The Role of WINE in Linux Desktop Migration Strategies

March 2005

CodeWeavers

Overview: *With the increasing interest in the value that Linux can bring to the enterprise, companies need to assess likely migration strategies. Technologies that allow the reuse of Windows applications, and Wine in particular, are a key component to such migrations. This White Paper examines the requirements for enterprise migration on the desktop. It examines a full range of available tactics, including Wine, and then suggests strategies for making the journey in ways that are pragmatic, economical, and customer focused.*

Contents

Part 1 Setting the Stage: The State of the Linux Desktop

Why would anyone (in their right minds) want to change to Linux on the desktop? The reasons are manifold:

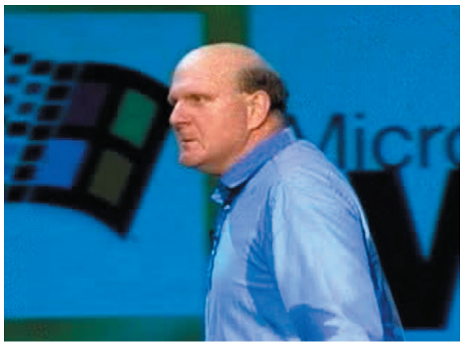
- It's cheaper
- It's more secure
- It gives users ultimate control over their own technology decisions.
- It removes the endless forced upgrade cycle that has been foisted on computer users for the past twenty years.

Desktop Linux adoption has been a slow, grinding battle.

Great. The question is, then, with all of this self-evident goodness, why hasn't the Linux desktop market gone crazy? Instead, what we've seen during the past three years has been a widening struggle to grow Linux' acceptance on the desktop, without much in the way of conspicuous success. Thus far, the two most readily discernible trends have been:

- It has been a slow, grinding battle. The rapid, explosive "take-off point" that all software vendors crave has thus far eluded desktop Linux providers. Total desktop Linux revenues thus far have been anemic—no one's buying any Ferrari's with their windfalls, that's for sure.
- Likewise, Microsoft, as is its tradition with any emerging threat to one of its core markets, has reacted with the near-pathological aggressiveness for which it is so justly noted. It forms no part of Microsoft's corporate doctrine to vie peacefully or fairly with competitors—they must be utterly crushed and humiliated, their houses knocked down, and salt sowed in the furrows.

  Thus far, while clearly falling short of this absolute victory, it's pointless to deny that Microsoft has succeeded in delaying the widespread adoption of desktop Linux through a combination of the technological maturity of its offerings, spending vast sums on marketing, establishing sales slush funds, and cleverly disseminating FUD in the marketplace. Oh, and when that doesn't work, they threaten to sue their customers. Nice guys.



It's safe to say that this man does not like Linux.

Let's face it: grinding, attritional battles are a drag. They're bad for morale. Everybody in the Linux community has been huddling in the trenches, staring at each other, and muttering, "When's it gonna *happen*? When will the skies open up, the heavens smile down upon us (virtuous and just warriors that we are), and the corollary money trucks start backing up to our gates?" That's a darned good question. And the likely answer is: Probably not any time soon.

Nevertheless, despite the bitter nature of this combat, certain long-term trends deleterious to Microsoft's position are discernable to the careful observer:

Attritional battles are a drag, and this is one we probably won't win any time soon. Nevertheless, there are some encouraging signs in the struggle.

- ❖ Despite Microsoft's best efforts, Linux on the desktop continues to grow. And not only that, it's improving. Today's Linux desktop offerings are vastly superior to the one available even two years ago. They're slicker, have nicer interfaces, and are better integrated. The switching costs for moving to Linux—measured in both pain and dollars—are coming down all the time.
- ❖ Microsoft is increasingly relying on the implied threat of intellectual property and other legal complications as a way of retarding Linux' growth. This is good. It means Microsoft is running out of legitimate ways of defending its market, and is resorting to threatening to sue its customers instead. That strategy may play for a time, but in the long-term it's a bankrupt approach to doing business. Microsoft's falling back to this legal defense line is truly a last ditch effort.
- ❖ All the marketing dollars in the world don't change the fundamental fact that personal computer operating systems are a commodity product and should be priced as such. Likewise, there's no amount of Microsoft price support slush funding that will keep Linux off the desktop indefinitely, particularly in developing nations. Third world countries rightly recognize that they have a national imperative to move large numbers of cheap computers into their populations in order to improve their standards of living. They don't care what Microsoft has to say about it—they're going to do that with Linux.
- ❖ Likewise, even wealthy nations such as France, Germany, and other European countries have looked to Linux as a way of removing a perceived U.S. technological hegemony from their borders, particularly as it relates to the operations of their national governments. These countries have mandated initiatives to go to Linux, Microsoft be damned.

The fact that Microsoft is relying more and more heavily on the implied threat of legal action against its own customers means that it is desperate. They are in last-ditch defense mode.

Bottom line: What we are witnessing is a slow, inside-out computer technological revolution, wherein events *outside* the traditional center of gravity in the IT industry (i.e. the U.S.) are going to eventually drive adoptions *inside* the U.S. as well. And while the battle right now is ugly and grim and bloody, Linux is winning. It may not *look* that way on the outside. And it sure as heck doesn't *feel* that way here in the trenches. But fundamental paradigm shifts are rarely self-evident or immediate. We're winning. And what's more, Microsoft is running scared.

