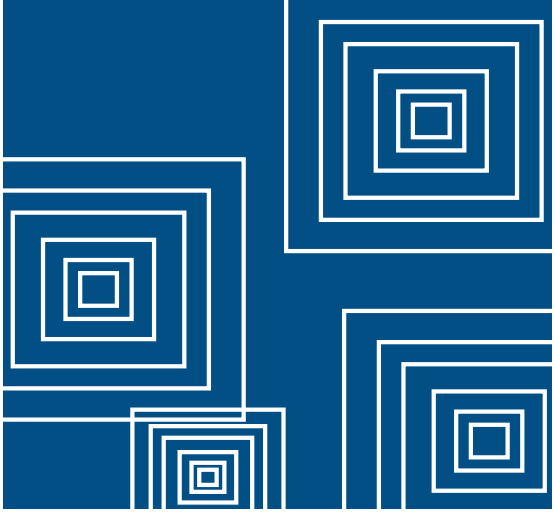
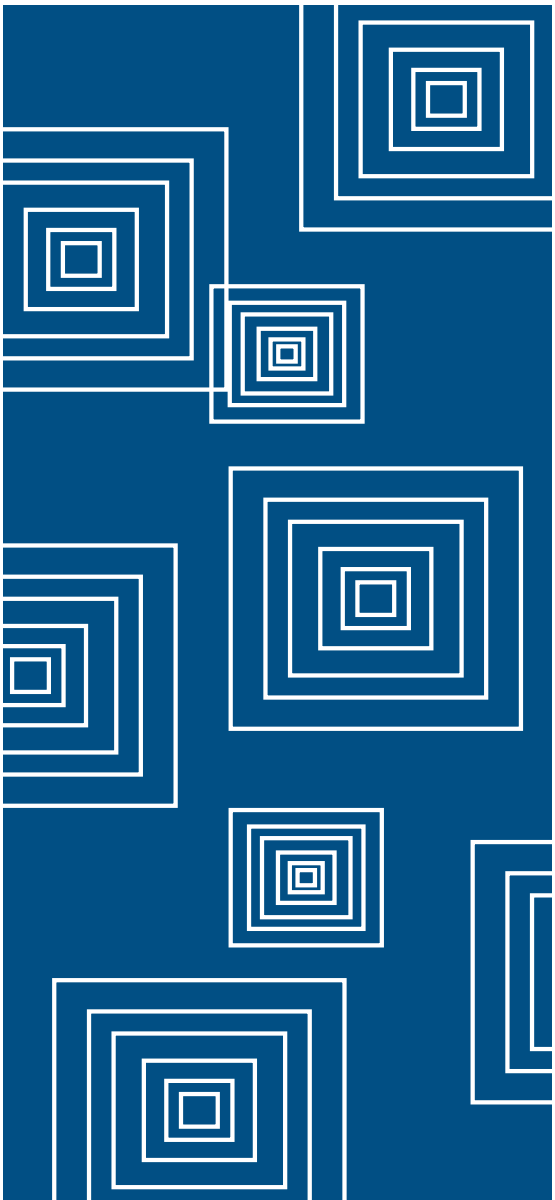


Anwendungsmigration mit Wine



Bei der Migration von Windows-Applikationen stößt man fast zwangsläufig auf Anwendungen, die nicht unter Linux lauffähig sind. Zudem ist dann oft auch deren Quelltext nicht verfügbar. Einen eleganten Weg, sie dennoch mit dem Pinguin zu verheiraten, bietet oft Wine. David M. Gümbel

Wine ist einem populären Missverständnis zum Trotz kein Emulator und wollte auch nie einer sein. Stattdessen handelt es sich um den Versuch, die Windows-API unter Unix nachzuimplementieren – also die Gesamtheit der Funktionen, die insbesondere das 32-Bit-Windows Applikationen zur Verfügung stellt. Wine unterstützt dabei nicht nur Linux, sondern grundsätzlich auch andere Unix-Derivate wie verschiedene BSD-Varianten oder Mac OS X.

