Using the delphi compiler in the Lazarus IDE

Michaël Van Canneyt

May 24, 2023

Abstract

The Lazarus IDE has some advantages over the Delphi IDE. For one thing, it works cross-platform: you can use Lazarus on your Mac or on Linux. So can you use it to work on your Delphi project and compile the result with the Delphi compiler ? In this article, we show you how.

1 Introduction

The Delphi IDE is suitable for windows only. Lazarus can be used on many platforms. So what if you wish to work on your delphi code on your mac, but still compile with Delphi ?

Theoretically, you can: the Delphi command-line compiler can easily be executed in wine, a platform that provides a windows-compatible API on unix systems, which can run windows binaries directly on Linux (and on Mac). Although it should be noted that not all versions of wine are suitable to run the Delphi command-line compiler (let alone the full IDE).

Even on windows, there can be reasons to prefer using the Lazarus IDE over the Delphi IDE: The Lazarus IDE is faster, and has superior code tools compared to the Delphi IDE: the code completion is far better.

You could edit in lazarus, and execute the delphi command-line compiler in a terminal or console window.

But Lazarus can do better. While it is of course primarily designed for pascal code, it can in fact also be used to edit and compile C code or Javascript - or indeed any compiler. Lazarus offers an API to help you call a compiler and analyse the output of that compiler. For example, there is a package that can analyse the output of the GCC (GNU C Compiler) compiler and jump to the right place in the code on GCC error messages.

2 The delphi tool in Lazarus

Delphi comes with a command-line compiler. Can the Lazarus IDE API not be used to execute the Delphi compiler in Lazarus ? The answer is: yes, of course !

Since some time, there is a package that does exactly that: the `delphitool` package. It is in the lazarus source tree, and will be shipped with the next major release. It allows you to call the delphi compiler, and will analyse the output messages of the Delphi compiler, and display them in the messages window in a manner that allows you to click on the message and the IDE will jump to the relevant source location

The package goes further than that: optionally, it will take the compiler command-line options used for the FPC compiler and convert them to equivalent Delphi command-line options: paths for units, generation of debug information or optimizations are just some of the options that are converted to delphi options.

The options are written to a configuration file, and you can use this configuration file in your command-line to invoke the Delphi compiler.

Since the only practical way to execute the Delphi command-line compiler on linux or mac is to use Wine, the Lazarus IDE will offer you to convert the filenames in output messages from windows notation to Unix notation: it will map drive letters to directories on your linux or mac machine, and changes backslashes to slashes. It will analyse the wine drive letter mapping to be able to correctly map the paths.

Installing this package is done in the usual manner:

The package is in Lazarus' source tree, in the development branch of the lazarus git repository. It is located in the components/compilers/delphi directory, and the package file is lazdelphi.lpk.

If you don't have a git version of Lazarus on your system, you can also simply download the files for this package from Gitlab and install the package in an older IDE: the APIs used by this package exist for quite some time.

Open the package file lazdelphi.lpk, and install it with the 'Use - install' context menu. After rebuilding the IDE and restarting the IDE you should have new pages in the IDE tools - options dialog and in the project options dialog.

3 IDE Configuration

Before you can use the Delphi compiler, you must configure the Lazarus IDE. In the `Tools - Options` dialog, there is a new frame 'Delphi compiler', see figure 1 on page 3. On this page, the following settings are available:

Delphi compiler executable This is the full path of the delphi command-line compiler.

Delphi configuration file extension When the IDE generates a configuration file for the Delphi commmand-line compiler, it will use the same name as the Lazarus project file, but with the extension indicated here.

