PORTING AND DEVELOPING APPLICATIONS

Raj Rajagopal

BUSINESS PORTING CONSIDERATIONS

If one has decided to port or rewrite one or more applications to another environment, then there are some business and technical considerations that need to be addressed. Porting considerations are addressed in this section. Rewriting applications is addressed later in this article.

Business Porting Considerations

There are some business benefits that can be derived by porting:

- The largest benefit of porting is that it permits a stepping stone approach to a potentially complex transition.
- Porting permits system administrators and their custom utilities to transition immediately and be productive in the new environment.
- Porting permits continuous use of development and maintenance skills while transitioning to the new environment.
- Depending on the applications, one may be able to get the applications working in the new environment with little effort.
- End users can start using existing applications in the new environment without having to relearn the applications.
- Unique features can be available in the new environment as a post-porting step, but this may introduce divergence in the source between sources in the old and new environments.

PAYOFF IDEA

Porting applications between environments can be accomplished using tools or manually changing the code. Article 50-20-41 covered operating system differences important to manual porting. Alternatively, users can use a porting tool that lets them port applications from one environment to another. Porting tools simulate the execution environment for which the application is the target environment.
introduce divergence in the source between sources in the old and new environments.

Business applications may have been developed from scratch, purchased as a package, or purchased as a package and modifications made to fit an individual environment. If a package was purchased, it will be necessary to check with the package vendor regarding a version that runs in the specific environment. Also check if the vendor offers porting assistance, particularly for the custom modifications that may have been made. An in-house application would be an appropriate porting candidate under the following conditions:

- The application has a long, useful expected life.
- The current development and maintenance programmer skill base dealing with the application would require a large learning curve for a new environment.
- The benefits of making the application work in the new environment outweigh leaving the application as is or rewriting it.
- The application size is large enough to gain some benefits when ported, but not too large that would make porting a long, error-prone, and risky process.

If a manager is wondering how serious a porting problem exists, then check code against the following list, making it easier to port:

- code in which end-user interface handling and business logic handling are isolated
- code in which vendor extensions to standards are isolated
- code in which dependencies on a specific operating system are isolated (particularly true for UNIX with its many flavors)
- overall program flow is separated into logical, modular blocks

The more code satisfies the above points, the better. However, regardless of how good the code is, expect some code changes. Be prepared to allocate time and resources for making source code changes when porting to another environment.

Keep in mind that sometimes an individual may be able to port and make the application work in a limited fashion in the new environment; but to take full advantage of the new environment, one may need to modify the code. For example, consider a UNIX program performing file access that is being ported to Windows NT. If the program uses standard C functions such as fopen, the program will still work when ported to Windows NT, although it may be unable to take advantage of advanced features like asynchronous I/O unless the fopen is replaced by an equivalent Win32 API call.
Technical Porting Considerations

Besides the application code itself, there are a few other items that need to be addressed when an application is ported:

• Any shell scripts that are used by the application either during development or execution need to be ported or rewritten.
• UNIX has built-in commands used in application development. If using tools to port code, tool vendors provide most of the commands on Windows NT. If manually porting code, one needs to write or get the equivalent tools or needs to find the Windows NT equivalent of the tools’ function.
• UNIX uses a number of utilities for application development, as well as system administration. Equivalent utilities in the new environment have to be used or the utilities need to be ported or rewritten. Many porting tool vendors include UNIX utilities equivalents with their products. These are covered with the tools later in this article. There are also some public domain utilities. Most of these are available from public ftp sites such as ftp.iastate.edu, ftp.cica.indiana.edu, sunsite.unc.edu, and ftp.uu.net. These include ci, ident, perl, rcsdiff, tic, co, lex, rcs, rcsmerge, yacc, cpp, merge, rcscheck, and rlog.
• Any documentation that may be available, such as help files and online manuals, may need to be ported.
• Testing scripts may also need to be ported that have been set up to perform automated and other testing.
• Daemons and other stand-alone utilities that perform some applications functions in the background need to be ported.
• UNIX shared libraries used by programs must be ported.
• Web-related unique application code using languages, such as PERL, should be ported.

Exhibit 1 summarizes some common UNIX types of UNIX programs and the equivalent type in Windows NT.

Following are the steps involved in porting application source.
Porting Steps

The following steps are involved in porting an application:

- Copy source code from the UNIX machine to the Windows NT machine (or, in some cases, Windows 95). There are a number of different ways to transfer the source and accessory files (these are covered later in this article).
- Copy compiling accessories to the source such as makefiles, resource files, and scripts.
- Make any changes required to the accessory files. Makefile changes may be required. Porting scripts is discussed later in this article.
- One can add Windows-specific code, such as Win32 calls, to take advantage of functions not available in UNIX.
- Compile the ported source into object modules using the appropriate language compiler in the new environment.
- Compile required accessories, such as resource files.
- Link the object modules with appropriate Windows and tool libraries.
- Test the ported application; identify problem areas in the port, if any.
- Repeat steps 3 through 8 until the ported application works satisfactorily.

Keep in mind that these steps are just a high-level overview. A number of details have to be addressed at each stage. While the major steps outlined above are applicable to any porting tool selected, the details will vary depending on the tool chosen.

Transferring Files from UNIX to Windows NT

Source and accessory files can be transferred from the UNIX machine to a Windows NT machine in a number of different ways:

- use floppy disks to copy
- use the built-in file transfer protocol (ftp) function in Windows to transfer files between the UNIX server and the Windows machine
- use built-in tftp function in Windows NT
- use the built-in rcp function in Windows NT
- use a serial link
- use a network operating system with built-in support to access UNIX server files
- use third-party network file system (NFS) packages

Porting C Programs

First considered are C programs without database or transaction processing calls. At first glance, it may be presumed that if the C program follows
C coding standards, it should compile and execute easily in the other environment. This is seldom the case. There are a number of reasons beyond standards for this:

- Path names are pretty standard in the UNIX world — header files are under /sys, binary executables are under /bin, etc. In Windows NT, the user selects most path names.
- Path names are passed as string arguments in many instances. UNIX supports only the forward slash (/), while Windows NT supports both forward and reverse slashes (except for the command prompt).
- UNIX is case sensitive, while NT is, for the most part, not case sensitive. (NTFS preserves case for directory listings. If viewing files in File Manager, for example, the case used is the same as when the file was created.)
- It is not uncommon for UNIX applications to hard code directory paths. For example, the code may be looking for a file in /usr/local/bin, presuming a standard UNIX environment. The ported code will almost certainly not be able to find the file in that path and thus fail. These paths have to be identified and taken care of before the ported application would work in the new environment.
- Vendor extension to standards — where there is a rule, there is an exception. Where there is a standard, there is a vendor extension. In most cases, it is difficult not to use the vendor extension, because the functionality of the standard by itself is limited. The “wrapper code” may provide support for the standard, but may not provide for all vendor extensions from different vendors.