Zu Wine gehört einerseits ein Loader, der unmodifizierte Programme in binärer Form in den Speicher lädt und ausführt. Andererseits enthält Wine auch eine Bibliothek – die sogenannte Winelib – die alle nachimplementierten APIs in C/C++-Programmen nutzbar macht. Wer über den Quelltext eines Windows-Programmes verfügt, kann sich so mit wenig Aufwand eine native Version erstellen.

Mit Wine lassen sich also Programme, die auf der 16- oder auf der 32-Bit-API von Windows aufbauen, in unmodifizierter Version direkt unter Linux ausführen (**Abbildung 1**). Dafür braucht man keine Windows-Lizenz und die Programme sind genau wie eine native Linux-Applikation mit ihren Icons, Fenstern, Menüs und der Zwischenablage in einen Gnome- oder KDE-Desktop integriert.

Stärken und Schwächen

Zu den unbestreitbaren Stärken von Wine gehört, dass es im Prinzip möglich ist, jede Windows-Applikation unter Linux zum Laufen zu bringen, und dies sogar in unmodifizierter, rein binärer Form. Zudem ist der Ressourcenverbrauch der Applikation durch den Betrieb unter Wine nicht oder unwesentlich erhöht, da sie Wine letztendlich nativ ausführt.

Wine lässt sich zudem intelligent mit anderer Software kombinieren: So ist es zum Beispiel möglich, ein durch Wine unter Linux ausgeführtes Programm per NoMachine-Terminalserver auf beliebige Clients zu transportieren. Nutzt man die GPL-lizenzierte Variante FreeNX, so hat man auf diese Weise einen lizenzkostenfreien Terminalserver für das Windows-Programm (**Abbildung 2**).

Eine der wesentlichen Schwächen von Wine ist allerdings ebenfalls prinzipbedingt: Da es sich um eine Nachimplementierung der Windows-API handelt, hinkt Wine grundsätzlich den neuesten Windows-Versionen hinterher. So sind von den circa 20.000 APIs in Wine nur etwa 7.000 überhaupt in irgendeiner Form nachprogrammiert, und auch das nicht unbedingt immer vollständig. Oftmals finden sich lediglich Teilimplementierungen oder gar

Historie des Projektes

Als das Wine-Projekt 1993 startete, war das erklärte Ziel, die Win16-API zu implementieren. Heute steht jedoch die Win32-API klar im Vordergrund. Lange Zeit befand sich das Projekt im Alpha-Stadium, weshalb kein Release veröffentlicht wurde. Es erschienen lediglich monatliche Snapshots, die als Versionsnummer ihr Datum trugen. Alle Releases seit dem 28. 2. 2002 stehen unter der LGPL-Lizenz. Zuvor benutzte Wine die liberale MIT-Lizenz, was dazu führte, dass verschiedene Forks entstanden, die aber wenig oder nichts von ihren eigenen Weiterentwicklungen an das Wine-Projekt zurückfließen ließen.

Ende 2005 erschien nach langer Zeit eine erste Beta-Version mit der Versionsnummer 0.9. Insgesamt 61 weitere Beta-Releases folgten, bis die Entwickler schlussendlich nach gut 15 Jahren Entwicklungszeit am 17.6.2008 die Version 1.0 freigaben.

Zu den vorher definierten Anforderungen, die das erste Wine-Release erfüllen sollte, gehörte vor allem die Ausführbarkeit bestimmter, besonders bedeutsamer Windows-Applikationen wie Photoshop CS2 oder MS Office-Viewer. Außerdem wollte man sicherstellen, dass künftige (stabile) Versionen von Wine möglichst nicht mehr an Applikationen scheitern sollten, die mit einer früheren Wine-Version bereits einmal lauffähig waren. Bei einem Projekt mit gut zwei Millionen Zeilen Code, das dabei auch noch die Kompatibilität mit Windows-Versionen zwischen 2.0 und Windows 2008 herstellen will, ist das keine ganz triviale Aufgabe.

