Migrating Applications From Windows NT to UNIX Fast

Chane Cullens (ChaneC@Bristol.com)
VP Engineering
http://www.bristol.com/Papers/NTtoUNIXFast

Overview

This document is a quick look that the major issues and technology that can be used to quickly move Windows NT applications to Unix. For detailed information see http://www.bristol.com/Papers.

Terminology

- Win32 - Windows 95 and NT application programming interface (API)
- MFC - Microsoft Foundation Class library, C++ application framework
- OLE - Object linking and embedding - a complete object orientation system
- COM - Component object model - foundation for OLE
- DCOM - Distributed component model
- ActiveX - OLE’s new name, ActiveX extended for the Internet and more!

Wind/U for HP-UX

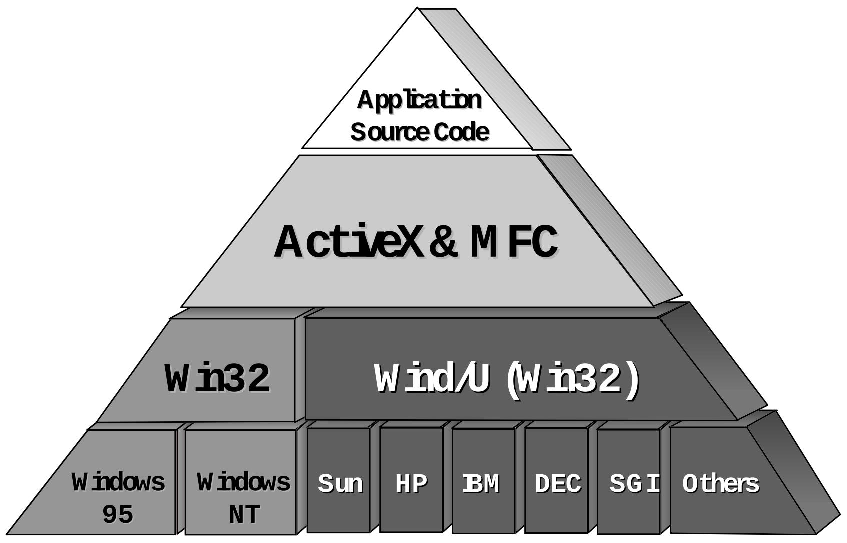
- Windows NT programming interfaces (Win32) for HP-UX
- Implemented using CLib, X, Motif and other Digital UNIX system interfaces
- Started shipping in 1993 on HP-UX
- Source level port - not binary emulation

Bristol Technology Partners

- Microsoft - "WISE" Partnership for Windows Source
- Digital - Affinity Program Packaging Partner. Wind/U runtime kit packaged on OpenVMS OS CDROMs
- IBM - Windows NT on OS/390 Partnership

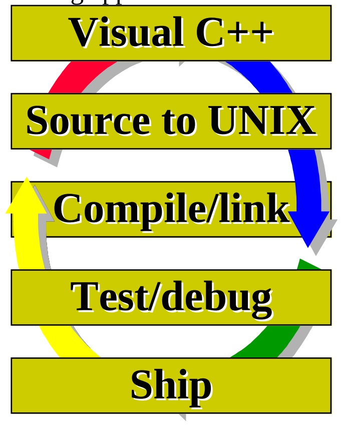
The Single-Source Code Solution

Wind/U enables you to maintain a single version of application source code for a variety of computing environments. Using a single source code base across platforms sharply reduces development and maintenance costs. It also greatly shortens development time—according to Bristol's customers, using Wind/U offers an average 80% time savings, compared with other development solutions. You can use this time and cost savings to focus on providing new solutions, rather than on reinventing existing solutions across multiple computer systems.



