be displayed in two modes: *Dialog.Open* opens a dialog for using it while *Dialog.Edit* opens it for editing. Dialogs can be created using an insert dialog (Figure 1). This dialog contains a list box which shows all items implemented so far. The user can select an item and insert it into a dialog viewer. Items can be moved, resized, deleted and copied with mouse clicks. It is even possible to associate an item with a command, which is called whenever a property of the item changes.

Oberon Dialogs can be used and understood in a few minutes. Therefore we believe that the Dialogs package is a good example for the flexibility of Oberon and object-oriented programming. Try it and give us your comments via e-mail.

Oberon Dialogs (with full source code) can be obtained via ftp (`Oberon.ssw.uni-linz.ac.at`, `/pub/Dialogs`). Extensive documentation is available. For questions or comments contact `knasmueller@ssw.uni-linz.ac.at`.

Oberon/F: Introducing a New Oberon System

CUNO PFISTER, Oberon microsystems.

Oberon microsystems, Inc., Switzerland, announces a new Oberon system called *Oberon/F*. Oberon/F will soon be available in versions for Windows 3.1 and for Mac OS 7. Oberon/F applications can be ported from one platform to the other simply by recompiling them, i.e. their application programming interfaces (APIs) are identical and platform-independent. However, the correct native look-and-feel is provided on each platform. Unlike the ETH Oberon systems, Oberon/F has no proprietary user interface. The design of Oberon/F has been strongly influenced by the experience with earlier Oberon systems like ETH Oberon System 3, V4, and Ethos, nevertheless it is a new design not directly compatible with any of its predecessors. It is fully based on the language Oberon-2. An extensible text editor is part of the standard distribution of Oberon/F, as is a forms editor and an integrated development environment. Documentation is available both in printed form and online. Oberon/F supports a compound document architecture which provides a seamless migration path towards OLE and OpenDoc; these standards will be supported in a later release. An educational version of Oberon/F (not for commercial use) will become freely available electronically, e.g. via anonymous ftp from `hades.ethz.ch` (129.132.71.5) `/pub/Oberon/NonETHSystems`.

The company's address is

Oberon microsystems, Inc.
Solothurnerstr. 45
CH-4053 Basle
Switzerland
phone (+41 61) 361 38 58
fax (+41 61) 361 38 46
e-mail `Oberon@applelink.apple.com`

Call for Papers

Oberon Track at the First Joint Annual GI-SI Conference 1995, Zurich, Switzerland, 18th-20th September 1995

Main Conference Theme: *Die Herausforderungen eines globalen Informationsverbundes für die Informatik*

Oberon Track Theme: *Oberon-Betriebssystem der Zukunft für globale Informationsdienste*

Submissions are solicited from present users of Oberon, both in education and in industry, relating their experiences with the language and system. Please send six copies of your paper (not exceeding about 10 pages single-spaced) to the Program Chair before the due date. Submissions are accepted in English or in German and will appear in a proceedings published by Springer-Verlag.

Key Dates:

13th January 1995 Submission Deadline
15th March 1995 Notification
15th May 1995 Camera-Ready Papers Due

Program Chair:

Michael Franz
Institut für Computersysteme
ETH Zurich
CH-8092 Zurich

Program Committee Members:

R. Crelier, Borland, Scotts Valley, USA
M. Franz, ETH Zurich, CH
H. Mössenböck, Universität Linz, A
C. Pfister, Oberon microsystems AG, Basle, CH
C. Szyperski, Queensland U of Technology, Brisbane, AUS
N. Wirth, ETH Zurich, CH

Statistics on a Voyage to Oberon

G. Sawitzki, Universität Heidelberg.

StatLab Heidelberg, the statistical laboratory at the Institut für Angewandte Mathematik, Universität Heidelberg, is developing a statistical data analysis and simulation system based on Oberon. The system provides an extensible base for classical statistics, but is focused on what is known in the trade as "interactive exploratory data analysis". It provides all the usual statistical interactive facilities like brushing in linked windows, selection and identification, or rotating 3d scatterplots. A first prototype has been demonstrated at the conference on computational statistics CompStat, Vienna 1994. The CompStat report is available as a postscript file by ftp or www from `statlab.uni-heidelberg.de`.

As part of this project, a library of statistical algorithms in Oberon is being build up and will be published on the same ftp/www site. If you need special algorithms like distribution functions, quantiles or random number generators now, you can contact me at `gs@statlab.uni--heidelberg.de`.

Recent Publications

The following Oberon related papers appeared in *Advances in Modular Languages*, the proceedings of the JMLC conference in Ulm (ISBN 3-89559-220X)[1]:

[1] The proceedings includes many more papers not directly related to Oberon.