Reaching the Tipping Point

What does this mean? It means that at some point in the relatively near future, a few large, high profile organizations will begin making the transition to Linux desktops. IBM and Novell both have major initiatives underway to convert thousands of their own seats to Linux. That's only to be expected of these vendors—they must "eat their own dogfood" in order to establish the credibility of their products with their potential customers. Beyond this, though, we also expect to see a small, but growing number of high-profile Fortune 1000 customers beginning to make their own enterprise migrations. That will represent a tipping point for adoption.

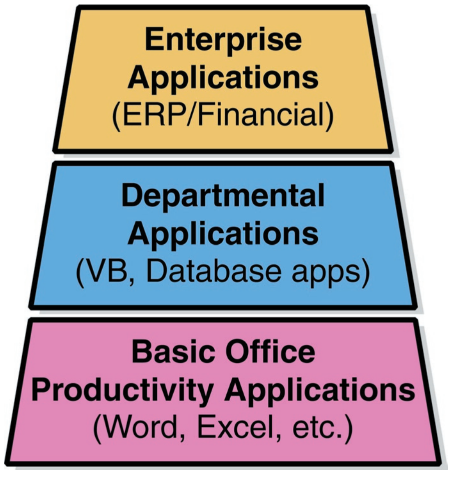
At some point in the relatively near future, a few large, high profile organizations will begin making the transition to Linux. When that happens, enterprises will begin wrestling with the issues associated with making their own journeys to the Linux desktop.

The Need for Legacy Support

If this tipping point is reached, then it is reasonable to assume that enterprises are going to begin wrestling with the issues associated with making their own journeys to the Linux desktop. One of the thorniest of these issues revolves around Windows legacy application support and integration across the entire stack of applications within the organization. It is inconceivable that the average enterprise is going to cast out all of their Windows applications wholesale. Instead, an intelligent strategy for rationalizing and prioritizing the disposition of these applications is called for. Building such a strategy comprises the bulk of this whitepaper.

Part 2 Evaluating The Enterprise Application Stack

The topic of Linux desktop migration is understandably very complex, as it can potentially touch every facet of the organization, thereby unleashing wide-ranging changes on both the company's user population, its IT infrastructure and staffing needs, and even the internal business processes of the company. For the purposes of this paper, though, we choose only to examine the core issues around migration, namely, figuring out the best migration choice(s) to take for a given set of applications, and turning that set of tactics into a sound migration strategy that works for the organization as a whole.


Enterprise Applications (ERP/Financial)
Departmental Applications (VB, Database apps)
Basic Office Productivity Applications (Word, Excel, etc.)

Enterprise application stacks will need to be addressed by tailored migration strategies.

Types of Legacy Applications

Desktop application migration will require attention to three basic levels of applications within the organization. These can be broadly described as:

- **Enterprise-wide applications**, including ERP, financial, and other specialized systems that can be ubiquitous within a given organization.
- **Departmental applications**, which are less widespread within the organization (help desk, provisioning, etc), but which often represent critical components of both the company's business processes, as well as its IT infrastructure.
- **Office Productivity applications**, including email, word processing, and other applications which perform the bulk of the mundane productivity needs of the organization. Practically all workers will have some of these applications on their computer.

Each of these basic application types has certain characteristics which impact the likely migration strategies that an organization can choose to pursue. These can be summarized as follows:

Enterprise Applications

- Can be widespread, and impact many functional areas within the organization. As such, these applications have a very wide range of audiences, with different needs within each audience
- Relatively deep functionality, rich interfaces
- Very high implementation/switching costs

- Typically provided by a packaged software vendor, and in some cases heavily customized by third-party vendors, but also often developed by in-house IT staff

Enterprise applications are a logical focal point for migration efforts, in that they represent the high-profile backbone of any company's IT infrastructure. Likewise, they are often very complex applications. Yet in many cases, as we shall see, they are easier to migrate than departmental apps.

Departmental Applications

- By definition, they are less widespread than either Enterprise or Productivity applications
- Often built on top of mid-range database technology—MS Access or SQL Server, FoxPro, Powerbuilder, etc.
- Often "home-grown", rather than packaged software
- Often locally supported, rather than vendor supported.
- Lower switching costs, but still vital to the department which they serve.

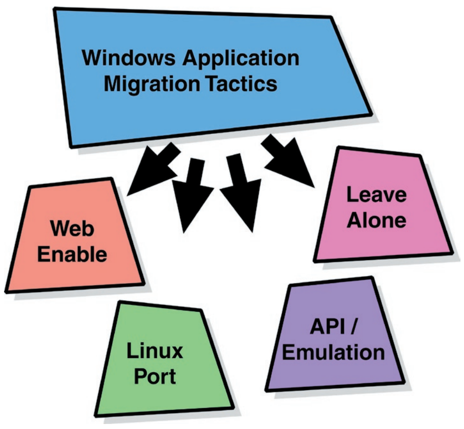
Departmental applications often present the stickiest migration problems. In a nutshell: how does a migration plan cope with the hordes of Visual Basic database applications, and older 2-tier client/server apps built in tools like PowerBuilder that are endemic to the corporate desktop? Most companies don't like to admit it, but the reality for many organizations is that these home-grown, departmental programs are sometimes more important to running their day-to-day operations than the vaunted ERP systems that supposedly form the core of their operations. How is one to reasonably to migrate all these applications to a new corporate Linux desktop without 1) breaking the bank, 2) taking until the mid-23rd Century to complete the migration, and 3) having one's IT staff kill themselves (or you) along the way? Answering these questions requires a skillful mix of migration tactics.