Cross-Platform Process

Porting applications from Windows to Unix consists of the following steps:



Preparing Win32 Code for Porting

The first step when porting Win32 code to other platforms is to make sure it compiles cleanly with Windows 95 or Windows NT compilers. Use the Windows compiler's highest warning level and enable strict ANSI C/C++ compatibility checking. Ensuring strict ANSI compliance in the original source code makes the compiling and debugging phases of the project much easier.
Next, copy the Win32 code to the target platform. Before compiling and linking, however, some minor portability changes need to be made. Wind/U provides a utility that processes all the source files, replacing Windows conventions with the target platform equivalents.

Compiling and Linking on Target Platforms

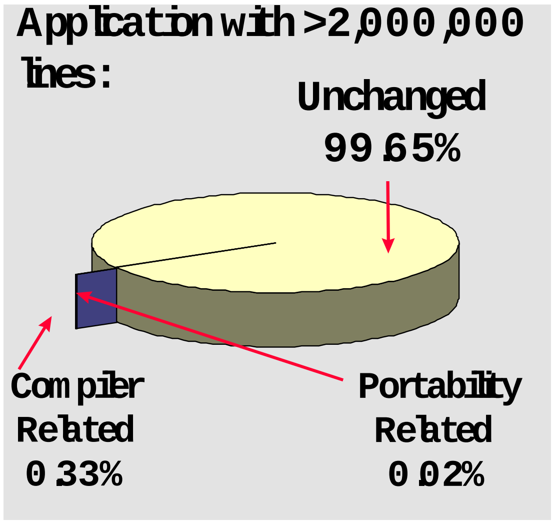
The next step is to create a makefile and build the application. Wind/U provides a utility that either converts a Visual C++ makefile to a UNIX, OpenVMS, or OS/390 makefile, or creates a new makefile for the application.
Building an application with the Wind/U makefile compiles the source files, compiles resource files with a UNIX version of the Windows resource compiler, and links the application to the Wind/U libraries. By default, the Wind/U generated makefile causes all objects and executables to be placed in an architecture specific directory.

Testing and Debugging

The most complex aspect of porting an application is testing and debugging on target platforms. Differences between compilers and architectures may expose problems in the application that were not important in the Windows environment. Similarly, differences between the graphics subsystems on different platforms may expose redundant API calls, unnecessary computations, and other wasteful practices that drastically impact performance on UNIX platforms, although the impact is hardly noticeable on Windows platforms. Therefore, it is critical that you thoroughly test all aspects of the application on all target platforms.
To assist with testing and debugging, Wind/U ships with a set of debug Wind/U libraries that help identify and resolve problems in your application. When an application is linked with the Wind/U debug libraries, Wind/U watches for any potential errors, such as incorrect handles, bad filenames, and invalid pointers.

Case Study Source Code Changes (>1,000,000 lines)

To see how portable Win32/MFC applications are to other platforms, let's examine a recent Wind/U porting project. This project is a large application with 2,400,000 lines of code, using the Windows API and MFC. This application use the latest Windows technology, including OLE, ActiveX, threads, shell APIs, and WinInet. Because this application was originally designed for Windows only, the original code was not written to maximize portability. Even so, only 0.38% of the lines required changes to run on other platforms.
What Changes Were Needed and Why?



Of the 8600 lines changed, 8020 (0.33% of the total lines of code) were the result of compiler differences on the various platforms. Although the Microsoft C++ compiler and the native C++ compilers supported by Wind/U all adhere to the ANSI C++ standard, there are some implementation differences that you may encounter when porting applications. In this application, changes were required because of differences in how the compilers handle the following features:

- C/C++ comments (0.06%)
- Pragmas (0.01%)
- Nameless unions (0.04%)
- #include case sensitivity(0.02%)
- vtbl padding (0.13%)
- C/C++ object references (0.08%)

Because some of these compiler issues, such as C/C++ comments, are very common, Wind/U provides a utility that scans your source code and automatically makes the required changes for you.

