Using Windows Subsystem for Linux with Dymola

Windows Subsystem for Linux (WSL) is a lightweight virtual machine supporting a range of Linux distributions. It allows us to combine the best worlds of Windows and Linux in one environment.

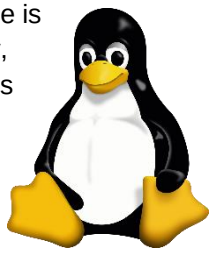
For the Dymola user, two key opportunities are available. First, to install a high-quality C compiler needed to compile and link the code generated by Dymola. Second, to use both environments in combination to generate Functional Mock-up Units that contain Linux and Windows executables in one convenient step.

Introduction

Every Dymola user needs a C compiler. The reason is that Dymola transforms the model we want to simulate into C code that it then compiles to into an executable code. Generating C code leverages existing compiler technology and allows the code to be compiled on many platforms.

However, the downside is obviously that the user must (once) go through the process of installing a suitable C compiler. Due to the complexity of Modelica models, the amount of generated C code by Dymola is a stress test for many compilers and only a handful will cope. On Windows, versions of the Microsoft C compiler (Professional or “community” editions) are popular. Linux systems have easy access to quality compilers.

By making Windows Subsystem for Linux (WSL) an integral part of Windows 10 and later versions, Microsoft has bridged the gap between the Windows and Linux worlds. For conventional, command line, usage of Linux the installation is quick and simple; installation of a graphical user interface is more complicated though. Fortunately, the simple command line interface plus a few optional modules are all that Dymola needs to compile models, and recent extensions in Dymola directly supports the integration.



However, the fundamentally new feature that WSL opens up is the generation of Functional Mock-up Units (FMUs) that contain executable code for both Windows and Linux platforms. With Dymola 2022 and later, the generation of such FMUs is automatic when suitable compilers are installed.

Installing WSL

The installation of WSL is simple, especially on Windows 10 version 20H2 which we will use as an example here. For earlier versions of Windows 10 an installation process in several steps is described by Microsoft. We have chosen Ubuntu 20.04 LTS (Long Term Support) as our Linux distribution.

► In Windows, open a Command Window in Administrator mode. Install WSL and Ubuntu:

```
wsl --install -d Ubuntu-20.04
```

You may be asked to reboot your computer to complete the installation.

► Create your user account when prompted.

► Then update your Linux system with the latest patches and install the C compiler and other tools:

```
sudo apt update
sudo apt upgrade
sudo apt install gcc g++ zip dos2unix
```

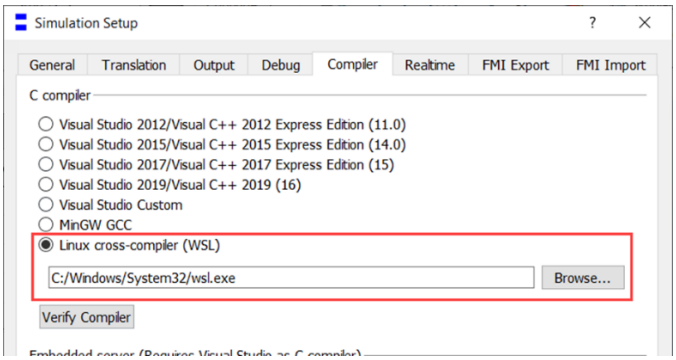
► To ensure that WSL can change file permissions, we need to edit the file `/etc/wsl.conf` so it has the following two lines (or create that file if it does not exist):

```
[automount]
options = "metadata"
```

Restart WSL or reboot the computer after any changes of this file.

Compiler setup in Dymola

Dymola 2022 supports the WSL-based compiler, so the setup process is limited to selecting the appropriate choice in Simulation>Setup>Compiler.



To test the WSL installation, click Verify Compiler. The verification will check that all needed Linux tools have been installed.

If you wish to use WSL gcc as the main compiler for Dymola on Windows, your setup is done here; translation and simulation work just as with any other compiler.

FMU generation for Windows and Linux

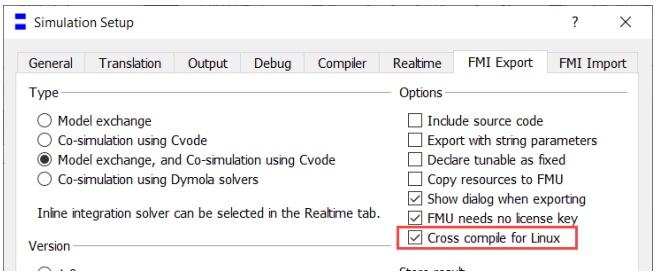
FMI has become a de-facto standard for exchanging executable simulation modules in a platform independent format. One particular advantage is that an FMU can contain binary code targeting several execution platforms; using Dymola 2022 generation of such cross-platform FMUs is easy.

First you need to set up compilers for both native Windows (e.g. Microsoft C), and WSL as described above.

Then a cross-platform FMU is easily generated; all you need is some very simple setup.

► Select the Windows compiler as your default compiler.

► Select cross-compilation in the FMI setup dialog.



Useful compiler options on Linux

Options for the C compiler are specified in Simulation>Setup>Compiler. When using WSL two options may be of particular interest.

`-O1` Moderate optimization that improves simulation speed without huge increases in compilation time.

`-pg` Standard Linux function profiling, which may help finding execution “hotspots” in external libraries or embedded C functions. After the simulation, the `gprof` command prints a profiling report.

The compilation scripts use the `cc` command to compile the C code. The `clang` compiler is an alternative, but that requires a small edit in the file dymola/bin/dsbuild.sh.

Conclusion

WSL bridges the gap between Windows and Linux, providing a command line environment with minimum setup. That makes it possible to use WSL as a free, high quality, compilation and simulation environment in Dymola.

The most significant benefit for the Dymola user on Windows is that cross-platform FMUs can be directly generated without any extra manual steps.