Reinen Wine einschenken

Linux, Wine und VirtualBox kommen als Open Source-Werkzeuge zum Einsatz, wenn Fachanwendungen, die ausschließlich für das Betriebssystem Windows vorliegen, auf dem Linux-Desktop eingesetzt werden müssen. Positive Nebeneffekte stellen sich dabei zudem ein. Harald Jele

Hintergrund
Die Klagenfurter Ausgabe Robert Musils ist 2009 als elektronische Edition auf DVD erschienen. Realisiert wurde diese mit der Windows-Anwendung *FolioViews*. Um den Kreis der potentiellen Benutzer zu erweitern, wurde entschieden, die 2013 erscheinende zweite Ausgabe, der umfangreiche inhaltliche Ergänzungen und Berichtigungen zum Werk Musils zugrunde liegen, mittels Linux und Wine auch als fertige Installation innerhalb einer VirtualBox anzubieten. Damit sollen vor allem auch die Benutzer aus dem Kreis der Linux- und Mac OS X-Anwender angesprochen werden. Zudem wird die Konfiguration so vorbereitet, dass eine möglichst einfache Integration in ein lokales Netz gelingen kann, ohne in zusätzliche Software investieren zu müssen. Eine weitere Beschreibung dazu findet sich unter [12].

Abseits der umfangreichen und bekannten Windows-Anwendungen sind am Markt unzählige vorhanden, die als ganz spezielle Fachanwendungen die Bedürfnisse kleiner bis mittelgroßer Nischen bedienen. Für den Fall, dass sich diese für Unternehmen und Institutionen als ihre primär unternehmenskritischen Anwendungen darstellen, kann der häufig zu beobachtende Umstand eintreten, dass genau diese eine Migration hin zu einem Linux-Desktop verhindern. An exakt diesem Dreh- und Angelpunkt setzen häufig Migrationsszenarien an, wenn es gilt, solche Anwendungen schrittweise in einem typischen Linux-Umfeld abzubilden.
Mit dem Einsatz der Systembibliotheken von Wine bieten sich einem Administrator bereits seit Jahren hinreichend erprobte, lizenzfreie und quelloffene Methoden an, durch deren Umsetzung ein solches Unterfangen gelingen kann.

Dekantieren

Das Ubuntu Repository zeigt sich für eine Installation von Wine gegenüber dem Anwender als besonders freundlich. Eine Installation des Meta-Pakets zu Wine über den Standard-Paketmanager „apt“ berücksichtigt nicht nur die notwendigen Libraries und Kernanwendungen, sondern liefert unter anderem auch gleich die erforderlichen Konfigurationswerkzeuge, eine Integration von Gecko zum Rendern von HTML-Anzeigen, sowie die im Windows-Umfeld häufig eingesetzten Schriftarten mit:

```
sudo apt-get install wine
```

Die Konfiguration von Wine hat sich im Laufe der Jahre völlig geändert. Im Wesentlichen wurde sie deutlich einfacher und übersichtlicher. Einfacher vor allem durch den Umstand, dass man sich nicht mehr um jedes Detail kümmern muss. Zu diesen Details zählen z. B. das automatische Einbinden entfernbarer Geräte (CD-ROMs, DVDs etc.) oder die Integration eines Drucksystems wie CUPS. Beides wird zuverlässig von den Prozessen des Wine-Servers wahrgenommen.
Eine bedeutende Änderung in der Architektur von Wine hat sich mit der Einführung sogenannter „Prefixe“ (auch „Bottles“ genannt) ergeben. Mit der Einrichtung eines Prefixes kann Windows-Software im Kontext dieses Prefixes installiert und betrieben werden, ohne in das Gehege anderer Software, die in weiteren Prefixen installiert wird, zu gelangen. Damit wurde auch ein relativ gefahrloses Experimentieren möglich, bei dem nicht sofort die gesamte Softwareinstallation in unmittelbare Gefahr kommt, rasch unbrauchbar zu werden. Ein „Zu-Tode-Installieren“ gehört damit der Vergangenheit an, und das paralle Installieren von Software, die sich nativ unter Windows am gleichen Rechner eigentlich gegenseitig ausschließt, wird möglich.
Beim „Umzug“ auf einen anderen Rechner bzw. beim Arbeiten mehrerer Benutzer in einer Mehrbenutzerumgebung kann ein erstellter und passend konfigurierter Prefix in die neue Umgebung (das neue Homeverzeichnis) übernommen bzw. weiteren Benutzern in ihre entsprechenden Homeverzeichnisse verteilt werden. Darauf zu achten ist, dass die Rechtestruktur der Dateien und Ver-