In addition to the compiler-related changes, this application also required 580 lines changed due to other portability issues. These changes (0.02% of the total lines of code) result from the following types of issues:

- Byte swapping
- Operating system differences
- Drive letter and pathname dependencies
- Runtime environment settings

Win32 - Windows API and Microsoft Foundation Classes

In cross-platform development, developers choose from a variety of strategies and technologies. However, the overwhelming popularity of the Microsoft Windows API makes it clear that most developers consider it the best development environment. Here are the reasons why:

MFC Framework

MFC is a complete framework of ready-to-use, compiler-independent application components that greatly improves engineering productivity for Windows 95, Windows NT, Macintosh System 7, UNIX, OS/390, and OpenVMS. Using these pre-built, reusable, and customizable C++ classes allows you to focus your resources on enhancing you application.

MFC offers over 120 reusable C++ classes, including general-purpose classes that support the non-graphical portion of an application, classes for the core Windows GUI features, architectural classes to help organize and structure application programming, and high-level abstractions and canned functionality that provide major building blocks.

Feature Rich API

Microsoft spends many millions of dollars adding advanced features to the Windows operating system, including complex graphic mapping modes, DLLs, OLE, sophisticated on-line help, a unified printing model, and advanced user interface components such as common controls and common dialogs. Wind/U provides support for all of these advanced features—something that you will not find with other cross-platform strategies.

Setting the Standard

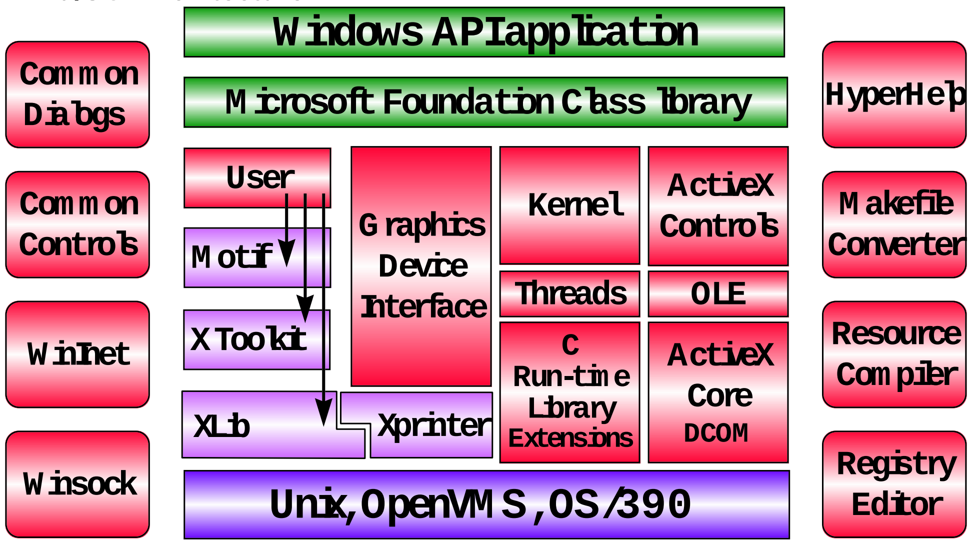
The Windows API is the de facto standard API for creating Windows applications. Wind/U lets companies use the Microsoft Windows API for all their software development, regardless of the computer system. Through Wind/U, the standard API on the world's most popular operating system, Microsoft Windows, is becoming the standard on UNIX, OpenVMS, and OS/390.

Training and Tools

Using the Windows API significantly reduces the cost of training associated with cross-platform development. First, there is a large base of available, experienced Windows developers. Second, Windows developers can access a wide range of readily available documentation and training from Microsoft and third party providers.

The enormous size of the Windows market has resulted in there being more software tools available for Windows API development than for any other platform. The intense competition in the Windows-based developer tools market has resulted in Windows being the most powerful and inexpensive development platform.

