

DAS VIRTUALISIERUNGS-BUCH

Herausgegeben von Fabian Thorns

Online-Kapitel zum Buch: Wine

Die Weitergabe dieses Kapitels oder das direkte Verlinken ist untersagt.



Computer & Literatur Verlag GmbH

Bibliographische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Alle Rechte vorbehalten. Ohne ausdrückliche, schriftliche Genehmigung des Herausgebers ist es nicht gestattet, das Buch oder Teile daraus in irgendeiner Form durch Fotokopie, Mikrofilm oder ein anderes Verfahren zu vervielfältigen oder zu verbreiten. Dasselbe gilt für das Recht der öffentlichen Wiedergabe.

Der Verlag macht darauf aufmerksam, daß die genannten Firmen- und Markenzeichen sowie Produktbezeichnungen in der Regel marken-, patent-, oder warenzeichenrechtlichem Schutz unterliegen.

Die Herausgeber übernehmen keine Gewähr für die Funktionsfähigkeit beschriebener Verfahren, Programme oder Schaltungen.

1. Auflage 2008

© 2008 by C&L Computer und Literaturverlag
Zavelsteiner Straße 20, 71034 Böblingen
E-Mail: info@CuL.de
WWW: <http://www.CuL.de>

Coverdesign: Hawa & Nöh, Neu-Eichenberg
Satz: C&L-Verlag

Dieses Buch wurde auf chlorfrei gebleichtem Papier gedruckt

ISBN: 978-3-936546-56-9

ONLINE-INHALT

0 Wine	5
0.1 Einschränkungen	6
0.2 Stärken	7
0.3 Wine installieren	8
0.4 Wine konfigurieren	8
0.4.1 Die Basiskonfiguration	8
0.4.2 DLLs einbinden	10
0.4.3 Das winetricks-Skript	11
0.4.4 Wichtige Umgebungsvariablen	12
0.4.5 Debugging	13
0.4.6 Drucken in Wine	14
0.4.7 TrueType-Fonts einrichten	16
0.5 Arbeiten mit Wine	16
0.5.1 Windows-Programme aufrufen	16
0.5.2 Windows-Programme installieren	18
0.5.3 Windows-Programme deinstallieren und archivieren	21
0.6 Winelib-Programme	21
0.7 Alternativen und andere Versionen	23
0.7.1 CrossOver Office	23
0.7.2 ReactOS	24
0.7.3 Cedega	24
0.8 Kommerzieller Support	25
0.9 Ausblick	25



ONLINE-KAPITEL

WINE

von David Gümbel

Das 1993 ins Leben gerufene Wine-Projekt (<http://www.winehq.org>) hat sich zum Ziel gesetzt, das Windows-API unter Unix nachzuprogrammieren. Dieser Ansatz erlaubt den Betrieb unmodifizierter Windows-Anwendungen unter Unix, ohne daß eine Windows-Lizenz benötigt wird. Es können aber DLLs einer gegebenenfalls vorhandenen Windows-Lizenz eingebunden werden, um Lücken in dem von Wine zur Verfügung gestellten Win32-API zu schließen. Unterstützt wird vor allem Linux, aber grundsätzlich auch Solaris, FreeBSD, OpenBSD, NetBSD und MacOS X.

Wine stand bis 2002 unter der MIT-Lizenz und wurde dann unter der GNU LGPL relizenziert. Windows-Programme, die unter Wine ausgeführt werden sollen, haben allerdings in der Regel andere, restriktivere Lizenzbestimmungen, die es selbstverständlich auch unter Linux und Wine zu beachten gilt.

Im wesentlichen besteht Wine aus zwei Bestandteilen:

- ◆ Einem Loader, der Windows-Executables (.EXE, .DLL, .OCX und so weiter) in den Speicher laden und unter Unix zur Ausführung bringen kann. Dieser Loader wird benötigt, da unter Windows mit dem sogenannten Portable Executable Format ein anderes Binärformat für ausführbare Dateien zum Einsatz kommt als unter Linux, das das ELF-Format benutzt.
- ◆ Einer Nachimplementierung des Win32-API unter Unix, die es zusammen mit dem Loader erlaubt, Windows-Programme in binärer, unmodifizierter Form unter Unix auszuführen. Die sogenannte Winelib erlaubt es zusammen mit den zugehörigen Entwicklungswerkzeugen, im Quelltext vorliegende Windows-Programme relativ einfach in native Linux-Programme umzuwandeln.

Der vom Projekt verfolgte Ansatz ist es, die einzelnen Funktionen, die das Betriebssystem Windows den Anwendungen zur Verfügung stellt, unter Unix nachzuimplementieren. Hierbei wird minutiös darauf geachtet, daß sich der so entstandene Programmcode exakt gleich verhält wie der von Windows. Die Entwickler sprechen in diesem Zusammenhang oftmals von einer »Bug-by-Bug Compatibility«, da das Windows-API selbst nicht immer fehlerfrei implementiert ist, und viele existierende Programme sich auf teilweise sehr subtile Fehlfunktionen von Windows verlassen.

Das bedeutet selbstverständlich auch, daß eventuelle Designfehler des Windows-API unter Wine ebenfalls bestehen. Ein Beispiel hierfür ist die Sicherheitslücke in der Behandlung von WMF-Bildern, die in Windows entdeckt wurde und in Wine ebenfalls vorhanden war¹.

0.1 EINSCHRÄNKUNGEN

Dieser in der Theorie sehr gute Ansatz stößt in der Praxis jedoch an verschiedene Grenzen. Einerseits sind nicht alle Windows-Bibliotheken umfassend dokumentiert. Bei der Nachimplementierung von Windows-Funktionalitäten sind die Programmierer von Wine daher manchmal auf intensive Tests und Reverse Engineering angewiesen. Prinzipbedingt hinkt zudem der Implementierungsstand des Windows-API unter Wine dem von Windows selbst hinterher, da die Wine-Programmierer ja immer erst nach dem Release eines neuen API durch Microsoft mit der Nachimplementierung beginnen können – so brachte beispielsweise in jüngerer Zeit Windows Vista einige technische Neuerungen, die Wine bisher noch nicht oder kaum unterstützt². Auch Fehler in der durch Microsoft online bereitgestellten Dokumentation sind in der Praxis hinderlich gewesen.

Produkt	Wine Version 1
Hersteller	Wine Projekt
Webseite (deutsch)	http://www.winehq.org/
Preis	Kostenlos (GNU Lesser General Public License V. 2.1). Kommerzielle Varianten mit Supportoption sind verfügbar.
Host-Plattform	Linux, BSD, MacOS X, Solaris
Gäste	DOS, Win32
Menü-/Hilfessprache	Diverse Konfigurationstools, teilweise übersetzt
Alleinstellungsmerkmal	Reimplementation eines großen Teils der Funktionen des Windows APIs auf X und OpenGL. Keine Emulation!

Tabelle 0.1: Steckbrief von Wine

Darüber hinaus sind auch die Implementierungsstände der einzelnen Windows-Funktionen in Wine unterschiedlich weit fortgeschritten. So gibt es beispielsweise APIs, die den Status »Stub« besitzen, also aus einer Funktion mit leerem oder fast leerem Rumpf bestehen. Ruft ein Windows-Programm eine solche Funktion in Wine auf, erscheint eine entsprechende Meldung in der Debug-Ausgabe. Einige APIs verhalten sich auch unterschiedlich, je nachdem, auf das Nachbilden welcher Windows-

¹ Zur WMF-Problematik siehe <http://www.heise.de/newsticker/Windows-Sicherheitsluecke-auch-in-API-Nachbau-WINE-fuer-Linux-Unix-Update--/meldung/68123>.

² Zu den technischen Neuerungen in Vista siehe die Wikipedia unter http://de.wikipedia.org/wiki/Microsoft_Windows_Vista#Technische_Ver.C3.A4nderungen und weitere Links. Beispielsweise ist die Windows Communication Foundation eine Weiterentwicklung des in der Vergangenheit mühevoll in Wine nachimplementierten DCOM.