zeichnisse erhalten bleibt bzw. in korrekter Weise neu angelegt wird. Daher sind solche Transfers am günstigsten mit dem *tar*-Kommando durchzuführen. Ein:

```
tar cfzv prefixname.tgz prefixname
```

komprimiert das erstellte Archiv aus dem vorbereiteten Prefix. Das Entpacken gelingt, wie bekannt, in umgekehrter Reihenfolge:

```
tar xfzv prefixname.tgz
```

Wird kein eigener Prefix angelegt oder die Angabe desselben bei einem Programmaufruf vergessen, so landen sämtliche Vorgänge im Verzeichnis `~/.wine` des Benutzers. Damit verliert dieser jedoch auch gleichzeitig die eben genannten Vorteile, die sich aus der Verwendung eines Prefixes ergeben.
Ein Wine-Prefix wird immer dann angelegt, wenn dieser dem Aufruf der Windows-Anwendung vorangestellt und noch nicht vorhanden ist. Ein solcher kann jedoch, und sollte wohl auch, vorab angelegt werden. Mit folgendem Befehl:

```
env WINEPREFIX=~/prefixname wineboot -u
```

wird durch den Aufruf des Prozesses „wineboot" der Wine-Prefix „prefixname" im Homeverzeichnis des Benutzers als Verzeichnisstruktur angelegt, wobei *wineboot* im Grunde hier jene Aktionen ausführt, die ein Windows-System bei einem seiner häufigen Neustarts erledigt. Die weiteren Optionen, die *wineboot* zu bieten hat, sind in [1] beschrieben.
Mit der auf diesem Weg angelegten Verzeichnisstruktur besteht bereits eine lauffähige Wine-Umgebung, die mit den Standard-Werkzeugen von Wine genauer inspiziert werden kann:

```
env WINEPREFIX=~/prefixname wine
C:\\windows\\explorer.exe
```

ruft den in Wine eingebauten „Explorer" auf und zeigt dem Benutzer die Sicht des Wine-Servers auf die eingehängten Filesysteme und den darin enthaltenen Dateien. Der Aufruf dieses Explorers ist immer günstig, um wahrzunehmen, ob beim Anlegen oder Updaten des aktuellen Wine-Prefixes die Filesysteme in der gewünschten Weise eingebunden, d. h. mit den richtigen „Laufwerksbuchstaben" der Windows-Welt versehen, wurden. CD-ROM/DVD-Laufwerke und Ähnliches werden ohne Eingriff nur dann angezeigt, wenn diese im Linux-System gemountet sind.
Eine parallele Erkundung des Prefix-Verzeichnisses mit Linux-Mitteln zeigt rasch die sich im Wesentlichen selbst beschreibende Struktur auf:

```
~/prefixname/drive_c
```

bildet die „gebootete" Festplatte einer typischen Windows-Installation ab,

```
~/prefixname/dosdrives
```

zeigt die angelegten Laufwerke mit den entsprechend zugewiesenen Buchstaben an.
Tieferliegende Verzeichnisse werden entlang jener Verzeichnisstruktur abgebildet, wie sie von einem Windows-System erwartet werden.
Mit dem Kommando:

```
env WINEPREFIX=~/prefixname wine
C:\\windows\\regedit.exe
```

startet der Benutzer das Wine-Tool „*regedit*", mit dem die für den Prefix spezifisch abgebildete Registry des Systems untersucht und bearbeitet werden kann. Die der Implementierung zugrundeliegenden Dateien sind:

```
~/prefixname/system.reg
~/prefixname/user.reg
~/prefixname/userdef.reg
```

