MONO FOR CROSS-PLATFORM CONTROL SYSTEM ENVIRONMENT*

H. Nishimura[1] and C. Timossi[2], LBNL, Berkeley, CA 94720, U.S.A.

Abstract

Mono is an independent implementation of the .NET Framework by Novell that runs on multiple operating systems (including Windows, Linux and Macintosh) and allows any .NET compatible application to run unmodified. For instance Mono can run programs with graphical user interfaces (GUI) developed with the C# language on Windows with Visual Studio (a full port of WinForm for Mono is in progress). We present the results of tests we performed to evaluate the portability of our controls system .NET applications from MS Windows to Linux.

.NET PLATFORM AND CONTROLS

Windows in an Accelerator Environment

Microsoft Windows is widely used as a platform for GUI based applications used for machine operations and physics studies. Windows is also a useful platform for instrumentation controls because of the wide industry support of specialized drivers for it. For these reasons Windows has become an essential platform for accelerator and beamline control systems.

The fundamental programming API for developing Windows software is changing from Win32 to .NET. The .NET framework has been evolving since .NET 1.0 in 2002, and reasonable backward compatibility with existing Windows software APIs such as Win32 and Active/X (COM) has been maintained. During this time we have been adapting our applications for the .NET framework on Windows.

EPICS on Windows at ALS

Our accelerator control applications use the Channel Access (CA) layer of EPICS[1] to access accelerator controls data. To support the widest variety of development tools on Windows (e.g. Delphi, C++ Builder, Visual Basic and LabView), we package CA as an ActiveX Control we call SCACOM[2]. This control is a thin wrapper around another library, Simple Channel Access (SCA)[3], that was developed to ease CA client development at the ALS[4].

Although .NET programs can use ActiveX controls directly, there is an advantage to repackaging the control as a .NET assembly. ActiveX is only available on Windows platforms whereas .NET was designed to be portable to other platforms. So, a .NET application - even a GUI application using WinForm - at least has the potential to run unmodified on a non-Windows OS. We've named this new assembly: SCA.NET[5]. In fact, we did much more than re-package SCACOM. We decided to spend some effort recoding some of the routines (in C#) to make better use of CA, thus improving data access performance.

*Work supported by the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

[1.] H_Nishimura@lbl.gov. [2.] CATimossi@lbl.gov

MONO AS .NET ON LINUX

Mono for Cross Platform Support of .NET

Mono[6] is an implementation of the .NET Framework originally developed by Ximian which is now under Novell. It can be run on multiple operating systems (including Linux, Mac OS X, Solaris, BSD, and Windows) on multiple hardware platforms (s390/s390x, SPARC, PowerPC, x86, x86-64, IA64 and ARM). We report on Linux running on x86 based PCs in this paper.

Compatibility

The Microsoft .NET Framework for Windows has evolved from version 1.0 in 2002, to 1.1 in 2003, to 2.0 in 2005 (version 3.0 will be delivered on Vista). Not surprisingly, Mono has lagged behind Microsoft and is currently delivering version 1.1. In addition to the basic framework, Mono also includes support for ADO.NET for database access and WinForm for GUI development.

In our experience, non-visual classes, such as ADO.NET, have been well supported on Mono.

On the other hand, GUI programming using WinForm is behind that in .NET 1.1 on Windows. When we use graphical libraries from third parties, we need to take extra steps to assure their availability/compatibility on Mono. For example, we use a popular open-source library ZedGraph[7] for plotting and charting. Its newest version 5.0 is for .NET 2.0. However, the previous version for .NET 1.1 required only minor modifications to run on Mono.

Third party library support can also be an issue. Generally speaking, these .NET libraries, which applications need to access with the Platform Invoke interface, are now moving to managed code rather than unmanaged DLLs. However, this move to managed code often occurs together with the migration to .NET 2.0, which is only partially supported on Mono.

At the time of this conference in October 2006, the version of Mono is at 1.1.17. This version basically covers .NET 1.1 and some of the new .NET 2.0 features. Better compatibility is expected with the release of Mono 1.2 and 2.0 in the near future.

Mono as Runtime Environment

Mono becomes a runtime environment for the .NET 1.1 programs developed on Windows, including those made using WinForm. In principle, they should run on Linux with Mono without rebuilding as long as run-time libraries are available. However, it is not unusual to modify and rebuild such programs to alleviate minor incompatibilities.

To start a .NET application with Mono:

```
$ mono WinApp.exe
```

where WinApp.exe is a .NET application developed either on Windows or with the Mono development tools.

