Oberon-A.doc

COLLABORATORS

	TITLE : Oberon-A.doc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		September 19, 2022	

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 1.4ß

    Created by: fjc (Frank Copeland)
     $Revision: 1.6 $
       $Author: fjc $
         $Date: 1994/08/08 21:30:19 $

Copyright © 1994, Frank Copeland.
```
_____

*** IMPORTANT NOTE FOR USERS OF OBERON-A 1.3 OR EARLIER VERSIONS ***

You are strongly advised to read the
              Updating
               section before proceeding
any further.
_____

Links marked with "+" are new, those marked with "*" have been changed.

                Oberon   *
                 What is Oberon?

                Oberon-A   *
                 What is Oberon-A?

                Distribution   *
                 Distribution and Copyright

                Requirements   *
                 What do I need to run Oberon-A?

                Installation   *
                 How do I install Oberon-A?

```
                Updating         *
                 How do I update an earlier version?
  Changes           *    Changes since the last release
  To Do             *    Bugs to fix and improvements to make


                Getting Started
                 How do I get Oberon-A running?

                Programming      *
                 How do I create a program with Oberon-A?

                Documentation    *
                 What else do I need to read?

                Resources        +
                 What other resources are there?


                The Author       *
                 Contacting the author

                Mailing List
                 Networking with like-minded people

                Bugs & Suggestions
                 Reporting bugs and suggestions


                Acknowledgements *
                 Who did what and why

                Bibliography
                 References used in developing Oberon-A

                Revision control
                 How revision control is handled in Oberon-A

                Release history
                 The history of Oberon-A
```

1.2   What is Oberon?

```
                The following are taken from  the  FAQ  file  for  the  comp. ←
                     lang.oberon
  Usenet newsgroup, Copyright © 1994 Michael  Gallo  and  reproduced  with
  permission.


                What is Oberon?