Porting Scripts
Scripts are utilities written in a scripting language that are part of an application in the same manner as programs written in a language like C. Programmers and system administrators typically use scripts to automate routine, repetitive tasks. UNIX operating systems include a shell which supports scripts, as does the Win 32 shell. Vendors providing support for porting code provide tools to create or port UNIX scripts to Windows NT. These scripts can run the ported UNIX programs, as well as native Win 32 programs.

Porting Applications from UNIX to Windows
Tool-based porting is considered here, including a brief look at some vendor tools. Manual porting is addressed later in this article.

COMMON PORTING PRODUCTS
This section presents an overview of some common porting products available. The information was gathered from different published sourc-
es, including vendors’ online Web pages. The tools are continuously being updated; check with the tool vendor for the latest information. Most vendors also offer free evaluations which are downloadable from their Web sites.

This article is not intended to help select one vendor over another. Migration/coexistence solutions should be selected that meet individual business requirements. Do perform an evaluation of vendor products. The inclusion of vendor information is provided to give an idea of the capabilities of the products and a quick reference for further information and software.

Common products that help in porting application from Windows to UNIX include NuTCRACKER from DataFocus Software and Portage from Consensys Computers.

**NuTCRACKER**

Information on NuTCRACKER is available online at http://www.datafocus.com.
NuTCRACKER is a product family for developers who want to port character-based and X/Motif UNIX applications to Windows NT or Windows 95.

Using NuTCRACKER, developers can recompile UNIX C, C++, or Fortran source code and link it to NuTCRACKER DLLs, resulting in native Win32 applications. A schematic of this process is shown in Exhibit 2.

NuTCRACKER supports Intel, Alpha, MIPS, and PowerPC platforms. If source exists on any of those platforms, one can get Windows NT for that platform along with the corresponding NuTCRACKER version and port the source. Please note that IBM recently decided not to support Windows NT on the PowerPC.

The NuTCRACKER Product Family

The NuTCRACKER product family currently consists of three products:

1. NuTCRACKER SDK: NuTCRACKER Software Development Kit (SDK) for porting character-based and daemon applications. It consists of the application program interface (API) (which includes system calls and library routines) and a comprehensive set of development utilities. NuTCRACKER SDK contains the user interface options, including curses and client/server-designed Win32 enhancements, for applications ported using the SDK.

2. NuTCRACKER: X/Software Development Kit (X/SDK) — provides the porting environment needed if the application is designed with an X/Motif user interface. If porting with the X/SDK, one may need to purchase NuTCRACKER X/Operating Environment (described below) for distribution with end-user applications. In addition to X11R5 and Motif API support to port X/Motif applications to Win32, NuTCRACKER X/SDK also includes the following features for porting X Window applications to Windows NT:
   - Windows NT versions of the X Window system and Motif libraries
   - OSF/Motif Toolkit and widgets, including the Motif user interface language (UIL) compiler and Window manager
   - Motif extensions for Toolkit widget creation and manipulation routines, message routines, font routines, and convenience routines that have been added to supplement those provided by the OSF/Motif Toolkit
   - Motif extensions to the Motif user interface language
   - An X-server that allows testing of X application on a Windows NT computer
   - A Motif replacement library, Wintif, that permits presentation of Motif applications with a Windows “look-and-feel”
The remote start-up service which simulates the rexec daemon and starts X applications located on a Windows NT computer from the NuTCRACKER X-server control panel.

3. NuTCRACKER X/Operating Environment (X/OE) — provides a 32-bit X-server environment for end-user X/Motif applications and the NuTCRACKER DLLs necessary to run ported applications (it may also be used to run X applications performing on UNIX platforms).

**UNIX Functionality Supported by NuTCRACKER**

**APIs.** Process Control — NuTCRACKER provides a true UNIX fork with proper inheritance of resources, such as handles. Processes and threads are slightly different between UNIX and Windows NT. This function allows UNIX programs with fork execute as they would in UNIX.

Shared Memory — Applications can use shared memory consistent with UNIX standards, which supports cross-platform interprocess communication and data access.

Security — NuTCRACKER preserves security permissions for user, group, etc. as set up in the UNIX environment and applications.

Networking — NuTCRACKER provides BSD sockets to support networked applications. Sockets are handled as file descriptors, which allows UNIX code to use other IPC mechanisms without modification. All of Windows NT’s supported protocols are available to the applications.

User Interfaces — The NuTCRACKER SDK API library includes integrated support for curses user interfaces, while the NuTCRACKER X/SDK provides integrated support for X/Motif APIs in the xlib, xt, and other libraries.

Windows NT and Motif end-user interfaces are similar. While the interfaces look similar from an end-user perspective, similar is not same and the differences may be annoying to the user who has to often switch back and forth between the systems. To overcome this problem, NuTCRACKER provides the Wintif technology which lets Motif applications be displayed with a Windows look-and-feel. Note that Wintif is optional and if there are users who still want the Motif interface, that is supported as well.

**Environment and Utilities.** User Environment — As in UNIX, the user environment is fully managed through environment variables set on either a permanent or temporary basis.

Build Environment — The build environment is set up as it is done in UNIX with UNIX make environment variables. With NuTCRACKER, one can also convert to nmake and Windows NT’s make if desired.

Utilities — A comprehensive set of UNIX commands and utilities (over 140 in all) that maintains productivity in the Windows NT environment. The commands and utilities provided address broad areas such as text
manipulation, recordkeeping, the program execution environment, editing, and arithmetic operations.

An exhaustive list of the APIs and utilities can be found in the NuTCRACKER Installation and Porting Guide, a 200+-page document. This document is available through the DataFocus Web page http://www.datafocus.com. The NuTCRACKER X/SDK includes X and Motif APIs in the xlib, xt, and other libraries.

NuTCRACKER includes utilities that are part of the MKS Toolkit. Besides the MKS Toolkit utilities, NuTCRACKER includes other utilities such as chmod, whoami, process, sdiff, etc.

PORTAGE

Information on portage is available online at http://www.consensys.com. (Be careful when you type in Consensys, as there is an other company with a similar name, Consensus and the address of http://www.consen-sus.com).

