NAME

o2c − Oberon-2 to C compiler

SYNOPSIS

o2c [*option*]... *filename*
o2c [**-M|--make**] [*option*]... *module* [*command*]

DESCRIPTION

o2c has two major modes of operation: compile and make.

compile No option **-M** or **--make** is set. Translates one or more files into C code. This is quite fast since no object files are generated, useful to check a given module for syntactic or semantic errors.
Example:
o2c Hello.Mod compiles the module **Hello**. If no errors are found, the files **Hello.c**, **Hello.OSym**, and **Hello.h** are generated (the latter two only if the symbol file has changed).

make Option **-M** or **--make** is set. All modules imported directly or indirectly by a given main module are recompiled if necessary, including compilation into *.o* files. In a final step the object files are linked together to form an executable.
Example:
o2c -M Hello World generates the executable file **Hello_World**. **World** is the name of a parameterless procedure exported by module **Hello**. It's called right after the module initialization. In the terminology of the Oberon System it would be a command.
o2c -M o2c generates the executable **o2c**. Since the startup code of module **o2c** doesn't terminate, no special command has to be executed.

Errors are reported like this:

```
In file lib/Hello.Mod:
14:139 ';' expected
```

The first number is the error's position in the source code, i.e. file *lib/Hello.Mod*. It's a character position, not a line number. After the colon follow the error number and the error message. If you prefer a different output format, e.g. if you'ld like to read error messages in the source text context, use the filter **o2ef** (see also *o2ef(1)*).

o2c uses the following suffixes for its input and output files:

.Mod	Oberon-2 source
.OSym	symbol file
.c	intermediate C code
.h	intermediate header file
.m	marker file, generated for EXTERNAL modules
.o	object file, generated by gcc

The file *~/.o2c.red* tells the compiler where to search for existing files and where to create new files, depending on their suffix. The file is a list of rules of the form

```
pattern {"," pattern} = path {";" path} [+RCS].
```

pattern is a string of characters that may contain **?** to match a single arbitrary character and ***** to match zero or more characters. E.g. **.Mod* would match all files ending in *.Mod*.
To locate a file *file* the path lists of the patterns matching *file* are searched from left to right. The first path containing the file *file* is used. If *+RCS* is set for a source file *M.Mod*, RCS files *RCS/M.Mod,v* or *M.Mod,v* will be found in addition to the perfect match *M.Mod*. The compiler will automatically retrieve the working file *M.Mod* by calling **co** and will use it as input file.
For files generated by the compiler (e.g. symbol files or C code) the first path in the list associated with the first matching pattern will be used to store the file.

If file paths stored in *~/.o2c.red* are relative to a directory, the compiler (and the browser) must be executed in this directory. Otherwise the compiler will complain about files not found.

OPTIONS

Single character options may be grouped together. **-MA** is equivalent to **-M -A** or **--make --all**.

--make, -M
: Make an executable. The main module and all modules it depends on are checked for consistency and are recompiled if necessary. **gcc** is called to generate the object files and to build an executable.

--all, -A
: Force (re-)compilation of all imported modules. Ignored unless **--make** is set.

--makefile <file>
: Write a makefile for the given module into file <*file*>. No compilation will be done.

--warn, -W
: Print additional warnings. Without this option only errors will be reported.

--redir <file>
: Use <*file*> as redirection table (instead of the default *~/.o2c.red*).

--help, -h
: Give a summary on how to call the compiler.

--version, -V
: Print compiler version.

--verbose, -v
: Print calls to **co** or **gcc** and recompilation steps. Useful in connection with **--make**.

-a Disable assertions. Any calls to the predefined function **ASSERT** will be ignored.

-R Normally the compiler will produce code to detect the following illegal program states:
- a function is left without a return statement
- a CASE expression does not match any label
- a NIL pointer is being dereferenced
- an array index is out of range
- a SET index is out of range
- RECORD assigment fails
- a type guard or test is applied to a NIL pointer
- a type guard fails
- all guards of a WITH statement fail

The option **-R** will disable all those checks.
Note: These checks are fairly expensive, since all of them are performed on C level.

-O Optimize code, i.e. set *-O2* when calling **gcc**.

--usegc Include support for **gc** package. This overrides the compiler's default setting. **o2c** has to be compiled with the correct path to the **gc** library. Note: It's dangerous to mix *.o* files that are compiled for **gc** with ones that are not.

--nogc Disable support for **gc** package. This overrides the compiler's default setting. Note: It's dangerous to mix *.o* files that aren't compiled for **gc** with ones that are.

-g Include debug information on C level, i.e. set *-g3* when calling **gcc**.

-s Strip executable. Don't use this together with **-g** since **-s** will cancel the effects of **-g**.

-p Include profiler information (for **gprof**).

--cflags <string>
: Pass <*string*> as parameter(s) to the C compiler.

--ldflags <string>
: Pass <*string*> as parameter(s) to the linker.

The options **-a**, **-R**, **-O**, **-g**, **-s**, **-p**, **--cflags**, and **--ldflags** will only take effect during a *make*, in particular only for those files that are compiled by **gcc** while these options are set. If you want to make sure that these options are in effect for all modules, use the options **--make --all**, or **-MA** for short, to recompile all modules.

FILES

~/.o2c.red	path list
ErrorList.txt	error messages
file.Mod	Oberon-2 source file
file.OSym	symbol file
file.c	intermediate C file
file.h	intermediate header file
file.o	object file
file.m	stores time of last compilation of EXTERNAL

DIAGNOSTICS

The exit status is non zero if and only if an error occured during compilation.

SEE ALSO

o2b(1), o2ef(1), o2whereis(1)

AUTHORS

Michael van Acken, Juergen Zimmermann