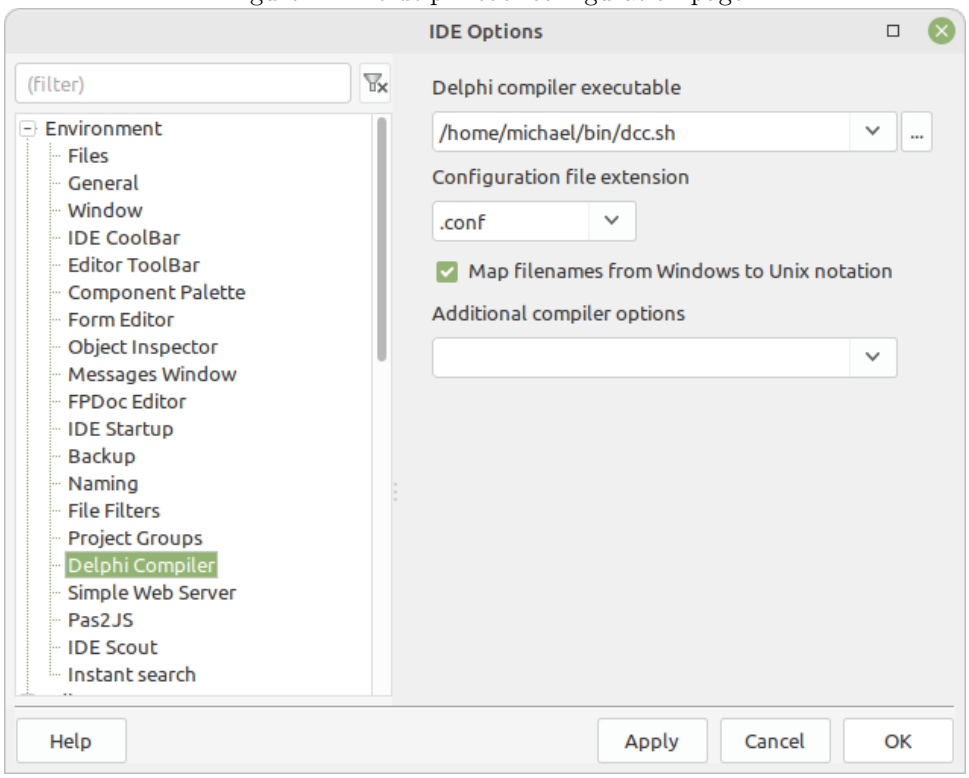
Map filenames from Windows to Unix notation This option is only available on linux or mac. When checked, the IDE will convert all filenames in the output of the Delphi compiler to unix notation, and will replace drive letters assigned by Wine with the correct directory on your unix system.

Additional compiler options these options will be passed on the command-line to the compiler in addition to the configuration file, for every project you wish to compile.

On Linux and MacOS, the compiler can be called through wine. Distributed with the delphi tool is a script dcc.sh that handles this for you: it will transform paths, and will make the generated binary executable (something which the delphi compiler does not do). You can point the IDE to this script instead of wine and the actual DCC binary.

The script can also be used on the command-line, it is not specific to lazarus.

Figure 1: The delphi tool configuration page



4 Project Configuration

In order to use the Delphi compiler for a project, you need to configure the project for this. This is of course done in the project options dialog.

There are 2 steps to configure. The first is general and can be configured in the frame 'Delphi compiler', see figure 2 on page 4. 2 options can be set here:

Generate Delphi config file based on FPC compiler options when checked, the IDE will - on every compile - generate a configuration file with the current options from the FPC compiler configuration.

Additional compiler options these options will be passed on the command-line to the compiler in addition to the configuration file, only for this project.

The second configuration is to adjust the compiler call. This is done in the 'Compiler commands' frame, the last frame under the compiler options. See figure 3 on page 4. 3 things must be done:

1. In the 'Execute before' box, enter the delphi compile command you wish to execute when giving the "compile" or "build" commands and check the 'compile' 'build' and 'run' checkboxes. More about the command below.
2. In the 'Execute before' box , check the 'Delphi compiler' in the list of Parsers. This tells the IDE that it should parse and analyse the output of the compile command with the 'Delphi compiler' parser tool. This tool has been registered by the 'lazdelphi' package.
3. In the 'Compiler' box, uncheck the 'compile' 'build' and 'run' checkboxes.