Version Wine konfiguriert wurde. Beispielsweise gibt es Funktionen, deren Verhalten unter den Consumer-Varianten von Windows (95, 98, ME) deutlich weniger komplex ist als unter den NT-Derivaten (NT, 2000 etc.), und die daher mit einem auf zum Beispiel Windows 98 konfiguriertem Wine korrekt und vollständig implementiert sind, bei einer Konfiguration auf Windows 2000 jedoch noch Lücken aufweisen.

Im allgemeinen läßt sich festhalten, daß Windows eine Fülle von mehreren zehntausend Funktionen zur Verfügung stellt. Kein Windows-Programm, und sei es auch noch so umfangreich, greift jedoch auf alle diese API-Aufrufe zurück. Selbst umfangreiche Software wie beispielsweise Microsoft Office XP nutzt lediglich etwa 1600 Funktionen des Windows-API. Für die Kompatibilität von einem Windows-Programm mit Wine ist also nicht entscheidend, wie viele Funktionen der Windows-API es nutzt, sondern ob die Funktionen, die es nutzt, unter Wine bereits hinreichend weit implementiert sind, um einen korrekten Betrieb des Programms gewährleisten zu können.

Um Lücken in der Nachimplementierung des Windows-API in Wine zu umgehen, kann man auf echte (»native«) Windows-DLLs zurückgreifen. Der in Wine integrierte Loader muß dann so konfiguriert werden, daß er nicht seine eigene Implementierung von zum Beispiel *Ole32.dll* lädt, sondern die von Microsoft bereitgestellte.

Das Einbinden von nativen DLLs erfolgt über das Kopieren der entsprechenden Dateien in das System- beziehungsweise System32-Verzeichnis von Wine. Welches Verzeichnis verwendet werden muß, hängt von der Konfiguration von Wine ab: Möchte man eine NT-basierende Windows-Version nachbilden, sollten die Dateien in das System32-Verzeichnis kopiert werden. Anschließend muß der Loader von Wine so konfiguriert werden, daß er das betreffende Modul anstelle seiner eigenen Implementierung der DLL lädt.

Grundsätzlich ist von einer Verwendung nativer DLLs abzuraten, da so die Suche nach Fehlern in Wine bei proprietären DLLs deutlich erschwert wird. Die Wine-Entwickler bearbeiten Bug-Reports von Fehlern, die nur bei Verwendung nativer Windows-DLLs anstelle ihrer Nachimplementierungen in Wine auftreten, in der Regel nicht oder nur mit sehr geringer Priorität, da das Hauptaugenmerk natürlich auf das Beseitigen von Fehlern in Wine selbst liegt. Berichte über Fehlfunktionen, die bei Einbindung von nativen DLLs verschwinden, werden aber natürlich gerne bearbeitet, weil diese auf Fehlfunktionen in Wine zurückzuführen sind.

0.2 STÄRKEN

Der größte Vorteil von Wine liegt in der nahtlosen Integration der Windows-Programme in den Unix-Desktop und das Betriebssystem. Grafische Programmelemente wie Fenster und Icons werden direkt auf dem Desktop unter X11 dargestellt. Alle Features des X-Servers stehen somit genau wie bei nativen Unix-Programmen zur Verfügung, allen voran natürlich das Forwarding von Fenstern auf einen anderen Rechner. Auch das Betreiben von Windows-Programmen unter einem freien Terminalserver wie FreeNX (<http://freenx.berlios.de>) ist so möglich. Sogar eine

Einbindung in Desktop-Umgebungen wie KDE oder Gnome beherrscht Wine. Beispielsweise werden Tray-Icons von Programmen wie WinAmp, die unter Windows unten rechts in der Taskleiste angezeigt würden, von Wine korrekt an die entsprechende Stelle in der KDE-Taskleiste gelegt.

Dadurch, daß kein zweites Betriebssystem gebootet werden muß, sind unter Wine ausgeführte Windows-Programme kaum speicherhungriger als unter Windows. Auch die Performance leidet in den meisten Fällen kaum. Zudem benötigt man bei Wine keine Windows-Lizenzen für den Betrieb von Programmen unter zum Beispiel Linux, sofern man auf native DLLs verzichtet.

0.3 WINE INSTALLIEREN

Wine liegt als freie Software immer auch im Quelltext vor und kann deswegen grundsätzlich auch immer selbst kompiliert und aus den Quellen installiert werden. In der Praxis ist hiervon jedoch abzuraten, da für den Übersetzungsprozeß eine ganze Reihe an Entwicklungstools installiert sein müssen, die in einer Standardinstallation von beispielsweise Ubuntu Linux nicht vorinstalliert sind. Zudem müssen auch für alle Teilsysteme von Wine nötigen Bibliotheken und Programme jeweils mitsamt ihren Headerdateien installiert sein, um eine voll funktionsfähige Wine-Version zu erhalten. Da Wine eine erhebliche Menge solcher Abhängigkeiten besitzt, ist dies eine durchaus komplexe Aufgabe, derer man sich auf einfache Weise entledigen kann, indem man die Binärpaket für die jeweils verwendete Distribution installiert.

Die Wine-Pakete (Binärpakete ebenso wie Quelltexte) können von der Webseite <http://www.winehq.org/site/download> heruntergeladen werden. Wer dennoch lieber selbst kompilieren möchte, findet im Wine-Wiki unter http://wiki.winehq.org/Recommended_Packages die wesentlichen Informationen für die wichtigsten Distributionen. Für Ubuntu steht sogar ein Shell-Skript zur Verfügung, das einem einen Teil der Installation von Abhängigkeiten abnimmt.

0.4 WINE KONFIGURIEREN

0.4.1 Die Basiskonfiguration

Wine benötigt für die korrekte Ausführung von Windows-Programmen unter Linux eine bestimmte Verzeichnis- und Dateistruktur, die im Regelfall unterhalb von `$HOME` im Verzeichnis `.wine` abgelegt wird. Dort befindet sich unter anderem auch die Registry, in der unter Wine ausgeführte Windows-Programme ebenso wie Wine selbst ihre Konfiguration speichern. Ebenfalls Bestandteil ist das Verzeichnis `dos-devices` mit den Laufwerksbuchstaben `c:` und `z:`. Diese Laufwerksbuchstaben sind aber nur Links von beispielsweise `C:` nach `$HOME/.wine/drive_c`. Unterhalb dieses Verzeichnisses befinden sich weitere Verzeichnisse, die auf jedem Windows-System vorhanden sind, insbesondere `windows` mit seinen Unterverzeichnissen, und `Programme`. Letzteres ist beispielsweise die Entsprechung des unter Windows bekannten `C:\Programme`.



Eingerichtet wird die Verzeichnisstruktur mit dem Befehl *wineprefixcreate*, der manuell auf der Kommandozeile aufgerufen wird. Er wird aber auch automatisch ausgeführt, falls man Wine startet, ohne daß ein *.wine*-Verzeichnis mit dem »Fake-Windows« vorhanden ist.

Die Konfiguration von Wine ist relativ komplex. Dies ist nicht zuletzt der Tatsache geschuldet, daß eine Kompatibilität mit nicht weniger als 14 Windows-Versionen herzustellen versucht wird, und man viele Elemente – wie zum Beispiel den Loader – konfigurieren kann oder sogar muß, an die man als normaler Anwender unter Linux oder auch Windows nie Hand anlegt. Zudem müssen verschiedene Windows-Eigenheiten auf ihre Äquivalente unter Unix gemappt werden. Dazu zählen die Zuordnung von Laufwerksbuchstaben (Windows) zu Verzeichnissen (Unix), die Zuordnung bestimmter Windows-Systemordner (»Eigene Dateien«, »Desktop« und so weiter) zu Verzeichnissen unter Unix, oder auch die Soundkonfiguration.