Portage is an integration of UNIX SVR4 with Windows NT, including over 125 standard UNIX utilities. It has a base and an SDK. There are four main functions provided by the Portage Base System:

• uses Portage UNIX utilities from the NT Command Prompt
• uses one of the Portage UNIX shells (ksh or csh) instead of the NT command prompt.
• uses the Portage Windows Interface to manage UNIX shells
• allows access of the Portage online manual pages as Windows Help

The main components of the Portage SDK are:

• 30 software development utilities, including ar, cc, ld, yacc, lex, make, and SCCS
• more than 700 system calls and subroutines

Besides the Portage SDK, one needs to use the Windows NT environment and C/C++ compilation tools to port applications.

Portage makes UNIX an integral part of Windows NT, sharing the same hardware. Portage utilities can operate on files and directories created by Windows NT programs, and Windows NT programs can operate on files and directories created by Portage utilities. The Portage shells ksh and csh can run Windows NT programs, and the Windows NT Command Prompt can run UNIX programs.

Developers can use the Microsoft Windows NT compilers and debuggers directly to develop UNIX programs, or use the standard UNIX ar, cc, ld and make commands included in the Portage SDK. The Portage SDK
also enables developers to create hybrid UNIX/Windows NT programs that combine the UNIX API with the Win32 API. Windows NT stores environment variables in the Registry. The Windows NT environment is read by the Portage Kernel when it starts up, and is imported into all Portage programs when they start up.

**Portage Base**
Portage Base supports the Windows command prompt and provides UNIX command prompt functions.

**NT Command Prompt**
All Portage utilities can be run from the NT command prompt or console. Since the command prompt passes command line arguments to programs unchanged (unlike UNIX shells, which perform wild-card expansion and much more), Portage programs are able to provide an overlapping set of UNIX-like and DOS-like syntax. In particular, from the NT Command Prompt, path names can use either UNIX style / or Windows style \ notation. Supported functions include:

- pattern matching is UNIX style, including '*', '?', single quotes, and double quotes
- the Portage environment variables (UNIXROOT, UNIXTMP, SYMLINKS, and CASENAMES) are in effect

Of course, shell-specific features like $X for environment variables and backquotes for subshells are not available from the command prompt. An example of a valid command prompt command line is `ls -lrt /bin/b*`.

There are several name conflicts between NT commands and Portage UNIX utilities. The echo command is built into the command prompt (as it is in UNIX shells), so it will always use the NT syntax. Other commands like **more** and **mkdir** are not built-in, so the user will get whichever one is found first by following the PATH environment variable.

When Portage is installed, the bin directory is added to the end of the PATH; therefore, by default, the user will get the NT version of these utilities. The PATH environment variable can be changed, but that may break existing batch and other Windows NT functions using the PATH. The simplest solution is to use one of the UNIX shells, where $PORTAGE/bin is added to the front of the PATH environment variable by default.

**The UNIX Shells**
Portage provides both the Korn shell (ksh) and the C shell (csh). Both of these shells run in the console. This means that a very large buffer size
can be set, 200 lines or more, and keep that much interactive history available for browsing. A ksh or csh can be launched from icons in the Portage program group. The user can also simply type ksh or csh at an NT command prompt. If one wants to run multiple shells (maximum of 20 can be active at any one time), the preferred method is probably to use the Portage Windows Interface to start them.

Once running ksh or csh, the user is in a UNIX environment. The shell window provides an emulation of a VT100 terminal (a VT100 subset), and the term environment variable will be set to VT100. The window also provides a limited tty line discipline, and stty(1) can be used to view the options in effect or to change a subset of them.

In general, a user should be unable to tell the difference between the behavior under one of the Portage UNIX shells and a native SVR4 implementation. A list of the few known differences is provided in the Portage documentation.

If graphical user interfaces are preferred, Portage includes the Portage Windows Interface which serves a number of important functions:

- allows easy start-up and management of multiple UNIX shells
- allows customization of the default settings of shell windows (e.g., size, colors, etc.)
- provides a dialog box interface to all Portage UNIX commands
- provides easy access to online manual pages for every command

A user can start up the Portage Windows Interface by clicking on its icon in the Portage program group. The program’s main feature is a set of 140 buttons providing dialog-box access to UNIX utilities. The Portage SDK has an even larger set of buttons (170) than the Base System. Screen shots are available at http://www.consensys.com.

**Portage SDK**
The Portage SDK includes utilities, system routines, etc.

**Portage SDK Utilities.** The Portage SDK provides all of the software development tools from standard SVR4 except the basic C/C++ compilation system (compiler, linker, archiver and debugger). In order to compile and link programs for Portage on Windows NT, Microsoft Visual C++ needs to be installed.

Portage does include SVR4-compatible versions of ar, cc, and ld, which call the relevant Microsoft utilities to perform the actual work. This means that existing UNIX makefiles can be used with only minor changes.

When compiling and linking existing UNIX SVR4 programs with the Portage SDK, one can either use the Portage make command or the Microsoft nmake command. Using nmake may require more effort com-
pared to Portage make. Portage includes standard makefile templates for both make and nmake that can be included at the beginning of individual makefiles. Examples and sample programs are available online at http://www.consensys.com.

Terminal-oriented programs linked with the Portage SDK display output in a console window within the Win32 subsystem (the same as the NT command prompt, and all Portage utilities). The Portage console window provides a (partial) emulation of a VT100 terminal to support full screen functionality, and the Portage SDK includes the complete curses library.

**System Calls and Subroutines.** The Portage SDK subroutines are supplied in a set of libraries that comes with the Portage product. Individual subroutines can be found in the same libraries as on native UNIX (for example, the math routines are in libm.lib). The standard SVR4 libraries provided are libc.lib, libcmd.lib, libcurses.lib, libgen.lib, libgenIO.lib, libm.lib, and libw.lib.

There are two cases where Portage does not implement UNIX features so they are accessible by non-Portage programs — symbolic links and case-sensitive file names. Portage has implemented both of these features in a file system-independent manner, but non-Portage Windows NT programs cannot follow symbolic links, nor create two filenames that differ only in case (e.g., makefile and Makefile). In both instances, Portage provides environment variables (SYMLINKS and CASENAMES) that allow each user to decide whether to enable or disable the feature.

Sometimes, Windows NT has greater functionality than UNIX. In those cases, Portage has enhanced the UNIX subroutines and utilities to take advantage of the extended capabilities, while maintaining complete backwards compatibility with UNIX SVR4. For example, Portage has extended the chmod and ls utilities and the related subroutine calls to allow users to view and modify a larger subset of the extended NT file permissions than the standard UNIX read, write, and execute.