Wenngleich in der Arbeitsweise von Administratoren das Patchen von Dateien zum Normalfall gehört, sollten, für ein möglichst fehlerfreies und durchaus komfortables Arbeiten, nicht die drei Dateien selbst, sondern diese über das vorgesehene Wine-Tool editiert werden. [2] liefert dazu eine praktikable Übersicht zu den Aufrufoptionen von „*regedit*".
Als eine zentrale Schaltstelle für die weitere Konfiguration des Prefixes zeigt sich das graphische Wine-Tool „*winecfg*" (Abb. 1):

```
env WINEPREFIX=~/prefixname winecfg
```

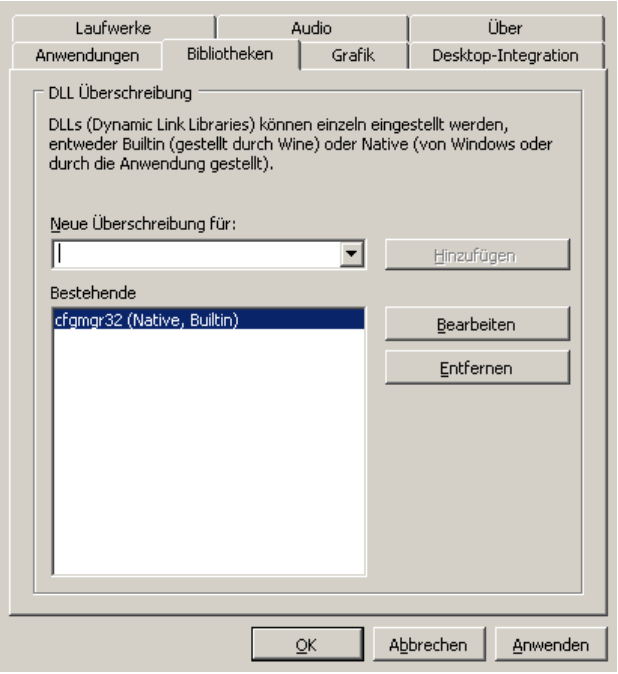


Abbildung 1: Die „Karteireiter" (Tabs) von *winecfg* zur Konfiguration eines Prefixes.

Über diese graphische Oberfläche können einerseits komfortabel Einstellungen zum Verhalten von Wine vorgenommen werden, andererseits dient diese Anbindung an Wine auch zum Starten bzw. Installieren von Anwendungen. Um jedoch die volle Kontrolle über den Installationsvorgang zu behalten, installiert und startet der gewissenhafte Administrator Programme im Weiteren jedoch besser auf der Kommandozeile.

Abb. 1 zeigt jenen Reiter, der sich für das Konfigurieren eines Prefixes unter Umständen als ein zentraler entpuppen kann: Wine ist in der Lage, Windows-Libraries (DLLs) in unterschiedlicher Reihenfolge zu laden. Mit der Installation von Wine gelangen eine durchaus erquickliche Fülle an solchen auf den Zielrechner, die überwiegend durch Reengineering von den Wine-Entwicklern selbst erstellt wurden. Da viele von ihnen bereits eine Fülle an Systemaufrufen abdecken, funktionieren in der Regel auch bereits viele Anwendungen ohne den Aufwand einer großen Nachbearbeitung.
Die Krux mit einer Wine-Installation beginnt jedoch immer dann, wenn die mitgelieferten Bibliotheken für eine spezifische Anwendung wesentliche Funktionen nicht abdecken. Dann ist es notwendig, die von Microsoft in einer Windows-Installation nativ bereitgestellten DLLs zu übernehmen und anstelle der von Wine ausgelieferten zu laden. Das Tool *winecfg* bietet dafür eine komfortable Schnittstelle. In Abb. 1 erkennt man, dass im hier willkürlich angenommenen Fall die DLL „cfgmgr32“ nicht „buildin“, sondern „native“ (also in der Microsoft-Version) von Wine geladen werden soll. Dazu muss die Datei „cfgmgr32.dll“ einer Windows-Instanz entnommen und Wine an der im Verzeichnisbau entsprechenden Stelle zugeführt werden.