Office Productivity Applications

- Ubiquitous—practically every knowledge worker within an organization has a word processor and email, at least.
- Overwhelmingly shrink-wrap software
- Relatively low switching costs in terms of dollar figures, but *high* switching costs in terms of user acceptance. Everyone has used MS Word, and few knowledge workers really want to switch these applications, even though in many cases they will accept a new departmental application without any fuss whatsoever.

Productivity applications are widespread. Thankfully, many of them have useful Linux equivalents. For those that don't, other approaches must be employed.

Part 3 Migration Strategy and Tactics



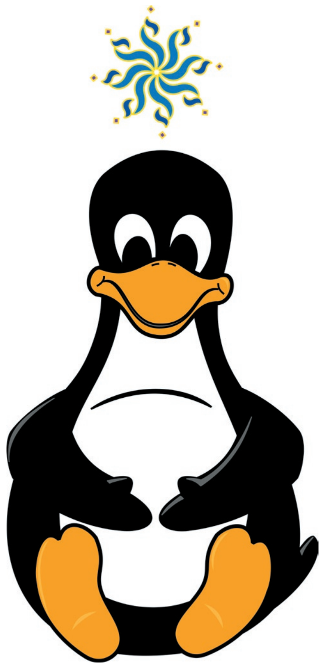
Corporations have four basic tactics they can use as part of the overall migration strategy.

A migration strategy can be thought of as a master game plan for moving the whole organization into a new desired state. As such, the overarching strategy comprises a set of individual approaches (which we'll call "tactics") for moving each needed application off of Windows and onto a Linux desktop. For any given Windows application, the enterprise has essentially four migration options available to them. These are:

- **"Webify" the application.** In other words, migrate the legacy application to a web-deployable form, thereby making it platform independent.
- **Perform a Linux port.** Develop a Linux client for the application and/or move the application wholesale onto a Linux platform.
- **Use an emulation or API technology** to allow the Windows client portion of the application to run on a Linux client. These solutions break down into two basic camps—emulators (such as VMWare), and API technologies, such as Wine. Each has pros and cons that are discussed later in this paper. But for the sake of analysis, we will assume that Wine is the emulation technology selected.
- **Continue running under Windows.** This can be either a transitional plan, or it can be a long-term solution to create pockets of Windows usage within the larger pool of Linux desktops. In other words, it is the mirror image of the current situation on the desktop, wherein pockets of Mac or Unix desktops co-exist within a sea of Windows desktops.

Each of these tactics makes sense in certain situations, and each has a part to play in a holistic migration strategy.

Part 4 Migration Philosophy



“In the world there are many different roads, but the destination is the same. There are a hundred deliberations, but the result is one.”
—Confucius, *I Ching*

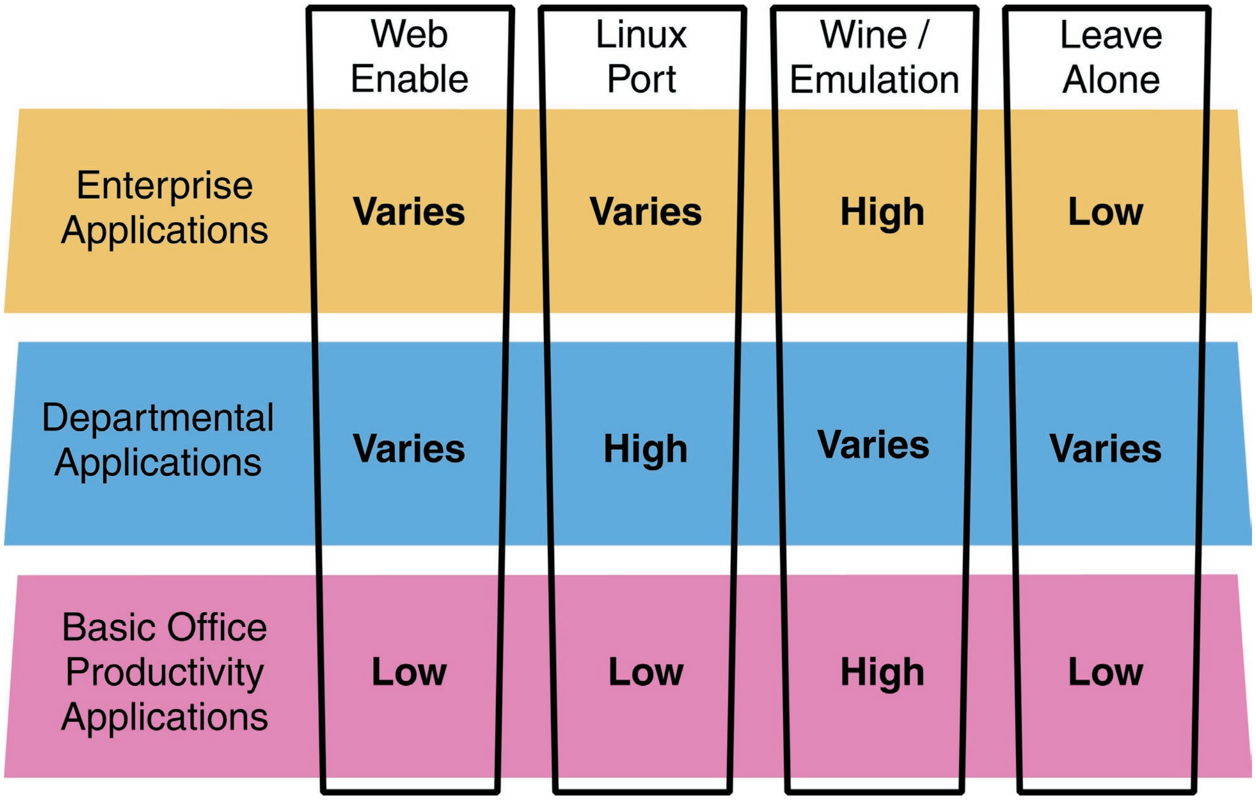
In CodeWeavers’ opinion, the desktop Linux marketplace, and technology consumers as a whole, are best served by a migration philosophy which is pragmatic, economical, and driven by customer (not vendor) needs. We believe that a careful, informed blending of the four basic migration tactics is the correct path to creating a high-quality corporate migration strategy.