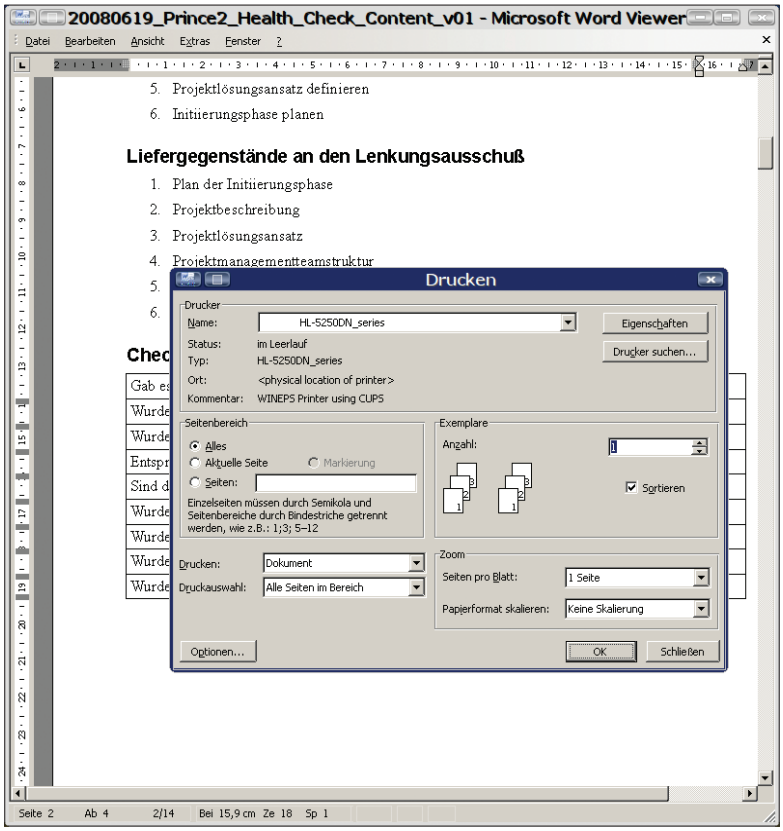


Abbildung 1: Der Windows Word-Viewer – hier mit geöffnetem Druck-Dialog – läuft dank Wine ohne Weiteres unter dem Linux-Desktop.

sogenannte Stubs – Funktionen mit leeren oder fast leeren Rümpfen.

Dieses Problem ist in der Praxis jedoch weit weniger gravierend, als es klingt: Erstens nutzen auch große Programme wie Microsofts Office XP lediglich rund 1.500 APIs. Zudem lassen sich die Lücken in der Implementierung bestimmter APIs oftmals durch eine weitere Funktion von Wine schließen: der Einbindung nativer Windows-DLLs anstelle der Nachimplementierungen. Dies ist nur bei nicht systemnahen DLLs möglich und setzt das Vorhandensein

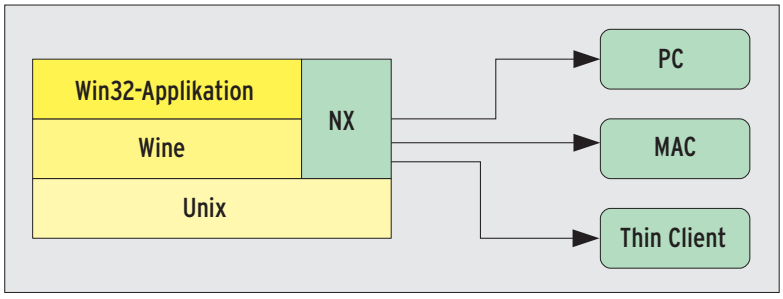


Abbildung 2: Mit FreeNX und Wine erhält man im Handumdrehen einen freien Terminalserver für nicht migrierbare Windows-Anwendungen.

einer Windows-Lizenz voraus – in der Praxis gibt es meistens keine Einschränkung, lief die zu migrierende Software ja zuvor auch unter einem lizenzierten Windows, das sich hierfür gewissermaßen recyclen lässt.

Vorher sortieren

Es lässt sich also festhalten: Viele Programme laufen unter Wine problemlos oder zumindest gut genug. Die Schwierigkeit besteht in der Regel darin, vorab abzuschätzen, wie gut ein Programm zu Wine kompatibel ist. Vollständige Gewissheit bieten letztlich aber nur eine Installation und ausführliche Tests.

Sinnvoll ist es jedoch, die Programme vor einem Test so zu sortieren, sodass man den nicht unerheblichen Aufwand nur für solche Kandidaten betreibt, bei denen die Erfolgswahrscheinlichkeit besonders hoch ist. Auf diese Weise erreicht man mit dem kleinstmöglichen Aufwand den größtmöglichen Nutzen.