Diese Datei aus dem Windows-Verzeichnis
`c:\windows\system32\cfgmgr32.dll`
landet im Prefix von Wine unter
`~/prefixname/drive_c/windows/system32`

Die Änderung, die mit *winecfg* hier durchgeführt wird, hinterlässt folgenden Eintrag in der Datei *user.reg* bzw. einen entsprechenden im Verzeichnis der Registry:

```
[Software\\Wine\\DllOverrides] 1320056438
"cfgmgr32"="native,builtin"
```

Interessant an dieser Stelle ist, dass Wine in der Lage ist, jenes Verhalten abzubilden, das mit Windows-XP eingeführt wurde und von Microsoft als „Side-by-side“ (SxS) bezeichnet wird [3]. Im Verzeichnis
`c:\windows\WinSxS`
liegen parallel mehrere Versionen ein und derselben Systembibliothek, gegliedert nach Unterverzeichnissen, nebeneinander und können von den Windows-Anwendungen in ganz bestimmten Versionen vom System angefordert werden. Zu beobachten ist, dass gerade Installationsroutinen von diesem Verzeichnis häufig Gebrauch machen, wenngleich die lauffähigen Anwendungen später zuweilen auf dort hinterlegte Systembibliotheken durchaus verzichten.
Sollte also das Installieren einer Anwendung fehlschlagen und keine besonderen Hinweise dafür vorliegen, so kann das Austauschen dieses Verzeichnisses gegen eines einer Windows-Installation, zumindest für die Dauer der Installation, sich als hilfreich erweisen.

Im Umgang mit Wine-Installationen ist der Administrator hier an einer Stelle angelangt, die, zumindest kognitiv, besondere Aufmerksamkeit verdient. Es gilt nämlich, nicht den Überblick zu verlieren. Systembibliotheken sind sehr schnell und einfach ausgetauscht, ganze Verzeichnisse sind rasch verändert und mit Werkzeugen, wie „*winetricks*“ werden in einem Rutsch eine Fülle an frei bzw. kostenlos verfügbaren, „originalen“ Microsoft-Dateien übernommen.
Nur sei an dieser Stelle auf die Warnung der Wine-Programmierer hingewiesen: Wer aus der Community Hilfe erwartet, sollte in einem ersten Schritt davon absehen, seine Installation wie eben beschrieben (vermeintlich) „aufzuwerten“ [4] und die Installation einer Windows-Anwendung beginnen, ohne weiter ins System einzugreifen.

Kredenzen

Der Aufruf eines üblichen Windows-Setup-Programms gelingt (hier von einem DVD-Laufwerk mit dem zugewiesenen Buchstaben „D“) in der Kommandozeile wie folgt:

```
env WINEPREFIX=~/prefixname wine D:\\Setup.exe
```

Wine kennt jedoch unterschiedliche Mechanismen, ein Programm zu starten.
Laut den Einträgen im Wine-Wiki ist der „offizielle“ Aufruf nach dem DOS-Schema:

```
env WINEPREFIX=~/prefixname wine start 'D:\Se-
tup.exe'
```

bzw. in der Unix-Schreibweise

```
env WINEPREFIX=~/prefixname wine start /Unix
/media/Setup.exe
```

Die Variante in der Unix-Schreibweise ist häufig dann anzuwenden, wenn bestimmte Pfadangaben bei der Installation bzw. in weiterer Folge beim Aufruf durch eine Anwendung nicht richtig umgesetzt werden können. Weitere Hinweise dazu finden sich in [5].