There are some differences between Portage and native UNIX SVR4; this list is included with the Portage documentation. The only significant UNIX SVR4 programs not provided in this release of Portage are the uucp family of utilities.

There are a handful of cases where Portage includes new subroutines to implement specific tasks that either must be done differently on Portage than on UNIX, or can be done more efficiently by a different method on Portage than on UNIX.

**OpenNT**

Information on OpenNT is available online at http://www.softway.com/OpenNT/.
A schematic of OpenNT Architecture is illustrated in shown in Exhibit 3.

OpenNT is based on a virtual subsystem design allowing for the hosting of a true conforming UNIX environment. By using the POSIX.1 subsystem built into Windows NT, users are given a secure environment that supports many of the features required by POSIX and UNIX systems. Some of the supported features include:

- proper case-sensitive file names within the NT file system (for example, files Report and report can coexist in the same directory)
- file links are supported (i.e., one file can have multiple names)
- background processing and job control via the OpenNT shell
- full control over file user ownership, group ownership, and access modes
- ability to run both UNIX/POSIX and Windows applications from the OpenNT shell

Softway Systems is committed to providing complete user and developer products for Microsoft Windows NT that conform to IEEE POSIX and X/Open specifications. As a demonstration, Softway Systems products will maintain the appropriate National Institute of Standards (NIST) certifications for FIPS 151-2 (POSIX.1) and FIPS 189 (POSIX.2) and the appropriate X/Open brands in future releases.
The OpenNT Product Family

The OpenNT product family includes:

- OpenNT Commands and Utilities
- OpenNT Software Development Kit
- OpenNT X11 Server
- OpenNT Telnet Server

Following is a brief look at these products.

**OpenNT Commands and Utilities.** OpenNT Commands and Utilities is a set of utilities that provide functions similar to UNIX, and a POSIX command shell for Windows NT. The product provides conformance to the IEEE POSIX.2 Execution Environment Utilities standard as defined in FIPS 189 (FIPS 189 is the Federal Information Processing Standard required by many federal government agencies and departments for operating system procurements).

OpenNT also adds a new POSIX.1 subsystem (which is meant as a replacement POSIX.1 subsystem for older Windows NT systems) with increased functionality to provide seamless integration of the POSIX environment with the other components of Windows NT. Additional UNIX utilities supported include vi, more, uuencode, etc.

**OpenNT Software Development Kit.** OpenNT Software Development Kit is a development system for porting and developing POSIX and UNIX applications on Windows NT. In combination with the OpenNT Command and Utilities product mentioned above, this development environment includes the tools necessary for providing a development platform to develop and run UNIX applications. Applications built with the OpenNT environment run as native, 32-bit Windows NT applications.

One can use third-party compilers and tools (for example, Microsoft Visual C/C++, GNU tools) within the development system. Utilities such as lex, yacc, make, and rcs are included as standard components in the product.

**OpenNT X11 Server.** The OpenNT X11 server for Windows NT (and Windows 95) is a PC X Display Server. The OpenNT X11 server supports bi-directional copy-and-paste of text and graphics between X clients and Windows NT/Windows 95 applications. From the same desktop, users can run:

- Win16 (Windows 3.1) applications
- Win32 (Windows 95 and Windows NT) applications
- POSIX and UNIX text-based applications via the OpenNT shell
• X11/Motif applications running natively on Windows NT via the OpenNT X11
• Windowing System and OpenNT Motif (By the time this article is read, the function should be available; check with the vendor on the timing for the availability of Motif support.)
• Remote network-based X Windows and Motif applications

UNIX window managers supported by the OpenNT server include Motif, OpenLook, VUE, and CDE. The OpenNT server also supports XD-MCP security through X authority. A full set of X11R6 fonts are provided with automatic font substitution and a choice of full-screen, single-window mode, or multiple-window mode.

**OpenNT Telnet Server.** The OpenNT Telnet Server is a telnet daemon service for Windows NT. Telnet users get full access to the Windows NT Common Command Shell or the OpenNT POSIX/UNIX shell. Multiple users can access one Windows NT server and simultaneously run a variety of Win32, DOS, OS/2, and POSIX character-based applications.

OpenNT Telnet Server provides a set of configuration options to the Windows NT system administrator, allowing user access and permissions to be defined for individual users and groups of users. OpenNT Telnet Server allows the Windows NT administrator to specify that a user or group of users be dropped directly into a custom application rather than a command line. Windows NT’s security features are maintained.

The telnet daemon allows access by telnet clients on any system connected to a TCP/IP network. Clients can be UNIX systems, Macintosh, Windows, DOS, or network terminals connected to the local area network. Dumb terminals connected to a terminal server on the network can also log in to the system. Clients can even access the shell from remote sites over the Internet.

The product allows telnet users to log on to the Windows NT server without interfering with the Windows NT console user or file server users.

OpenNT Telnet Server provides ANSI/VT100 terminal emulation to the network client that logs in to the system. Certain keystroke sequences map to special PC function keys and can be passed through by terminal emulators and interpreted by applications. This feature allows users to make use of function keys, arrow keys, and ALT key combinations in PC programs while logged in to the system. Full-screen applications can be run and support is provided for character attributes, color, and line drawing characters.

The product supports the connection of multiple dumb terminals when used with a network terminal server. With OpenNT Telnet Server users can access the server to perform text mode command-line operations and run any character-based application.
OpenNT Telnet Server can drop a user directly into a custom application or shell after logging in. This allows users on dedicated terminals to see only their application, shielding them from the computer system or a shell.

When the user exits the program, the telnet client is disconnected from OpenNT Telnet Server. A feature of OpenNT Telnet Server is its integration with the rest of the OpenNT environment. Users can select the OpenNT UNIX shell as their default shell and log in directly to an environment that is consistent with today's UNIX operating systems.

OpenNT Telnet Server uses the security features of Windows NT. OpenNT Telnet Server abides by access restrictions imposed by the server. Each user runs in their own appropriate security context, with all their own permissions in place, just as if they had logged in from the console. OpenNT Telnet Server preserves all of the C2 security features that come with Windows NT.

OpenNT Telnet Server features a graphical user interface application for administration and set-up. The system administrator can control:

- user access and permissions
- maintaining a log of all events and user activities
- starting, stopping, and pausing the telnet service
- terminal emulation features and color control
- user shell options and default directory
- time slice and priority administration
- program licensing and registration

Following is an overview of another porting product, eXcursion from Digital.