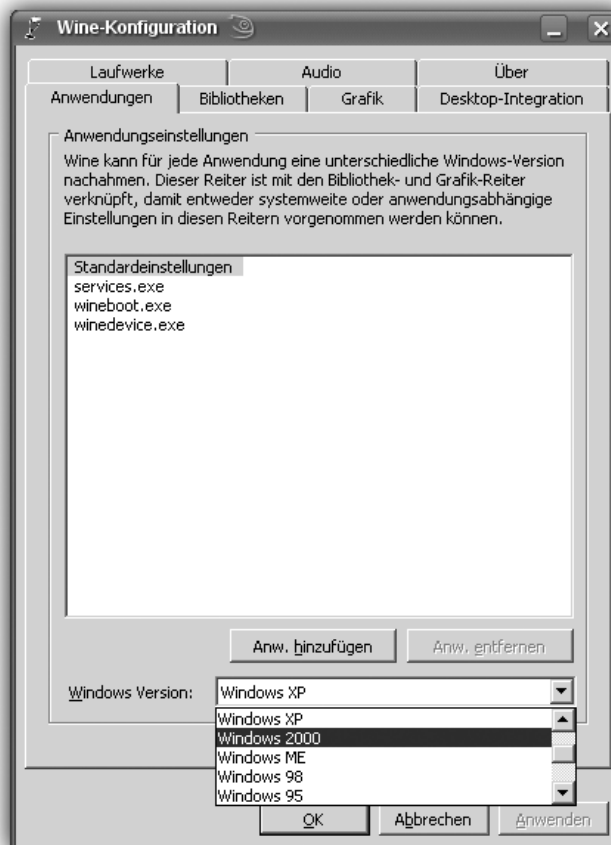


Bild 0.1: Zuordnung der simulierten Windows-Version mit winecfg

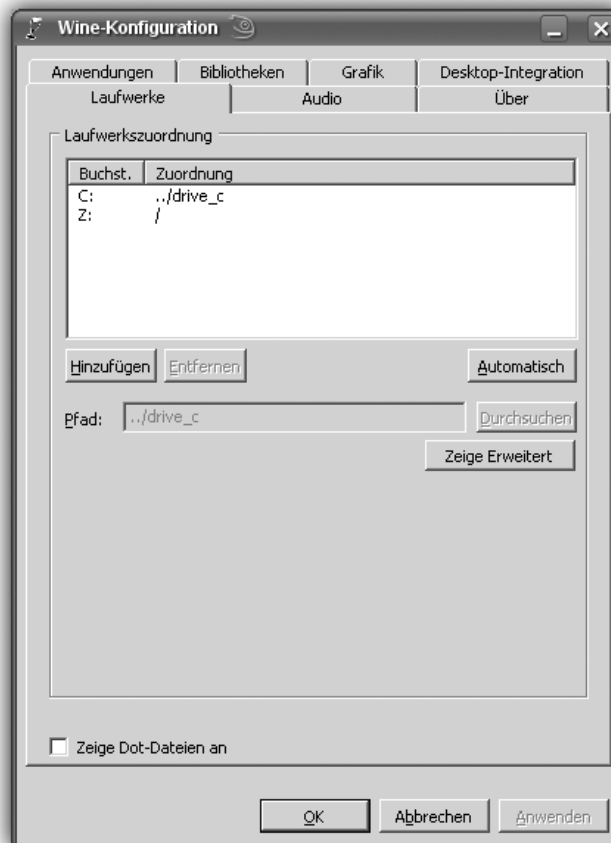


Bild 0.2: Laufwerkzuordnung in Wine

Um diese Konfiguration einfach zu gestalten, enthält Wine das grafische Konfigurationsprogramm *winecfg*. Es empfiehlt sich, mit diesem Programm Einstellungen vorzunehmen und permanent zu speichern. Diese können zum Teil auch applikationsspezifisch sein. Darüber hinaus kann man mit einem dem Registry-Editor unter Windows nachempfundenen Programm namens *regedit* auch direkt Einstellungen in der Registry lesen und schreiben.

0.4.2 DLLs einbinden

Aus rein technischer Perspektive ist das Einbinden von DLLs beispielsweise aus einer Windows-Installation recht einfach: Die jeweilige Datei muß lediglich in das Äquivalent des Windows-Systemverzeichnisses von Wine kopiert werden. Im Regelfall befindet sich dieses unter `$HOME/.wine/drive_c/windows/system`. Anschließend muß der Loader von Wine so konfiguriert werden, daß er das entsprechende Modul an-

stelle seiner eigenen Implementierung lädt. Diese Einstellungen nimmt man über das Konfigurationsprogramm *winecfg* vor.

Die Verwendung von nativen DLLs wird aber auch von den Wine-Entwicklern nicht gerne gesehen, da sie das Debugging von Wine deutlich erschweren. Bugreports, die sich auf Konfigurationen mit aktivierten nativen Windows-DLLs beziehen, werden daher nicht oder nur mit sehr niedriger Priorität von der Community bearbeitet. Reports über Fehler, die bei nativen DLLs verschwinden, sind aber selbstverständlich gerne gesehen, weisen sie doch auf Fehlfunktionen in Wine hin.

0.4.3 Das *winetricks*-Skript

winetricks ist ein kleines Skript, das Benutzern von Wine dabei behilflich sein soll, bestimmte Zusatzpakete zu installieren, die gegebenenfalls für den Betrieb von Programmen unter Wine benötigt werden. Zum großen Teil handelt es sich bei diesen Paketen um DLLs oder andere Module von Microsoft. Typische Beispiele sind die Runtime-DLLs für Visual-Basic-Programme, die Microsoft Core Fonts oder der Windows Scripting Host. *winetricks* beherrscht ebenfalls in dem jeweils dafür notwendigen Umfang die Umkonfiguration des Loaders von Wine, damit er die neu installierten DLLs anstelle der eigenen nutzt.

Das Skript wurde vom Programmierer Dan Kegel geschrieben und wird regelmäßig erweitert. Es benötigt selbst lediglich das Programm *cabextract* und natürlich *wget*, die beide in den üblichen Linux-Distributionen enthalten sind und somit einfach nachinstalliert werden können. Die jeweils aktuellste Version befindet sich unter <http://www.kegel.com/wine/winetricks>. Der einfachste Weg, die jeweils aktuellste Version zu beziehen ist es also, sie in ein Verzeichnis eigener Wahl herunterzuladen und ausführbar zu machen:

```
wget http://www.kegel.com/wine/winetricks && chmod 700 winetricks
```

Zu *winetricks* gibt es im offiziellen Wiki des Wine-Projekts ebenfalls eine Dokumentationsseite, zu finden unter <http://wiki.winehq.org/winetricks>. Dort wird auch noch einmal erwähnt, daß gemeldete Bugs von den Wine-Entwicklern in der Regel nicht bearbeitet werden, wenn Fremd-DLLs von Microsoft installiert wurden. Ausgenommen hiervon sind explizit nur die Installation der Gecko-Engine, von Mono als .NET-Implementierung und das Setzen eines Registry-Eintrags, der Programme vorgaukelt, der Internet Explorer 6 sei installiert. Das Reporten von Fehlverhalten von Wine, das durch die Einbindung von nativen DLLs verschwindet, ist aber natürlich stets erwünscht.

Das Skript wird dann aus dem Verzeichnis, in das es heruntergeladen wurde, mit dem Befehl

```
./winetricks
```

aufgerufen. In einem Dialogfenster werden die verschiedenen Optionen angezeigt. Alternativ kann *winetricks* auch direkt von der Kommandozeile aus aufgerufen werden. Beispielsweise lassen sich, mit

```
winetricks vb6run
```

die Runtime-DLLs für mit Visual Basic 6 erzeugte Programme installieren.