                The programming language Oberon

                The programming language Oberon-2
```

```
        The 'Oberon family' of languages

        Bibliography                    +
```

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. There are a number of
useful modules, including the beginnings of an object-oriented
application Framework.

The Oberon-A Interface is a collection of library modules that
provides a complete interface to the Amiga's operating system. This is
based on Commodore's 40.15 interfaces, for release 3.1 of the operating
system.

The Oberon-A archive contains a programming environment utility, the
compiler, an error lister (OEL, by Johan Ferreira), a pre-link utility,
a recompilation utility, and the source code for the Library and
Interface modules. 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

Oberon-A and the Oberon-A Library are:

  Copyright © 1993-1994, Frank Copeland

Oberon-A is free software; you can redistribute it and/or modify it
under the terms of version 2 of the GNU General Public License as
published by the Free Software Foundation.

The Oberon-A Library is also free software; you can redistribute it
and/or modify it under the terms of version 2 of the
GNU Library General Public License.

Oberon-A and the Oberon-A Library are distributed in the hope that they
will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public Licenses for more details.

You should have received copies of the GNU General Public License and
the GNU Library General Public License along with Oberon-A; if not,
write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge,
MA 02139, USA.

The Oberon-A Interface is:

 Copyright © 1994, Frank Copeland

The Oberon-A Interface is also free software, but it is not subject to
the GNU licence. Instead, it may be freely distributed, but only as
part of the Oberon-A archive for use in creating software using the
Oberon-A compiler.

Programs created with Oberon-A may be distributed under any terms their
creator desires. However, as all such programs must be linked with one
or more Oberon-A Library modules, the programmer should be aware of the
requirements of clause 6 of the GNU Library General Public License.
Oberon-A is intended mainly for personal use, and for the creation of
other free software. Commercial programmers may prefer to use a
commercial Oberon compiler.

Parts of the Oberon-A compiler and Library are based on source code
developed at ETH Zuerich. Permission to use, copy, modify and
distribute this software is granted by ETH (see the file
ETH-Copyright.txt).

The Oberon-A Error Lister (OEL) is:

  Copyright (C) 1994 Johan Ferreira

OEL is free software and is distributed under the GNU General Public
Licence. See OEL.guide for details.

The archive file containing Oberon-A also includes a number of other
freely-distributable programs that are used by Oberon-A. These
programs are copyrighted by their authors and distributed under
conditions set by those authors. These programs include:

  BLink, Copyright © 1986, The Software Distillery.
  arp.library, Copyright © 1987/1988/1989 by Scott Ballantyne
  intuisup.library, Copyright © 1992, Torsten Jürgeleit.

This document and others in the archive are in Commodore's AmigaGuide
hypertext format. A shared code library and reader program are included
in the archive. AmigaGuide, AmigaGuide.info and amigaguide.library are:

  (c) Copyright 1992 Commodore-Amiga, Inc. All Rights Reserved.
  Reproduced and distributed under license from Commodore.

  AMIGAGUIDE SOFTWARE IS PROVIDED "AS-IS" AND SUBJECT TO CHANGE; NO
  WARRANTIES ARE MADE. ALL USE IS AT YOUR OWN RISK. NO LIABILITY OR
  RESPONSIBILITY IS ASSUMED.

1.9 What do I need to use Oberon-A?

Oberon-A requires a Commodore Amiga personal computer with at least 1MB
of memory (more is recommended), running Release 2.04 (Kickstart 37+)
or later of the Amiga operating system. Most of the individual programs
will still run under Release 1.3, but this will not last much longer.

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. Unfortunately, this linker is poorly

behaved and produces numerous Enforcer hits on high-end systems. It
will be replaced in time. Commodore's ALink linker (distributed with
the Native Developer's Kit) and Matt Dillon's dlink (distributed with
the DICE compiler) have been shown to work with Oberon-A. It is not
known if either causes Enforcer hits.

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 AmigaOS 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. Permanent installation involves modifying your
startup sequence.

Temporary installation
 Temporarily installing Oberon-A

Library modules
 Compiling the library modules

Permanent Installation
 Permanently installing Oberon-A

Uninstalling
 Removing Oberon-A from your system

1.11   Temporarily installing Oberon-A under Kickstart 2.0+

Run the script in the file Install, either by typing "Execute
Oberon-A/Install/Install" at the CLI prompt or by double-clicking
its icon. The script sets up a few logical assignments and makes the
libraries used by Oberon-A available with the ADD option of the AmigaDOS

Assign command.

1.12 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.

1.13 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. The "OLIB:" assignment is
automatically searched by the compiler and some other programs, so it
should be created even if it is not used.

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.

1.14 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
                .

1.15   Updating an earlier version of Oberon-A

                    The Oberon-A compiler in Release 1.4 generates linker labels in ←
                    a
slightly different form than previous compilers. Object files
generated by earlier versions of the compiler are therefore OBSOLETE.
This, combined with the new Amiga Interface modules, means that once
more all existing modules must be recompiled with the new compiler.

Existing source code does not need to be changed, with two exceptions.
The SYSTEM.SETCLEANUP standard procedure now accepts only one
parameter, which must be a parameterless, typeless, procedure or
procedure variable. A procedure variable is no longer needed to hold
the previous cleanup procedure. Oberon-A now uses square brackets to
indicate additional information for procedure variants, such as
register parameter specifications. The source code of library interface
modules must be changed to replace any braces with square brackets. See
OC.doc for details.

A number of files have been renamed, and some have been removed from
the archive. Renamed files are replaced with a dummy file containing
message pointing to the new file.

See Changes.doc for a more complete listing of the changes made in this
release.

Updating an existing Oberon-A installation should be done as follows:

  1. If you have modules of your own that will need to be recompiled,
     run the ORU utility to create lists of modules that can be used
     with the compiler's BATCH option to automate the process. See
     ORU.doc for details.

  2. Unpack the sub-archives in the Oberon-A archive into the Oberon-A
     directory.

  3. Recompile all the library modules, using the
              CompileLibs
               script
     in the Install directory.

  4. Make the changes mentioned above in any modules you have written
     and recompile them as well.

   5. Delete any files that have been renamed or removed. See
      Changes.doc for a list of the files affected. A script file called
      DeleteOld can be found in the Install directory. This will delete
      old versions of files that have been renamed or removed. Please
      study this script carefully before using it. If you are in any
      doubt, don't use it, and make any necessary changes by hand.

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. Many modules are "library" modules, which can be
 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.
 If the compiler detects any errors, it produces an error file
 indicating their location and nature. The error lister (OEL) can then
 produce a description of the error and the context in which it occurs.

 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 a linker
 program, such as BLink. The linker 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 the linker 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 needing work DO
    REPEAT
      Edit the module source code
      Run OC to compile the module
      IF there are errors THEN
        Run OEL to generate an error report
      END
    UNTIL no more syntax errors are reported
  END
  Run OL to generate information for BLink [*]
  Run the linker 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.
```

                      Using a linker with Oberon-A +

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
 OEL          +     Using the error lister
 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

Oberon-A was mostly written by Frank Copeland.