Wind/U® Architecture



GDI Layer - Display/Printer Graphics

- Logical Coordinate Mapping
- Printer Support
- Logical Font Mapping
- Simplified Painting Model
- Device-Independent Bitmaps
- GDI Performance
  - Attribute cache, color cache, dynamic font cache

Kernel Layer - Non-GUI Functions

- File I/O
  - CreateFile, DeleteFile, OpenFile, ...

- Memory Management
  - HeapCreate, HeapAlloc, LocalAlloc, ...
- Dynamic Link Libraries
  - LoadLibary, GetProcAddress, FreeLibrary
- Utility functions
  - Atoms, System Information, Profile, …
- Threads

User Layer/Window Management

- Creating, manipulating and destroying windows
- Translate Windows classes into equivalent Motif widgets
- Translate widget callbacks into Windows messages
- Translate X events into Windows messages

Windows Standard Controls to Motif Widget Mappings

- List box ==> XmList
  - Subclassed for tabs and owner drawn
- Scrollbar ==> XmScroll ==> XbScroll
  - Motif scrollbar enhanced
- Button (radio, check, etc.)
  - XmToggle, XmRadio
- Edit ==> XmText == > XbText
- Static ==> XmLabel
- Group box ==> XmFrame

Custom Wind/U Widgets

- Multiple Document Interface - MDI
  - Child windows, nested and clipped to parent window
  - Look and feel derived from Motif windows
- Combo box
  - Composed of 3 widgets, Edit, Button and List
  - Consistent look and feel to Motif 2.0 Combo box
- 3-state button

Common Controls

- Enhanced IE 3 Common Controls
- Coolbar, comboex, date, drag list box, header, hot-key, image list, list view, progress bars, rich edit, status bar, tab, tooltip, trackbars, tree-view, up-down

Win95 Common Dialogs

- File Open/Save As
- Find/Replace
- Printing/Printer Setup
- Choose Font
- Choose Color
- Runtime option to use Motif File Open dialog

Clipboard

- Interchange text data with X/Motif clipboard
- Interchange any data with other Wind/U apps
- Supports standard Windows clipboard types
  - text, bitmap, metafile
- Supports user defined clipboard formats

X Resources

- Allow end user to modify attributes
  - Foreground/background colors
  - Fonts
  - Window placement
  - System colors
- Supplements Microsoft Windows registry and .ini files

Debug Library

- Modeled after Windows debug libraries
  - Link time option, ARCH=xyz-g
- Runtime type checking
  - Invalid arguments, window handles, objects,
  - Over 150 detailed messages printed to stdout
- Built with compiler -g, debug symbols
  - Stack traces, break on wuDebugWarning()
- -SYNC for X-server synchronous mode

Portability Issues

Hardware Issues

- CPUs
- Intel, PowerPC, Alpha, HP-PA, SPARC, MIPS
- Word size, 32 or 64
- Byte ordering
- Little-endian (Intel) vs Big-endian
- Type alignment
- 1, 2, 4 or 8 bytes
- Structure packing
- Saving space vs performance

Operating System Issues

- Several choices
- Windows, Windows 95, Windows NT, UNIX, OpenVMS, OS/390
- File systems
- Various syntax and length restrictions
- Feature set
- Threads, memory mapped files, preemptive multitasking
- Portability Guidelines

Follow these guidelines when writing your Win32/MFC application to make the porting process as easy as possible:

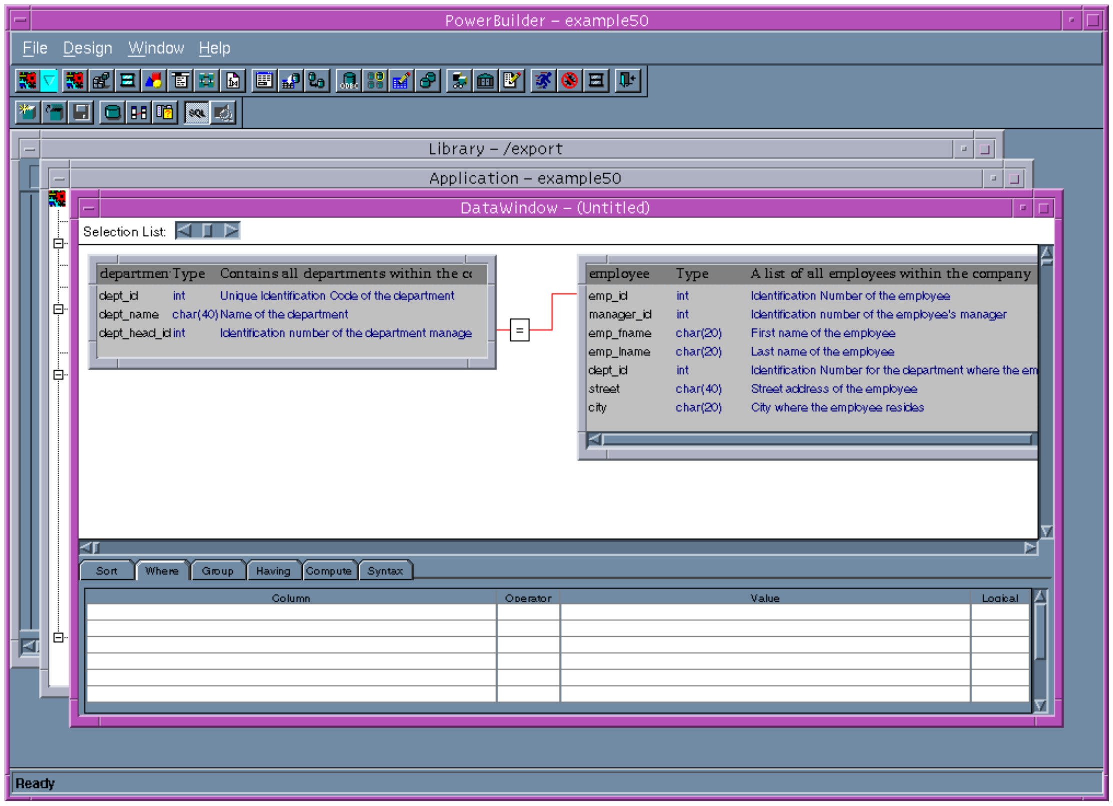
1. Use MFC to create the application. Since MFC provides an additional level of abstraction, it is typically much more portable than writing directly to the Win32 API.
2. Make sure the code is ANSI compliant. In Visual C++, use the /Za compiler option to enable ANSI compliance checking. Although there may be some differences in the way various compilers implement the standard, any ANSI compliant source code should theoretically be completely portable.
3. Make sure that “include” files are the same case as the filename and that functions are defined and used with exactly the same case, since UNIX is case-sensitive.
4. Always use the fields of a structure or class instead of making assumptions about their location or size in a structure. Different machine architectures store data in different order and use different primitive type sizes, so any assumptions about storage order and alignment that work on one architecture may not work on a different platform.

5. Use the HIBYTE, LOBYTE, HIWORD, and LOWORD macros instead of making assumptions about byte ordering. Some processors place the least significant byte first, while others place the most significant byte first.
6. Always use the sizeof operator to obtain the size of data to be read or written. Differing data type sizes or alignment schemes may alter the size of a given structure.
7. Design the product so that individual subsystems can be isolated for unit testing prior to full integration testing of the entire application.
8. Verify availability of any third party components across all target platforms prior to incorporating them in the application.
9. Make sure that the code to be ported compiles cleanly under Windows. With Visual C++, use the /W4 or /W3 and /WX compiler options to enable the highest warnings and to treat warnings as errors.
10. Do not use undocumented functions.