Conversely, we *deeply* mistrust plans of attack which rely too heavily on a “rip and replace” tactics. We believe that these are primarily motivated by vendors’ desires to sell new hardware, software, and consulting services, rather than by a genuine desire to meet customer needs. While replacing old systems is sometimes useful, it should also be viewed as a last resort, not as the universally to-be-preferred option.

Now, obviously, CodeWeavers would like to sell some software, too. On the other hand, we’d also be the first to tell you that Wine has its limitations, and is far from being some sort of magic silver bullet that slays all the Linux migration dragons you’re likely to encounter on your journey. Frankly, *all* of the basic migration tactics have their drawbacks, and there is no one-size-fits-all tactic that will get the job done. As such, it is important to approach the strategic migration process honestly and carefully, examining the strengths and weaknesses of the various tactics in relation to specific scenarios.

Part 5 A Matrix of Migration Options

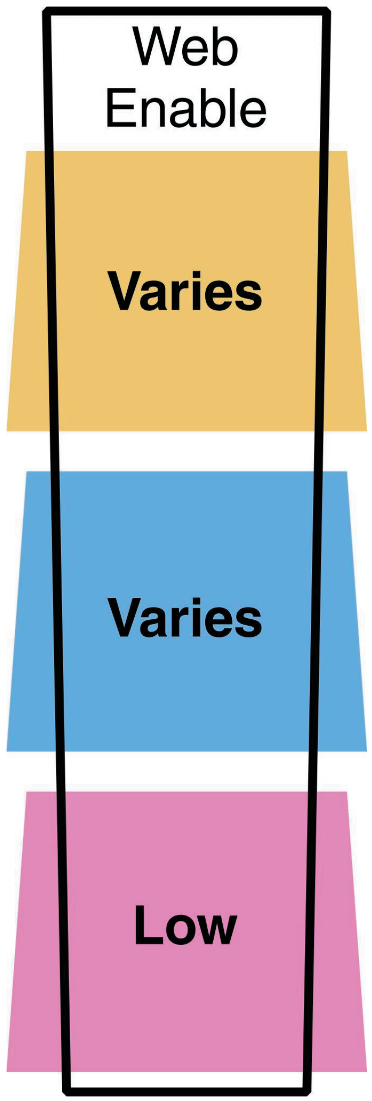
Having now described the basic types of applications within the corporation, the basic migration tactics that can be used, and our overall philosophy towards crafting a migration strategy, we turn to how applications interact with specific tactics. The following matrix illustrates these relationships, and provides a general rating of the applicability of the tactic to the application type.



	Web Enable	Linux Port	Wine / Emulation	Leave Alone
Enterprise Applications	Varies	Varies	High	Low
Departmental Applications	Varies	High	Varies	Varies
Basic Office Productivity Applications	Low	Low	High	Low

Obviously, as with any set of generalizations around a topic as complex as this one, values like “High” and “Low” don’t adequately describe the solution set. Nor do they account for the inevitable exceptions to the rule. We elaborate on each of the scenarios in more detail below.

Web Enablement Tactic



Web enablement is the preferred migration tactic of larger vendors like IBM. Funniest thing; they sell a lot of WebSphere to their Linux customers. Now, we think that Web enablement is a great way to go for many applications. But it is not a be-all end-all solution, and it needs to be used appropriately, not in shotgun (wedding) fashion.

Enterprise Applications

Enterprise applications can be broken down into two broad categories—commercially available packaged software applications, such as SAP, Oracle, etc., and home-grown applications that are unique to the customer. The applicability of Web enablement varies according to which type is being discussed.

Packaged Software: Web clients for many enterprise applications are already available. These interfaces run the gamut from lousy to great. Due to the constraints of the medium (i.e. the vagaries of Javascript, or the tradeoffs inherent in using large quantities of proprietary ActiveX controls, etc.), they may not be as rich as the native Windows client. In some cases this may not be an issue, but it is not uncommon for users to resist using such client software if they feel that they are receiving less robust functionality than they are used to. Generally, though, using the vendor-provided Web client is an inexpensive solution to the migration issue.