Tatsächlich lässt sich die Kompatibilität und damit die Erfolgswahrscheinlichkeit einer Migration mit Wine anhand von Indikatoren abschätzen. So können APIs aus älteren Windows-Versionen, auf die sich mittlerweile eine größere Zahl von Programmen stützen und bei denen die Entwickler genug Zeit und vermutlich auch Anreiz für die Nachimplementierung hatten, als erfolgversprechender gelten. Gleiches gilt für Programme, die unter vielen Windows-Versionen laufen: Sie nutzen mit großer Wahrscheinlichkeit nur die Schnittmenge der unter allen Varianten verfügbaren Funktionen.

Wer eine Migration plant, sollte die zu migrierenden Programme mit allen dazugehörigen Informationen in einer Tabelle erfassen und dann nach einem Ampelschema gruppieren: Grün für Programme, bei denen sich kein Inkompatibilitätsgrund erkennen lässt, sodass sich der Aufwand für eine Testinstallation mit großer Wahrscheinlichkeit lohnt. Gelb gilt für Produkte, bei denen mindestens eine potenzielle Schwäche zum Vorschein kam. Hier muss eine Bewertung im Einzelfall folgen. Rot klassifiziert die Programme, die sich mit großer Wahrscheinlichkeit als Show-Stopper erweisen.

Etwas anders sieht es in dem seltenen Fall aus, dass eine in C/C++ verfasste Eigenentwicklung zu migrieren ist. Hier ist ein Abklopfen der Indikatoren ebenfalls sehr sinnvoll. In diesem Fall können aber die Entwickler noch genauere Auskünfte zu Schwachstellen geben.

Womöglich kann man auch um die eine oder andere Klippe herumprogrammieren, etwa durch die Verwendung von Unix-APIs oder -Bibliotheken mit vergleichbarer Funktion. Sofern vorhanden, ist es auch sehr sinnvoll, Selbsttests des Programmes unter Wine ablaufen zu lassen, um potenzielle Fehler schon im Vorfeld einer Testinstallation genauer identifizieren zu können.
Für die als rot und gelb klassifizierten Programme gilt es, einen alternativen Migrationspfad zu finden. Je nach Infrastruktur und Zielsetzungen könnte das beispielsweise eine Virtualisierungslösung oder ein Terminalserver sein.

Gründlich testen

Bei den Kandidaten, die durch das Sortieren in der Gruppe der aussichtsreichen Kandidaten landeten, ist eine Testinstallation unumgänglich. Hier eignet sich am besten das aktuellste stabile Wine-Release. Der Anwender sollte dabei das Programm mit denselben Parametern installieren, mit denen es auch auf dem Produktivsystem läuft. Sinnvollerweise löscht oder verschiebt man das Verzeichnis ».wine« im Home-Verzeichnis des Testbenutzers, um eine unbenutzte Installations- und Testumgebung zu erhalten:

```
mv ~/.wine ~/.wine-old
```

Üblicherweise startet man dann das Programm mithilfe der Installations-CD von Wine:

```
wine /media/cdrom/SETUP.EXE
```

Es empfiehlt sich, alle Schritte wenigstens stichpunktartig in einer Datei oder noch besser in einem Wiki zu dokumentieren. Die Zahl der Zwischenschritte und einzelnen Konfigurationseinstellungen ist immens, und ohne genaue Aufzeichnung ist schon nach wenigen Wochen der Test nur noch schwer nachzuvollziehen, weil der genaue Ablauf in Vergessenheit gerät. Dies gilt umso mehr, wenn es viele Applikationen zu bearbeiten gilt.
Für die Dokumentation und die Nachverfolgung von Problemen bietet sich nach der Erfahrung des Autors ein Ticketsystem wie Trac (1) an. Zwischenschritte sollte man regelmäßig durch eine Kopie des ».wine«-Verzeichnisses sichern, sodass man beim Experimentieren jederzeit zu einem definierten Zwischenstand zurückkehren kann.

