Building a Wine environment for Windows applications

Versatile Vintage

More and more Windows applications run on Linux thanks to Wine. If you spend a little time on configuration and troubleshooting, you won't be stuck in Windows – even with applications that no one dreamed would run on Linux. *By Harald Jele*

Beyond the easily replaceable large office and graphics packages, the market bustles with countless, typically smaller programs that cater to the needs of small- to medium-sized niches. Commonly, companies or government offices create their own custom, mission-critical applications.

The famous Wine system libraries [1] provide a means for running Windows applications from within Linux. Right now, version 1.4 is the stable release, and development release 1.5.6 is also available.

Configuring Wine has completely changed in the course of the years. Although Wine was once regarded as complicated, it has become significantly easier and clearer, and it handles many annoying details that were formerly left to the admin. These details that once caused grief and are now relatively painless include integrating removable devices such as DVDs or integrating the CUPS printing system.

Figure 1: A graffiti portrait of Robert Musil at the "Musilhaus" in Klagenfurt.

An equally significant change in the architecture of Wine was the introduction of prefixes (since 2003, also known as bottles). Setting up a prefix means you can install and operate Windows software within a pre-defined context without conflicting with other software.

A recent project at the Robert Musil Institute of the University of Klagenfurt provides an interesting case study for Wine in the real world. The institute recently produced a new edition of the works of Austrian author Robert Musil (Figure 1). The electronic version of this new edition is based on the Folio Views application, which only runs on Windows. We used Wine to create a version that can easily launch from Linux. (See the box titled "Virtual: Robert Musil in Klagenfurt.")

Decant and Install

Wine is quickly installed on any popular distribution, and Ubuntu's repository proves particularly friendly.

AUTHOR

Dr. Harald Jele works for the University of Klagenfurt and is currently engaged with the technical aspects of putting Klagenfurt author Robert Musil's works on Linux, Mac OS X, and the network.

Setting up the `wine` metapackage with the Apt package manager takes into account the necessary libraries and core applications and also provides the necessary configuration tools, integrating Gecko to render HTML views and frequently used fonts from the Windows environment.

For a list of available versions, search with `apt-cache search wine` or `aptitude search wine` (Listing 1).

Creating a Prefix

A prefix creates a virtual environment with custom settings for running a Windows application on Wine. Wine always creates a prefix if a prefix is needed but not available. However, the admin can – and should – explicitly create a prefix in advance. Wineboot uses the command

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

to set the Wine prefix *name_of_prefix* as the directory structure in the user's home directory. Creating a prefix makes it relatively safe to experiment, without the user running the risk of trashing the entire software installation with a simple action. Careful use of a prefix puts the dreaded Windows system "installation death battles" firmly in the past. Also, parallel use of software that would native conflict using Windows on the same computer is now possible because each program believes it has exclusive use of the system.

Once a prefix is suitably created and configured, you can easily copy it to your new home directory when you move to a different computer – and this also works in multiuser operations. The admin just has to maintain the structure of the files and directories, or else correctly create a new structure. The best approach is to issue the command

```
tar cfzv name_of_prefix.tgz name_of_prefix
```

to prepare the directory with the prefix for the move. Then, unpack by running `tar xfzv` in the home directory of the desired user.

If you do not create your own prefix, or if you forget to point to a prefix when you launch a program, the results of

VIRTUAL: Robert Musil in Klagenfurt

The Klagenfurt edition of Robert Musil's works was released in 2009 as an electronic edition on DVD – the project used Folio Views, a Windows application (Figure 2). To extend the group of potential users, the institute decided to offer the second edition, which will be published in 2013 with major additions to the content and corrections, on Linux, with Wine as a pre-built installation in a VirtualBox [16].

In particular, the project stakeholders aimed to address users who prefer Linux and Mac OS X. Additionally, the configuration supports simple integration into a local area network, without the need to invest in additional software.