Mono as Development Environment

Mono includes a C# compiler (MCS) that enables .NET development on non-Windows platforms. Although there are more than 10 programming languages already available on Mono[8], including Java, Visual Basic.NET and Python, we currently focus on C# on Mono.

MonoDevelop[9] is the integrated development environment (IDE) for Mono, running primarily on Linux. It can import a Visual Studio 2003 solution containing .NET 1.1 projects in C# developed on Windows. It supports GUI development with several GUI tool kits including GTK# with visual designer. However, WinForm is not currently supported in this manner.

MONO FOR EPICS CLIENTS ON LINUX

SCA.Net for Mono

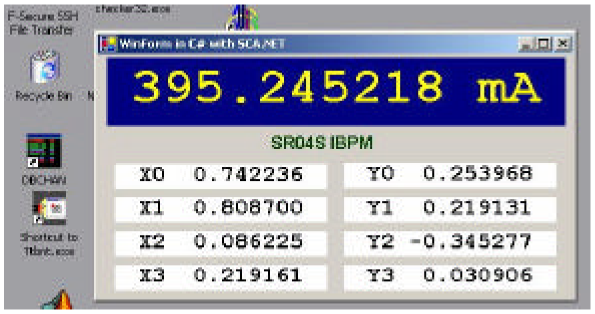
SCA.NET wraps the Channel Access shared libraries as a .NET assembly that calls into CA using the Platform Invoke API (also known as P/Invoke). For example, on Windows:

```
public unsafe class Ca
{
    . . .
    [DllImport("ca.dll")]
    public static extern
          short ca_field_type (IntPtr ChanID);
    . . .
}
```

On Linux, we would expect to need to replace the reference to “ca.dll” with “ca.so” (the shared library for Linux-x86). However, if this class is built with “Any CPU” option, this is not required. It picks up “ca.so” properly at runtime on Linux. Therefore, there is no need to modify the source code of SCA.NET.

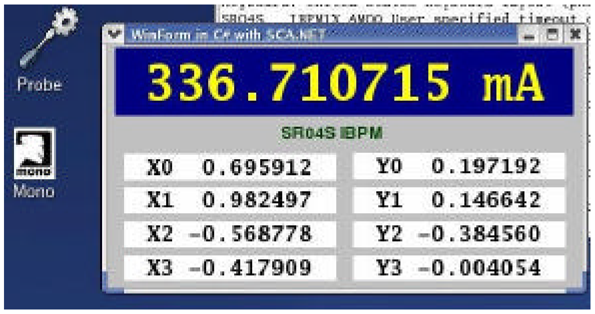
Here is an example of a client program running on Windows XP (Fig.1) and Linux (Fig.2); there is no need for source code changes. It reads the beam current and beam locations (X, Y) at 4 locations through SCA.NET. Its binary is identical on both platforms. Here ca.dll is in the PATH on Windows and ca.so is in the LD_LIBRARY_PATH on Linux.

Thus, by carefully limiting the use of WinForm 1.1 controls for GUI programming, Linux is seamlessly supported at run-time for .NET 1.1 programs with EPICS access.



```
395.245218 mA
SR04S IBPM
X0  0.742236     Y0  0.253968
X1  0.808700     Y1  0.219131
X2  0.086225     Y2 -0.345277
X3  0.219161     Y3  0.030906
```

Fig.1. EPICS Client Program on Windows XP



```
336.710715 mA
SR04S IBPM
X0  0.695912     Y0  0.197192
X1  0.982497     Y1  0.146642
X2 -0.568778     Y2 -0.384560
X3 -0.417909     Y3 -0.004054
```

Fig.2 EPICS Client Program on Linux

AKNOWLEDGEMENTS

The authors thank A. Biocca and D. Robin for their support, T. Scarvie for useful advices, and C. Ikami and T. Kellogg for their technical support.

REFERENCES

[1] L. R. Dalesio, et al., ICALEPCS '93, Berlin, Germany, 1993.http://www.aps.anl.gov/epics

[2] C. Timossi and H. Nishimura, IEEE PAC’97, 0-7803-4376-X/98, p805, 1998

http://www-controls.als.lbl.gov/epics_collaboration/sca/win32

[3] http://www-controls.als.lbl.gov/epics_collaboration/sca

[4] LBL PUB-5172 Rev. LBL,1986

A. Jackson, IEEE PAC93, 93CH3279-7(1993)1432

[5] H. Nishimura and C. Timossi, PCaPAC 2005, Hayama, Japan, 2005.

[6] http://www.mono-project.com

[7] http://zedgraph.org

[8] http://www.mono-project.com/Languages

[9] http://www.monodevelop.org