Ist der Test aus technischer Sicht erfolgreich, das heißt, zeigen sich keine Programmfehler oder gar Abstürze, dann sollte die Applikation ein Fachbenutzer aus seiner Perspektive begutachten. Nicht selten zeigen sich hier noch Mängel, die einem Techniker entgangen wären. Erst wenn die Fachabteilung das Programm zur Benutzung abgenommen hat, kann es der Admin paketieren.

Informationsquellen

Die Homepage des Projektes (2) selbst liefert in der Regel eine gute, allgemeine Dokumentation sowohl aus Entwickler- als auch aus Anwendersicht. Es ist aber ratsam, die Dokumentation auf Aktualität zu überprüfen und gegebenenfalls auf der Entwickler-Mailingliste oder im IRC

Der „Haar-in-der-Suppe"-Ansatz

Die Sortierung von Programmen nach dem Kriterium Kompatibilität mit Wine fußt auf zwei Annahmen:

- grundsätzlich läuft jede Applikation mit Wine,
- genau die Applikation, die mir gerade zur Analyse und Migration vorliegt, kann aber Schwierigkeiten machen.

Aus diesem Grund ist es sinnvoll, vor einer Testinstallation nach Faktoren zu suchen, die den Betrieb des Programmes beeinträchtigen könnten – das sprichwörtliche Haar in der Suppe. Solch ein wichtiger Hinderungsfaktor ist die Anbindung von Spezial-Hardware, zum Beispiel USB- oder Parallelport-Dongles, oder Geräte, für die ein spezieller Treiber zu installieren ist. Grundsätzlich schwierig gestalten sich auch Programme, die relativ neue Windows-Versionen zwingend voraussetzen: Sie benutzen mit hoher Wahrscheinlichkeit APIs, die unter Wine noch nicht weit genug nachimplementiert sind.
Immer wieder zu Problemen führen auch Applikationen, die etwa in Java geschrieben, aber durch die Verwendung des Java Native Interface dennoch nicht plattformunabhängig sind. Vorsicht ist oftmals auch bei .Net-Applikationen geboten: Meistens handelt es sich um Win32/.NET-Hybridwesen, die beide APIs nutzen. Wine stellt selbst nur die Win32-API zur Verfügung, sodass man allenfalls durch ein natives Nachinstallieren von .NET-Runtimes Erfolg haben kann.
Applikationen, die nicht zufriedenstellend mit Wine zu migrieren sind, benötigen in der Praxis den größten Aufwand für einen Test, der dann doch nur unbefriedigende Resultate liefert. Der „Haar-in-der-Suppe"-Ansatz hat hier den großen Vorteil, dass er viel unnötige Arbeit vermeidet. Er hat aber auch den Nachteil, dass er potenziell zu negativ ist, und Applikationen aussortiert, die doch mit etwas Mehraufwand zufriedenstellend zu betreiben wären.
Trotz der absichtlich pessimistischen Schätzung kann man in der Regel gut die Hälfte der Applikationen für eine Testinstallation freigeben. Die Wahrscheinlichkeit einer zu positiven Klassifizierung liegt bei gründlicher Voranalyse erfahrungsgemäß nicht höher als 10 Prozent.

nachzufragen, wenn man Zweifel hat. Insbesondere der IRC ist bei den Wine-Entwicklern sehr beliebt, sodass man hier oft Aufklärung erhält.

Eine gute Quelle von aktuelleren Informationen ist zudem das Wiki des Projektes (3). Hier finden sich oft auch Einträge zu bestimmten Themen, die die Weiterentwicklung von Wine betreffen, und so mancher Eintrag zu bestimmten Windows-Programmen, die den Entwicklern besonders am Herzen liegen (4). Ebenso sind hier viele der für die Konfiguration oder für das Debugging wichtigen Registry-Einträge dokumentiert (5).

Zu guter Letzt ist bei vielen Programmen auch bereits ein Eintrag in der Application Database vorhanden (6). In dieser Datenbank dokumentieren Wine-Anwender ihre Erfahrungen. Die Einträge sind von sehr unterschiedlicher Qualität, weshalb man sie mit einer gewissen kritischen Distanz lesen sollte. Oft bewerten Anwender die Programme dort etwas zu positiv oder testen sie nicht gründlich genug, sodass selbst bei hervorragend eingestuften Programmen noch Problemstellen lauern können. Einen guten Anhaltspunkt liefern die Einträge jedoch allemal und oft enthalten sie auch wertvolle Hinweise für das Umschiffen von Schwierigkeiten. Natürlich ist jeder aufgerufen, diese Informationsquelle der Community durch eigene Beiträge weiter zu verbessern.