Läuft die Installationsroutine ohne abzubrechen durch, stehen die Zeichen für einen nahen, ge-

glückten Aufruf gut. Wine hinterlegt in diesem Fall den Aufruf des Programms wie es Windows täte: Sollte die Setup-Routine dies vorsehen, wird ein Eintrag am Desktop über ein verknüpftes Icon getätigt. Dem Menü des Linux-Desktops wird ein Punkt Wine hinzugefügt, unter dem im Weiteren die installierten Programme aufzufinden sind.

Degustieren

Wenn das Setup des Windows-Programms jedoch nicht zu einem vielversprechenden Ende gelangt, gilt es, den Ursachen auf den Grund zu gehen. Wobei natürlich davon auszugehen ist, dass einige der hier genannten Aspekte durchaus berücksichtigt werden sollten, bevor eine Installation angegangen wird.
Die Mittel erster Wahl sind gegebenfalls:

- das Überprüfen, ob sich in der Anwendungsdatenbank von CodeWeavers weiterführende Einträge finden [6]
- eine gründliche Recherche im Web. Viele Installationsversuche, die geglückt oder misslungen sind, finden sich dokumentiert, jedoch bei CodeWeavers nicht eingetragen
- die parallele, kontrollierte Installation auf einer nativen Windows-Instanz
- ein näheres Kennenlernen des Wine-Debuggers [7]

Auf die beiden erstgenannten Aspekte muss hier nicht weiter eingegangen werden. Sie zählen zu den üblichen Handlungsweisen eines Administrators, um Hilfestellungen oder gar Dokumentation zu lukrieren.
Für die Arbeit an einer parallelen Windows-Instanz ist jedoch einiges an Vorbereitung unverzichtbar, denn der Installationsvorgang und der Aufruf der Anwendung sollten durch die eingesetzten Hilfsmittel möglichst detailgenau erfasst werden. Das Tool, das der Administrator letztlich nach freier Wahl bestimmt, sollte während der Installation mitprotokollieren, welche

- Verzeichnisse und Dateien angelegt,
- Einträge in der Registry vorgenommen,
- Bibliotheken dem System hinzugefügt werden.

Letztlich kann dies auch durch Handarbeit einigermassen gut erledigt werden; für mehr Komfort sorgen jedoch entsprechende Programme wie *TrackWinstall* als Freeware [8] oder ähnliche.

Besondere Aufmerksamkeit bei der Wahl des einzusetzenden Werkzeugs gilt der Überprüfung, ob nach einer „kontrollierten Installation“ eines Programms die angelegten Verzeichnisse und Dateien, die hinzugekommenen Einträge der Registry, sowie die relevanten Systembibliotheken eventuell auch sofort exportiert und somit direkt in eine möglichst „frische“, unangetastete Wine-Umgebung übernommen werden können.

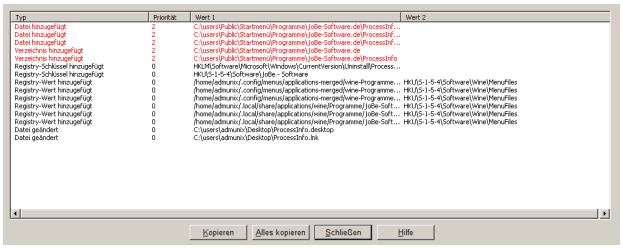


Abbildung 2: Ergebnisliste der gefundenen Änderungen an einer Windows-Installation von *TrackWinstall*.

Ist auf diesem Weg die Installation umschifft, startet der Administrator frank und frei die Anwendung über die üblichen, oben angemerkten Startoptionen von Wine.

Wenn auch dieser Weg scheitert, das Windows-Programm starten zu können, ist eine genauere Inspektion der laufenden Anwendung in einer parallel gehaltenen Windows-Installation notwendig. Ein erster Schritt dazu ist das Starten der Anwendung und das Überprüfen jener Systembibliotheken, die die Anwendung nach deren Aufruf ladet.
Dazu reichen einfache Werkzeuge, die als Freeware oder zum Installieren in einer Demo-Version zur (freien) Verwendung vorliegen. *DLL Show* [9] zählt mittlerweile wohl zu den Veteranen, leistet jedoch immer noch gute Dienste und zeigt zuverlässig die in den Arbeitsspeicher des Betriebssystems geladenen Anwendungen und nach deren Auswahl die zugehörigen Bibliotheken an. Ähnliches leistet auch *ProcessInfo* [10], das als Demo-Version kostenlos für einen Zeitraum von 30 Tagen verfügbar ist und dessen Einschränkungen gegenüber der Vollversion für diese Belange nicht weiter stören. Von weiterem Nutzen erweist sich das Programm auch durch den Umstand, dass es zudem in einer Wine-Umgebung installier- und ausführbar ist.