Process Visualisation with Oberon System 3 and Gadgets — E. Templ, A. Stritzinger, G. Pomberger
Compiler Optimizations Should Pay for Themselves — M. Franz
Building an Optimizing Compiler for Oberon: Implications on Programming Language Design — M.M. Brandis
Post Mortem Debugger for Oberon — M. Hof
Using Oberon to Design a Hierachy of Extensible Device Drivers — P.J. Muller
Object-Oriented Distributed Programming in the Oberon-PVM Environment — E. Bugnion, M. Gitsels, B.A. Sanders
Design of a Distributed Oberon System — S. Traub
A Distributed Real-Time Architecture in Oberon-2 — B. Kirk, L. Nigro
Oberon Perspectives of Evolution — J. Gutknecht
Towards End-User Objects: The Gadgets User Interface System – J.L. Marais
Native Oberon on the PC Compatible (ISA) Platform — F. Arickx, J. Broeckhove, T. Van den Eede, L. Vinck
Alpha AXP/Open VMS (Modula—Oberon)-2 Compiler Project — G. Dotzel
Bringing the Oberon Language to the Macintosh — J. Gesswein, R. Ondrus, O. Schirpf

The following Oberon related papers appeared elsewhere:

Inside Oberon System 3, Johannes L. Marais, Dr. Dobb's Journal October 1994
A comparison of object-oriented programming in four modern languages., Software - Practice and Experience, 24, 11, 1077-1095 (Nov. 94).

Accessing The Internet By E-Mail

The following text is copyrighted by "Doctor Bob" Rankin. Please read the copyright notice in the document refered to by this text.

If your only access to the Internet is via e-mail, you don't have to miss out on all the fun! Maybe you've heard of FTP, Gopher, Archie, Veronica, Finger, Whois, WAIS, World-Wide Web, and Usenet but thought they were out of your reach because your online service does not provide those tools. Not so! And even if you do have full Internet access, using e-mail servers can save you time and money.

This special report will show you how to retrieve files from FTP sites, explore the Internet via Gopher, search for information with Archie, Veronica, or WAIS, tap into the World-Wide Web, and even access Usenet newsgroups using E-MAIL AS YOUR ONLY TOOL.

If you can send a note to an Internet address, you're in the game! This is great news for users of online services where there is partial or no direct Internet access.

I encourage you to read this entire document first and then go back and try out the techniques that are covered. This way, you will gain a broader perspective of the information resources that are available, an introduction to the tools you can work with, and the best methods for finding the information you want.

This document is now available from an automated mail server. To get the latest edition, send e-mail to either address below.

```
To: LISTSERV@ubvm.cc.buffalo.edu (US)
```

Leave Subject blank, and enter only this line in the body of the note:

```
GET INTERNET BY-EMAIL NETTRAIN F=MAIL
```

Or:

```
To: MAILBASE@mailbase.ac.uk (Europe)
```

Leave Subject blank, and enter only this line in the body of the note:

```
send lis-iis e-access-inet.txt
```

You can also get the file by anonymous FTP at either of these sites:

```
At: ubvm.cc.buffalo.edu
cd NETTRAIN
get INTERNET BY-EMAIL

Or: mailbase.ac.uk
cd pub/lists/lis-iis/files/
get e-access-inet.txt
```

Oberon Source Code

Stefan Ludwig, Institute for Computer Systems.

If you login on our ftp-server `ftp.inf.ethz.ch` (also known as `neptune.inf.ethz.ch`, 129.132.101.33), you will find many interesting programs in the `/pub/Oberon/Tools` directory. Among them are an extended text-editor for the Oberon V4 system, called *XE*, which features convenient enhancements for programmers. For instance, text-selection works incrementally: Subsequent clicks on the same location select a character, a word, a name (e.g. *a.b.c*), and a whole line. Also, you can middle-click at a selected text, interclicking the left key in the target viewer, and thereby moving the text stretch to another location (drag and drop). A flexible compile command lets you compile programs containing folded texts, and you can append a command and options overriding standard options to the compile command (e.g. *Analyzer.Analyze* instead of *Compiler.Compile*). Error elements showing possible errors are inserted into the text automatically. By middle-right clicking at a text stretch, the command *Doc.Open* is called with the clicked-at text as an argument. This way, you can open any kind of document according to its extension with an installed command (e.g. for **.Text XE.Open* may be called, for **.Graph Draw.Open*, etc.). *Doc* is also in that directory among other programs, such as *AsciiCoder*, *CaptionEdit*, *Find*, *Folds*, *LineSorter*, and *Macro*. If you are using these programs, please write your comments to the authors, which are listed in the respective tool texts. Enjoy!

Programmieren in Oberon — Das neue Pascal

The German edition of *Programming in Oberon — Steps beyond Pascal and Modula-2* by N. Wirth and M. Reiser is available from Addison-Wesley as *Programmieren in Oberon — Das neue Pascal.* The German edition has been prepared by Josef Templ, a native German speaker with 5 years of experience in teaching and using Oberon at ETH Zurich. ISBN 3-89319-657-9, hard cover, 330 pages, DM 69,90 incl. tax. Discounts up to 35% have been reported for ordering more than 200 pieces.