Abbildung 3: Unter den verschiedenen kommerziellen Wine-Abkömmlingen hat sich Cedega auf Spiele spezialisiert: hier der Kreaturen-Editor der Evolutionssimulation Spore unter Linux.

Paketierung

Hat man eine lauffähige Installation der Windows-Software unter Linux erzeugt, wartet noch eine weitere Aufgabe auf den Administrator: Er muss das Programm in das Paketverwaltungssystem seiner Distribution integrieren. Dadurch lässt es sich dann sauber auf mehreren Rechnern ausrollen und aktualisieren oder auch wieder entfernen. Die Paketierungsmechanismen der einzelnen Distributionen sind unterschiedlich – grundsätzlich gilt es jedoch, einige Wine- und Windows-Spezifika im Auge zu behalten.

Zunächst erwartet Wine seine Einstellungen und auch die installierten Programme in einem ».wine«-Verzeichnis. Dieses orientiert sich in seiner Verzeichnisstruktur an dem Laufwerk »C:« von Windows und liegt im Normalfall im Home-Verzeichnis des ausführenden Benutzers. Dieser Pfad lässt sich durch das Setzen der Umgebungsvariable »WINEPREFIX« zwar problemlos verändern, jedoch erwartet Wine, dass das jeweilige Verzeichnis dem ausführenden Benutzer gehört. Eine generische Installation unter »/opt« scheidet somit aus. Zudem verlangen Windows-Programme oft Schreibrechte auf bestimmte Verzeichnisse, die man als Administrator nicht allzu freigiebig außerhalb der Home-Verzeichnisse vergeben möchte.

Es empfiehlt sich daher, das im Test erstellte ».wine«-Verzeichnis im Tar-Format in das Paket aufzunehmen, und den Tarball mittels eines kleinen Skripts jeweils beim ersten Start des Programmes in ein Unterverzeichnis von »$HOME«, etwa »$HOME/.wine-*Programmname-versionsnummer*« zu entpacken. Eine Optimierungsmaßnahme wäre es, im Einzelfall Teile dieser Verzeichnisstruktur doch systemweit unter »/opt/*programmname*« abzulegen und in dem besagten Skript lediglich Symlinks zu erstellen. Ob und für welche Verzeichnisse dies möglich ist, hängt jeweils von der zu migrierenden Windows-Software ab.

Zudem ist es erfahrungsgemäß trotz der deutlich gestiegenen Abwärtskompatibilität von Wine empfehlenswert, genau die Wine-Version, mit der auch der Test erfolgte, mit in das Programmpaket aufzunehmen. Diese muss das Startskript dann beispielsweise unter »/opt/*programmname*/wine« installieren. Das zu erstellende Programmpaket sollte dann natürlich die

Abhängigkeiten von Wine als eigene Abhängigkeiten aufnehmen.

Support

Wie bei allen Open-Source-Produkten stellt sich auch bei Wine früher oder später die Frage nach kommerziellem Support. Ein wichtiger Anbieter ist die Firma Codeweavers (7) in den USA, bei der auch ein großer Teil der aktiven Wine-Entwickler zumindest in Teilzeit beschäftigt ist. Die kanadische Firma Lattica (8), eine Gründung eines ehemaligen Wine-Entwicklers, offeriert ebenfalls Unterstützung. In Deutschland und Frankreich bietet die ITOMIG (9) Support und Beratungsleistungen rund um Wine an. In Deutschland und England lässt sich zudem Support vom Open Source Support Center von der Firma Credativ einkaufen (10). Der Entwickler des Winetricks-Skripts, Dan Kegel, führt auf seiner Website eine Liste mit Anbietern von Consulting- und Unterstützungsleistungen für Wine (11).