Figure 2: The project delphi configuration page

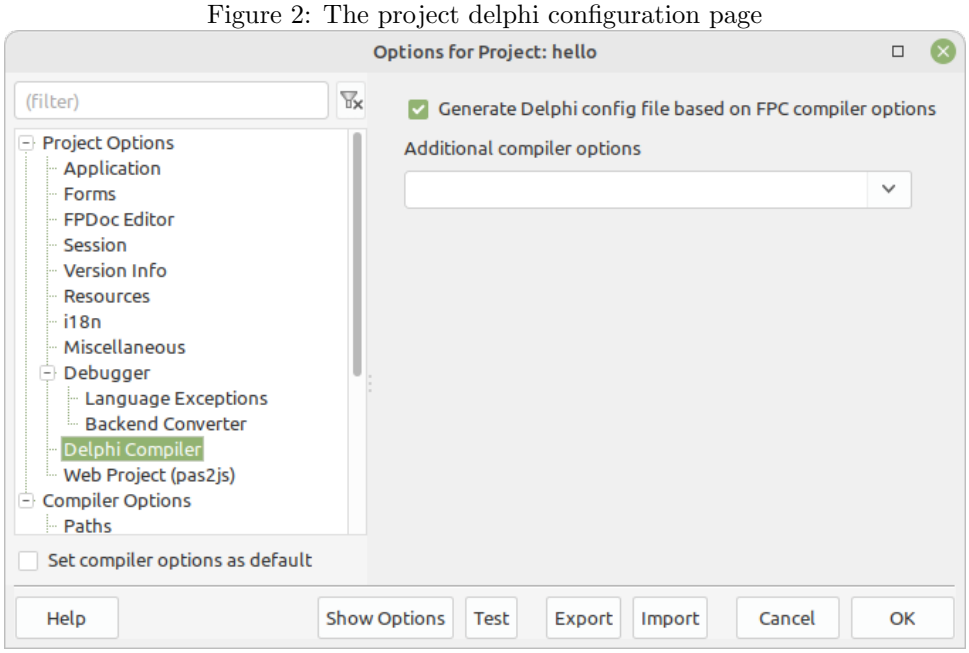


Figure 3: The project compiler commands page

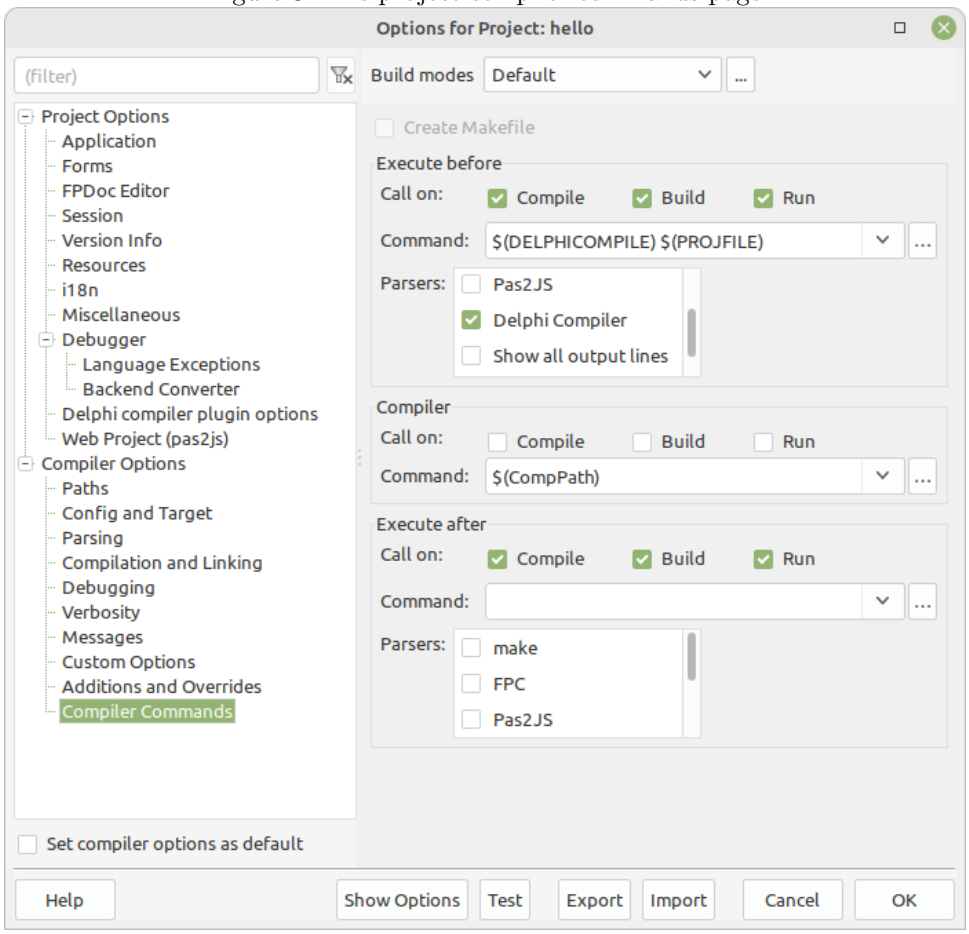
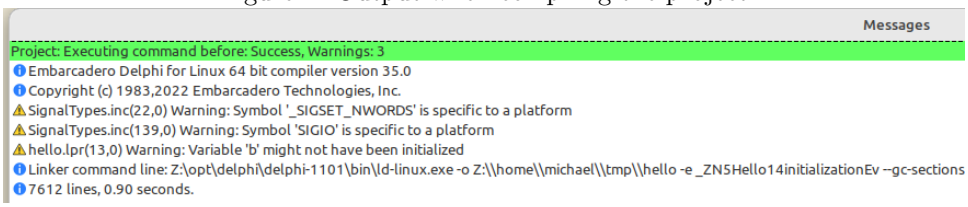


Figure 4: Output when compiling the project



After this, when compiling or building the project in the IDE, the 'execute before' command-line will be executed and the FPC compiler will not be called.

