Oberon-A.doc

COLLABORATORS			
	TITLE : Oberon-A.doc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 17, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Oberon-A.doc

1.1 Oberon-A

```
                    $RCSfile: Oberon-A.doc $
Description: Overall documentation for Oberon-A, release 1.0ß

 Created by: fjc (Frank Copeland)
  $Revision: 1.2 $
    $Author: fjc $
      $Date: 1994/05/19 23:31:09 $

Copyright © 1994, Frank Copeland.
```

1.2 What is Oberon?

The following are taken from the FAQ file for the comp.lang.oberon Usenet newsgroup,

1.3 WHAT IS OBERON?

From "The Oberon Guide"

> Oberon is simultaneously the name of a project and of its outcome. The project was started by Niklaus Wirth and [Jrg Gutknecht] late in 1985 with the goal of developing a modern and portable operating system for personal workstations. Its results are an implementation of the system for the Ceres computer and a programming language.
>
> The development of the language Oberon needs perhaps a short justification. It became quite inevitable because the type-system of available languages turned out to be too restrictive to express the desired data model in a natural and safe way.

Easy introductions to both aspects of the Oberon project can be found in back issues of BYTE magazine. The operating system is overviewed in "Oberon: A Glimpse at the Future", volume 18 number 6 (May 1993). The Oberon language is examined in "Oberon", volume 16 number 3 (March 1991). Both articles are by BYTE's European correspondant, Dick Pountain.

1.4 THE PROGRAMMING LANGUAGE OBERON

From "From Modula to Oberon"

> The programming language Oberon is the result of a concentrated effort to increase the power of Modula-2 and simultaneously to reduce its complexity. Several features were eliminated, and a few were added in order to increase the expressive power and flexibility of the language. This paper describes and motivates the changes. The language is defined in a concise report.
>
> Whereas modern languages, such as Modula, support the notion of extensibility in the procedural realm, the notion is less well established in the domain of data types. In particular, Modula does not allow the definition of new data types as extensions of other, programmer-defined types in an adequate manner. An additional feature was called for, thereby giving rise to an extension of Modula.
>
> . . . .
>
> The evolution of a new language that is smaller, yet more powerful than its ancestor is contrary to common practices and trends, but has inestimable advantages. Apart from simpler compilers, it results in a concise defining document, an indispensable prerequisite for any tool that must serve in the construction of sophisticated and reliable systems.

Among the eliminations in the move from Modula-2 to Oberon are variant records, opaque types, enumeration types, subrange types, the basic type CARDINAL, local modules, and Modula's WITH statement. The major addition to Oberon is the concept of type extension (i.e., single inheritance) for records.

1.5 THE PROGRAMMING LANGUAGE OBERON-2

From "Differences between Oberon and Oberon-2"

> Oberon-2 is a true extension of Oberon. . . .
>
> One important goal for Oberon-2 was to make object-oriented programming easier without sacrificing the conceptual simplicity of Oberon. After three years of using Oberon and its experimental offspring Object Oberon we merged our experiences into a single refined version of Oberon.
>
> The new features of Oberon-2 are type-bound procedures [i.e., virtual methods], read-only export of variables and

record fields, open arrays as pointer base types, and a with statement with variants. The for statement is reintroduced after having been eliminated in the step from Modula-2 to Oberon.

1.6 THE 'OBERON FAMILY' OF LANGUAGES

Object Oberon is a now defunct, experimental extension of Oberon featuring "classes", structures somewhere between modules and records. It evolved into Oberon-2.

Seneca was also an experimental extension of Oberon. It focused on numerical programming on vector computer architectures. It evolved into Oberon-V.

Oberon-V is an experimental dialect (but not a superset) of Oberon. It is concerned with issues of numerical computing, array processing, and code verification. Since it was originally aimed at vector architectures in general and the Cray Y-MP in particular, no Oberon-V compiler has yet been implemented for the Oberon System.

1.7 What is Oberon-A?

Oberon-A is an Oberon-2 compiler and associated utilities for the Commodore Amiga personal computer. The Oberon-A compiler translates programs written in Oberon or Oberon-2. It also supports a number of language extensions that assist programming in the Amiga's unique environment. It produces MC68000 machine code directly without an intermediate assembly language stage. The object files it creates are in standard AmigaDOS format.

The Oberon-A Library is a collection of library modules that can be linked with programs created with Oberon-A. It includes a complete interface to the Amiga KickStart v2.04. A number of other useful modules are also part of the Library, including the beginnings of an object-oriented application Framework.