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

```
  Email : fjc@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 :-).

Note that the e-mail address has been changed to my private mailbox, to
avoid confusion with the mailing list server.

I also regularly read the comp.sys.amiga.programmer newsgroup, and
will respond to any Oberon-A related posts I see there.

1.20   The Oberon-A mailing list

A mailing list has been set up to provide support for users of Oberon-A
and allow discussion of the compiler and the Oberon language in general.
To find out more, send e-mail to:

  oberon-a-request@wossname.apana.org.au

The Subject: line may be empty, and the body of the message should
contain only the following lines, starting in the left-hand column:

  HELP
  HELP oberon-a

The listserver software will reply with information about its commands,
how to subscribe, and a description of the list.

1.21   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.

1.22 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.

The following people have contributed modules and programs to Oberon-A:

  Terje Bergstr\vm has contributed the interface module for the
  Universal Message System.

Johan Ferreira has contributed the Oberon-A error lister, OEL.

Helmuth Ritzer has contributed interface modules for Nico Francois'
ReqTools library and the TextField Boopsi gadget.

Albert Weinert has contributed Classface.asm to allow access to
Intuition's Boopsi functions.

Carsten Ziegeler has contributed the interface module for his GuiEnv
library.

Apologies to anyone I have inadvertantly overlooked.

Thanks to those who have reported bugs and made suggestions for
improving Oberon-A.

1.23 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.

1.24 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 two-part
release number. Partial releases containing bug fixes will have an
additional update number. For example, the second major release of
Oberon-A will be known as "Oberon-A release 2.0". The first minor
release of release 2 will be known as "Oberon-A release 2.1". The
second update of release 2.1 will be called "Oberon-A release 2.1
update 2".

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

  <version>.<revision>

The version number will change whenever substantial changes are made
to the program. The revision 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.

1.25 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.

1.2   More bug fixes.

1.3   New compiler with varargs and improved symbol table handling.

```
               Heavily revised Amiga interface modules.

  1.4          Steady improvements to the compiler and utilities.
               Upgraded the Amiga interface modules to Release 3.1.
               Added more third-party interface modules.
```

1.26   Useful resources for Oberon-A Programmers

```
                  Apart from the documentation provided with Oberon-A, there are a ←
                       number
of other resources that you may find useful.

  Books

    See the
                  bibliography
                   in the comp.lang.oberon FAQ for a list of books
    on the programming language and the Oberon System. Reiser and Wirth's
    "Programming in Oberon" is a good introduction to the language,
    pitched at a level suitable for students as well as more advanced
    programmers. Mössenböck's "Object-Oriented Programming in Oberon-2"
    is exactly what the title says :-).

  UseNet newsgroups

    comp.lang.oberon

      From the comp.lang.oberon FAQ:

      "The Comp.lang.oberon newsgroup is a forum for discussing
      Oberon, both the programming language and the operating system,
      and any related issues.  Although not strictly accurate, this
      newsgroup is part of the Comp.lang.* hierarchy because it began as
      a spin-off of Comp.lang.modula2."

      The FAQ is posted in the newsgroup periodically, and can be
      obtained by anonymous ftp from rtfm.mit.edu in the /pub/usenet
      directory.

    comp.sys.amiga.programmer

      This group is for discussing Amiga programming in general, and
      there is occasional mention of Oberon and Oberon-A.

  Internet ftp sites

    ETH, where Oberon was developed, maintains an ftp site for
    distributing Oberon-related material, including the various versions
    of the Oberon System. Connect by anonymous ftp to
    neptune.inf.ethz.ch, and look in the /pub/Oberon directory.

    Oberon-A is primarily distributed through AmiNet, a network of ftp
    sites covering most of the Western world. Almost anything you could
    wish for as an Amiga programmer is available somewhere in the
    archive. The following sites are currently part of AmiNet:
```

```
    USA (MO)      ftp.wustl.edu           pub/aminet/
    USA (CA)      ftp.cdrom.com           pub/aminet/
    USA (TX)      ftp.etsu.edu            pub/aminet/
    Scandinavia   ftp.luth.se             pub/aminet/
    Switzerland   ftp.eunet.ch            pub/aminet/
    Switzerland   litamiga.epfl.ch        pub/aminet/
    Germany       ftp.uni-erlangen.de     pub/aminet/
    Germany       ftp.uni-paderborn.de    pub/aminet/
    Germany       ftp.uni-kl.de           pub/aminet/
    Germany       ftp.cs.tu-berlin.de     pub/aminet/
    Germany       ftp.uni-oldenburg.de    pub/aminet/
    Germany       ftp.coli.uni-sb.de      pub/aminet/
    Germany       ftp.uni-stuttgart.de    cd aminet
    UK            ftp.doc.ic.ac.uk        pub/aminet/
    Australia     splat.paxnet.edu.au     pub/aminet/
    Australia     archie.au               pub/micros/amiga/aminet (*)

    * archie.au is probably a better bet than splat.etc.
```

The Amiga Modula-2 and Oberon Club Stuttgart produces a collection of disks containing freely-distributable code written in, oddly enough, Modula-2 and Oberon. The Oberon code is naturally for the AmigaOberon compiler, but there may be stuff that can be adapted for Oberon-A. The collection is currently available by anonymous ftp from ftp.rz.uni-wuerzburg.de.

1.27 Bibliography