In-house Developed: Webifying home-grown applications is feasible, but a high level of effort is typically required, and the resulting interfaces often are relatively kludgy. Older legacy applications that were never developed with the Web in mind are notoriously difficult for developing new interfaces. Likewise, enterprise apps tend to be big and tricky, with lots of screens, making their redeployment tedious. On the other hand, a variety of bridgeware and screen-scraping tools have alleviated this problem somewhat.

Departmental Applications

Just as with enterprise applications, departmental systems come in two flavors—packaged, and home-grown. The same general comments apply to their Webification as with their big brothers.

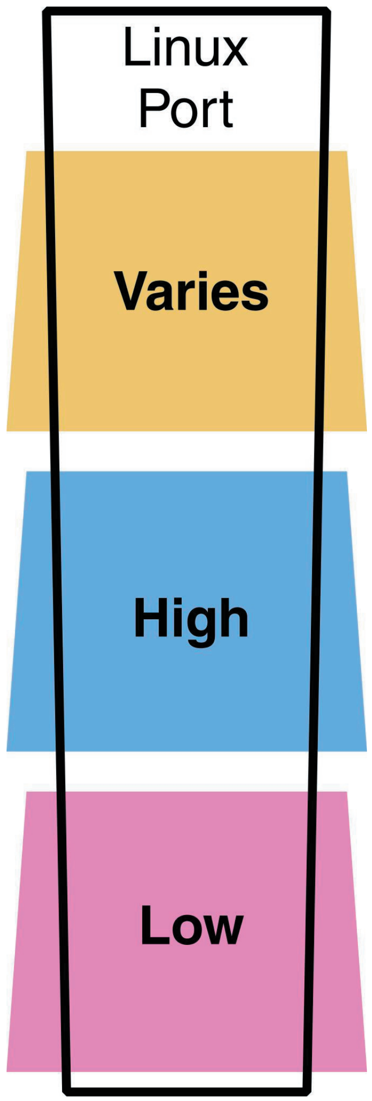
Packaged Software: A Web client may already be available from the vendor, otherwise developing your own is nearly impossible. The same hurdles to user functionality pertain as with larger applications—the Web client may be quite good, or it may not offer the same level of functionality. However, if the Web client is robust, this can be a very solid choice.

The applicability of Webification in many cases depends on the underlying architecture of the application, and whether or not a new, and sufficiently robust presentation layer can be bolted on economically.

In-house Developed: This can be a very good option, too, particularly if the apps' original architecture is sound. Departmental apps are typically smaller in scope, and therefore require less effort to Web-enable with a new presentation layer. Hopefully. On the other hand, if this application doesn't serve the needs of that many users, this can be expensive on a per/head basis.

Office Productivity Applications

These applications are mass-produced by well-known Independent Software Vendors (ISVs), such as Microsoft, Lotus, Adobe, etc. Only a fool would try to replicate their extremely deep functionality on a new home-grown Web client of some sort. Now, some of these apps (such as Notes) already come with a web client (with the same caveats as noted above), but most do not. Basically, you're out of luck on this one.



Linux Portation Tactic

Moving existing applications to a native Linux format brings with it obvious benefits. The applications are built from the ground up to operate in the Linux environment, and take advantage of Linux's benefits. This, too, can be a very viable option for a number of different types of applications.

Enterprise Software

Packaged Software: Unless a Linux version of the application is available from the vendor, this option isn't really possible. The vendor isn't typically going to give you the source code, after all. Many of the larger applications will already have Linux *server* components, of course, but Linux client versions have been much slower in coming. If you've got major clout with the vendor, by all means use it, but don't get your hopes up.

In-house Developed: Linux portation of existing applications is feasible, but a high level of effort is typically required. Unless the application was developed with good abstraction between the business logic and the database layers, such efforts will require not only a re-write of both, as well as a massive data-portation effort. All in all, this typically requires more time and money than trying to Webify the app, because you're having to deal with at least two, and possibly three layers of the app—Logic, Data, and Presentation. On the other hand, if the application is valuable enough to the organization, a re-write in a new environment may bring with it new functionality and features (as well as better stability, uptime, scalability, etc.) which may justify the effort. All in all, though, this road is not to be undertaken lightly.

Departmental Applications

Packaged Software: Unless a Linux version of the application is available, this option isn't really possible.

In-house Developed: The same caveats apply as to in-house enterprise apps. Upside: you potentially get exactly what you want. Downside: it can take a lot of work to get that, and you need to balance those costs with the likely benefit to the target user base, and the company as a whole.

