How to port the OpenMI from Windows to Linux

Table of contents

1. Introduction

The document describes how to generate **OpenMI dlls and executables** on a Linux machine. This is done by using **Mono, a .NET Open Source software** for different platforms. Most of the OpenMI functionality has been ported:

- the definitions of the **OpenMI Standard**,
- the source development kit **SDK**,
- running OpenMI from command line or in the **ConfigurationEditor** (GUI),
- some LinkableComponents of the examples.

2. Technical prerequisites

2.1. Machine and OS of the test system

- workstation with an Intel Xeon processor
- 64bit openSUSE 11.0

2.2. Mono

- Mono v. 1.9.1. for openSUSE 11.0 in 64 bit mode. V. 1.9.1. is also referred to as Mono 2.0.
- The standard and the sdk sources should be compilable with previous versions. But the GUIs require Windows.Forms, which has been shipped with v. 1.9.1. in Oct. 2008.
- Mono includes NUnit version 2.2.0 for UnitTests

3. The Mono compiler gmcs

3.1. Recommended compiler options

Command for generating a dll:

gmcs -target:library -out:<target>.dll -r<linkedLib_1>.dll,<linkedLib_k>.dll -pkg:<package_1>.pc,<package_m>.pc <source_1>.cs <source_n>.cs

-r:<linkedLib_?>.dll : reference to linked shared libraries (path and name)

-pkg:<package_?>.pc : ASCII files (path and name), that refer to linked shared libraries <linkedlib_?>.dll. They contain the path and the version of the dll.
Example for Windows.Forms:

prefix=/usr/lib/mono/2.0
exec_prefix=${prefix}
libdir=${exec_prefix}

Name: WindowsForms
Description: Windows Forms
Version: 2.0
Libs: -r:${libdir}/System.Windows.Forms.dll

Command for generating an exe file:

gmcs -out:<target>.exe -r<linkedLib_1>.dll,<linkedLib_k>.dll -pkg:<package_1>.pc,<package_m>.pc <source_1>.cs <source_n>.cs

3.2. Mono and GUIs (Grapical User Interfaces)

The GUI sources use Windows.Forms, that works with resource files in order to design the graphical elements. The Windows resource files (*.resx) are generated automatically in the Microsoft Visual Studio IDE. Linux applications do not process them and create a "resource not found" exception during runtime. Thus, the resources have to be converted with the Mono tool resgen / resgen2. The following command generates a Linux readable resource <name>.resource:
resgen <name>.resx generates a Linux readable resource <name>.resource .
Visual Studio assigns automatically a resource file to a C# file. On a Linux machine naming conventions guarantee the correct assignment. The C# source and its resource file must have the same prefix. This first part of the prefix must be the namespace.

Example for ElementSetViewer
namespace: Oatc.OpenMI.Gui.Controls
Linux source name: Oatc.OpenMI.Gui.Controls.ElementSetViewer.cs
Linux resource name: Oatc.OpenMI.Gui.Controls.ElementSetViewer.resources

The original Windows resources contained *.bmp graphics with a 24bit colour depth. Mono on Linux could not process them. The solution was to convert the BMPs externally to 8bit GIFs, before adding them to the resource in the Visual Studio IDE.

Mono can not process all type of GUIs. Mono 2.0 has nearly the full functionality of Windows.Forms. It does not support the WindowsPresentationFoundation WPF, shipped since .NET 3.0, which generates resources in XAML style or logical resources. Thus, it is highly recommended that the **developers of OpenMI GUIs use Forms elements** instead of WPF elements.

3.3. Help

Mono Forums are very helpful: http://www.go-mono.com/forums/
Error messages are often the same as in Microsoft Windows .NET. You can look them up on the MSDN homepage.

4. OpenMI on Mono in five steps

4.1. Download from SourceForge

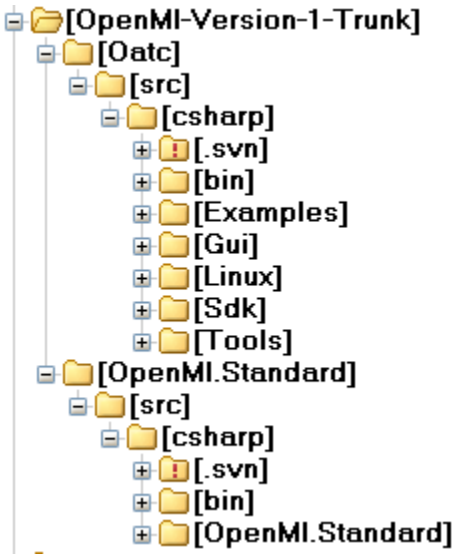

Fig. 1: Directory structure

After the first step you should have a copy of the OpenMI sources on your machine with the directory structure displayed in Fig. 1. OpenMI-Version-1-Trunk is in this examples the name for the starting directory $OPENMI_DIR. There are two new subdirectories:

- ./Oatc/src/csharp/bin: linux dlls / exe files
- ./Oatc/src/csharp/Linux: shell scripts for building dlls / exe files and running UnitTests; *.pc files

Use Subversion (svn) for download of the C# sources:

- create $OPENMI_DIR/Oatc/src/csharp and $OPENMI_DIR/OpenMI.Standard/src/csharp.
- go to $OPENMI_DIR/Oatc/src/csharp
- check sources out from http://openmi.svn.sourceforge.net/svnroot/openmi/branches/OpenMI-Version-1-Trunk/Oatc/src/csharp/.
- go to $OPENMI_DIR/OpenMI.Standard/src/csharp
- check sources out from http://openmi.svn.sourceforge.net/svnroot/openmi/branches/OpenMI-Version-1-Trunk/OpenMI.Standard/src/csharp/.

4.2. Building ...

4.2.1. ... with shell scripts

- Go to $OPENMI_DIR/Oatc/src/csharp/Linux/

- Edit the package files *.pc:
- drawing.pc: replace the path for System.Drawing.dll
- nunit-framework.pc: replace the path for nunit.framework.dll
- windows-forms.pc: replace the path for System.Windows.Forms.dll

- Run build.Standard.sh
- Run build.Sdk.sh and the Sdk including UnitTest will be built.
- Run build.Gui.sh and the Gui including UnitTest will be built.
- Run build.Examples.ModelComponents.SpatialModels.sh will generate test models.

The dlls and exe files will be copied to $OPENMI_DIR/Oatc/src/csharp/bin/.

4.2.2. ... with MonoDevelop

The IDE MonoDevelop is an alternative to the scripts. It imports Microsoft projects by loading the *.sln file. The author had some compilation problems with MonoDevelop 2.0 Alpha without clear error messages. Experienced developers may find a solution. Furthermore, MonoDevelop will be constantly improved in 2009, s. http://www.mono-project.com/Roadmap

4.3. Environment variables

It is recommended to extend two environment variables:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OPENMI_DIR/Oatc/src/csharp/bin/
export MONO_PATH=$MONO_PATH:$OPENMI_DIR/Oatc/src/csharp/bin/
```

4.4. Running the applications

Mono applications do not run directly on the Linux system, they are interpreted in a runtime engine, comparable to the virtual machine in java. Thus, the command in the terminal window command line has to start with mono:

- mono Oatc.OpenMI.Gui.ConfigurationEditor.exe
- mono Oatc.OpenMI.Gui.CommandLine.exe

4.5. UnitTests

- Go to $OPENMI_DIR/Oatc/src/csharp/Linux/
- runNunit.Sdk.sh for all Sdk dlls
- runNunit.Gui.sh for Oatc.OpenMI.Gui.Core.dll
- runNunit.Examples.ModelComponents.SpatialModels.sh e.g. for a RiverModel and a GroundWaterModel

You have done it.🙂

5. Miscellaneous

5.1. Further platforms

The generated applications will probably run on further platforms. On non-SUSE Linux derivates it may work without re-compilation. But it has not been proved yet. Interested persons are invited to test. A short notice to the community, e.g. to the OpenMI Forum on SourceForge, would be appreciated.

5.2. Can I use the same sources for Linux and Windows?

Yes, you can generate a set of dlls and exe files for Linux and a set for Windows with the same sources. The method Gui/Core/Utils.IsRunningOnMono decides during runtime, whether it runs on Mono or not. The build.*.sh scripts are tailored for Linux, for Windows you can use the Visual Studio projects.

5.3. Todo list

- portation of OpenMI Tools
- portation of SimpleRiver example

- ConfigurationEditor: Help page
- ConfigurationEditor ConnectionProperties dialogue: Buttons "Use Element Type Filter" and "Use Dimension Filter" do not work -> there is no filtering in the current version
- ConfigurationEditor: RegisterFileExtensions can not work on Linux.

- test on further platforms supported by Mono
- merging OpenMI-Version-1-Trunk (Linux) and OpenMI-Version-1-4-Trunk
- download of compiled dlls and exe files