Wind/U Customer Applications

PowerSoft

Powersoft's award-winning PowerBuilder 4.0 made its debut as the leading cross-platform client/server development tool by releasing both UNIX and Macintosh versions based on their original Windows source base. Wind/U provides Powersoft with rapid application redeployment on various UNIX platforms, while maintaining the advanced Microsoft Windows functionality that PowerBuilder exploits.



departmen	Type	Contains all departments within the c
dept_id	int	Unique Identification Code of the department
dept_name	char(40)	Name of the department
dept_head_id	int	Identification number of the department manage

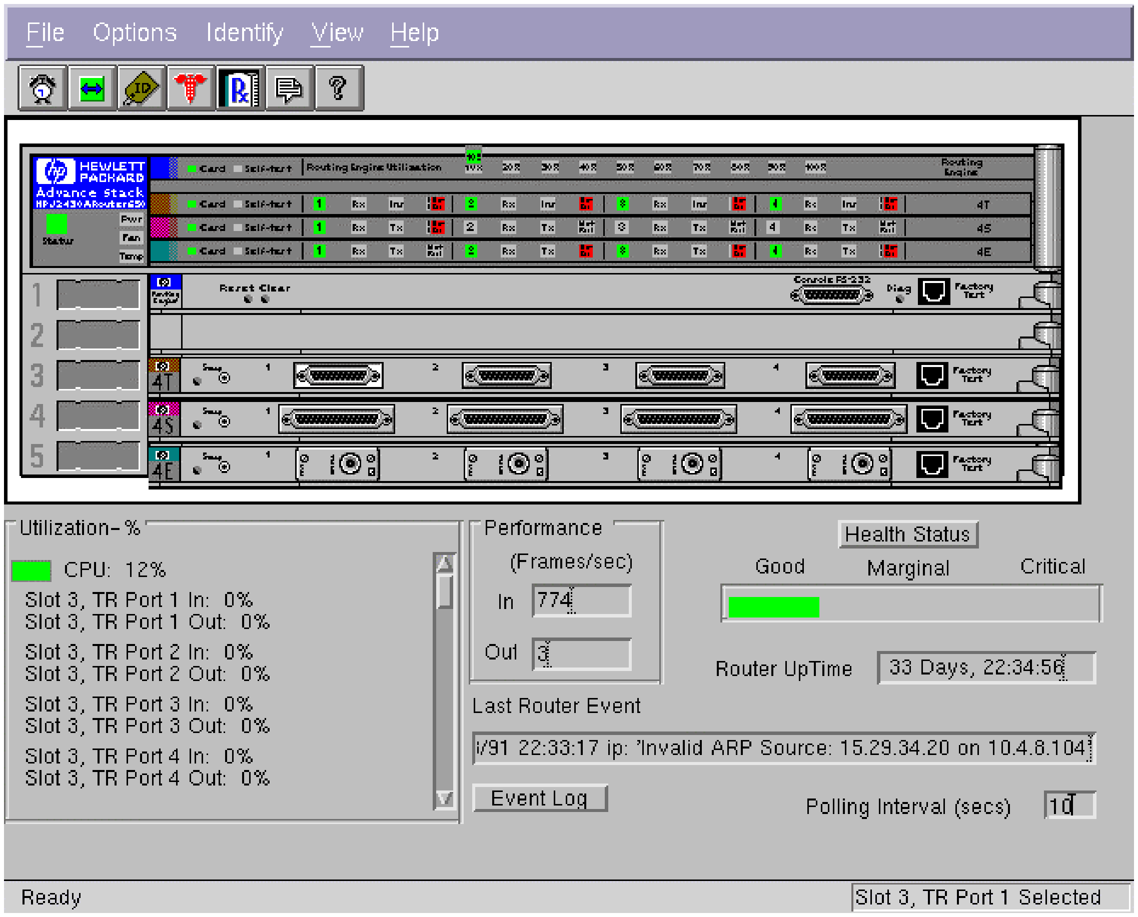
employee	Type	A list of all employees within the company
emp_id	int	Identification Number of the employee
manager_id	int	Identification number of the employee's manager
emp_fname	char(20)	First name of the employee
emp_lname	char(20)	Last name of the employee
dept_id	int	Identification Number for the department where the em
street	char(40)	Street address of the employee
city	char(20)	City where the employee resides

Powersoft uses Wind/U to provide its customers with benefits such as integrated enterprise-wide computing and application consistency across multiple operating systems.