The Oberon-A archive contains a programming environment utility, the compiler, a pre-link utility, a recompilation utility, a full interface to KickStart 2.04 and a number of other library modules and example programs. Full source code is included for all modules and programs where available. The archive also contains other software needed to use Oberon-A, most importantly the BLink linker.

The compiler and utilities at present run only from the CLI. However, included in the package is FPE, a programming environment. This is a fully Intuition-based program that allows you to run the compiler, utilities and other tools by simply clicking on a button.

1.8 Distribution and Copyright

1.9 What do I need to use Oberon-A?

To use Oberon-A requires an Amiga personal computer with at least 1 MB of RAM, running Kickstart version 1.3 or greater. A conscious effort has been made to make sure that the core Oberon-A programs will run under Kickstart 1.3, but they were developed under Kickstart 2.04 and it was not possible to test this. It is likely that at some future date Oberon-A will require Kickstart 2.04 or greater. If this is a problem for you, please contact the author.

It may be possible to run Oberon-A from floppies, but this has not been tested and a hard disk is highly recommended. The contents of the archive when decompressed will take up 2-2.5 MB of disk space. Further disk space will be required to develop programs.

A linker is needed to create executable programs from the object files generated by the compiler. A freely-distributable linker, BLink, is included in the Oberon-A archive. The ALink linker, available from Commodore, should also work (it has not been tested). Other third-party linkers such as PhxLink may also be suitable.

The FPE program requires arp.library and intuisup.library. These are included in the Oberon-A archive.

The documentation files are written in Commodore's AmigaGuide hypertext format. Users of Kickstart 2.1(?) or greater already have the software needed to view these files. A minimal AmigaGuide installation is included in the Oberon-A archive for users of earlier Kickstart versions. The complete AmigaGuide distribution can be found on AmiNet in the text/hyper directory, or in the Fred Fish collection on disk ??.

There is no editor provided with Oberon-A. Many suitable editors are available as freely distributable software. The Memacs editor distributed with the operating system can also be used.

1.10 How do I install Oberon-A?

Oberon-A can be installed temporarily while you evaluate it. This mainly involves assigning logical device names and possibly copying shared libraries to your libs: directory. The Oberon-A library modules must also be compiled the first time Oberon-A is installed. The temporary installation process must be repeated each time you re-boot your Amiga. In order for this to work correctly, the Oberon-A directory *must* be in the root directory of the currently active disk volume. Permanent installation involves modifying your startup sequence.

1.11 Temporarily installing Oberon-A under Kickstart 1.3

[I am unable to test this under AmigaDOS 1.3. If it fails to work, please let me know. fjc]

Run the script in the file Install1.3, either by typing "Execute :Oberon-A/Install/Install1.3" at the CLI prompt or by double-clicking its icon. The script will check for arp.library and intuisup.library in your libs: directory and ask if you wish to install them or replace earlier versions.

1.12 Temporarily installing Oberon-A under Kickstart 2.0+

Run the script in the file Install2.0+, either by typing "Execute :Oberon-A/Install/Install2.0+" at the CLI prompt or by double-clicking its icon. The script works the same as Install1.3 except that it does not copy the libraries; it uses the ADD option of the AmigaDOS 2.0+ Assign command instead.

1.13 Compiling the library modules

To save space, the symbol and object files for the Oberon-A library modules are not included in the distribution. A script file is provided which will compile the modules and place the symbol and object files in the OLIB directory. To run the script, double-click on the CompileLibs icon. This script will take a considerable time to complete, depending on your hardware. You will be given a chance to back out if you want.

1.14 Permanently installing Oberon-A

The installation is mainly for the benefit of the FPE utility. If you decide not to use it, you may still wish to set up the "OBERON-A:" and "OLIB:" assigns described below.

If you use it, FPE must be able to locate it's setup files. When extracted from the archive, these are placed in the directory Oberon-A/S. FPE looks for its setup files in the directory "FPE:S", so the temporary installation assigns "FPE" to the Oberon-A directory. If you keep the same directory organisation, simply place the line

```
"Assign FPE: Oberon-A"
```

in your startup sequence. If you move the Oberon-A/S directory you will need to adjust this accordingly.

