



Ways to Migrate A Windows Application onto UNIX/Linux.

Abstract

Some few reasons to migrate a Microsoft Windows application to UNIX are scalability, server consolidation, customer demand and other business factors. This white paper investigates the choices available and provides a roadmap to completely migrate from Windows to UNIX and Linux. The scope of this paper is limited to Win32 and VC++ applications utilizing all features of Windows including all of the Win32 API, Registry, COM, and OLE.

Executive Overview

This white paper list a few options and their accompanying advantages and disadvantages and further helps the reader to make an intelligent choice by offering a roadmap to completely migrate Windows application written in C/VC++ to UNIX and Linux. The pros and cons of each option and the recommendations made are to suit those specific situations.

Glossary

API - Application Programming Interface

WIN32 API - C API for writing Windows Applications

MFC - Microsoft Foundation Classes

VC++ - Microsoft Visual C++. Also used to mean the frameworks supplied such as MFC, template libraries etc.

Existing Solutions

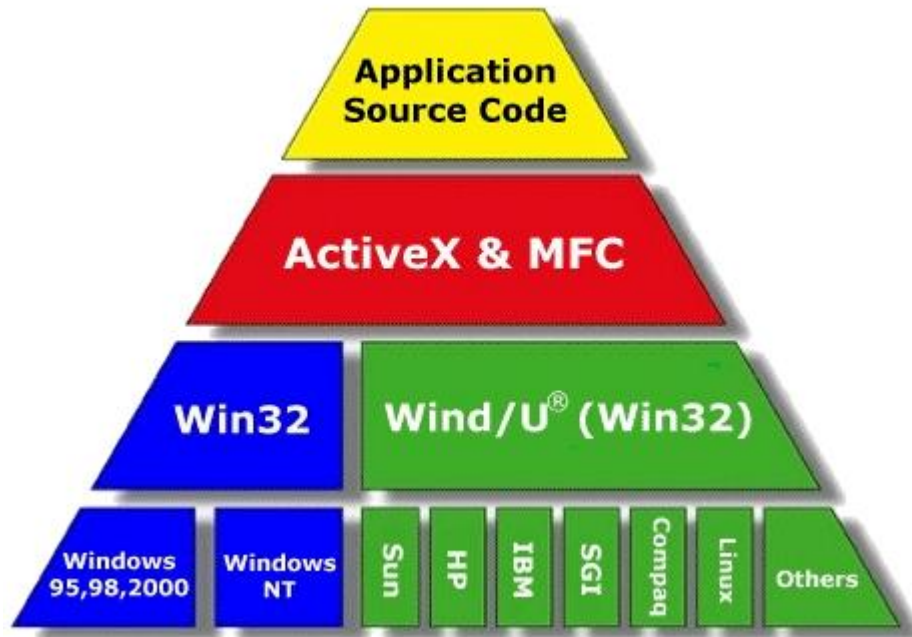
Currently, there are four options worth pursuing to migrate a Win32 based application to UNIX.

1. Using a source code emulator such as Wind/U or MainWin.
 2. Using Wine, a binary emulator.
 3. Partially rewriting the application using some cross-platform toolkit (for example, Qt).
 4. Completely rewriting the application using a totally different technology like Java or Web technology.
- Platform vendors sometimes have toolkits to assist migrations to their platforms. For instance, Sun Microsystems has a Win32 Emulation Library, but it is geared towards Win32 server applications and does not cover many portions of the API. But these options are not as powerful as the ones listed above.

MainWin or Wind/U

MainWin from MainSoft and Wind/U from Bristol Technology are commercial source level emulation toolkits. Here is a brief description of Wind/U. MainWin and Wind/U are competing products that solve the problem in the same way Wind/U is a source code level emulation tool for migrating Win32 based applications onto all flavors of UNIX (Solaris [sun4, x86 and SPARC versions], HP-UX [PA-RISC, Itanium], AIX, SGI, AS400, Linux (Redhat and SuSE) etc. The support goes beyond the Win32 APIs and includes Registry, OLE, COM, MFC, Stingray APIs, SCM, ATL, and service APIs.

The figure below shows a schematic of Wind/U.