Note that you can perfectly have 2 build modes: one to compile your project with Delphi, one to compile your project with FPC.

So how to specify the delphi compile command in the 'Execute before' edit box ? You can specify the compile command completely yourself, for example:

```
c:\delphi\bin\dcc32.exe -V c:\projects\myproject.dpr
```

Or you can use macros. The lazdelphi package defines several macros, which you can use to compose your command as you see fit:

DCC This macro is expanded to the Delphi compiler binary path as set in the IDE options.

DCCCONFIG This macro is expanded to the delphi compiler configuration file generated by the IDE for your project. The filename is preceded by an sign if it is non-empty. (the sign is how the config file is specified on delphi compiler command-line)

'DCCARGS' This expands to the concatenation of the 'additional compiler options' specified in the global and project-specific settings.

DELPHICOMPILE this macro is in fact equivalent to the three above macros combined:

```
$(DCC) $(DCCARGS) $(DCCCONFIG)
```

it exists for convenience.

So the simplest compile command is:

```
$(DELPHICOMPILE) $(PROJFILE)
```

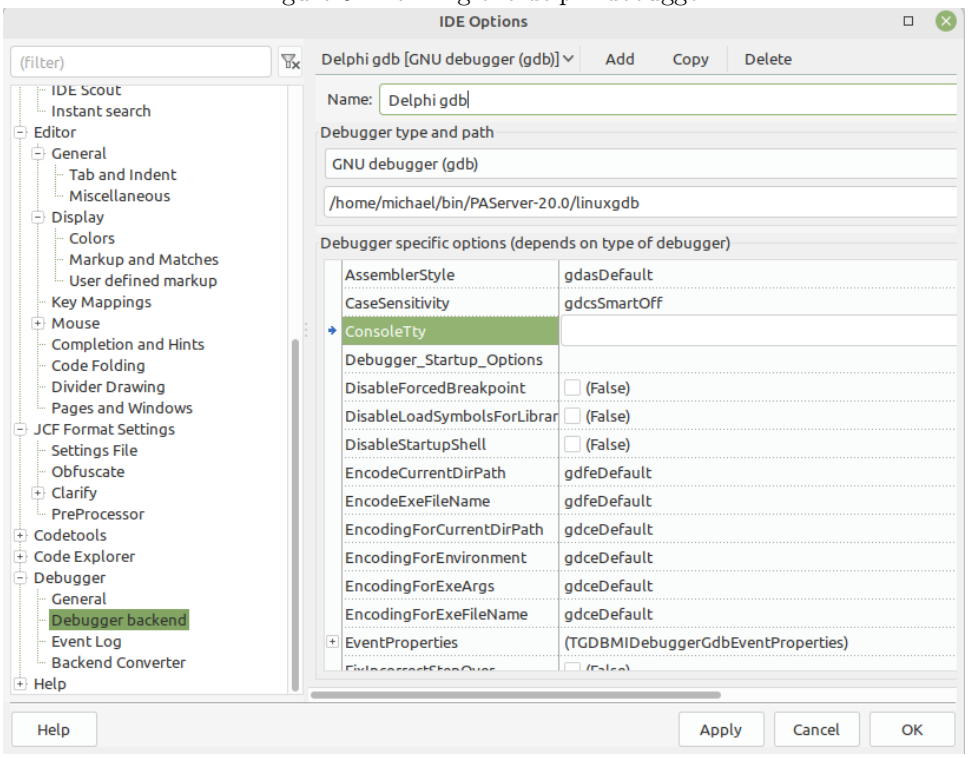
Exactly as shown in figure 3 on page 4.

Once this is all done, you can compile your project with Delphi, and the result can look like what you see in figure 4 on page 5. Clicking on the warning will take you to the correct location in the project.

5 Debugging

The lazarus IDE of course offers debugging, and the debugger has lots of advanced features (see the recent contributions by Martin Friebe in Blaise pascal magazine).