Darüber hinaus gibt es noch kommerzielle Varianten von Wine, die ihre Hersteller in der Regel mit Support vertreiben. Die am weitesten verbreitete Software verkauft die Firma Codeweavers unter den Namen Crossover Office. Es handelt sich im Wesentlichen um eine Wine-Version, die mit zusätzlichen Patches und Hacks versehen ist, um bestimmmte Programme unter Linux lauffähig zu machen. Sie enthalten zudem einige meist grafische Tools, um beispielsweise die Installation bestimmter Windows-Programme für den Anwender einfacher zu gestal-

Tools und Konfiguration

Oftmals ist es hilfreich, gezielt bestimmte Windows-Komponenten nachzuinstallieren, um die Kompatibilität eines Programmes zu testen oder zu verbessern. Dan Kegel hat ein mittlerweile recht umfangreiches Skript geschrieben, das bei der Installation von typischen Kandidaten sehr hilfreich ist: »winetricks«, im Quelltext erhältlich auf seiner Website (14). So lassen sich beispielsweise die Visual Basic 5 Runtime-DLLs mit einem einzigen Aufruf nachinstallieren und Wine auch gleich für ihre Verwendung konfigurieren:

```
./winetricks vb5run
```

Grundsätzlich gilt: Der Admin konfiguriert Wine – wie Windows auch – über Einstellungen in der Registry. Damit er nicht stets zum eingebauten »regedit« oder zum Texteditor greifen muss, liefert Wine eine grafische Oberfläche für die wichtigsten Konfigurationseinstellungen mit (Abbildung 4), die sich mittels »winecfg« aufrufen lässt. Hier ist beispielsweise auch einstellbar, welche Windows-Version Wine nachstellen soll, oder ob bei bestimmten DLLs die Implementierung von Wine (»builtin«) oder eine eventuell vorhandene native Version (»native«) zu verwenden ist.

Viele der Konfigurationseinstellungen lassen sich auch temporär für nur einen Aufruf von Wine mithilfe von Umgebungsvariablen überschreiben. So kann man beispielsweise den Loader mittels »WINEDLLOVERRIDES« anweisen, eine etwa vorhandene native Version von »OLE32« der eigenen Implementierung vorzuziehen:

```
WINEDLLOVERRIDES="ole32=n,b" wine
notepad
```

Wine verfügt über sehr ausführliche Logging-Mechanismen, die sogenannten Debug-Channels (15). Mit ihrer Hilfe lassen sich für bestimmte Subsysteme gezielt mehr oder auch weniger Logmeldungen erzeugen. Hilfreich ist es zum Beispiel, nachzuvollziehen, welche Module (DLLs) nativ, welche builtin, und welche eventuell auch gar nicht in Gebrauch sind. Genau das leistet der Channel »loaddll«, der sich über die Umgebungsvariable »WINEDEBUG« aktivieren lässt:

```
WINEDEBUG=loaddll wine notepad
```

Dabei ist zu beachten, dass die Log-Ausgaben je nach Channel und Programm außerordentlich anwachsen können. So zeigt beispielsweise der »relay«-Channel jede aufgerufene Funktion an. Ein entsprechendes Logfile kann selbst für ein Hello-World-Programm recht unübersichtlich sein.

Oftmals ist es auch im Nachhinein interessant zu wissen, zu welcher Zeit genau die jeweiligen Ausgaben erfolgt sind. Das kleine Perl-Skript »winestart« hilft dabei, die Debug-Ausgaben mit einem Zeitstempel versehen auszugeben oder in eine Logdatei zu schreiben (16).

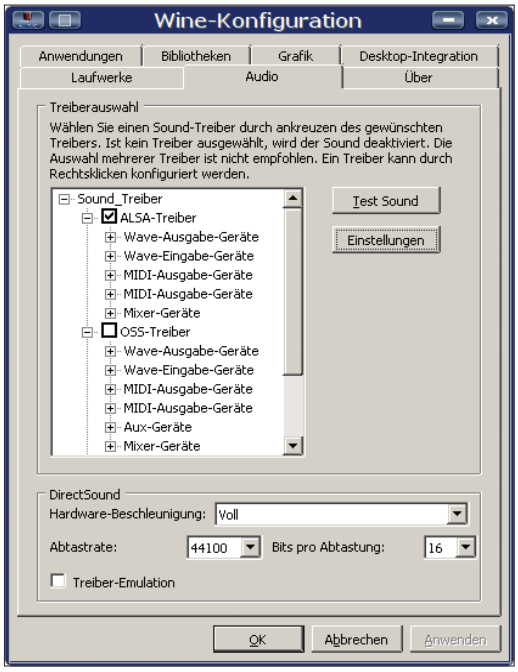


Abbildung 4: Ein grafischer Konfigurationsdialog erleichtert die Einstellarbeiten bei Wine.