Wind/U libraries implement the Microsoft Win32 API, including features such as ActiveX, COM, OLE, WinInet, common controls, and common dialogs on major UNIX platforms, Compaq's OpenVMS operating system and IBM's OS/390 operating system. Wind/U also provides a number of utilities for configuring your cross-platform development environment, making common portability changes in your source code, and building the cross-platform versions of your application. Using Wind/U not only results in cross-platform applications with a native look-and-feel and superior performance, but also has the functionality found in Windows applications.

Wine

Wine is an Open Source implementation of the Windows API on top of X and UNIX. Think of Wine as a compatibility layer for running Windows programs. Wine does not require Microsoft Windows, as it is a completely free alternative implementation of the Windows API consisting of 100% non-Microsoft code, however Wine can optionally use native Windows DLLs if they are available. Wine provides both a development toolkit for porting Windows source code to Unix as well as a program loader, allowing many unmodified Windows programs to run on x86-based Unix-es, including Linux, FreeBSD, Mac OS X, and Solaris.

Comparison Matrix

The following matrix compares the different alternatives.

Feature	Wind/U or MainWin	Wine
Support for Server side APIs	Yes	Yes
Support for GUI	Yes	Yes
Support for COM, OLE, ActiveX	Yes	Partial
Support for COM, OLE, ActiveX	Yes	Partial
Cost of migrating	High	Low
Runtime costs	High	High

Disadvantages of Wind/U and MainWin

While Wind/U and MainWin provide the most complete solution among the available options, the costs are very high. Along with the high cost of purchasing developer licenses, there is runtime cost also. MainWin and Wind/U have licensed source code from Microsoft to provide a more complete solution. Due to this, the end customer has to pay royalty fees to Microsoft for every copy of the application that is shipped.

Usually, this leads to an increase in the price of the UNIX version of their product. Also, given the nature of this migration, the tools cannot accomplish complete automated porting. Typically, customers engage in Professional Services offered by Bristol and MainSoft to ensure that their application is completely ported to UNIX.

Sometimes, a considerable chunk of this professional services effort can go into adding features or fixing bugs in their product.

Disadvantages of Wine

While Wine is an open source solution and available free of cost, it lacks feature support. Wine is still at version 0.9.31, though the project started in the open source world way back in 1993.

Simple applications can be easily made to run under Wine. However, complex applications are difficult to port. A company named CodeWeavers is running its business based on this fact. Even common applications like Microsoft Office cannot be run off the shelf; one needs to purchase "CrossOver Linux" from CodeWeavers.

Also, the cost of getting defects in Wine fixed by CodeWeavers is prohibitive. The following is from their services page at <http://www.codeweavers.com/services/engagements/>

"The work that we do on Wine is time consuming and hard, which unfortunately also makes it fairly expensive. For example, we have a rough working metric that each bug in Wine costs us between \$1,000 and \$1,500 to fix." There are also concerns regarding the performance of Wine based applications as it an emulator and does run raw machine code.

Partially rewriting the application using Qt

The third option to migrate the application is to rewrite it. Qt is a very popular and high-featured cross platform C++ framework. By using Qt, you can reuse many portions of your existing source code in C/C++. Qt also has some support for converting MFC applications, but it is not a completely automated solution.

There are two versions of Qt. The first is a free open source version which can be used if you choose to open source your application. The other is a commercial version supplied by Trolltech. The licensing model offered - a license per developer per operating system. There are no runtime costs.

Completely rewriting the application

Another option is to completely rewrite the application in some cross-platform technology like Java or Web technologies. The disadvantage in this approach is that most of the current investments will be lost. If you use Java, you can reuse your non UI portions of C++ code by using mechanisms such as the Java Native Interface (JNI), but these come with performance and security compromises.

Conclusion

There are multiple ways to migrate a Windows application to Linux. This White paper not only discusses these available options but also the pros and cons of employing the same. Thus, helps companies to make an informed decision based on their situation. Migration can be made risky and expensive if handled by inexperienced developers.

Seemingly small things can lead to hard to debug issues and cause performance problems. Hence services of professional migration service providers who have executed similar projects are recommended.

An interesting article (reference 1) in the Redmond magazine narrates why such migration efforts fail with the help of a live example. The sentiment of this article is echoed in a Gartner report (reference 2) on software migration, which says:

“Companies can try and execute the migration on the cheap but can end up spending considerably more in hidden costs and lost productivity.”