Office Productivity Applications

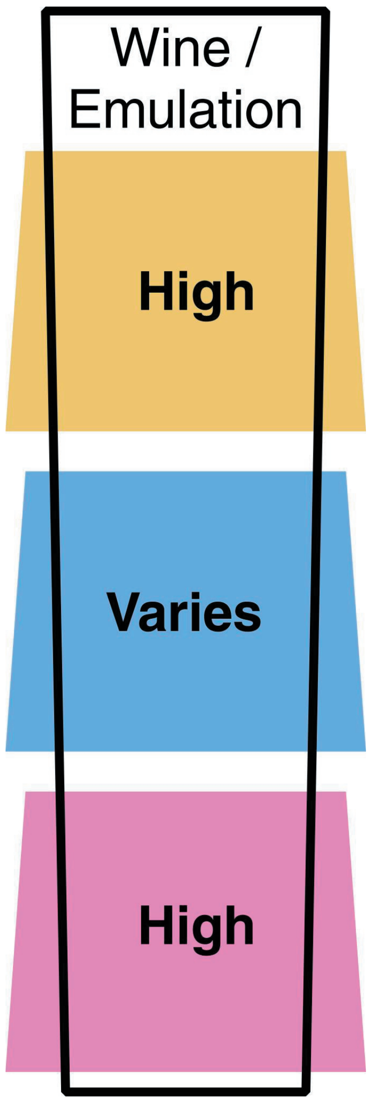
In this case, we're really talking about using a native Linux application of some sort, because (just as with Web enablement) you probably don't want to develop, say, a Linux word-processor from scratch. Now, for some applications, using a native Linux version can be the way to go. For instance, for basic office word-

Native Linux ports can provide some of the best upside, but in many cases also entail the most work.

processing, spreadsheets, etc., using OpenOffice or StarOffice is a very viable option. Many other apps, though, don't have Linux versions (ahem, that's why CodeWeavers is in business). As such, you may have to learn to live with an open-source equivalent (i.e. use The Gimp instead of Photoshop, or Firefox in place of IE). In some cases, that may be feasible, in other cases, not.

An even more esoteric option is to get involved with an open-source project that is developing the application you need (such as The Gimp) and contribute enough effort to it that you can turn it into the product that *you* need. This can require developing soft skills in cat-herding, not to mention accepting a certain amount of elbow grease, but the results can be very satisfying, not to mention beneficial to the open source community.

Wine/Emulation Tactic



Again, for the sake of argument, we're going to assume that you're interested in potentially using Wine as your bridge tactic. Why? Because all the other emulators require you to have a fully licensed version of Windows running on your Linux box. Like as not, if you're interested in undertaking this journey to the Linux desktop, you probably want to put as much distance between yourself and Microsoft's DARK, EVIL KINGDOM as possible. Leaving Windows on all of those boxes seems to sort of defeats the purpose, don't you think? However, for the sake of fairness, we discuss the tradeoffs of Wine vs. other emulation solutions at the end of this paper.

Enterprise Applications

Packaged and In-House Developed. This can be a very feasible option. Indeed, in many cases, the cost of getting the client application to run under Wine will be far less than Webifying it or redeveloping the application client from scratch. Indeed, for packaged software, this may be the only way to go if you're determined to get the depth of functionality you've come to expect from the native Windows client.

On the other hand, be aware that Wine is very tricky and expensive to develop in. CodeWeavers estimates that fixing a single bug in Wine costs at least $1,000. So if you have a lot of application screens, things can become quite costly. Not only that, but it is usually very difficult to achieve absolutely the same level of perfection in the Wine-enabled client as the native Windows client. You'll need to make some cost/benefit calculations—can you live with that one pink checkbox that blinks occasionally on that one screen, or will the AP department strangle you? If you have a large user population, you can probably justify a higher degree of perfection, because you can amortize that cost across a greater number of users. Generally, the less complex the application is, and the larger the user base, the better the case for Wine becomes.

Departmental Applications

The same caveats apply to Departmental as to Enterprise applications, except that the cost/benefit choices become trickier. The same cost structure for Wine development remains in place, i.e. it can be quite expensive. Yet the likely user population is smaller by definition. The question is finding a healthy balance between amortized per/user cost, functionality, and overall fit with the corporate Linux migration strategy. In other words, Wine can be a very solid choice in this arena, or a complete non-starter.

Users should also note that home-grown applications have at least one advantage over packaged software in this respect—access to the application source code. When developing in Wine, having the ability to understand what the source code is trying to do can be a huge advantage. In many cases, a quick phone call to a developer that understands the code, and can clarify what FunctionCallX is expecting as a return value, can save a Wine developer many frustrating hours of trial and error trying to reverse engineer what is happening instead of FunctionCallX's little black box.

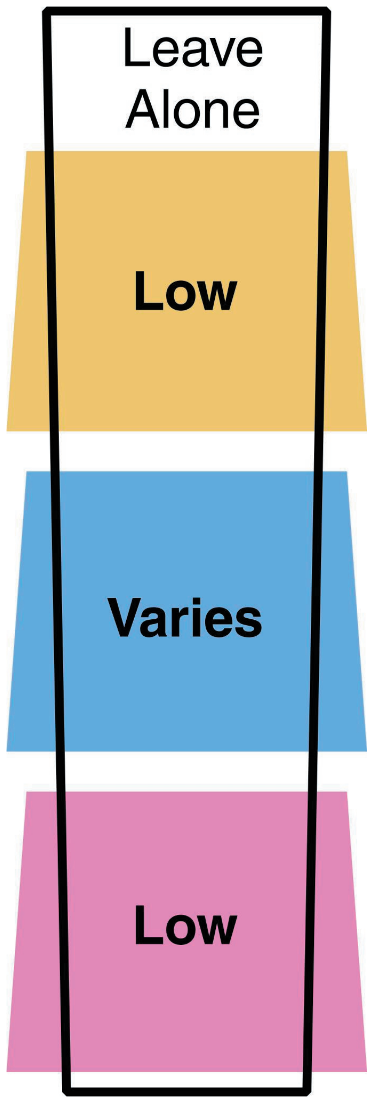
WINE (Wine Is Not an Emulator) is a crucial technology for Linux desktop migration. In the right situation, it can provide the best performance and most pleasing user experience of any of the migration options.

