$2.50

MUUGlines

December 1996 Volume 9, Number 4

Special Christmas Event! Free Stuff!

Yes, it's time for MUUG's annual Christmas extravaganza, and this year we're getting into the gift-giving excitement early. Come to our meeting to be elegible to win fabulous prizes! Attending members (applications will be acceptedduntil the draw!) will be entered in the draw for three great Linux goodies: ApplixWare, an office suite valued at approximately $400; the Linux Bible, 4th edition; and a one year free subscription to the Linux Journal.

In addition, non-members as well as members will be entered in the door prize draw. Among the items are the Linux Internet Archive (an 8 CD set of a vast array of Linux software and related documentation!) and the Linux Bible, 3rd edition. The highlight of the evening (yes, there's more!) will be a demonstration of ApplixWare by Red Hat Software. Providing the demo will be MUUG board member Arne Grimstrup.

Don't miss the other Christmas program this month: An Obfuscated C contest winner to put you in the festive spirit. This must be seen to be believed. Find it at http://www.muug.mb.ca.

Where To Go

Our fourth meeting of this year (and Christmas spectacular) will be at IBM Canada's offices in the TD Centre building at the corner of Portage and Main. We'll be meeting at the lobby on the main floor, and Steve Moffat will take us up to the meeting room just before the meeting starts.

This month's meeting is on December 10th at 7:30 PM. Please arrive a little early for the meeting, as it will take some time for Steve to get people up to the meeting room.

Parking is available either in the parkade behind the TD building (off Albert St.), or in the ground level lot just north of the TD building. Entrance to the lot is from Albert Street, behind the parkade. Either way, parking is $1.25 flat rate for the evening.

PC WEEK: Up Periscope

Linux: Microsoft's real competition? October 7, 1996

Oh yeah, sure. A freeware operating system knocks off Windows NT? If IBM, with all its might and 11 million OS/2 users, can't get more than a footnote in the record books, what chance does Linux have? Admittedly, not much. Still, the Redmondians could learn a thing or two from the Linux phenomenon.

Considering that in 1991 Linux was something between a hobby and an intellectual exercise for Linus Torvalds, a computer science student in Helsinki, Finland, Linux has made enormous progress. When Torvalds started, he didn't know much about the X86 architecture, but he got up to speed quickly, publishing his code on the Internet as he went. His project captured the imaginations of programmers all over the world, who pitched in with code, drivers, testing and documentation. Today, only about half of the kernel is Torvalds' code, but he still is the guiding light for Linux development. He also crafted the copyright and license agreement that allows free distribution (although packagers can charge for their value added and support).

The key to Linux's development is that Torvalds was utterly straightforward when he showed his code to the world and admitted, "I don't know what I'm doing." As Torvalds said in a recent interview, "No question about it. Without Net access, the project would never have even gotten off the ground." Experts pitched in and created the best Unix for the Intel platform, even while making it less machine-dependent and more portable. "The SCSI drivers, the networking code and the new floating-point emulator code are completely written by others," says Torvalds.

Can you imagine Microsoft, IBM or Sun doing that? Neither can I, although Sun's handling of Java debugging and security issues comes close.

This spirit of cooperation has yielded some surprising benefits for Linux users. Linux became the first server platform with a workable defense for SYN attacks, the hacker's trick of tying up a server by sending a continuous barrage of packets. Linux International (http://www.li.org) now serves the needs of users and provide information and links to software and support, and it has the backing of big (and small) companies. Linux is rapidly becoming one of the most popular Unix variants, especially on Web servers. It's robust, fast and capable. Recent compatibility enhancements let you simply recompile most Unix programs for Linux.

Caldera has announced an application suite for Linux, and forthcoming Java applications won't care what platform is running under them, so Linux is becoming more feasible for the desktop, too.

Still, there remains the question of Linux's goals. "I hate to admit it," Torvalds says, "but Linux development has never had any real well-defined goals. ... Features have been added when somebody has been interested enough to write the code — and I've felt the result was worthy."

If you venture into Linux land, you'll have to learn how to pronounce it. Most people say "LINE-ux;" Torvalds pronounces his name "LEEN-oos" and pronounces his baby's name "LEEN-ux." So you can one-up everybody and pronounce it the way he does. It's quite continental.