**eXcursion**

Information on eXcursion is available online at http://www.digital.com.

The eXcursion Software Developer's Kit from Digital contains the libraries and headers needed to port Motif and X applications from the UNIX operating system to Windows NT. The eXcursion SDK includes X Window System and Motif libraries for Windows NT, Motif Toolkit and widgets, the Motif User Interface Language (UIL) compiler and Motif Window Manager (MWM), DECwindows Motif extensions, example X and Motif applications for Windows NT, a platform-independent imake utility, and the eXcursion X-server.

The eXcursion Software Developer's Kit supports Microsoft's Visual C++, and can operate with Microsoft's TCP/IP (included with Windows NT) and DECnet (included with PATHWORKS for Windows NT).
eXceed XDK
Hummingbird’s X Development Kits incorporate all of the X Window System standard libraries and toolkits that are required to develop X applications on the PC, which include Xlib, Xt intrinsics, Xaw, and Xm.

Also available are the OSF/Motif and UIL compilers, in addition to sample source and sample applications.

PC X-server software provides users with a full X Window System implementation typically found on workstations. Available for the Windows, Windows 95, Windows NT (Intel and Alpha), and OS/2 platforms, the Hummingbird eXceed X Development Kits (XDKs) give today’s PCs powerful workstation-like functionality by transforming them into a client-host and server, all at the same time.

Local X support permits X clients to be started and displayed locally or remotely on other X servers in the network. Local X support also permits local X clients to run on stand-alone PCs with no network support.

Hummingbird’s eXceed 3D for Windows NT includes additional libraries for developing OpenGL X applications on PCs.

PORTING APPLICATIONS FROM WINDOWS TO UNIX
The previous section covered porting applications from UNIX to Windows. In this section, porting in the opposite direction is reviewed, including applications written for the Windows environment and the tools that allow the application execute in the UNIX environment.

Windows applications can be run on UNIX in three ways:

In one way (which is actually a rewrite and not a port), some vendors saw a market opportunity in providing Windows look-alike native UNIX applications for popular office products. The idea was that customers need not buy any Windows operating environments or software, but still use office applications using the familiar Windows graphical interface on the same desktop with production UNIX applications. Applications ported in this manner have generally not been very successful, such as Quorum’s ports of Microsoft Word and Excel.

A second way is to port the source manually. If ported manually, care must be taken of the operating system differences between Windows NT and UNIX. Attention should be focused on programming language differences, such as C language differences. In addition, if one has old Windows sources written for the 16-bit world, the following must be considered:

- direct hardware calls or access to memory space (that would not normally be allowed in 32-bit applications)
- near and far pointers
- compiler options specific to 16-bits, such as NEAR or Pascal
A third way would be to provide tools that assume the role of Windows in UNIX systems. These tools would provide the function calls, messages, device contexts, and controls on top of the UNIX operating system that Windows provides for application programs executing in the Windows environment. Besides providing support for the operating system functions, the tools also provide support in the UNIX environment for Windows’ unique features such as OLE, MFC, etc.

### COMMON PORTING PRODUCTS

Some of the tools that help in the porting from Windows to UNIX include Wind/U from Bristol Technology, MainWin Studio from MainSoft, and the Willows Toolkit from Willows Software.

**Wind/U**

Information on Wind/U is available online at http://www.bristol.com.

Wind/U is an implementation of the Microsoft Windows API under UNIX. Wind/U leverages Microsoft Windows source code, licensed to Bristol Technology as part of the Microsoft WISE program. Wind/U supports Win32 with Microsoft Foundation Classes (MFC). **Exhibit 4** shows where Wind/U fits in when applications are developed using Visual C++ and MFC in the Microsoft environment to run them in the UNIX environment. The Visual C++ and MFC application runs on Windows NT and Windows 95 by using the Win32 and MFC libraries. The same applica-
tions can also run on UNIX machines, such as those from SUN, HP, and DEC using Wind/U libraries.

The following advanced Windows features are supported to make Windows applications that use them portable to UNIX:

- GUI features such as multiple document interface (MDI), combo boxes, common dialogs, common controls, palettes, and graphical device interfaces (GDIs), including logical coordinate mapping
- architectural APIs for dynamically linked libraries (DLLs)
- dynamic data exchange management library (DDEML)
- object linking and embedding (OLE)
- Windows sockets
- kernel layer APIs for all non-GUI APIs in the Windows environment, including file i/o, memory management, memory-mapped files, etc.
- advanced printing capabilities through Bristol’s Xprinter
- PostScript and PCL Language Library — together, Xprinter and the GD1 layer provide transparent access to the display and PostScript and PCL printers
- online help through Bristol’s Microsoft Windows-compatible help subsystem, HyperHelp. HyperHelp allows use of the same rich-text format (RTF) source files from the Windows version to create an equivalent help system in the UNIX environment

All of these features are provided as libraries to which the application is linked after recompilation in the native UNIX environment. The result is a truly native application that maintains identical functionality with the Windows version.

The toolkit also includes several tools for the programmer:

- Wind/U Wizard — automatically verifies installation, checks the environment set-up, and contacts Bristol support via e-mail
- Wind/U Documentation On-line — provides the complete content of Wind/U printed documentation in a cross-referenced, browsable format
- make_windumakefile — scans the original Windows source code to create an appropriate make file for the UNIX environment
- prepare_source — scans the original Windows source to automatically replace DOS format characters (end-of-line, etc.) with equivalent UNIX or OpenVMS characters
- resource compiler — a ported version of the Microsoft resource compiler that accepts Windows resources as input
- regedit — a ported version of the Microsoft Registry Editor that enables you to change setting in your system registry
- dllink — creates a dynamic link library from a set of objects and a definition file
• windu_clientd — a color management daemon that enables sharing of Wind/U color resources across Wind/U applications
• windu_registryd — manages communication between applications and the Wind/U registry
• windu_scmd — manages interaction between OLE processes and the Wind/U registry

Wind/U is currently available for the following platforms: HP 900 Series 700 and 800, Sun SPARC (SunOS and Solaris), IBM RS/6000, Silicon Graphics, Digital UNIX, and Digital OpenVMS.

MainWin Studio

Information on MainWin is available online from Mainsoft at http://www.mainsoft.com.

MainWin Studio is a complete set of development components that enable a software developer to port an application written to the Microsoft Windows API onto all major UNIX platforms. Exhibit 5 shows an overview of the MainWin XDE environment.