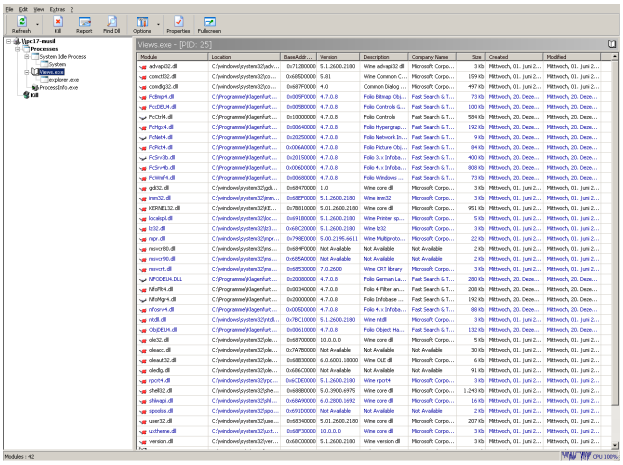


Abbildung 3: Anzeige der mit einer ausführbaren Windows-Datei verknüpften DLLs am Beispiel von *ProcessInfo*.

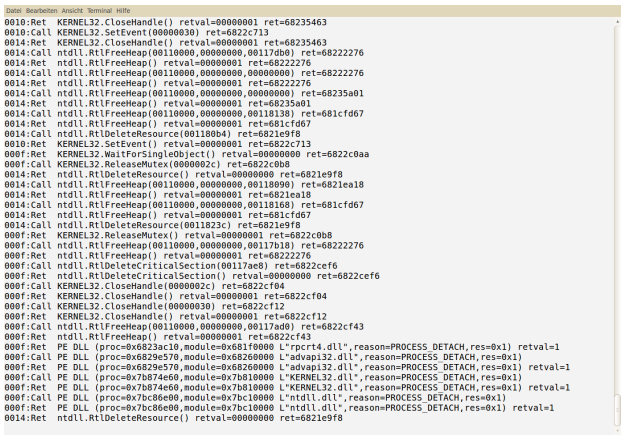
Anhand der Liste jener Bibliotheken, die durch die Software geladen werden, erhält man einen ersten Eindruck davon, worauf man in weiterer Folge möglicherweise sein Augenmerk legt. Ob jedoch tatsächlich der Umstand zutrifft, dass eine („build-in“) Wine-Bibliothek einer Anwendung nicht alle relevanten Funktionen anbietet und deshalb durch eine native Version ersetzt werden muss, verrät einem eventuell bereits die Ausgabe auf der Startkonsole oder in jedem Fall der Aufruf des Wine-Debuggers beim Start der Anwendung.

„Cuvée-rieren“

Dass das fortgeführte Ersetzen von Libraries letztlich einen Ansatz darstellt, der nur eingeschränkt zum Ziel führt, zeigt sich in jenem Umstand, den Windows-Administratoren in der Vergangenheit als die „DLL-Hölle“ beschrieben haben: Nicht alle Versionen der Systembibliotheken vertragen einander. An diesem Punkt angelangt, erfordert das Vorgehen ein wenig Geschick, Recherche und Glück.

Abbildung 4: Starten einer Anwendung durch Wine und gesetzter Umgebungsvariable „WINEDE-BUG=+relay“.