Office Productivity Applications

Using Wine for these applications is quite feasible. Wine already runs many of the more popular office apps, such as MSOffice, Lotus Notes, Dreamweaver, etc. Not only that, but it runs them quite nicely in many cases. So if your users are going to string you up by your thumbs unless they can use Outlook for their email, well, hey; give the people what they want.

The downside, of course, is that there are many applications that Wine doesn't support (at least not yet). These include any number of applications such as utilities, development tools, and so on that may have a very high utility for a small number of people, but don't really justify the cost of Wine. This is a conundrum—it's hard to make the transition to Linux without being able to drag along a great number of these applications with you. The good news to this, though, is that Wine is getting better all the time. As a result, more and more applications are spontaneously beginning to work under Wine without any conscious effort being made—as Wine's water level rises, more and more applications will begin running "out of the box." The only question is how long you might have to wait.

"Leave It Alone" Tactic



As we mentioned previously, we think that leaving a certain number of Windows workstations intact in the company can be a perfectly valid approach to the larger topic of migration. Unlike Microsoft, we don't feel there's any need to hunt down and annihilate all traces of competing operating systems. We won't send the Penguin Police after you. After all, the open-source movement is all about giving people choices again. Likewise, you've probably been living with a certain number of Macs and/or Unix workstations in your company for years. Furthermore, using a product such as Windows Terminal Server and/or Tarantella or Citrix, you can even choose to isolate the legacy Windows applications in a very managed environment. The question is, though, where do you draw the line in terms of numbers of Windows machines?

Enterprise Applications

Chances are, if you're going Linux on the desktop, you've got to move your enterprise applications over somehow. Otherwise, what's the point, as you are going to be leaving a huge number of desktops out there running Windows? For that reason, leaving enterprise applications alone really isn't a viable option.

Departmental Applications

Departmental applications can be a different story from enterprise apps, in that they involve smaller user populations, as well as sometimes very thorny development issues for the other migration tactics (web enablement, portation, and Wine). In selected cases, then, it may make good sense to keep a group of Windows users where they are, at least until a suitable migration tactic can be identified. This choice, of course, must be weighed against such factors as support costs, hardware costs if Windows upgrades will be required in the future, etc. In general, this tactic should be applied only sparingly, and only to those users for whom the other migration tactics would impose an undue burden.

There's no need to be dogmatic about eliminating all traces of Windows. The Penguin Police won't come after you. The trick is drawing the line intelligently.

Office Productivity Applications

Just as with enterprise applications, the whole point of the Linux migration process is to move as many users onto Linux as possible, and leaving everyone on Windows so that they can run Outlook defeats the purpose. Given both the open-source solutions available, as well as the applicability of Wine for running many of these applications, there should be a strong incentive to find solutions that don't require leaving vast swathes of the user population on Windows.

Part 6 Choosing the Right Emulator

As we've seen, emulation solutions play a valuable role across all three levels of software applications. They can be a cost-effective strategy for porting enterprise applications. Likewise, on the lower end of the spectrum, they can reduce switching / re-training costs for workers by allowing them to use the same software that they're already familiar with. The question then becomes, which emulation strategies make the most sense?

There are two major "flavors" of emulation—virtual machine, and Wine. Each has pros and cons.

Contrasting Types of Emulation

There are two major competing flavors of emulation—virtual machine-based emulation, and API re-implementation. Virtual machine emulation is predicated on running a copy of software on top of a fully licensed version of the Windows operating system while being hosted in a different environment (like Linux). Examples of this approach are products like Win4Lin and VMware. API reimplementation (at this point in time) essentially means using Wine, which is a re-implementation of the Win32 API under Linux. There are pros and cons to each approach, depending on the application that needs to be run.

Technology Approach

Both VMWare and Win4Lin are true emulators. In other words, Windows applications are actually run under Windows in a separate virtual machine on the client PC, and the emulator handles displaying the applications under Linux. This arrangement can be thought of as a "box within a box," wherein Windows runs inside a virtual sandbox within Linux.

Wine, on the other hand, is a complete re-implementation of the Win32 API under Linux. As such, running Windows applications does not require having a Windows OS running in the background. Instead, Windows applications run "as if natively" directly under Linux. This provides the user with an immediate cost savings associated with the purchase of the Windows OS, and has other implications as well.

Memory Allocation

Most emulation technology requires a dedicated block of memory to run the emulator. It is not uncommon, for instance, for a VMWare session to soak up 128MB of system RAM. This

RAM is used exclusively by the emulator as long as it is run, regardless of whether any applications are being run or not. Wine, on the other hand, consumes RAM dynamically, based on the application's needs at the time.

Ease of Use / Convenience / Integration

Wine has the advantage of less overhead, superior performance, and much lower cost. However, it doesn't run as many applications (at least not yet).