The setup files provided for FPE assume the existance of two other assigns, OBERON-A: and OLIB:. OBERON-A: is assigned to the Oberon-A directory and OLIB: is assigned to the Oberon-A/OLIB directory. If you wish to keep this arrangement, add the following two lines to your startup sequence:

```
"Assign OBERON-A: Oberon-A"
"Assign OLIB:     OBERON-A:OLIB"
```

FPE also requires arp.library and intuisup.library. These must be copied to the SYS:libs directory, or to another directory to which LIBS: has been assigned.

1.15 Removing Oberon-A from your system

THIS SPACE INTENTIONALLY LEFT BLANK (just kidding)

If you decide not to permanently install Oberon-A, it is a simple matter to remove all traces of it from your system. First, delete the Oberon-A directory and its contents. Next, if you copied arp.library and/or intuisup.library to your LIBS: directory, delete them. Finally, remove any Assign statements you put into your startup sequence.

Even if you are not impressed by Oberon-A, I would like to hear your comments and assessment of it. See
Bug reports and suggestions
.

1.16 How do I get Oberon-A running

After reading the rest of this document, the best place to start is with the programming environment utility, FPE. Read the chapter Getting started in FPE.doc, then run the program.

Part of the process of setting up FPE for the first time involves telling it which editor you want to use. Before starting FPE, select an editor and install it on your system. Make sure you know how to run it from the CLI so that it will edit a specific file.

1.17 Creating a program with Oberon-A

An Oberon program consists of one or more modules, each of which is compiled seperately. Some modules are "library" modules, which are re-used many times in different programs. Each program has a "main program" module which acts as the entry point to the program. Any module may be used for this: there is no specific language construct to indicate that a module is the main program module.

Each module is contained in a single text file. Modules may be edited by any standard text editor that produces plain ASCII files. The Oberon-A compiler (OC) is then used to check the module for syntax errors and to translate it into an object file containing machine code.

A program is created out of its component modules by linking it. This combines all the object files into a single file and resolves any references between modules. This function is performed by the BLink program. BLink must be told which module is the main program module and which other modules are to be included in the program. The Oberon-A pre-link utility (OL) is used to generate the information BLink requires.

All these programs may be run from the CLI. Unfortunately, none can presently be operated from the Workbench. The good news is that the FPE program provides an Intuition-based interface that allows any of these programs to be called at the push of a button.

The entire process can be summarised as follows:

```
REPEAT
  FOR each module in the program DO
    REPEAT
      Edit the module source code
      Run OC to compile the module
    UNTIL no more syntax errors are reported
  END
  Run OL to generate information for BLink [*]
  Run BLink to link the program
  Test and evaluate the program
UNTIL satisfied with the result
```

* This is only necessary the first time the program is linked and whenever modules are added to or removed from the program.

1.18 Other documents you should read

The following documents will be the most immediately useful to you:

Oberon-2 Report	The Oberon-2 language report
FPE	Using the programmer's environment
OC	Using the compiler
Error codes	Error codes output by the compiler
OL	Using the pre-link utility
BLink	Using the linker

The AmigaGuide file Index.doc in the main Oberon-A directory contains an index of all the documentation and source code files distributed with Oberon-A.

1.19 The Author

All bug reports, suggestions and comments can be directed to:

Email : oberon@wossname.apana.org.au

Snail Mail :

Frank J Copeland
PO BOX 236
RESERVOIR VIC 3073
AUSTRALIA

Remember the J. It saves a lot of confusion at my end :-).

1.20 Reporting bugs and suggestions

This version of Oberon-A is a beta-test version. That means that it is basically complete, but has not been rigorously tested. Bug fixes and suggestions from users of this version will be incorporated in future releases.

You are encouraged to report any bugs you find, as well as any comments or suggestions for improvements you may have. I am also happy to answer any questions about the language itself. For information about Amiga programming in general you should consult the relevant Commodore and third-party documentation first. I can help if you have trouble translating examples written in C into Oberon.

Before reporting a suspected bug, check the file ToDo.doc to see if it has already been noted. If it is a new insect, clearly describe its behaviour including the actions necessary to make it repeatable. Indicate in your report which release of Oberon-A you are using. Include an example of a program or short fragment of code that demonstrates the bug. If you discover a bug in BLink, please report it but there is nothing that can be done except find a workaround. The original authors no longer support BLink.

I would like to hear your opinion of Oberon-A, even if you decide not to use it. Suggestions for improvements and additions are also most welcome. I am especially interested in the following areas:

* Compatibility with different versions of the Amiga hardware and operating system. So far it has only operated on a stock A500 with AmigaDOS 2.05 and a 20MB hard disk.

