How Much Interoperability Do You Want?

By Doug Akers, Product Manager, MKS Toolkit

So you're thinking of bringing Microsoft Windows into your UNIX environment. Well you are not alone. According to research by International Data Corporation (IDC), a leading industry analyst firm, global companies have invested in more then 1.5 million Windows server licenses.

Over the past few years, working in a heterogeneous environment has become the norm rather than the exception. Your decision lies not with whether to integrate UNIX and NT but rather how and to what extent do you integrate these two systems. More specifically, you must ask yourself a few questions with respect to interoperability:

1. Do you simply want these systems to talk to each other? *Accessibility.*
2. Do you want users to have the same level of functionality with your enterprise applications? *Portability.*
3. Do you want to merge these applications with your Windows systems? *Integration.*
4. Do you want to exploit the new technology (i.e. COM, Java, etc) of other Windows applications and move forward into the future? *Modernization.*

And what do each of these degrees of interoperability involve?

Application Accessibility

At the low end of platform interoperability is application accessibility, the ability to connect to and access UNIX file systems and applications from a Windows system. Application accessibility is really a discussion of two separate issues, user compatibility and platform interoperability. Each sub-section has its own set of tools and technologies, as well as benefits, and essentially solves a different aspect of application accessibility.

User Compatibility

When working in a heterogeneous UNIX and Windows environment, wouldn't it be nice if you didn't have to re-train your UNIX users to use Windows? Wouldn't it be nice if your machines were user compatible?

This concept of UNIX-Windows user compatibility falls into two general categories. The first is traditional UNIX commands and utilities that follow the same standards and operate in the same way as their UNIX counterparts but are designed, written, tested, implemented and available for the Win32 system. The availability of familiar tools will allow your users to be instantly productive in their new environment, and allows the organization to retain the training investment and reap these productivity gains.

The second category of UNIX-Windows user compatibility is comprised of commands and utilities developed in the UNIX fashion specifically for the Windows environment. Tools that take advantage of Windows technologies like the registry, ACLs, user and group administration, and database and internet infrastructure are a natural extension to the base UNIX command set. These features allow you to not only manipulate these technologies from the command line or a script, thereby automating many routine tasks, but also retain the UNIX mind set and way of approaching problems and formulating solutions in the Windows environment.

Having the ability to solve a problem by firing up a Korn shell and creating a script which uses both traditional UNIX commands as well as Windows specific UNIX-like utilities is an incredibly powerful thing to offer Windows users. It is also quite a comfort for a traditional UNIX user to be able to type `'find . -name "*.c" | grep "myfile"'` and not get `"The name specified is not recognized as`

`an internal or external command, operable program or batch file."` (By the way, without User Compatibility tools, typing something as simple as 'ls' would give the same error.)

Platform Interoperability

In today's heterogeneous world, you need interoperability between your UNIX and Windows machines, including file exchange, data exchange, application exchange, remote access and remote administration.

File exchange capabilities, whether using NFS (Network File System), SMB (Server Message Blocks), or some other solution, allows UNIX and Windows systems to access files physically located on other systems. This access, tying in the concept of data exchange, allows for the central storage of information that decreases the duplication and replication efforts required between disconnected systems.

The concept of application exchange, whether character based UNIX applications accessed using Telnet or graphical UNIX applications accessed using X technology, enables organizations to leverage existing software assets while gaining the productivity benefits of the Windows platform.

There are a couple of facets to remote access and remote administration. The first, covered off by Telnet, NFS, and FTP, is the ability to access not only the file system but the configuration and administration data as well. Being able to administer the entire hybrid system from a single workstation, any workstation, saves time, removes the expense of investing in two systems on a desktop, and most important, alleviates the frustration of working in two platforms. The second area is the ability to administer both systems using a common tool set, or at the very least, a common mind set. Having a set of commands and utilities that operate consistently across both UNIX and Windows allows companies to leverage the training and skills of systems administrators.

Platform interoperability is all about access and making the back-end systems transparent to the users, and allowing organizations to re-coup the costs associated with their existing UNIX systems.

Organizations are not choosing one operating environment over another, specifically opting for Windows NT instead of UNIX, but rather a mix of the two, as each has its own strengths. Knocking down the integration hurdles of the two systems at the user compatibility and platform interoperability level is an essential victory for those that want to succeed in this heterogeneous environment.