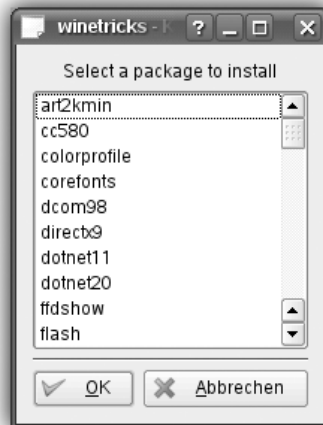


Bild 0.3: Dialogfenster des winetricks-Skripts zur Installation von Windows-Systemteilen

0.4.4 Wichtige Umgebungsvariablen

Das Verhalten von Wine kann mit verschiedenen Umgebungsvariablen beeinflusst werden. Beispielsweise wird Wine angewiesen, die normalerweise unterhalb von $\$HOME/.wine$ befindliche Verzeichnisstruktur in einem anderen Verzeichnis zu suchen. Das geschieht mit der Umgebungsvariable $WINEPREFIX$. So startet der folgenden Befehl das Programm *winecfg*, nutzt aber für die Konfiguration nicht die Registry unter $\$HOME/.wine$, sondern unter $/tmp/winetests/$:

```
WINEPREFIX=/tmp/winetests/ winecfg
```

Darüber hinaus können Einstellungen des Loaders temporär, das heißt einmalig, für die Ausführung eines Programms überschrieben werden. Die hierfür zuständige Variable heißt $WINEDLLOVERRIDES$. Die Syntax für diese Anweisungen folgt dem Schema $[Modulname]=[Wert]$. Zulässige Modulnamen sind alle Namen von DLLs, zum Beispiel *ole32* oder *vbrun6*.

Der Wert *n* steht für »native« und weist den Loader an, eine im Dateisystem befindliche DLL des entsprechenden Namens anstelle der eingebauten Nachimplementierung von Wine zu nutzen. Kann diese DLL nicht gefunden werden, bricht Wine beim Programmstart mit einer Fehlermeldung ab. Das Gegenstück ist der Wert *b*, der für »builtin« steht und für das so konfigurierte Modul immer die Wine-Version nimmt. Möglich sind auch Kombinationen wie *n,b*. Als Beispiel sei der folgende Befehl genannt:



```
WINEDLLOVERRIDES="ole32=n,b" wine BEISPIEL.EXE
```

Dieser Befehl ruft das Programm *BEISPIEL.EXE* unter Wine auf. Das Modul *Ole32.dll* wird dabei, sofern es von dem Programm benötigt wird, in der nativen Version (das heißt, der von Microsoft) geladen. Falls das fehlschlägt, ruft Wine seine eingebaute Implementierung von *Ole32.dll* auf.

0.4.5 Debugging

Um Fehlern bei der Ausführung von Programmen mit Wine gezielt auf den Grund gehen zu können, gibt es in Wine sogenannte Debug-Channels. Sie werden mit der Umgebungsvariable *WINEDEBUG* aktiviert und gesteuert. Jeder Debug-Channel gibt nur Meldungen zu einem bestimmten Teilsystem oder einem bestimmten Modul in Wine aus. Beispiele für Debug-Channels sind *ole*, *winsock* oder *file*. Eine vollständige Liste findet sich im Wiki des Projekts unter <http://wiki.winehq.org/Debug-Channels>. Zu jedem Channel gibt es vier sogenannte Classes, mit denen gesteuert wird, welche Informationen aus dem jeweiligen Channel an die Standardfehlerausgabe geschrieben werden sollen. Diese Classes lauten *err*, *warn*, *fixme* und *trace*. Die Syntax der Channels folgt dem Schema *[Class][+/-][Channel]*. Es können beliebig viele Channels aktiviert werden, die Angabe einer Class ist optional. Wird sie weggelassen, werden alle Meldungen aus dem entsprechenden Channel angezeigt.

Der folgende Befehl führt das Programm *BEISPIEL.EXE* aus und gibt dabei Meldungen aus dem Channel *loaddll* auf der Standardfehlerausgabe aus:

```
WINEDEBUG=loaddll wine BEISPIEL.EXE
```

Es ist zu beachten, daß die Ausgabe der Channels sehr unterschiedlich groß ausfällt. Während *loaddll* meistens relativ überschaubar bleibt, da nur geladene und entladene Module von Loader ausgegeben werden, sind zum Beispiel im Channel *file* sämtliche Dateioperationen einsehbar. Ein Debug-Log wird so sehr schnell unübersichtlich groß und kann unter Umständen selbst in kurzer Zeit mehrere hundert MByte Text erzeugen.

Für die Zuordnung eines Programmfehlers zu Fehlermeldungen aus dem Debug-Channel ist zudem oft ein genauer Zeitstempel jeder Meldung von Wine hilfreich oder sogar – bei umfangreichen Logdateien – unumgänglich. Eventuell zu Testzwecken mittels *WINEDLLOVERRIDES* nativ eingebundene DLLs sind bei der Fehlersuche ebenfalls von Belang, genauso wie die Versionsnummer der beim Testen installierten Wine-Release und andere Umgebungsvariablen.

Bei all diesen Aufgaben unterstützt das GPL-lizenzierte Perl-Skript *winestart*, das unter <http://freshmeat.net/projects/winestart/> zum Download bereitsteht. Wird ein Windows-Programm mit *winestart* aufgerufen, gibt das Programm einen Header mit Umgebungsvariablen und anderen wesentlichen Informationen aus. Jede Zeile Debug-Ausgaben von Wine wird zudem mit einem Zeitstempel versehen. *winestart* beherrscht auch das Schreiben dieser Informationen in eine Datei und die simultane Ausgabe in eine Datei und auf die Standardausgabe.

Alle Optionen des Programms sind mit dem Befehl

```
./winestart.pl --help
```

einzusehen. Der folgende Befehl startet das Programm *BEISPIEL.EXE*, dokumentiert alle Ausgaben und den Programmstart in einer Datei in *\$HOME/.wine/ drive_c/*, dupliziert diese Ausgabe auch auf der Konsole, und setzt zuvor *WINEDEBUG* und *WINEDEBUGLLOVERRIDES* entsprechend:

```
./winestart.pl --program=BEISPIEL.EXE --to-file --double-out \  
--winedebug="loadll" --winedlloverrides="ole32=n,b"
```

Das Namensschema für die angelegten Protokolldateien lautet dabei *Programmname --Wine-Versionsnummer--Datum--Zeit.txt*.

0.4.6 Drucken in Wine

Unter Linux installierte und eingerichtete Drucker stehen unter Wine ausgeführten Windows-Programmen zur Verfügung. Die Druckaufträge werden von Wine an das jeweilige Drucksystem weitergeleitet und von diesem behandelt.

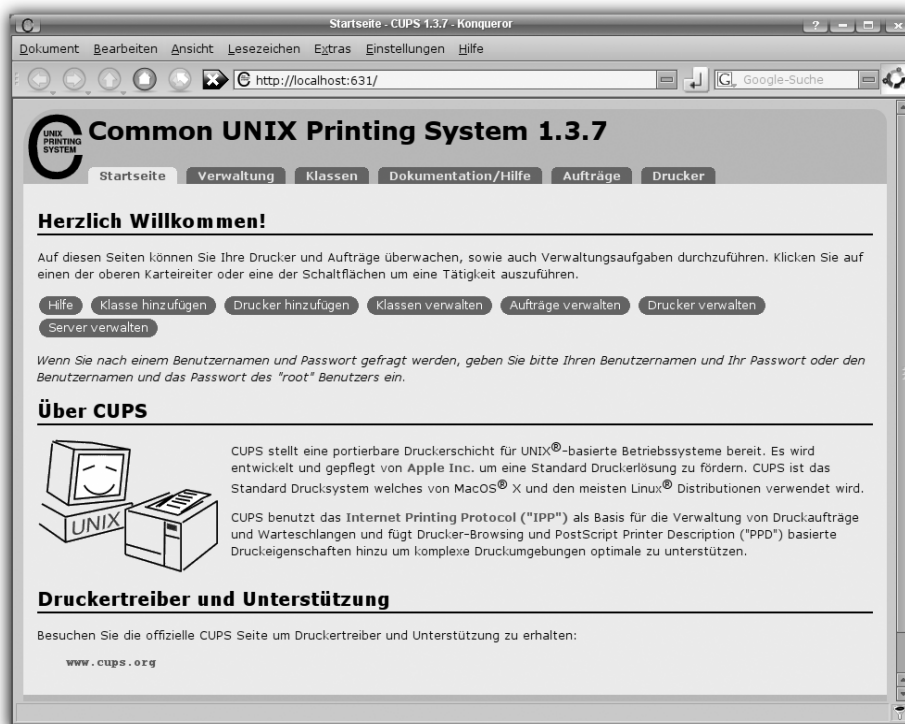


Bild 0.4: Konfiguration der Drucker über die Weboberfläche von CUPS

In der Regel kommt unter Linux das Drucksystem CUPS zum Einsatz. In Ubuntu Linux ist es bereits in der Standardinstallation enthalten. Mit dem Kommando

```
dpkg --get-selections | grep cups
```

kann man auf Ubuntu und anderen APT-basierten-Systemen an der Kommandozeile überprüfen, ob CUPS bereits installiert ist. Auf RPM-basierten Systemen lautet das entsprechende Kommando

```
rpm -qa | grep cups
```

Falls nötig, kann man es auf APT-basierten Systemen mit

```
aptitude install cups
```

nachinstallieren. Bei RPM-basierten Systemen lautet der Befehl

```
rpm -Uhv cups-*
```

Die Administration der einzelnen Drucker erfolgt über den Webbrowser auf der URL <http://localhost:631>.