* How good/useful/helpful/complete the documentation is.

* How suitable it is for use by programmers with varying levels of experience, from beginners to hackers.

* How correct and useable the operating system interface modules are. These modules were translated from the C header files provided by Commodore. The translation was done quickly and only a fraction of the modules have been tested in any way.

* How useful the library modules provided are, and suggestions for additional modules.

* Departures from the language specification.

* Extensions to the language supported by the compiler.

* Memory management. I am unable to use Enforcer or similar utilities on my A500, so I would like people who can to report any Enforcer hits they get. I am also concerned about possibly excessive memory fragmentation caused by the run-time memory allocator.

1.21 Acknowledgements

The Oberon-A compiler is a port of a compiler written for the Ceres workstation by Niklaus Wirth. The book "Project Oberon" written by Wirth and Jürg Gutknecht contains a description of this compiler and the full source code for it. The original source can also be obtained by anonymous ftp from neptune.inf.ethz.ch. Many thanks to Professor Wirth for making this source code available.

The machine code generator for early versions of the compiler was a port of the corresponding parts of Charlie Gibb's A68K assembler. This code is no longer part of the compiler, but it was extremely useful in the early stages of development and debugging.

arp.library is used to fill in deficiencies in the version 1.3 AmigaDOS. It is the work of Scott Ballantyne et al of the AmigaDOS Replacement Project.

Torsten Jürgeleit's intuisup.library is used to create and manage the user interface for FPE.

1.22 Bibliography

The following works have proved useful in developing Oberon-A:

* N. Wirth. The programming language Oberon (Revised Report). Institut für Computersysteme, ETH Zürich. (See Oberon-Report.doc).

* N. Wirth. From Modula to Oberon. Institut für Computersysteme, ETH Zürich. (See ModToOberon.doc).

* N. Wirth and J. Gutnecht. Project Oberon. Addison-Wesley, 1992.

* H. Mössenböck and N. Wirth. Differences between Oberon and Oberon-2. Institut für Computersysteme, ETH Zürich.

* H. Mössenböck and N. Wirth. The Programming Language Oberon-2. Institut für Computersysteme, ETH Zürich.

* Commodore-Amiga, Inc. Amiga ROM Kernel Reference Manual: Libraries. Third Edition. Addison-Wesley, 1992.

* Commodore-Amiga, Inc. Amiga ROM Kernel Reference Manual: Libraries & Devices. Second Edition. Addison-Wesley, 1990.

* S. Kelly-Bootle. 680x0 Programming by Example. Howard W. Sams, 1988.

* Commodore-Amiga, Inc. The AmigaDOS Manual, 3rd Edition. Bantam, 1991.

* Commodore-Amiga, Inc. The AmigaDOS Manual, 2nd Edition. Bantam, 1987.

* R. Bornat. Understanding and Writing Compilers. MacMillan, 1979.

* A. V. Aho and J. D. Ullman. Principles of Compiler Design. Addison-Wesley, 1977.

* B. S. Gottfried. Programming with C. McGraw-Hill, 1990.

* S. Ballantyne and C. Heath. The Final ARP.Library Tour. Amiga Transactor, 1, 4, 44-57.

* J. Toebes. The Art of Assembly Language. Amiga Transactor, 2, 1, 38-43.

1.23 Revision Control

All the source code and document files in Oberon-A are managed using the freely-distributable HWGRCS package. This is a port of the Un*x RCS system.

Each new release of the entire package will be identified by a single release number. Partial releases containing bug fixes will have an

additional patch number. For example, the second release of Oberon-A will be known as "Oberon-A release 2". The first patch release of release 2 will be known as "Oberon-A release 2.1".

Individual programs are identified with a two part number of the form:

<version>.<patch>

The version number will change whenever substantial changes are made to the program. The patch number will initially start at 0 and will change whenever a bug-fix patch of the program is released.

Each module and documention file has a two-part revision number, of the form:

<version>.<revision>

All the component modules and documentation files of a program will have the same version number as the program. The revision number will change whenever the file's text is modified. The version number of a documentation file not associated with any one program will be the same as the current overall release number.

1.24 Oberon-A Release History

0.0	Initial version, written in Modula-2. Never released.
0.1 - 0.3	Intermediate versions, written in Oberon. Never released.
1.0	Initial beta-test release. Compiler upgraded to Oberon-2.
1.1	Bug fixes and some improvements to the compiler and utilities.