ten. Die Professional-Variante enthält obendrein Mechanismen zum erleichterten Deployment von Windows-Software unter Wine. Es werden zudem Varianten des Produktes speziell für Spiele beziehungsweise für Mac-Rechner mit Intel-Hardware angeboten.

Im Spielebereich ist das Programm Cedega **(12)** mit dem Crossover Office vergleichbar: Es ist eine kommerzielle Version der Firma Transgaming, die den bis 2002 noch unter der MIT-Lizenz veröffentlichten Wine-Quelltext zur Basis hat (**Abbildung 3**). Hauptziel ist hier die Realisierung von DirectX-Funktionalitäten. Die im Laufe der Zeit entwickelte, eigene DirectX-Implementierung lizenziert Transgaming unter dem Namen SwiftShader an kommerzielle Kunden.

Relativ neu am Markt ist das Produkt der Bordeaux Group **(13)**, die, vergleichbar mit Crossover Office, eine vorgefertigte Wine-Variante anbietet, die bestimmte wichtige Windows-Programme unterstützt. Dazu zählen insbesondere die Microsoft Office Suiten und der Internet Explorer. Bordeaux ist auch in einer BSD-Variante erhältlich.

Fazit

Wer erwartet, mit Wine und ein bisschen Experimentieren seine Windows-Applikationen problemlos migrieren zu können, wird mit Sicherheit enttäuscht. Zu groß ist die Vielfalt der Programme und das Spektrum der Konfigurationsmöglichkeiten von Wine, zu klein das Wine-Entwicklerteam, das lediglich aus etwa 50 Aktiven besteht – kein Vergleich mit den Entwicklerzahlen bei Microsoft.

Wählt man jedoch ein pragmatisches Vorgehen und sortiert die Programme zuvor anhand von Indikatoren, dann ist es realistisch, insgesamt etwa die Hälfte bis Dreiviertel mithilfe von Wine bei relativ geringem Aufwand umstellen zu können. Kombiniert man dies mit dem NoMachine-Terminalserver oder seinem freien Pendant FreeNX, hat man für diese Windows-Applikationen einen lizenzkostenfreien Terminalserver zur Verfügung, was sowohl mit Blick auf die Lizenz- als auch die Wartungskosten attraktiv ist. *(jcb)*

Infos

(1) Trac: (http://trac.edgewall.org)
(2) Wine-Homepage: (http://www.winehq.org)
(3) Wine-Wiki: (http://wiki.winehq.org)
(4) Tipps zu bestimmten Applikationen: (http://wiki.winehq.org/AdobeApps)
(5) Doku zu Wine und Registry Keys: (http://wiki.winehq.org/UsefulRegistryKeys)
(6) Application Database: (http://appdb.winehq.org)
(7) Codeweavers: (http://www.codeweavers.com)
(8) Lattica: (http://www.lattica.com)
(9) ITOMIG: (http://www.itomig.de/Wine-Support.84.0.html)
(10) Credativ: (http://www.credativ.de/support/open-source-support-center)
(11) Liste von Wine-Supportern: (http://www.kegel.com/wine/isv/#consultants)
(12) Cedega: (http://www.cedega.com)
(13) Bordeaux Group: (http://bordeauxgroup.com)
(14) Winetricks: (http://www.kegel.com/wine/winetricks)
(15) Debug-Channel: (http://wiki.winehq.org/DebugChannels)
(16) Winestart: (http://freshmeat.net/projects/winestart)

Der Autor

David M. Gümbel ist Geschäftsführer der ITOMIG GmbH und berät Firmen bei der Einführung und wirtschaftlichen Nutzung von Open Source und offenen Standards.