Unter KDE steht alternativ ein umfangreiches und komfortables Konfigurationsprogramm zur Verfügung, das über das K-Menü oder über den Befehl `kcmshell printers` aufgerufen werden kann.

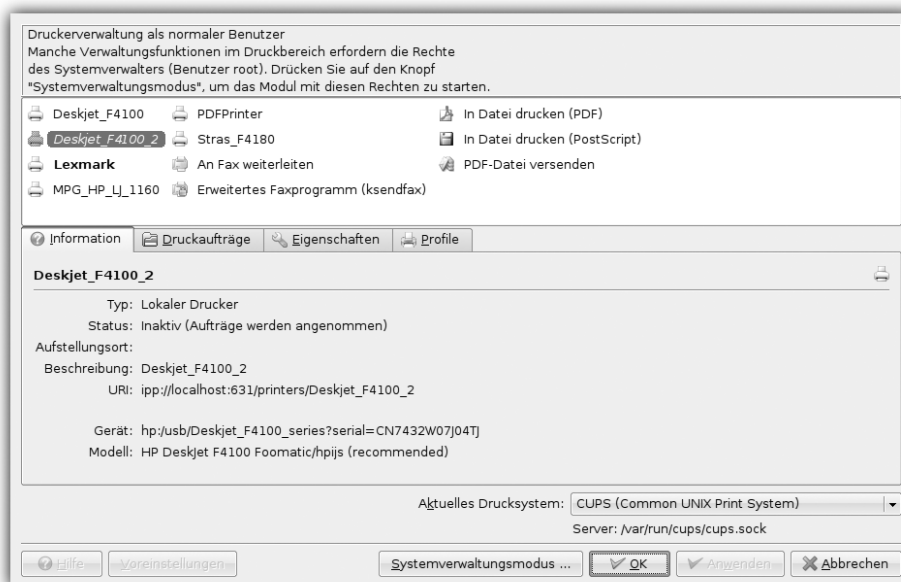


Bild 0.5: Konfiguration der Drucker mit kprinter

0.4.7 TrueType-Fonts einrichten

Speziell für Wine installierte Fonts befinden sich standardmäßig unterhalb des Verzeichnisses `$HOME/.wine/drive_c/windows/fonts`. Der Befehl

```
ls $HOME/.wine/drive_c/windows/fonts/
```

listet folglich alle installierten Fontdateien auf. Im Normalfall ist dieses Verzeichnis leer. Die besonders weit verbreiteten TrueType-Fonts von Microsoft (Core Fonts) finden sich im Internet unter <http://sf.net/projects/corefonts/>. Sie können einzeln heruntergeladen und dann zum Beispiel mit dem Aufruf `wine arial32.exe`

installiert werden. Die Installation mittels `winetricks` ist aber deutlich komfortabler. Man ruft einfach den folgenden Befehl in dem Verzeichnis auf, in das man `winetricks` heruntergeladen hat:

```
./winetricks corefonts
```

0.5 ARBEITEN MIT WINE

0.5.1 Windows-Programme aufrufen

Windows-Programme können auch mit installiertem Wine nicht direkt von der Kommandozeile gestartet werden. Statt dessen muß Wine aufgerufen und der Name des zu startenden Programms als erster Parameter übergeben werden.

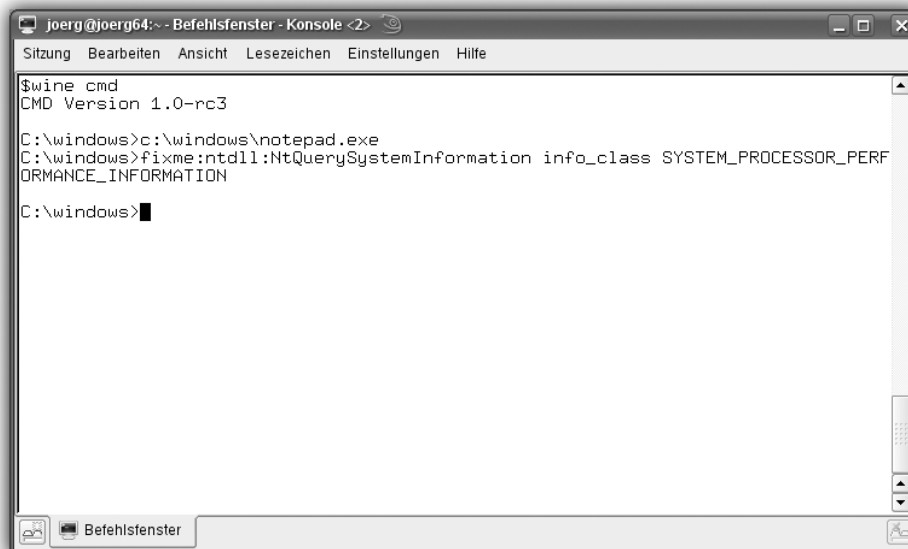


Bild 0.6: Die Kommandozeile von Wine



In Wine sind einige Standard-Windows-Programme wie Minesweeper oder Notepad enthalten. Letzteren startet man an der Kommandozeile beispielsweise mit

```
wine c:\\windows\\notepad.exe
```

Man kann dabei mit der von Windows gewohnten Notation mit Laufwerksbuchstaben arbeiten. Der doppelte Backslash ist erforderlich, da ein einfacher Backslash in der Shell als Escape-Zeichen dient und so `\n` beispielsweise als Zeilenumbruch interpretiert würde. Ebenfalls möglich und deutlich eingängiger ist die Unix-Notation – die gewohnte Angabe von Unix-Pfadnamen:

```
wine $HOME/.wine/drive_c/windows/notepad.exe
```

Muß man Parameter nicht an Wine, sondern an das unter Wine auszuführende Windows-Programm übergeben, werden diesem Parametern zwei Bindestriche (`--`) vorangestellt:

```
wine <PROGRAMM.EXE> -- [Parameter 1] [Parameter 2] ... [Parameter n]
```

DOS-Programme werden mit

```
wineconsole <PROGRAMM.EXE>
```

aufgerufen.