Um unkompliziert an Debug-Informationen zu gelangen, wird Wine in einem ersten Schritt die Umgebungsvariable „WINEDEBUG“ beim Start übergeben. Der Wert „+relay“ regelt im Debug-Modus von Wine das Verhalten in einer Weise, dass sämtliche Funktionsaufrufe, sowie die Bezeichnung der zugehörigen Bibliothek auf der Konsole ausgegeben werden. Damit erhält man erste Informationen darüber, bei welchem Funktionsaufruf Wine scheitert und welcher DLL diese zugeordnet werden kann. Die Standard-Ausgabe sollte (günstigerweise) zudem in ein Logfile umgeleitet werden,



um anschließend den Startprozess anhand dieses Files in Ruhe analysieren zu können:

```
env WINEPREFIX=~/prefixname WINEDEBUG=+relay
wine anwendung.exe &>start_anwendung.log
```

Deutlich mehr Informationen liefert schließlich das Setzen des Wertes „+all“ für die Umgebungsvariable „WINEDEBUG“.

```
env WINEPREFIX=~/prefixname WINEDEBUG=+all
wine anwendung.exe &>start_anwendung.log
```

Dieser Wert bewirkt, dass Wine sämtliche Informationen auf die Standard-Ausgabe schreibt, die im Debug-Modus zugänglich sind. Der Startvorgang der Anwendung verzögert sich dadurch zwar erheblich; mit einiger Übung kann man sich auf diesem Weg jedoch einen deutlichen Eindruck von den implementierten Mechanismen und den daraus resultierenden Stolperfallen verschaffen, die es anschließend zu umschiffen gilt.

Neben dem Setzen der Umgebungsvariable „WI-NEDEBUG“ besteht die Möglichkeit, eine Anwendung nicht direkt mit Wine zu starten, sondern mit dem eingebauten Wine-Debugger.

```
env WINEPREFIX=~/prefixname winedbg
anwendung.exe
```

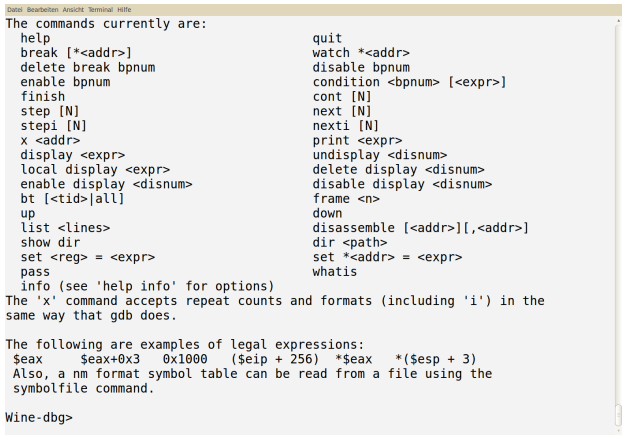


Abbildung 5: Die aktuellen Befehle des Wine-Debuggers.

Dieser bringt die üblichen Eigenschaften eines Debuggers mit und integriert sich passend in die Umgebung der Systembibliotheken von Wine. Anhand des Debuggers sind einem umfangreiche Methoden zugänglich, mit denen Anwendungen kontrolliert gestartet werden können. Die Befehle des Wine-Debuggers bilden ein Subset jener des GNU Project Debuggers (GDB). Damit ist die weitere Arbeitsweise zumindest jenen auf Anhieb vertraut, die bereits Erfahrungen im Debuggen mitbringen. Die Web-Site der Entwickler zum Wine-Debugging [11] gibt einen guten Überblick zum schrittweisen Annähern an ein vorerst unbekanntes Problem, das sich beim Ausführen einer Windows-Anwendung in einer Wine-Umgebung einstellt.