Hewlett-Packard

As one of several HP applications ported with Wind/U, Hewlett-Packard’s Router Manager is a strong addition to its latest line of advanced network configuration and monitoring tools. A quote from HP: “HP required an increase in productivity; we've ported thousands of lines of Windows code, consisting

of SNMP networking and interprocess communications with minimal source code changes. Rewriting this code from scratch was unthinkable."
This Hewlett-Packard group began its relationship with Bristol when they integrated an impressive HyperHelp system into a native UNIX/Motif application. They cite technical support responsiveness as a deciding factor in their expanded use of Bristol software products.



Utilization-%

CPU: 12%
Slot 3, TR Port 1 In: 0%
Slot 3, TR Port 1 Out: 0%
Slot 3, TR Port 2 In: 0%
Slot 3, TR Port 2 Out: 0%
Slot 3, TR Port 3 In: 0%
Slot 3, TR Port 3 Out: 0%
Slot 3, TR Port 4 In: 0%
Slot 3, TR Port 4 Out: 0%

Performance (Frames/sec)
In 774
Out 3

Router UpTime 33 Days, 22:34:56

Last Router Event
i/91 22:33:17 ip: 'Invalid ARP Source: 15.29.34.20 on 10.4.8.104

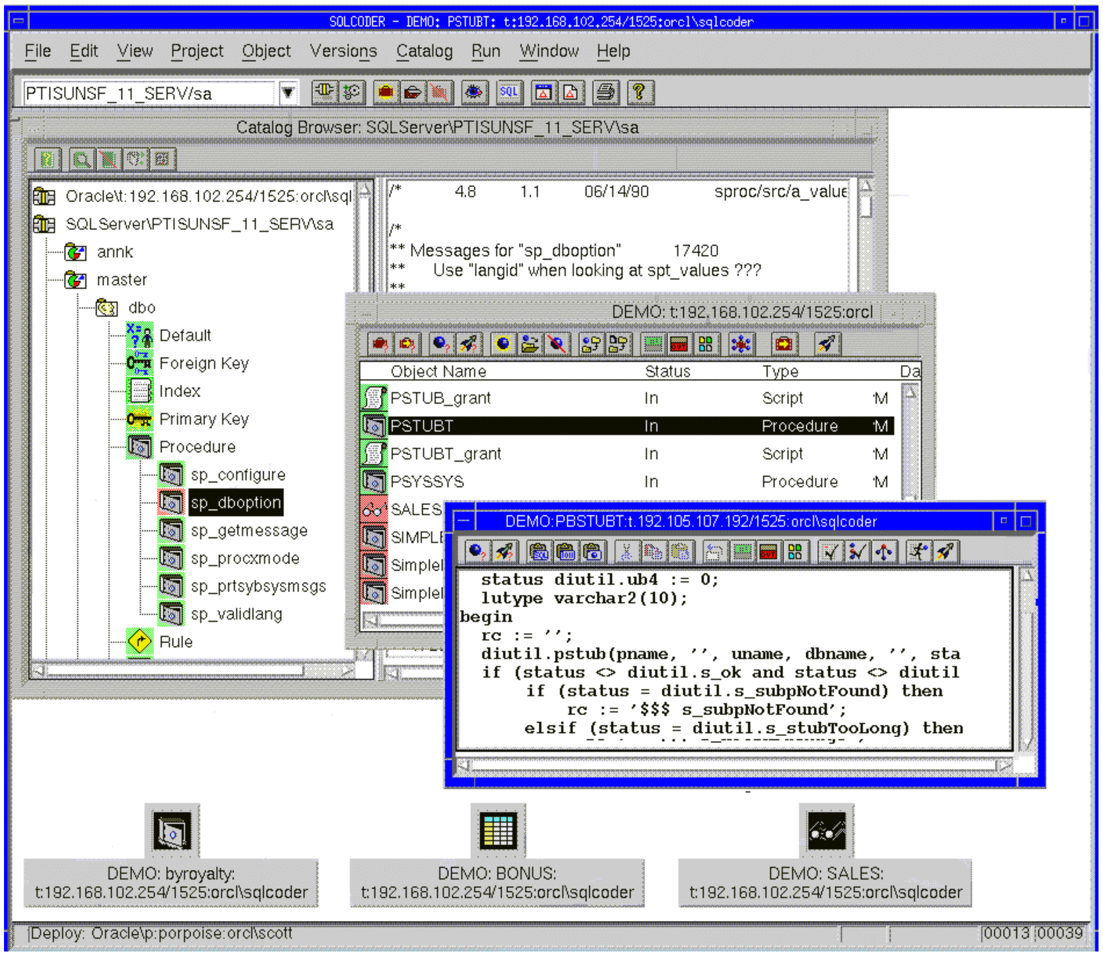
Polling Interval (secs) 10

Ready
Slot 3, TR Port 1 Selected

Hewlett-Packard uses Wind/U to provide cross-platform versions of networking products from a single source code base.

Platinum Technology

Platinum Technology's SQL-Coder is one of many Platinum applications that use Wind/U. It provides a comprehensive development and management environment for creating, editing, executing, managing, and deploying server-side SQL code across the open enterprise. Product features include a robust query facility, graphical catalog browser, reverse engineering of SQL objects, and project metaphor. Additional