In Wine enthalten ist auch eine Kommandozeile (Shell), die über

```
wineconsole cmd
```

gestartet wird. In dieser Shell können dann wie unter DOS auch Befehle abgesetzt und Programme aufgerufen werden. Mit

```
C:\windows\notepad.exe
```

startet man aus dieser Shell beispielsweise die Notepad-Implementierung von Wine. Durch die Tastenkombination `[Strg][C]` in der Shell kann man das jeweils dort gestartete unter Wine laufende Programme terminieren. Der Befehl

```
wineserver -k
```

hält sämtliche unter Wine laufende Programme an. Die unsanftere Alternative dazu ist

```
killall -9 wine-preloader
```

0.5.2 Windows-Programme installieren

Photoshop CS2

Ein relativ mächtiges Programm, von dem leider keine Linux-Version verfügbar ist, ist Adobe Photoshop. In der Version CS2 läßt es sich unter Wine installieren und benutzen. Dazu muß zunächst – wenn keine Vollversion verfügbar ist – die 30-Tage-Testversion vom Adobe-Server heruntergeladen werden¹. Der Download ist etwa 290 MByte groß. Das heruntergeladene ZIP-Archiv kann anschließend in ein Unterverzeichnis *TryOut* extrahiert werden:

```
unzip PS9_Tryout_d.zip
```

Mit dem folgenden Befehl wird die Installationsroutine von Photoshop aufgerufen. Man folgt dem Wizard bis zum Ende der Installation. Wenn gewünscht, kann man sich auch noch die README-Hinweise von Photoshop anzeigen lassen.

```
cd TryOut; wine setup.exe
```

Die Installation benötigt aufgrund der Größe der Software selbst auf relativ leistungsfähigen Rechnern ein wenig Zeit. Nachdem sie durchgelaufen ist, kann man an der Kommandozeile das Photoshop-Programm starten:

```
cd $HOME/.wine/drive_c/Programme//Adobe/Adobe\ Photoshop\ CS2  
wine Photoshop.exe
```

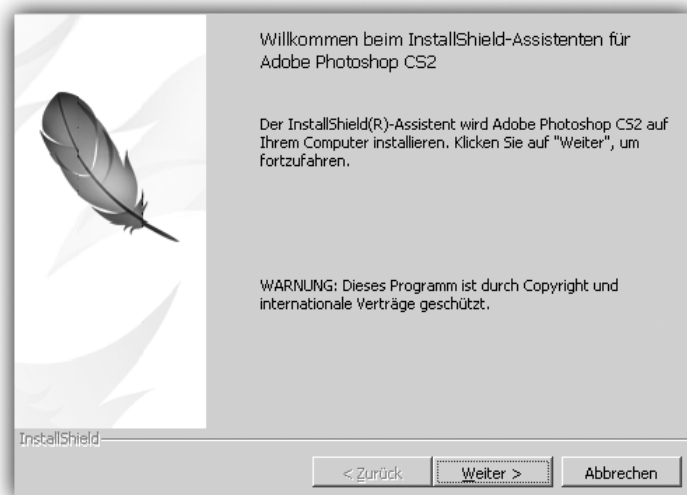


Bild 0.7: Installations-Wizard von Photoshop CS2

¹ Der Download-Link ist ftp://ftp.adobe.com/pub/adobe/photoshop/win/cs2/PS9_Tryout_d.zip

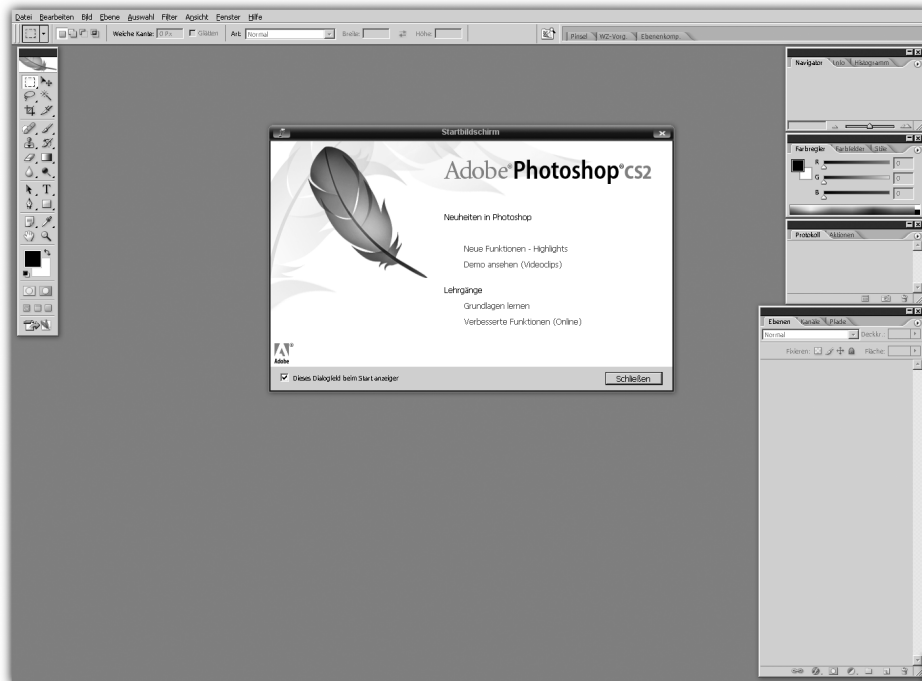


Bild 0.8: Das Hauptfenster von Photoshop CS2 unter Wine

Spiele

Als weiteres Beispiel wird nun die kostenlose Demo-Version des Spiels *SpongeBob Collapse* (http://www.download-free-games.com/puzzle_game_download/) installiert. Dazu wird das Installationsprogramm *SBCollapseSetup.exe* im */tmp*-Verzeichnis gespeichert und die Installation mit

```
cd /tmp; wine SBCollapseSetup.exe
```

aufgerufen.

Für die korrekte Ausführung des Spiels unter Wine wird die Gecko-Engine benötigt. Man installiert sie zum Beispiel mit *winetricks*. Beim ersten Start des Spiels bemerkt Wine jedoch selbständig die Abwesenheit der Engine, die dazu benötigt wird, das API des Internet Explorers unter Linux für Windows-Programme zur Verfügung zu stellen. Ein Dialogfenster erlaubt dann, per Mausklick Gecko herunterzuladen und zu installieren. Voraussetzung ist selbstverständlich eine funktionierende Internet-Verbindung.



Bild 0.9: Der Gecko-Installationsdialog

SpongeBob wird im Verzeichnis *C:\Programme\Sponge Bob Collapse* installiert, wiederfinden wird man es Verzeichnis unter *~/wine/drive_c/Programme/Sponge Bob Collapse*. Bei der Installation wird unter KDE zudem ein Icon auf dem Linux-Desktop abgelegt, über das das Spiel per Mausklick gestartet wird.

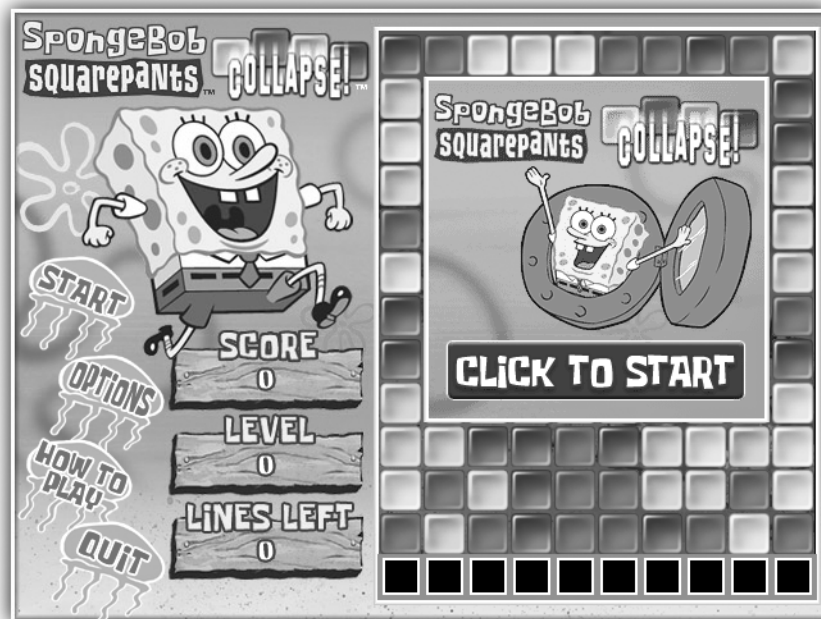


Bild 0.10: Das Spiel SpongeBob Collapse unter Wine

0.5.3 Windows-Programme deinstallieren und archivieren

Wine bietet eine einfache grafische Oberfläche für das Entfernen von installierten Programmen. Dies wird mit dem Befehl

```
wine uninstaller
```

gestartet. Vollständig gelöscht werden alle installierten Windows-Programme mit- samt der dazugehörigen Wine-Konfiguration durch das Löschen des *.wine*- Verzeichnisses:

```
rm -rf $HOME/.wine
```

Selbstverständlich ist es auch möglich, das *.wine*-Verzeichnisse zu archivieren, zum Beispiel mit

```
mv $HOME/.wine $HOME/.wine-archiv-PROGRAMM_XY && \  
tar -cjf wine-Verzeichnis-PROGRAMM_XY.tar.bz2 \  
$HOME/.wine-archiv-PROGRAMM_XY
```

Dieser Befehl verschiebt das *.wine*-Verzeichnis nach *.wine-archiv-PROGRAMM_XY* und legt dann ein komprimiertes tar-Archiv mit dessen Inhalt in der Datei *wine-Verzeichnis-PROGRAMM_XY.tar.bz2* an.