Fazit

Wenn es gelingt, eine Windows-Anwendung mit Wine unter Linux betreiben zu können, so kann dies dazu beitragen, dass möglicherweise eine Anwendung weniger störend ins Gewicht fällt, wenn es gilt, einen Linux-Desktop zu etablieren. Zudemkönnen sich damit auch durchaus positive Nebeneffekte einstellen: Mit Linux als Betriebssystem zeigen sich einem Administrator sofort eine Reihe von Möglichkeiten, diese Anwendung nicht nur lokal, sondern auch als Netz-Installation einzurichten, sowie diese in Umgebungen zu etablieren, in denen ein Mehrbenutzerbetrieb gewährleistet werden soll. Darüber hinaus stellt die Kombination Wine-Linux-VirtualBox eine Open Source-Lösung dar, mit der auch Linux fremde Anwendungen quasi in einen Container gepackt und für einige Zeit konserviert werden können.

Kurze Geschichte von Wine
1993: Inspiriert vom Produkt Wabi (auf Sun Solaris) begann die Entwicklung von Wine. Erste Arbeiten an einem Tool zum Laden von 16-bit-Windows Programmen und einer Fensterverwaltung.
1994: Das große Echo zu den Entwicklungen führt zu einer Diskussionsliste im Usenet. Die Netzwerkunterstützung wird eingeführt. Das Lesen und Schreiben der Windows-Registry wird vervollständigt.
1995: Anfänge in der Unterstützung von 32-bit-Windows-Code und einer automatischen Konfiguration von Wine-Installationen.
1996: Microsoft Word und Excel können erstmals ausgeführt werden.
1998: Corel und später CodeWeavers unterstützen die Entwicklung. Dadurch wird eine Vielzahl an Desktop-Anwendungen installierbar.
2001: Effektiver DirectX-Support unterstützt das Ausführen von Spielen.
2002: Wine wird parallel zum X11-Lizenzmodell auch LGPL-lizenziert.
2003: Einführung von Präfixen/Bottles zur Trennung der installierten DLLs pro Anwendung ("DLL-Seperation").
2008: Wine wird in der Version 1.0 freigegeben.
2012: Das Projekt beinhaltet mehr als 1.4 Mio. Zeilen Quellcode und mehr als 700 Beteiligte.
Aktuelle Projekte: Mac OS X QuartzDriver, Direct3D-10, Multiuser Wineserver, USB-Support
Weitere Ziele für die Version 1.4: DIB engine, Sprachunterstützung von rechts nach links (z.B. für Häbräisch und Arabisch), Redesign des Audiosupports

Quellen

[1] Kurzbeschreibung zu *wineboot* im entsprechenden Wiki:
[http://wiki.winehq.org/wineboot]

[2] Kurzbeschreibung zum Winetool *regedit* im entsprechenden Wiki:
[http://wiki.winehq.org/regedit]

[3] Microsofts Beschreibung von „Side-by-side Assemblies“
[http://msdn.microsoft.com/en-us/library/aa376307.aspx]

[4] Kurzbeschreibung zu *winetricks* im Wine-Wiki:
[http://wiki.winehq.org/winetricks]

[5] Aufrufoptionen von Wine auf der Kommandozeile:
[http://wiki.winehq.org/FAQ#head-3b297df7a5411abe2b8d37fead01a2b8edc21619]

[6] CodeWeavers CrossOver Compatibility Center:
[http://www.codeweavers.com/compatibility]

[7] Using the Wine Debugger:
[http://www.winehq.org/docs/winedev-guide/wine-debugger]

[8] Die Freeware *TrackWinstall* zur kontrollierten Installation von Windows-Anwendungen:
[ftp://ftp.heise.de/pub/ct/ctsi/trackwinstall_111.zip]

[9] *DLL Show* als Freeware:
[http://www.gregorybraun.com/DLLShow.html]

[10] *ProcessInfo* als Demoversion:
[http://www.jobe-software.de]

[11] Web-Site zum Wine-Debugging:
[http://www.winehq.org/docs/winedev-guide/wine-debugger]

[12] Jele, Harald: Robert Musil und das Matroschka-Prinzip:
[http://www.linux-magazin.de/2012/03/plus]

Der Autor

Dr. Harald Jele ist Mitarbeiter an der Universität Klagenfurt und beschäftigt sich zur Zeit (auch) mit den technischen Aspekten der Klagenfurter Ausgabe Robert Musils.