Bill Machrone is vice president of technology for Ziff-Davis Publishing Co. He can be reached at bill_machrone@zd.com.

UnixWorld Online: Review No. 005

PowerBuilder for Unix

By Raghuram Bala.

Questions regarding this article should be directed to the author at rbala@i-2000.com.

PowerBuilder from Powersoft owns about 50% of the market in the client-server tool market on Microsoft Windows. It has gained mass appeal with an easy-to-use scripting language, high degree of object orientedness, tight database integration, and support for team development. Now Powersoft has released Unix and Macintosh versions of its flagship product for cross-platform development.

Will it draw converts in the Unix arena? Who is Powersoft targeting? Who competes with it and how does PowerBuilder stack up? Answers to these questions and more below...

PB's Roots

The move to client-server computing began in the late 1980s, and what started as a trickle has turned into a flood with the vast majority of corporations right sizing their computer systems. The days of proprietary systems were replaced by open systems bearing industry standard hardware and software. Among the catalysts for this paradigm shift were cheaper personal computers and workstations, and increasingly computing power on the desktop as a result of leaps in microprocessor technology.

In the business computing arena, the Windows operating system from Microsoft captured the giant's share of the market leaving OS/2, DOS, Unix and others to share among the spoils. Graphical User Interfaces (GUIs) have been around since the late 1970s resulting from pioneering research from Xerox's PARC Labs and eventually gaining commercial appeal in the form of Apple's Macintosh line of computers. However, it was Microsoft that helped bring GUIs to the masses aided by low cost PCs bundled with the Windows environment.

The native language of Windows is the Windows API, and early development on Windows was limited to C-language programs with API calls. As many discovered soon, it was an onerous and arduous task to develop even simple programs using this technique. In addition, many corporate entities had bigger needs like accessing large databases from their desktops, creating reports, sharing data between applications and so on. Hence, the need arose for industrial strength tools to mask the complexity of the Windows API, and to provide powerful mechanisms such as tight integration with relational databases, transaction management, and configuration management for team development. These were features that mainframe users had come to expect and PCs had to catch up or lose their appeal. A wave of client-server tools on PCs emerged in 1992 that included Visual Basic from Microsoft, SQLWindows from Gupta, and PowerBuilder from Powersoft.

All these products use proprietary scripting languages that are easy to learn and have powerful capabilities suited for programmers of business applications in enterprises.

Two-Tier Client-Server

In the last four years, PowerBuilder, Visual Basic, and SQLWindows together with new entrant Delphi from Borland International have dominated the client-server tools arena on Windows. In the same period of time, many have discovered the strengths and weaknesses of the client-server architecture as a whole, and likewise the limitations of the above mentioned products. All of these tools thrived in two-tier client-server architectures. In the purists' viewpoint, client-server refers to two pieces of software, namely the client, which makes a request that is handled by a another piece of software on the server. In most implementations in corporate America however, client-server has translated to a GUI program that handles presentation and application logic "talking" to a database server. This architecture is also known as the "fat" client architecture.

This architecture has its weaknesses on PC platforms, namely:

Scalability It is not very scalable because the PC platforms were not as powerful as say, the Unix workstations, because their computing power was more limited. In addition, PowerBuilder, Delphi, Visual Basic, and SQLWindows were tools that ran only on Microsoft Windows. **Robustness** The Windows environment as a whole was not very robust and even till today does not exhibit full pre-emptive multitasking (with the exception of Windows NT).**Performance** Because the GUI tools had to deal not only with the presentation layer but also the application logic, the front-end performed tasks such as database transaction management. This resulted in poor performance, network bottlenecks, and other maladies. Performance boosts have been achieved to a certain degree by use of stored procedures that reside on databases. However, because stored procedure languages are non-standard there is little appeal for them as long-term strategic choices.**Cross Platform Development** Because application logic was embedded in the GUI front-end, re-use of code by in-house application development was nearly impossible when it came to cross-platform application development.