0.6 WINELIB-PROGRAMME

Die Winelib ist eine Bibliothek, die alle von Wine selbst nachimplementierten APIs aus Windows enthält und als Unix-Systembibliothek zur Verfügung stellt. Bestandteil sind zudem verschiedene Entwicklungswerkzeuge für das Übersetzen von Windows-Programmen unter Unix und Wine.

Voraussetzung für das Einbinden der Winelib ist, daß das zu nutzende Programm im Quelltext vorliegt und in C beziehungsweise C++ geschrieben ist. Im Idealfall genügt dann ein einfaches Neukompilieren des Quelltexts, um eine zum Beispiel auf Linux einsatzfähige Version eines bestehenden Windows-Programms zu erhalten – und das, ohne am Quelltext etwas zu verändern. Da es sich beim Resultat um ein natives Unix-Programm handelt, kann man im Quelltext bei Bedarf auch Systemaufrufe aus zum Beispiel Linux setzen, um beispielsweise bestimmte Funktionalitäten besser in ein Linux-System zu integrieren. Auf diese Weise läßt sich auch unter weitgehender Beibehaltung des Original-Quelltextes um eventuelle Lücken in der Wine32-API-Nachimplementierung von Wine »herumprogrammieren«.

Grundsätzlich gelten für Winelib-Programme allerdings auch die gleichen Einschränkungen wie für alle Programme unter Wine: Daß eine bestimmte Funktion von Windows unter Wine bereits vorhanden ist, heißt noch nicht automatisch, daß ihr Implementierungsstand auch schon so weit fortgeschritten ist, daß sie im Einsatzszenario des zu portierenden Programms perfekt funktionieren wird. Zu empfehlen sind daher Testläufe. Sollte ein API in Wine schlicht fehlen, fielen dies allerdings bereits beim Kompilieren und Linken auf.

Damit bisher unter Windows übersetzter Quelltext unter Unix kompiliert werden kann, müssen verschiedene Kriterien erfüllt sein, die oftmals syntaktische Änderungen am Quelltext erfordern: Unter Unix wird bei Dateien und Pfadangaben Groß-/Kleinschreibung unterschieden, so daß eventuell Pfadangaben korrigiert werden müssen. In Include-Anweisungen stehende Backslashes (»\«) müssen in Slashes (»/«) gewandelt werden – diese funktionieren mit standardkonformen Compilern auch unter Windows. Zudem müssen gegebenenfalls Zeilenenden und -umbrüche von CRLF auf LF umgeschrieben werden. Anstelle des unter Windows verwendeten Compilers oder des *gcc* beziehungsweise *g++* unter Unix müssen *winegcc* und *wineg++* – Wrapper um die jeweiligen Compiler – zum Einsatz kommen.

Um diese Anpassungen weitgehend zu automatisieren, liefert Wine das Perl-Skript *winemaker* mit. Es paßt nicht nur Quelltexte an, sondern fertigt, sofern nötig, auch Sicherungskopien der Original-Dateien an und erstellt ein neues Makefile, das mit GNU make abgearbeitet werden kann.

Ein einfaches Hello-World-Programm, das das Windows-API nutzt und unter Windows lauffähig wäre, sieht beispielsweise wie folgt aus:

```
#include <WINDOWS.H>

int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE hPrevInstance,
                  PSTR CmdLine, int CmdShow) {
    MessageBox(NULL, "Hello World", "Hello World", MB_OK);
    return 0;
}
```

Speichert man diesen Quelltext in einer Datei *HELLOWORLD.CPP* in einem neu angelegten Verzeichnis *Winetest*, kann man nach Wechsel in dieses Verzeichnis an der Kommandozeile *winemaker* über dieses Mini-Projekt laufen lassen:

```
cd Winetest; winemaker .
```

Beim Beispiel des Hello-World-Programms muß die Include-Anweisung in der ersten Zeile auf Kleinbuchstaben umgestellt werden, und der in Großbuchstaben gehaltene Dateiname von *HELLOWORLD.CPP* wird analog ebenfalls in Kleinbuchstaben gewandelt. Vor der Änderungen in Dateien selbst legt *winemaker* eine Sicherheitskopie an. Zudem erstellt er ein Makefile. Der Verzeichnisisinhalt von *Winetest* sieht also nach dem Aufruf von *winemaker* wie folgt aus:

```
main.cpp main.cpp.bak main.o Makefile winetest.exe.so
```

Der Aufruf von

```
make && wineg++ -o helloworld winetest.exe.so
```

stößt die Kompilierung des Quelltextes an und man erhält ein Shellskript *helloworld*, das man einfach an der Kommandozeile aufrufen kann, um das Programm zu starten:



```
./helloworld
```

Das funktioniert allerdings nur, wenn auch die Datei *windows.h* gefunden wird. Das geschieht aber nur, wenn außer dem Wine-Paket selbst auch das entsprechende Devel-Paket installiert ist.

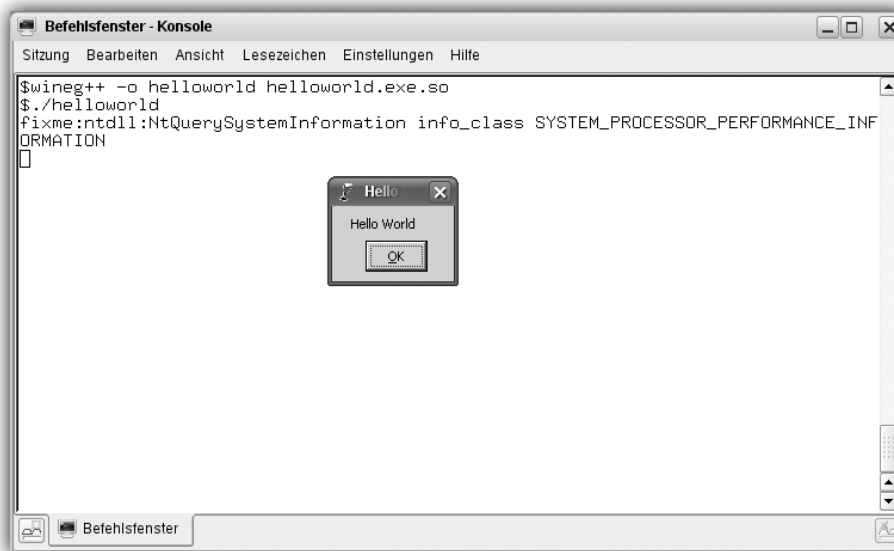


Bild 0.11: Ein Win32-Hello-World unter Linux

Ausführlichere Auskünfte zur Winelib gibt die entsprechende Dokumentation auf <http://www.winehq.org/site/winelib> sowie der Winelib User's Guide unter <http://www.winehq.org/site/docs/winelib-guide/index>. Auf Sourceforge befindet sich zudem ein Programm, das Programmierern beim Beheben typische Fehler beim Übersetzen von Windows-Programmen mit der Winelib behilflich sein soll. Das *cherubi* getaufte, Python-basierte Projekt scheint allerdings derzeit nicht aktiv weiterentwickelt zu werden (<http://sourceforge.net/projects/cherubi>).

0.7 ALTERNATIVEN UND ANDERE VERSIONEN

Von Wine gibt es verschiedene Derivate und Versionen. Nachfolgend sind die wichtigsten kurz aufgeführt und erklärt.