Application Portability

There are essentially two types of applications that live in the UNIX world, script-based and binary. Script-based applications are those that are predominantly run from the back-end by shell (Korn shell and C shell being the most popular), awk, and/or Perl scripts that manipulate user input, files, and text-based data. Binary applications, just like in the Windows world, need to be coded, compiled, and linked to produce executables that are then run by the end-user and interact in one form or another with the appropriate data.

Porting UNIX Scripts

It is very likely that the majority of your UNIX scripts do not exist as part of a script-based application but rather in bits and pieces in the `~user/bin` directories of you and your users. Script porting effects everyone in the enterprise and not just the IT and infrastructure departments. Administrators, developers, QA, and even the less technical around us may use scripts to automate various tasks and simplify data processing--some even without knowing it.

There are several key issues you have to be aware of when talking about or implementing a script porting effort:

- Path name differences between Windows and UNIX. Some points you might need to address in your scripts include:
    - the existence of drive letters on Windows;

    - case-preserving rather than case-insensitive path names;
    - spaces in path and filenames;
    - different directory separator characters (`/` vs `\`); and
    - different path-element separators (`:` vs `;`).
- The difference in the end of line character(s) used on UNIX (carriage-return) and Windows (carriage-return/line-feed).
- System specific assumptions that are made (i.e. the location of and use for /tmp or `/var/tmp` vs `/WinNT/Temp` or taking into account the Windows registry).

Another item to keep in mind is... be thorough. You may take for granted some of the commands you have used for years and not realize exactly how painful a job was before you simplified it with a script. What I am trying to say is... port them all!

Porting UNIX Applications

Binary UNIX applications will not run directly on Win32. This fact is obvious (I hope) to anyone who develops, tests or is involved in the creation of software in any way. Though the development process is vastly similar, the underlying systems are not. System APIs, file systems, device drivers etc. are system dependant and in order to move applications from UNIX to NT this supporting infrastructure has to be available. (Unless you WANT to rewrite the entire application.)

As organizations are not abandoning UNIX for NT but rather opting for a mix of the two systems, the best porting strategy will allow you to maintain a single source base for your application regardless of the target system. (Break out the bucket of `#IFDEF`'s?) Rest assured though, there are solutions available that will allow you to CRACK that N*u*T.

The basic process of porting UNIX applications to Windows would look something like this:

- Gain access to the source code from your Windows system;
- Examine the make files, header files, and source code for potential problem areas;
- Compile and link;
- Address any compilation and linkage issues;
- Test and Debug your Win32 application;
- Port supporting scripts;
- Test and retest; and
- Distribute the application.

Some of the caveats that you will need to watch out for when considering a porting effort:

- Ensure that you have a suitable build environment and the appropriate build tools with which to maximize the effectiveness of your port.
- Ensure that you a) have adequate testing facilities or b) also port your testing application and environment to Win32 (the section on porting UNIX scripts may be of interest if your testing environment is heavily script based).
- Keep in mind the way UNIX and Windows systems deal with process management, security, files and I/O... (really there are too many points to list here, what I suggest is to check out the *Cross Platform Developers Guide* (www.mkssoftware.com). Not only does it offer comprehensive “How to” but also additional sources of information.)

And of course there will be subtle nuances to the porting effort when attempting to move daemons, threaded applications, or shared libraries. Not to mention if Windows 9x along with Windows NT/2000 is part of your porting strategies...

Integrating Applications With Win32

Each of the operating systems, Windows and UNIX, has its own unique strengths. The ability to capitalize on those strengths, within your environment and applications, is paramount to achieving this level of interoperability within your organization. The Windows security model, user and group structures, and most importantly, interaction with the abundance of third party applications available for Win32 are examples of natural integration points within the environment. In addition to the porting solutions you will require in moving applications to Win32, you will also need these tools to easily integrate the various APIs of these applications and the operating system itself.

Evolution Of Legacy Systems

At the apex of UNIX / Windows interoperability is not only the ability to build mission-critical Windows components from existing UNIX code, and not only integrate those components with current technologies, but also update, interact and embed these components in the infrastructures of tomorrow. COM, HTML and the Internet, and Java are taking the IT world by storm. You need the ability within your software and within your software development tools to completely integrate the Win32 world, as well as branch out and use these dynamic constructs in order to carry your organization and your enterprise applications forward.

Each organization has its own distinct set of criteria for its information systems and you will have to decide how far to delve into interoperability, both now and in the future. There are a number of solutions available but you must consider complexity, productivity, and cost issues involved with any choice you make. All you can do is remain informed on the issues and the solutions to help you make the right choice for your organization.