features are enhanced control of SQL code with revision control, check in/out and deployment from a central repository. The intuitive Motif GUI provides a friendly server code editing and management environment. SQL-Coder is the only feature rich SQL development tool that supports more than one RDBMS server type.


SQLCODER - DEMO: PSTUBT: t:192.168.102.254/1525:orcl\sqlcoder
File
Edit
View
Project
Object
Versions
Catalog
Run
Window
Help
PTISUNSF_11_SERV/sa
Catalog Browser: SQLServer\PTISUNSF_11_SERV\sa
Oracle\t:192.168.102.254/1525:orcl\sql
SQLServer\PTISUNSF_11_SERV\sa
annk
master
dbo
Default
Foreign Key
Index
Primary Key
Procedure
sp_configure
sp_dboption
sp_getmessage
sp_procxmode
sp_prtsybsysmsgs
sp_validlang
Rule
DEMO: t:192.168.102.254/1525:orcl
DEMO:PBSTUBT:t.192.105.107.192/1525:orcl\sqlcoder
DEMO: byroyalty: t:192.168.102.254/1525:orcl\sqlcoder
DEMO: BONUS: t:192.168.102.254/1525:orcl\sqlcoder
DEMO: SALES: t:192.168.102.254/1525:orcl\sqlcoder
Deploy: Oracle\p:porpoise:orcl\scott
00013
00039

```
/*      4.8     1.1     06/14/90        sproc/src/a_value

/*
** Messages for "sp_dboption"       17420
**      Use "langid" when looking at spt_values ???
**
```

Object Name	Status	Type	Da
PSTUB_grant	In	Script	M
PSTUBT	In	Procedure	M
PSTUBT_grant	In	Script	M
PSYSSYS	In	Procedure	M
SALES			
SIMPL			
Simple			
Simple			

```
  status diutil.ub4 := 0;
  lutype varchar2(10);
begin
  rc := '';
  diutil.pstub(pname, '', uname, dbname, '', sta
  if (status <> diutil.s_ok and status <> diutil
      if (status = diutil.s_subpNotFound) then
          rc := '$$$ s_subpNotFound';
      elsif (status = diutil.s_stubTooLong) then
```

Platinum Technology uses Wind/U to deliver Windows functionality such as common controls on UNIX platforms.