```
THE PROGRAMMING LANGUAGE
```

"Type Extensions" by N. Wirth; ACM Transactions on Programming Languages and Systems; 10, 2 (April 1988) 204-214.

"From Modula to Oberon" by N. Wirth; Software: Practice and Experience; 18,7 (July 1988) 661-670.

"The Programming Language Oberon" by N. Wirth; Software: Practice and Experience; 18,7 (July 1988) 671-690.

"Variations on the Role of Module Interfaces" by J. Gutknecht; Structured Programming; 10,1 (January 1989) 40-46.

"Object Oberon -- A Modest Object-Oriented Language" by H. Mssenbck and J. Templ; Structured Programming; 10,4 (April 1989) 199-207.

A New Approach to Formal Language Definition and Its Application to Oberon by M. Odersky; Verlag der Fachvereine Zrich; 1989.

"Oberon" by Dick Pountain; BYTE; March 1991.

"The Programming Language Oberon-2" by H. Mssenbck and N. Wirth; Structured Programming; 12,4 (April 1991).

Programming in Oberon: Steps Beyond Pascal and Modula-2 by M.
Reiser and N. Wirth; ACM Press; 1992.

"A Systematic Approach to Multiple Inheritance Implementation" by
J. Templ; ACM SIGPLAN Notices; Volume 28, Number 4 (April 1993).

Object Oriented Programming in Oberon-2 by H. Mssenbck; Springer-
Verlag; 1993.

A Programming Language for Vector Computers by R. Griesemer; Swiss
Federal Institute of Technology (ETH Zurich); Dissertation Number
10277, 1993.

THE OPERATING SYSTEM

"Designing a System from Scratch" by N. Wirth; Structured
Programming; 10,1 (January 1989) 10-18.

"The Oberon System" by N. Wirth and J. Gutknecht; Software:
Practice and Experience; 19,9 (September 1989) 857-893.

The Oberon System: User Guide and Programmer's Manual by M. Reiser;
ACM Press; 1992. This was reviewed in Computing Reviews articles
9109-0679, 9209-0651, 9209-0652, and 9207-0443.

Project Oberon: The Design of an Operating System and Compiler by
N. Wirth and J. Gutknecht; ACM Press 1992.

"Oberon: A Glimpse at the Future" by Dick Pountain; BYTE; May 1993.

"Implementing an Operating System on Top of Another" by M. Franz;
Software: Practice and Experience; 23,6 (June 1993) 677-692.

Distributed Object-Oriented Programming in a Network of Personal
Workstations by Spiros Lalis; Swiss Federal Institute of Technology
(ETH Zurich); 1994 (in preparation).

1.28   "

Oberon-A does not require a specific linker. Any linker that can
process standard AmigaDOS object files may be used, with some
limitations. The BLink freely distributable linker is included in the
Oberon-A archive and is presumed to be the default linker. However, it
causes Enforcer hits on Amiga's equipped with MMU's, making it almost
unusable on those systems. Commodore's ALink, and Matt Dillon's DLink,
have both been shown to work with Oberon-A, with varying degrees of
success. PhxLnk is not suitable for use with Oberon-A.

Before a linker can create an executable program, it must first be told
which object files are to be included. Linkers distinguish between
object files proper, which are copied whole into the executable, and
library files, which are scanned so that only code that is actually
used is included in the executable. In an Oberon-A program, only the

main program module needs to be copied complete, and all other modules
can be treated as libraries.

The modular nature of Oberon tends to produce a lot of object files,
making the task of determining which ones are to be included too
difficult to do by hand. The OL utility solves this problem by scanning
the symbol file of the main program module, and those of any modules it
imports, to determine which modules form part of the program. It
outputs a list of modules to a file, which can then be read by the
linker. This depends on the linker having an option which allows file
lists to be passed as files. ALink, BLink and DLink all have such an
option. PhxLnk does not, making it unsuitable for use with Oberon-A.

Currently, OL understands the file list formats expected by ALink,
BLink and DLink, each of which is different. The programmer need only
tell OL which linker is being used. The list file has the name of the
main program module, and ".with" as an extension.

DLink has an unfortunate restriction that makes it less suitable than
the other linkers. It treats all files with ".o" or ".obj" extensions
as object files, and all files with ".l" or ".lib" extensions as
library files. This is unfortunate because OC uses the ".obj" extension
for all object files it produces. A future version of the Oberon-A
compiler will include an option to use a ".lib" extension instead of
".obj". Until then it is probably best not to use DLink, however for a
possible workaround, see OL.doc.

Once OL has produced an object file list, the linker may be called. The
command line used depends on the linker being used.

    ALink : "ALink WITH <module>.with [options]"
    BLink : "BLink WITH <module>.with [options]"
    DLink : "DLink @<module>.with [options]"

Any of the options available for the given linker may be used, but
there should not be any object or library files listed on the command
line.