• MainWin XDE (eXtended Development Environment), which includes MainWin Help (the Microsoft Help engine on UNIX)
• MainWin Test (testing tool to generate test cases and port test scripts across platforms, equivalent of Microsoft Test)
• Visual SourceSafe for UNIX (source code manager, equivalent of Microsoft Visual SourceSafe for Windows)

EXHIBIT 5 — MainWin XDE Environment

MainWin XDE Environment Diagram

- Common Application Source
- Win32 SDK
- Windows NT
- MainWin XDE
  - Windows Application
  - MainWin XDE Dedicated Library
  - UNIX X Windows System
- MainWin XDE
  - Windows Application
  - MainWin XDE Dedicated Library
  - UNIX X Windows System
MainWin XDE
MainWin XDE provides a complete package of development tools for cross-developing Windows applications to run on multiple UNIX platforms. It includes support for MFC libraries, common controls, resource compiler; help engine, automatic makefile generator; specialized source files to include with application programs, and a set of tools that implement the cross-development environment on the target systems.

The MainWin XDE Dedicated Library is required, to enable the Windows application programming interface (API) on each individual UNIX workstation. MainWin supports both a Windows and Motif look, selected "on-the-fly" by the end user.

The MainWin XDE Dedicated Library is based on Microsoft Windows source code and essentially replaces Motif. MainWin has source code license agreements with Microsoft for Windows, Windows 95, NT, Microsoft Test, and Microsoft Visual SourceSafe. MainWin XDE Dedicated Library improves performance by directly interfacing to Xlib.

Mainsoft has ported the Microsoft Foundation Class (MFC) library to work on UNIX platforms. MFC is included as part of MainWin XDE. While MFC DLLs can be shared extension DLLs or statically linked USER DLLs, MainWin supports only the extension DLL model.

Extension DLLs are more resource efficient as they link to a single instance of the MFC library. MainWin does not support USER DLLs as many UNIX linkers do not discriminate between the symbols in multiple instances of MFC libraries. Support for MFC DLLs with "state" information (DLLs with AFX_MANAGE_STATE) is planned for a future release of MainWin XDE.

MainWin Test
MainWin Test is an automated testing tool that permits retesting of applications on multiple platforms. MainWin Test leverages the investment in test script development by allowing the test scripts to be used for different platforms. MainWin Test is compatible with test scripts developed using Microsoft Test on the PC.

MainWin Test has its own scripting language (TestBasic) and Bristol is a Microsoft Test source code licensee. Platforms supported by MainWin Test include DEC, HP, IBM, SCO, SGI, and Sun. A syntax editor and debugger included provide an interactive environment for test script development; a trap error handler allows the unattended collection of test results.

Visual SourceSafe for UNIX
Visual SourceSafe for UNIX is a project-oriented version control system that provides a graphical user interface. Mainsoft is a Microsoft SourceSafe source code licensee and Visual SourceSafe for UNIX is compatible with
Microsoft SourceSafe (both can share a single repository). Visual SourceSafe for UNIX features drag-and-drop file sharing, user-configurable options, visual file difference displays, and a central code repository.

Visual SourceSafe for UNIX supports file sharing which allows source code modules to be reused across projects. Changes to shared code checked into one project are automatically propagated. Visual SourceSafe supports UNIX platforms from DEC, HP, IBM, SCO, SGI, and Sun.

**Willows Toolkit from Willows Software**

Willows Toolkit allows applications to the Win APIs run on UNIX and Macintosh. Additional information on the Willows Toolkit is available online at http://www.willows.com.

The 16-bit version of the Willows Toolkit was known as TWIN APIW. Willows Toolkit Version 2.0 added Win32 Support. Willows Toolkit is royalty-free.

**PORTING DATABASE APPLICATIONS**

If porting applications that access a database, aside from the C language calls, the POSIX or Win32 API calls, other operating system calls, the calls to the database have to be ported as well.

In the Windows environment, the common method of accessing databases is using Open Database Connectivity (ODBC). ODBC is an attempt to shield an application from variations in databases. If using direct structured query language (SQL), one may need to change the SQL and recompile and relink the application, if needing to change databases. ODBC has two main components — an ODBC driver manager and ODBC drivers.

ODBC allows the application to make the same calls, and handles the variations of databases through ODBC drivers, typically one driver for each database. The application calls the ODBC driver manager, which interacts with the correct ODBC driver and returns the result to the program. ODBC drivers shield the database specifics in much the same way as device drivers (for example, printer drivers) shield the specifics of the device from the application.

Support for porting database calls depends on the porting vendor. For example, Wind/U provides a stubbed ODBC library that compiles and links successfully but returns failure codes at runtime. Bristol Technology is working with third-party vendors to provide the actual ODBC driver manager and database drivers in future versions of Wind/U.

Third-party vendors that provide ODBC on UNIX include:

- Visigenic Software, Inc.
- Q+E Software/Intersolv
Visigenic Database Connectivity Software


Database Connectivity Software from Visigenic Software includes VisiChannel, VisiODBC DriverSets, and VisiODBC SDKs.

VisiODBC Channel

VisiChannel is the overall architecture (see Exhibit 6).

VisiChannel is based on the ODBC standard. VisiChannel is database-independent and supports the JDBC API for Java applets and applications.

VisiODBC DriverSets

With the VisiODBC Drivers (formerly Visigenic ODBC Drivers), one can provide cross-platform access to the most popular SQL databases, including CA-Ingres, IBM DB2, Inofrmix, Microsoft SQL Server, Oracle, Sybase DBLib, and Sybase SQL Server from any ODBC-enabled application. The VisiODBC drivers are available for Windows, Windows NT, Windows 95, ATT GIS, IBM AIX, HP-UX, SCO, Solaris, Sun OS, Macintosh and Power Macintosh, and OS/2.

VisiODBC SDKs

VisiODBC SDKs (formerly Visigenic ODBC SDKs) are SDKs for developers to develop vendor-independent database applications and ODBC-compliant drivers. Visigenic has an agreement with Microsoft that gives it the right to license and port the Microsoft ODBC Software Development Kit (SDK) to all non-Windows platforms. Visigenic has ported the Microsoft ODBC SDK to ATT GIS, IBM AIX, HP-UX, SCO, Solaris, Sun OS, Macintosh and Power Macintosh, and OS/2.
Information about DataDirect is available at http://www.intersolv.com.

INTER SOLV’s DataDirect includes both client-based and server-based ODBC data access, connecting to most current databases, on most standard server platforms. See Exhibit 7.