One of the practical outcomes of an emulator approach is that the emulator by necessity "carves out" an environment that is somewhat separate from the Linux OS on the host PC. For instance, in order to establish a VMWare session, a user literally has to boot Windows in a separate partition and wait for it to load before then loading the actual application. Given the memory issues just discussed, it is impractical for many users to simply keep VMWare running in the background at all times, meaning that the user suffers this startup penalty whenever s/he uses the software. By contrast, Wine suffers from no start-up lag for the OS—the only delay is the time it takes for Word to load.

Similarly, in terms of file integration, an emulator solution lives in its own sandbox as far as file-sharing is concerned. As a result, moving files in and out of VMWare requires mounting Samba drives and so on. With Wine, the applications run natively under Linux file system, and files can be saved and moved around normally within the OS.

System Performance

Emulation technology typically imposes a penalty in terms of performance. Wine does as well, but tends to exact a smaller premium, meaning that it offers very good performance to the average office user. In most cases, this performance will not be as good as native Windows. However, the code is being run at native speed, and hence in theory can be made to run just as quickly as native Windows. The same is not true of emulation solutions, which typically have much slower overall performance.

Cost

Emulators like VMWare tend to be pricier than Wine (which is free) or commercialized versions of Wine such as CrossOver Office. VMWare retails at around $300 per workstation; Win4Lin for around $90. In addition, each of these solutions requires a WindowsXP license, which retails at $200. In other words, complete emulation solutions retail for between $290-$500. Basic Wine is free, and some versions of CrossOver Office retail for as little as $40 per workstation, with no need for a copy of Windows.

Application Footprint

This is the current major shortcoming of Wine, which does not yet support as wide a range of Windows applications (nor with the same degree of fluidity) that emulators do. This is because emulators truly run Windows applications natively, whereas Wine currently has not completely re-implemented the Win32 API. As a result, emulators are still often a better choice when a user needs to run a very wide range of Windows software.

For this same reason, CodeWeavers has focused its efforts on running a discreet set of high-value office productivity applications like MSOffice. The currently supported applications for CrossOver Office includes:

- ❖ Microsoft Office 97, 2000, and XP, including Microsoft Word, Excel, Powerpoint and Outlook
- ❖ Internet Explorer
- ❖ Microsoft Visio
- ❖ Intuit Quicken
- ❖ Lotus Notes 5.5 and 6.5
- ❖ Microsoft Viewers (Word, Excel, Powerpoint)
- ❖ Macromedia Dreamweaver
- ❖ Apple iTunes
- ❖ Adobe Photoshop and Framemaker

Wine's application footprint is expanding all the time. Any Windows application can be made to run well under Wine, and eventually *all* apps will run.

Many other applications run quite well, but are not officially supported. More information on CodeWeavers current list of supported and unsupported applications can be found at:

http://c4.codeweavers.com

The bottom line is that *any* application can be made to run under Wine. Not only that, but eventually pretty much *all* of them will run pretty well—Wine continues to improve all the time, and its application footprint expands accordingly. We're still a ways away from that hallowed land, but it's coming.

Making the Decision

Not surprisingly, picking the right emulation technology involves making some of the same choices around migration tactics as a whole. These factors include:

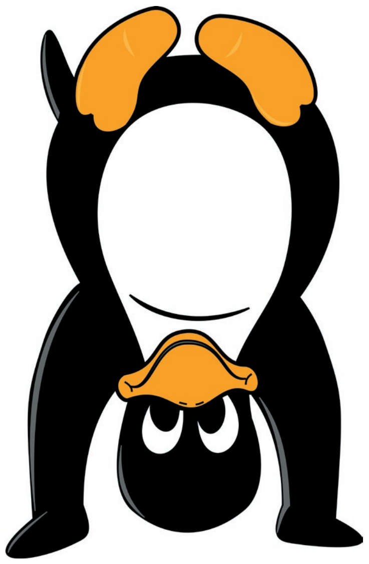
- ❖ Licensing costs and support costs
- ❖ Performance requirements of the user population (i.e. can they live with the slower performance the stems from emulator solutions?)
- ❖ Technical proficiency of the user population (i.e. do they feel comfortable using an emulator, or would direct integration of the Windows applications into the Linux desktop (a la CrossOver Office) provide a superior user experience?)
- ❖ The breadth of the application set to be supported

It should also be borne in mind that different emulation solutions may make sense in different parts of the organization. Wine or CrossOver Office may work well for 80% of the organizations knowledge workers, leaving a smaller group with a very wide range of Windows software needs either to run VMWare, or perhaps be left on Windows.

Part 7 Conclusion

Like any major enterprise IT transition, migrating to the Linux desktop presents both opportunities and challenges. With a proper mixture of tactics, and a pragmatic approach, your organization can experience the benefits of Linux without being sucked into rigid, expensive rip-and-replace exercises. In our opinion, pragmatism should override dogmatism in all cases. Making good choices for the particular needs of *your* organization is far more important than adhering to some sort of technological agenda. Open-source technology is all about choice, and you should use this opportunity to develop the maximum power for your organization. Go forth and conquer!

If you'd like to talk to CodeWeavers about how we can help you with your own migration needs, or how Wine and CrossOver can be a part of your larger Linux migration strategy, give us a call. We're always happy to talk.



Pragmatism should override dogmatism. Adopting open-source is all about obtaining choices, and returning power from vendors to buyers. It's turning the IT world upside down!

Jeremy White
President/CEO
jwhite@codeweavers.com

Jon Parshall
COO
jparshall@codeweavers.com