Figure 5: Defining the delphi debugger



Thus the question whether the compiled executable can be debugged in the lazarus IDE is of course relevant. The answer to this question is 'Yes, but...': On linux it is definitely possible to debug the delphi-generated executable.

In order to do this, GDB (the GNU debugger) must be used. To get the best results, a specially modified version of GDB is needed. Delphi ships this gdb version as part of it platform assistant 'PAServer' for linux.

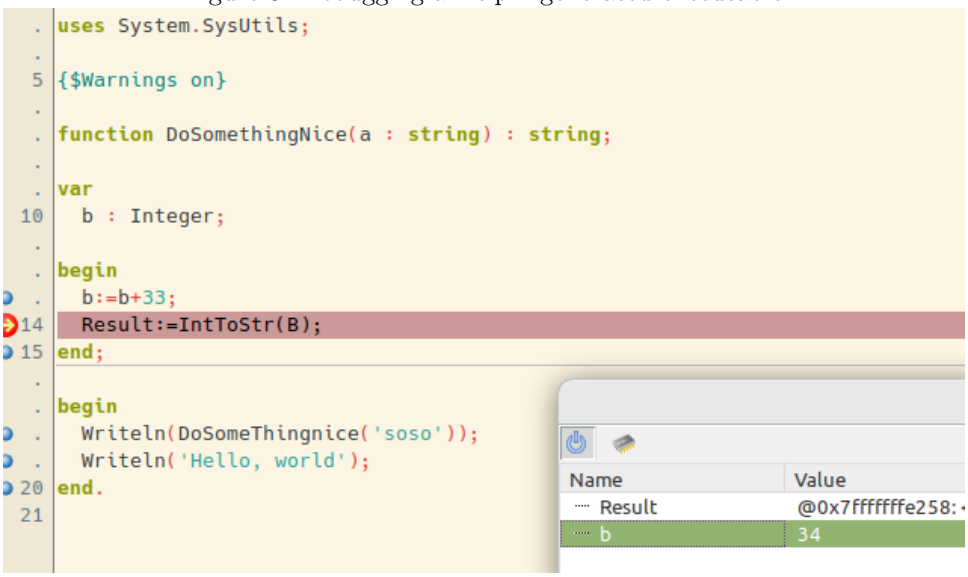
Fortunately, lazarus can use this GDB executable. You can use set it up in the Lazarus IDE in the tools - Options dialog under 'Debugger - debug backend'. The IDE can work with multiple debug backends, so what we must do here is define a new debugger backend. To do so, the following actions must be performed:

1. click 'Add' at the top of the dialog.
2. Enter a new name in the 'Name' edit.
3. For debugger type, select 'GNU debugger (gdb)'.
4. Select the 'linuxgdb' binary that is part of PAServer.

The result will look like figure 5 on page 6. Once this is done, you can run your application in the debugger: you can set breakpoints, you can watch variables, as shown in figure 6 on page 7 The debug experience is not yet flawless: Delphi encodes certain Pascal types differently than FPC does, and the Lazarus IDE is (not yet) aware of this info. So to see the value some types, some typecasts may be necessary.

The author has not tested debugging on MacOS and Windows. Based on available knowledge about the tools on Windows and MacOS, it is most probable that

Figure 6: Debugging a Delphi-generated executable



- On MacOS lazarus can be used for debugging with lldb (the debugger used on Mac) using a similar technique as on Linux.
- On Windows the expectation is that debugging will not be possible, since Delphi uses a proprietary format for debug info on Windows.

6 Conclusion

The delphi tool can be used to compile your free pascal or delphi code with the Delphi compiler, right from within the Lazarus IDE. The tool works as it is now, but some improvements are still planned: the quick fixes that exist for the FPC compiler can also be implemented for the delphi compiler. Additional ideas for improvement are of course always welcomed by the lazarus maintainers.

A word of warning: While coding is not a problem, one should take care when editing visual forms: When you add a component (say, a `TEdit`) to a form, there are properties in the LCL that do not exist in Delphi. These properties will be written to the .dfm file by the Lazarus IDE. When you run the program and the form is created at runtime, there will be a streaming error: the components that are actually instantiated will be the Delphi components, which may be missing some properties. Inversely, when loading a Delphi form in the Lazarus IDE, there may be VCL properties (or even complete components) in the form that have no counterpart in the LCL and the lazarus IDE will show an error, asking you what should be done with these properties.