In addition to the Oberon book, Addison-Wesley offers an Oberon CD-ROM, which contains all ETH-Oberon implementations, a number of example programs in source form, demo versions of commercial Oberon products and — due to the lack of German documentation — a German introduction into using the Oberon system. Since all ETH-Oberon distributions contain installation guidelines and online documentation in English, the CD-ROM will also serve the needs of the English speaking reader. ISBN 3-89319-791-5, DM 69,90 incl. tax.

Miscellaneous Oberon Software

Johannes L. Marais, Institute for Computer Systems.

The *Compress* package[2] from Emil Zeller allows you to compress Oberon files into a single archive. The resulting archive is useful for the distribution of Oberon files as the longer Oberon filenames are stored inside of the archive. The Compress source code is portable between V4 and System

[2] This package is not compatible with the UNIX utility with the same name.

3 on all Oberon ports and can be obtained from `hades.ethz.ch` in the `/pub/Oberon/Sources` directory.

Alan Freed has released a basic Oberon Math library for REALs and LONGREALs (sqrt, sin, cos, cot etc). It is based on the existing Math modules of Oberon with an interface providing error messages for exceptional conditions. It can be downloaded as the files `Maths.Mod` and `MathsL.Mod` from the ftp server `hades.ethz.ch`. Alan can be contacted at `al@sarah.lerc.nasa.gov`.

Literature

Several books have been written about the Oberon System and Language. We recommend these books for serious Oberon users. However, if you want to try out Oberon before buying a book, most Oberon releases have enough online information to get a new user started with Oberon.

N. Wirth and M. Reiser: *Programming in Oberon — Steps beyond Pascal and Modula.* Addison Wesley, 1992, ISBN 0-201-56543-9. Tutorial for the Oberon programming language and concise language reference.

N. Wirth and M. Reiser: *Programmieren in Oberon — Das neue Pascal.* Addison Wesley, 1994, ISBN 3-89319-657-9. The German translation of *Programming in Oberon.*

M. Reiser: *The Oberon System: User Guide and Programmer's Manual.* Addison Wesley, 1991, ISBN 0-201-54422-9. User manual for the programming environment and reference for the standard module library.

N. Wirth and J. Gutknecht: *Project Oberon.* The Design of an Operating System and Compiler. Addison Wesley, 1992, ISBN 0-201-54428-8. Program listings with explanation for the whole system, including the compiler for NS32000.

H. Mössenböck: *Object-Oriented Programming in Oberon-2.* Springer, 1993, ISBN 3-540-56411-X. Principles and applications of object-oriented programming with examples in the language Oberon-2.

How to get Oberon

Oberon is available free of charge from our Internet FTP server `ftp.inf.ethz.ch` in the `/pub/Oberon` directory. The sub-directory `System3` contains the Oberon System 3 versions. Oberon V4 is available for Amiga, DECStation, MS-DOS, Microsoft Windows and Windows NT, HP 700, Mac II, IBM RS6000, SPARC and Silicon Graphics machines. Oberon System 3 Version 1.5 is currently only available for MS-DOS, Linux, and SPARC computers. If you do not have access to Internet, you can order diskettes from the address below. We charge a fee of Sfr 50.00 to cover our costs. We accept payment via Eurocard/Mastercard or VISA. To order by credit card, specify your credit card number, expiration date, and your name exactly as it appears on the card. If you already purchased an Oberon version from us, we will upgrade you to a newer version for Sfr 20. The upgrading policy applies only to versions of the same architecture; this means you cannot upgrade from an older Oberon V4 for Windows version to a newer DOS-Oberon version or vice-versa.

Note that the ftp server `hades.ethz.ch` also archives a number of Oberon ports (notably Linux and OS/2) and several example programs and packages. The official archive of the newsletter is `ftp.inf.ethz.ch` in the `/pub/Oberon/Newsletter` directory. If you don't have ftp access we can add you to our address list.

Institut für Computersysteme
ETH Zentrum
CH-8092 Zürich
Switzerland

Telephone +41 (1) 632 73 11
Fax +41(1) 632 12 20
e-mail `oberon@inf.ethz.ch`

The Oberon User Group
Bergstrasse 5
CH-8044 Zürich
Switzerland
e-mail `oberon-user@inf.ethz.ch`

Acknowledgements: Many thanks goes to Stephan Gehring, Jürg Gutknecht, Taylor Hutt, Dominique Lebègue, and Stefan Ludwig for proof-reading the newsletter.

On the backpage of the newsletter you see a desktop snapshot of Oberon System 3 and Gadgets. The left bottom corner shows a panel by Emil Zeller to play music CDs from an attached CD-ROM drive. On the right we have the Oberon System 3 World-Wide Web browser from Ralph Sommerer with inbuilt formula support (Textfield gadgets are used to "register" Oberon at the White-House). In the background we have the *TeleNews* panel generated by the teletext database.