N-Tier Client-Server

With two-tier client-server computing hitting a performance and scalability wall, many of the tool vendors quickly realized the need for scalability in their products. This led to N-tier client-server architectures where the application logic is no longer bundled in with the presentation layer, but instead is kept separate. The communication between the presentation layer and application layer is through remote procedure calls or mes-

saging. This architecture has the advantage that one could leverage the best presentation tool without affecting the application logic. A case in point would be the sudden popularity of the World Wide Web as a presentation layer. Companies using the N-Tier architecture could now use Web browsers as their presentation layer and hook into existing application logic easily without major reinvestment.

PB Strategy

Powersoft's focus has always been on corporations and the need to build client-server applications rapidly with solid database integration. It is not a tool aimed at research laboratories, academic institutions, or other sectors of the market. That does not mean that it does not have the capability to build non-database applications.

With N-Tier client server gaining immense popularity, Powersoft has attacked the problem in two ways:

Going Cross-Platform This enables PowerBuilder code modules to be built in Windows 3.1, Windows 95, Windows NT, MacOS, and Solaris 2.4.**Code Distribution and Objects** With PowerBuilder 5.0, one can now write PowerBuilder code for the presentation layer and also write the application logic in PowerBuilder on the application server. For example, a company could deploy PCs for presentation and have PowerBuilder run there, and deploy Sun SPARCs with Solaris 2.4 as application servers and have PowerBuilder on that machine. PowerBuilder on the client would call PowerBuilder objects on the application server either through RPCs or message-oriented mechanisms that comply with CORBA standards.

With the Unix client-server market, PowerBuilder faces off with Unify Vision from Unify Corporation, Progress ADE from Progress, JAM from JYACC, and Elements Environment from Neuron Data. Although these tool vendors have been around for a number of years, they have not enjoyed the media coverage, popularity, and industry support that PowerBuilder has garnered in the last few years. This is not to say that their products are inferior, but the market usually follows the market leaders.

With Sybase acquiring Powersoft in 1994, both companies benefited from the strengths of the other. The Powersoft acquisition made a Sybase a "complete" client-server company providing client application development tools as well as back- end database server tools. Powersoft, on the other hand, gained Sybase's marketing muscle, distribution channels, and a company that has a solid reputation for technical excellence. Also, on Unix platforms, Sybase's database products are extremely strong which aids Powersoft's move to Unix.

What about conventional X Toolkits such as Xlib, Xintrinsics, Motif, and others? When coding for sheer performance, the X toolkits would definitely beat PowerBuilder or any other similar tool. However, in business applications, development cycles tend to be short and rapid application development with tight database support is a must. In this scenario, tools like PowerBuilder do very well.

Powersoft's approach on the Unix platform is three-fold:

1.If any of their existing clients wanted to migrate from PC clients to Unix workstations, then PowerBuilder for Unix would be the answer as its code is almost fully compatible with the PC version. The only minor changes are commands relating to DDE, OLE, and other features found only under Windows. 2.If existing PowerBuilder users had many client platforms, then all of their development could be done using PowerBuilder. This scenario occurs in large corporations that have Unix workstations, PCs, and Macs. 3.Looking to the future, N-Tier client server needs to be supported. With Distributed PowerBuilder, a feature in Version 5.0, an enterprise can build stand-alone objects on application servers using PowerBuilder. So existing clients can migrate their application logic "stuck" in two-tier architectures to an N-Tier architecture without recoding in another language, for instance, C or C++.

PB Architecture

PowerBuilder's application architecture consists of seven classes:

Application Every application has exactly one application class associated with it. An application object is instantiated during run time. The application class has attributes such as the name of the application, the fonts to use, the libraries to use, and so on. Window The window is the focal point in any GUI application. Although, not absolutely necessary for every application, most GUI applications have at least have one window. Microsoft Windows contain controls such as static text, listboxes, drop-down listboxes, checkboxes, radio buttons, and others. One unique control to PowerBuilder is the data window that is a smart data-aware control with links to databases. Every data-window control is linked to a Datawindow class. Data-window controls have a rich set of methods for manipulating data. **Menu** Menus are associated with windows and cannot stand on their own right. Every menu item can also be associated with a toolbar bitmap. In this sense, developers can get a toolbar for "free" as part of menus. **Datawindow** This is a class that is unique to PowerBuilder. Datawindows are input-output mechanisms for data stored in data sources. These data sources could be flat files or databases, among others. Datawindows have excellent graphing capabilities based on data retrieved from data sources. Datawindows have a number of presentation styles including grid, tabular, freeform, crosstab, and many more. Datawindow controls in windows are associated with a Datawindow class. **Structure** These classes are similar to structures in the C language and records in Pascal. Function These are global functions similar to functions in any other language. They can take arguments and can have return

values. **User Objects** User objects follow the "parts" or "component" model of programming where pre-fabricated code modules can be easily plugged into a window and used. There are many types of user objects:

Class User Object These are non-visual user objects that are akin to C++ classes with attributes and methods. C++ This allows code written in C++ to be closely integrated with PowerBuilder. **Simple Visual User Objects** These are simple controls, such as radio buttons or command buttons, that can be specialized and "componentized." For example, you can make a "close" button that can be used in several windows. **Complex User Objects** This is a whole set of components that are "componentized." These complex user objects can also have encapsulated methods and attributes making them "objects."

Object-Oriented Design

PowerBuilder supports object-oriented programming by providing mechanisms for encapsulation, inheritance, polymorphism, and dynamic binding:

Encapsulation Windows, menus, user objects, and application classes support user-defined attribute and method creation. There are shared attributes and instance attributes. Shared attributes are like static attributes in C++ classes where the storage allocated for that attribute is shared among all objects of that class. Instance attributes are "regular" attributes for which storage is allocated separately for each instance of that class. Methods can be created within windows, menus, user objects, and applications. Both attributes and methods have access- modifiers, such as private, public, and protected similar to C++. **Inheritance** Class inheritance is supported for windows, user objects, and menus. Inheritance speeds up development by allowing common traits among objects to be preserved in an ancestor class and specializations of the ancestor reserved for descendants. **Polymorphism** In windows, menus, and user objects, the methods that are declared can be polymorphic. First, the descendant methods can have different implementations than their ancestors. Second, the methods in descendants can override their ancestor methods by taking different number and types of parameters.

Function overloading is also supported in that more than one function with the same name can exist within a class. For example, we could have both:

```
void foo(char a)
int foo(int a, double b)
```

Dynamic Binding Also known as late binding, dynamic binding enables decisions like which method would be called for an object during run time instead of compile time.

Weaknesses of PowerBuilder

One of the weaknesses of PowerBuilder is that the development environment is controlled by the tool. For example, if you inherited Window B from Window A, and later realize that you need to inherit Window C from Window A and Window B from Window C, there is no simple way to accomplish this inheritance. In a true object-oriented language, such as C++, one would easily do this by changing the source code. In PowerBuilder, inheritance is achieved via the development environment and one actually does not directly interact with the code that is generated. In addition, the code that is generated does not have a published grammar. This prevents one from building code generators and conversion utilities to and from PowerBuilder. For example, if you wanted to convert a PowerBuilder screen to an HTML Web Page there is no simple method because there is no PowerBuilder grammar whereas HTML is a published standard.

Hence, rapid application development without the ability to manipulate the underlying code and development environment directly has severe limitations at times. The fact that Powerscript, the language of PowerBuilder, is a 4GL and non-standard has drawbacks in terms of performance and limiting its appeal to a wider audience.

Conclusion

PowerBuilder for Unix is a solid product and should capture a significant market share on Unix for business application development. Competition in the Unix marketplace stacks up well against PowerBuilder in technology. However, the marketing muscle of Sybase and Powersoft and the fact that PowerBuilder's ease of use, object-oriented programming paradigm, and rapid application development capabilities would be hard to overcome.

Contact Information

To obtain more information on PowerBuilder for Unix, you can contact:

Powersoft Corporation 561 Virginia Road Concord, MA, 01742-2727, USA General Info: (508) 287-1500 Sales: (800) 395-3525 Web site: http://www.powersoft.com/

Edited by Becca Thomas, Online Editor, UnixWorld Online, editor@unixworld.com

Contact Information

To contact the MUUG board for membership information or anything else, send e-mail to board@muug.mb.ca. We have a Web presence as well, at http://www.muug.mb.ca/, where you can find all kinds of information, including details of upcoming and past meetings and presentations and references related to them.

To contact the newsletter editor (and I know you want to shower him with dozens of well-written article submissions), e-mail editor@muug.mb.ca.