0.7.1 CrossOver Office

CrossOver Office ist ein Produkt der Firma Codeweavers (<http://www.codeweavers.com>) und besteht im wesentlichen aus einer Wine-Version, die mit einigen zusätzlichen Patches und Hacks versehen ist, um bestimmte Programme wie Microsoft

Office oder Adobe Photoshop unter Linux lauffähig zu machen. Zusätzlich enthalten sind noch einige meist grafische Tools, um beispielsweise die Installation bestimmter Windows-Programme für den Anwender einfacher zu gestalten. Die Professional-Variante enthält obendrein noch Mechanismen zum erleichterten Deployment von Windows-Software unter Wine.

Da Wine unter der LGPL steht, ist auch diese speziell gepatchte Wine-Variante unter der gleichen Lizenz zu bekommen und steht auf der Website der Firma zum Download bereit. Sie muß jedoch aus dem Quelltext übersetzt werden, da die Binärpakete selbstverständlich kostenpflichtig erworben werden müssen. Der Kompilier- und Installationsprozeß läuft anders ab als bei Wine selbst, weil auch dieser Teil von Codeweavers gepatcht worden ist. Die Wine-Version, auf der die jeweils aktuelle CrossOver-Office-Version basiert, liegt meist um etwa ein halbes Jahr hinter der gerade aktuellen Wine-Release.

Von Codeweavers gibt es auch Varianten ihrer Produkte speziell für Spiele und für mit Intel-Hardware ausgestattete Macintosh-Rechner.

0.7.2 ReactOS

ReactOS (<http://www.reactos.org/de/index.html>) ist eine freie Implementierung eines zu Microsoft Windows NT kompatiblen Betriebssystems. Das GPL-lizenzierte Projekt startete bereits 1996, damals unter dem Namen »FreeWin95« und mit dem Ziel einer Kompatibilität mit Windows 95. Es ist weit von einer Fertigstellung entfernt und befindet sich trotz in der Vergangenheit erreichter erheblicher Fortschritte noch in der Alpha-Phase. ReactOS arbeitet bei den Teilen des Betriebssystems, die das Win32-API betreffen, eng mit dem Wine-Entwicklungsteam zusammen. Bei ReactOS handelt es sich im Gegensatz zu Wine um ein vollständiges Betriebssystem, nicht um eine Kompatibilitätsschicht für Windows-Applikationen unter Unix.

0.7.3 Cedega

Cedega (<http://www.cedega.com/>) ist eine kommerzielle Wine-Version der Firma Transgaming. Sie basiert auf einer Weiterentwicklung des Teils des Wine-Codes, der unter der MIT-Lizenz stand. Diese Lizenz verlangt es nicht zwingend, Änderungen am Quelltext auch zu veröffentlichen, so daß Transgaming Weiterentwicklungen verkaufen konnte, ohne sie an das Projekt zurückfließen lassen zu müssen. Das Wine-Projekt selbst änderte aus diesen Gründen die Wine-Lizenz zur LGPL. Der Cedega-Quelltextbaum von Wine hat sich dadurch seither immer weiter von dem von Wine selbst entfernt, weil aus lizenzrechtlichen Gründen Cedega nicht von den unter der LGPL stehenden Verbesserungen in Wine selbst profitieren konnte.

Hauptziel von Cedega ist die Realisierung der DirectX-Funktionalitäten von Windows und hatte auch daher früher den Namen »WineX«. DirectX ist eine API zur Darstellung von 2D- und 3D-Grafiken unter Windows. Die im Laufe der Zeit entstandene Implementierung von DirectX lizenziert Transgaming unter dem Namen SwiftShader an kommerzielle Kunden. Zudem gibt es mit Cider ein Wine-basierendes Produkt, das Spieleherstellern bei der Portierung von PC-Spielen auf



den Mac helfen soll – in etwa analog zur Mac-Version von CrossOver-Office, nur mit anderer Zielgruppe.

0.8 KOMMERZIELLER SUPPORT

Für Wine ist kommerzieller Support verfügbar. Hauptanbieter ist dabei die Firma Codeweavers in den USA, bei der auch ein Großteil der Wine-Entwickler zumindest in Teilzeit tätig ist. In Kanada ansässig ist die Firma Lattica (<http://www.lattica.com>), die einem ehemaligen Wine-Entwickler gehört und ebenfalls Support leistet. In Deutschland bietet die Firma ITOMIG (<http://www.itomig.de>) Support an.

Der Entwickler des winetricks-Skripts, Dan Kegel, führt auf seinen Webseiten eine Liste mit Anbietern von Support für Wine. Diese Liste ist unter <http://www.kegel.com/wine/isv/#consultants> zu finden.

0.9 AUSBLICK

Durch den vom Projekt gewählten Ansatz der *Nachimplementierung* des Windows-API liegt der Stand von Wine prinzipbedingt immer hinter dem aktuell von Microsoft als De-facto-Standard vorgegebenen. Kursierte lange Zeit der Witz in der Community, daß Wine seit zehn Jahren stets ein halbes Jahr vor der ersten stabilen Release stünde, so hat sich insbesondere seit Freigabe der ersten Beta-Version 0.9.0 im Oktober 2005 in Wine einiges stark verbessert.

Ein wichtiger Fortschritt ist, daß zu allen wesentlichen Funktionen des Windows-API inzwischen Regressionstests als Bestandteil von Wine vorliegen. So wird verhindert, daß bei der Implementierung neuer Funktionen in bereits fertiggestellte Teile Fehler eingeschleppt werden. Zudem wird so verifiziert, daß sich die nachimplementierten Funktionen in Wine genau wie ihre Vorbilder unter Windows verhalten, denn die Testsuite ist auch auf Windows lauffähig.

Die Stabilität von Wine hat sich seit 0.9.0 deutlich verbessert, da auch in immer stärkerem Maße seitens der Entwickler der Fokus auf die Bereinigung von Fehlern gelegt wurde. Auch wurden erhebliche Anstrengungen unternommen, um die Lauffähigkeit von möglichst vielen Installern wie InstallShield unter Wine zu gewährleisten. Zudem wurde die Kompatibilität bestimmter Schlüsselanwendungen wie beispielsweise Photoshop deutlich verbessert. Zum 15. Jahrestag der Geburtsstunde von Wine im Jahre 1993 erschien daher die Version 1 im Juni 2008.



STICHWORTVERZEICHNIS

B	
Binärformat der ausführb. Dateien	5
C	
Cadega	24
Codeweavers	23
CrossOver Office	23
D	
DirectX-Funktionalitäten v. Windows	24
E	
ELF-Format	5
N	
Native Win.-DLLs in Wine einbinden	7
P	
Portable Executable Format	5
R	
ReactOS	24
S	
Stub-API	6
W	
Windows-Anwendungen unter Unix	5
Windows-API unter Unix	5
Windows-Executables, Loader	5
Windows-Lizenzen, Wine	8
Wine installieren	8
Wine konfigurieren	8
Wine, Core Fonts	16
Wine, CUPS	15
Wine, Debugging	13
Wine, Derivate	23
Wine, DLLs einbinden	10
Wine, DOS-Programme aufrufen	17
Wine, drucken	14
Wine, Gecko-Engine	19
Wine, Implementierungsstand	6
Wine, kommerzieller Support	25
Wine, Laufwerksbuchstaben	8
Wine, Loadereinstellungen überschreiben	12
Wine, Photoshop	18
Wine, regedit	10
Wine, Runtime-DLLs für VB-Programme	11
Wine, Spiele	19
Wine, Standard-Windows-Programme	17
Wine, TrueType-Fonts	16
Wine, Umgebungsvariablen	12
Wine, Unix-Systembibliothek	21
Wine, Verzeichnisstruktur einrichten	9
Wine, Windows Scripting Host	11
Wine, Windows-Lizenzen	8
Wine, Windows-Programme aufrufen	16
Wine, Windows-Programme de-installieren	21
Wine, Windows-Programme installieren	18
Wine, Windows-Systemordner	9
Wine, Winelib-Programme	21
Wine, Zusatzpakete installieren	11
winecfg	10
Winelib	5
WINEPREFIX	12
wineprefixcreate	9
winetricks-Skript	11