DataDirect’s client-based connectivity implementation offers ODBC access from clients, including all Microsoft platforms, Mac, OS/2, and many flavors of UNIX through client-based drivers. DataDirect includes connectivity to 35 databases, right out of the box.

Server-based connectivity offers the same functionality of client-based ODBC, without the need for database-specific client middleware. DataDirect server-based connectivity includes direct point-to-point or multi-tier platform data access to different databases and networks and can be accessed concurrently through DataDirect Sequelink. Server-based connectivity has a single ODBC driver on the client and a thin server listening piece.

Exhibit 8 summarizes the criteria to choose between client-based and server-based connectivities.
If one is using databases from database vendors such as Oracle, including some of the advanced versions such as the Oracle parallel server, check with the vendor to see if the same version is available in the ported-to environment (such as Windows NT).

By the time of publication, Oracle will have announced a parallel version of its database, Oracle Parallel Server, for Microsoft’s Windows NT operating system. The Windows NT version makes it possible to divide database workload across multiple servers running Windows NT.

Oracle has already been selling Parallel Server versions for UNIX platforms and Digital Equipment’s VMS operating system for a few years. Compared to the UNIX versions, which can run across a large number of UNIX servers, the first versions of the NT version of the parallel server would be able to use two to eight servers.

PORTING TRANSACTION PROCESSING APPLICATIONS

Most transaction processing applications are developed using programming interfaces such as X/Open organization’s X Application to Transaction Manager Interface (XATMI) or the remote procedure call interface — X/Open’s TxRPC interface.

Porting the transaction processing part of an application should be relatively straightforward if the following conditions are met:

- the transaction processing monitor software used supports standards such as TxRPC and ATMI
- the code is fairly standards-compliant
the TP vendor has a version that runs in both the current environment and the environment one wants to port to and supports an (almost) identical programming interface in both environments. Most of the major TP products, such as Tuxedo, Top End, Encina, etc., come with versions supporting a fairly standard API for Windows and UNIX.

**PORTING 3GL/4GL APPLICATIONS**

So far discussion has been about programs written in programming languages with additional database or transaction processing calls embedded in them. If an application has been developed using application development tools like PowerBuilder, then the best bet is to check if the application development tool vendor provides a version that runs in the target environment. An application source should be portable, but check with the vendor about platform-specific differences and whether they provide support in porting an application’s source to the target environment.

PowerBuilder 5.0, for example, has a UNIX and a Mac version. If using Tivoli for Application Management, then Tivoli has a UNIX version as well.

**PORTING WEB APPLICATIONS**

Web applications are similar to other applications with some differences. The languages used to develop Web applications include hypertext markup language (HTML), PERL, and Java. HTML standards exist although they are being updated with a frequency quite uncommon for standards.

There should not be a major problem porting HTML source between UNIX and Windows NT environments. Java, by design, is intended to be
platform-independent and should not present a problem porting Java code. Java, being relatively new, does not have a whole lot of porting references in the industry. PERL has been used for a while in UNIX Web applications and porting tool manufacturers, such as NuTCRACKER, support porting PERL applications.

**Porting Distributed Applications**

If porting distributed applications, the area to focus on is porting the Remote Procedure Calls (RPC) that are in distributed applications. While DCE RPC is a very common flavor of RPC, there are some applications that use Sun Microsystems’ Open Network Computing (ONC) RPC. Some UNIX systems support ONC RPC. Windows NT provides RPC support that is compatible with DCE RPC. Tools that allow porting of ONC RPC applications to DCE RPC include RhaPC-d from Intergraph.

**RhaPC-d.** Information on RhaPC-d is available online at http://www.intergraph.com.

RhaPC-d (pronounced “rhapsody”), Intergraph’s Remote Procedure Call development toolkit lets programmers transfer applications across a client/server network without learning network protocols or sockets programming. RhaPC-d operates over both TCP and UDP transport protocols.

On Windows NT, the user can link with the RhaPC-d RPC library while compiling applications. RhaPC-d supplies the protocols needed for exchanging data using ONC RPC and for encoding data using the eXternal Data Representation (xdr) standard.

RhaPC-d provides interoperability with existing RPC-based client/server applications and facilitates their porting to the Windows NT environment. The RhaPC-d port mapper and the Windows NT Service Control Manager combine to allow Windows NT to operate as both a client and a full-function server.

RhaPC-d provides a static library that eliminates the need to deliver additional libraries or DLLs with the application program. RhaPC-d also provides an API interface to create network RPC packets in xdr format. The RhaPC-d RPC protocol compiler generates C code from an RPC protocol description file.

**PORTING INTERNATIONAL APPLICATIONS**

If applications have been written for international use using different code pages, Multi Byte Character Sets (MBCS), or UNICODE (see below), Windows NT and many UNIX systems support code pages and MBCS. Windows NT also supports UNICODE, although not all UNIX systems support UNICODE.

The Win32 API and Windows NT support UNICODE. Windows NT uses UNICODE extensively in its internal operations. For example, all text strings in graphics device interface (GDI) APIs are in UNICODE,
New Technology File System (NTFS) uses Unicode for file, path, and directory names; object names and all system information files are in Unicode.

The Windows NT subsystems take care of many of the conversions. For example, the Win32 subsystem converts ASCII characters it receives into Unicode strings and converts them back to ASCII, if necessary, for output. Microsoft Visual C++ and the Microsoft Foundation Class (MFC) library support the most common form of MBCS-DBCS (two bytes per character).

Digital UNIX supports XPG4-compliant internationalization and includes 22 multibyte versions, primarily for the Asian market.

**Unicode**

Developed by the Unicode Consortium, a nonprofit group sponsored by a number of computer companies, Unicode is a fixed-width encoding scheme where each character is represented by 16 bits. The number of characters that can be represented by Unicode is thus $2^{16}$ or 65,536.

Using Unicode, all the characters can be represented from character sets like ANSI, characters from the Cyrillic character set, special-purpose characters such as publishing characters, and mathematical symbols. In short, everything can be represented to date. For a complete description of the Unicode standard, the characters represented, etc., refer to *The Unicode Standard: Worldwide Character Encoding* by Addison-Wesley Publishing Company: ISBN 0201567881.

**Porting OS/2 Applications to Windows NT**

If OS/2 applications need to be ported to Windows NT, most of the porting considerations that have been discussed, such as source code issues and scripts, apply to OS/2 to Windows NT porting as well.

If doing a manual porting, OS/2 to Windows NT porting may be a little easier than UNIX to Windows NT porting. Earlier versions of OS/2 and Windows were developed with collaboration between IBM and Microsoft and many concepts between OS/2 and Windows are similar. HPFS and NTFS have many similar features. In addition, unlike the UNIX world, there is only one flavor of OS/2 from one company.

Third-party assistance is available for OS/2 to Windows port. Tandem, for one, offers OS/2 to Windows NT porting services. Tandem uses a two-phase approach. The first phase scans OS/2 programs for code that is not compatible with Windows NT and must be rewritten. The scanning phase gives an idea of the amount of code to be rewritten and also helps create project requirements.

The second phase ports OS/2 source code that can be ported to Windows NT. The second phase also tests and validates the system.
OTHER PORTING CONSIDERATIONS
Object models are at the root of many programming technologies. The object model specifies the rules for object interaction. Examples of object models include the Component Object Model (COM) and Distributed Component Object Model (DCOM) from Microsoft and System Object Model (SOM) and Distributed System Object Model (DSOM) from IBM. The object models become important because of the technologies supported by the model. For example, COM is the basis for Microsoft’s ActiveX and OLE technologies.

Microsoft recently announced that Digital Equipment and Hewlett-Packard will support Microsoft’s COM on their operating systems in a year. The HP support will be in HP-UX while Digital Equipment will provide COM support in Digital UNIX and OpenVMS.

Once the underlying model is supported in the operating system, an individual will be able to develop or buy applications using the model-based technologies on these operating system. Porting OLE and ACTIVE X applications to UNIX will become easier.

REWITING APPLICATIONS
As a manager, it may be decided that rewriting an application might be better than porting it. Applications that match the following criteria qualify for rewriting:

- application is close to its life expectancy
- no longer have the source code for some or all of the application
- no longer have the programmers who wrote or have knowledge about the application
- application is a maintenance nightmare
- application uses obsolete technology (language, architecture, hardware, etc.) that needs to be replaced

If people with knowledge of the application are not available and nor is the source, then a significant rewriting effort will be necessary. If the source is available, but people knowledgeable in the application are unavailable, there are some tools, such as DISCOVER, that may help in capturing the business knowledge in the application.

DISCOVER
DISCOVER scans all C and C++ code and identifies every inter- and intradependency in software, and then builds a complete information model of the entire software system. The information model is a highly associative repository that keeps track of every dependency, regardless of the size of the application, allowing all changes to the software to be immediately reflected.
The knowledge stored in the information model is leveraged by application sets designed for specific development tasks, providing overall improvements in software process and quality.

There are five major application sets that make use of the DISCOVER Information Model. Each of these tool sets is a value-added application. The sets include:

- **DEVELOP/SET** consisting of PROGRAM/sw, DESIGN/sw, DEBUG/sw, and DEBUG+/sw
- **REENGINEER/SET** consisting of PACKAGE/sw, PARTITION/sw, EXTRACT/sw, and SIMPLIFY.H/sw
- **CM/SET** consisting of CM/sw.ATRIA, CM/sw.CONTINUUS, CM/sw.CVS, CM/sw.RCS, and CM/sw.SCCS
- **ADMIN/SET** consisting of ADMIN/sw
- **DOC/SET** consisting of DOC/sw.FRAME and REPORT/sw

PROGRAM/sw provides software comprehension through its browsing, navigation, query, editing, and graphical views. In addition, PROGRAM/sw supplies automated risk assessment and management capabilities through Impact Analysis, which reports on the systemwide impact of any proposed change before the change is submitted. Impact Analysis gives organizations the information necessary for making accurate development decisions, improving software quality during development, not after a release.

Change Propagation, another PROGRAM/sw capability, automatically checks out all needed files from the existing configuration management system and implements the approved change. This unique capability automates a usually tedious and error-prone aspect of development, freeing up time for more creative work, while also increasing the quality of the software produced.

DESIGN/sw provides the ability to create and modify graphical views — data charts, entity relation diagrams, and class inheritance diagrams. Modifying any of these diagrams results in automatic modification (or incremental generation) of the associated code to reflect those changes, speeding development and ensuring accuracy.

DEBUG/sw is an integrated debugger interface to industry-standard debuggers such as GDB, DBX, and DDE. The interface emulates the most popular debuggers and translates familiar commands and conventions to the installed debugger. DEBUG+/sw enhances the chosen debugger by the addition of mixed-mode debugging capability.

CM/SET provides seamless integration with many popular configuration management systems, including Atria’s ClearCase, Continuuus’ Continuuus/CM, CVS, RCS, and SCCS.
REENGINEER/SET, used alone or in tandem, breaks large, monolithic applications into smaller, more manageable modules or components that incorporate a subset of the functionality of the original software.

EXTRACT/sw module allows users to create and extract subsystems from existing source code, either manually or automatically. This module also provides dormant code analysis, identifying those functions, variables, structures, and classes not utilized by the software system. Automatic elimination of dead code enables organizations to easily remove software that is no longer called.

PACKAGE/sw subdivides an application into logical components based on the dynamic relationships of the functions and data in the program.

PARTITION/sw performs application restructuring. Teams can physically rearrange software into a new structure that accurately reflects the logical structure. Using this approach, teams can reorganize existing software into more manageable pieces, to delete unnecessary code, or to divide the software into two or more applications.

SIMPLIFY.H/sw uses the information contained in the Information Model to pare down header files to a necessary minimum, resulting in fewer compilations needed after a change. In addition, compilations that are done proceed more quickly since the compiler does not waste time reading and parsing nonessential information.

DOC/SET consists of DOC/sw.FRAME and REPORT/sw. DOC/sw.FRAME provides multiple methods for establishing direct links between source code and documents, while REPORT/sw generates systemwide “as-built” software documentation. Both applications enable organizations to keep software and documentation synchronized.

ADMIN/sw includes software administration utilities for set-up and usage of DISCOVER. These utilities include makefile readers, batch mode support, a Project Definition File generator, and debugger.

**CONCLUSION**

Porting applications have been reviewed from UNIX to Windows NT and vice versa. Some common porting products available to help with porting have also been considered. Porting is not limited to C or C++ programs. Porting applications that access database, Web applications, transaction processing applications, and others have been presented.

---

Raj Rajagopal has more than 17 years of professional experience in all phases of software development, including requirements, architecture, high-level design, low-level design, coding, testing, maintenance, and project and line management. He is currently a Senior Computer Scientist at the Illinois Institute of Technology Research Institute (IITRI), and is involved with technology assessments. Prior to IITRI, he worked at IBM for nearly ten years.