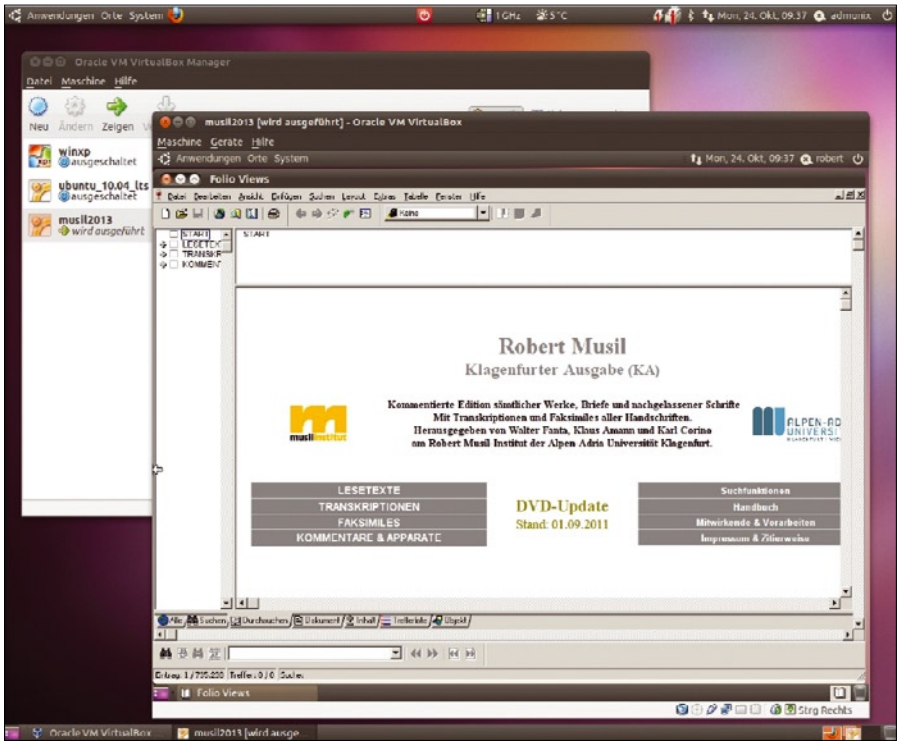

Figure 2: The digital version of the entire works of Klagenfurt writer Robert Musil, published by the University of Klagenfurt with Folio Views, Wine, Linux, and VirtualBox.

LISTING 1: Wine Packages on Ubuntu 11.10

```
01 Playonlinux 01 - Frontend for wine
02 02 Wine1.2-gecko - Microsoft Windows compatibility layer (web browser)
03 03 Wine1.2-gecko - Microsoft Windows compatibility layer (web browser)
04 :
05 05 q4wine - Qt4 GUI for wine (W.I.N.E)
06 06 Wine - Microsoft Windows Compatibility Layer (Meta Package)
07 07 Wine1.2 Microsoft Windows Compatibility Layer (Binary Emulator and Library)
08 08Wine1.2-dbg - Microsoft Windows Compatibility Layer (debugging symbols)
09 09 wine1.2-dev - Microsoft Windows Compatibility Layer (Development files)
10 :
11 11 winetricks - Microsoft Windows Compatibility Layer (winetricks)
12 12 gnome-exe-thumbnailer - Wine .exe and other executable thumbnailer for Gnome
13 13:separate wine settings
14 14 Wine1.3 - Microsoft Windows compatibility layer (Binary Emulator and Library)
15 15 wine1.3-dbg - Microsoft Windows Compatibility Layer (debugging symbols)
16 16 wine1.3-dev - Microsoft Windows Compatibility Layer (Development files)
17 :
```

all activities will land in the `.wine` subdirectory of the user's home directory (`~/.wine`), and the user will have to live without prefix benefits.

Simulated Restart

Wineboot basically performs the same actions a Windows system would. The full range of Wineboot options is described at the Wine project website [2]. The directory structure generated by Wineboot gives you a working environment that the admin can inspect more closely with standard Wine tools. The command

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

executes the built-in Explorer in Wine and presents the view of the Wine server on the mounted filesystems along with the files it contains (Figure 3).

Wine Explorer gives the admin an option for verifying that the filesystems are mounted in the desired way when creating or updating the current Wine prefix – that is, when specifying the correct drive letter from the Windows world. Without

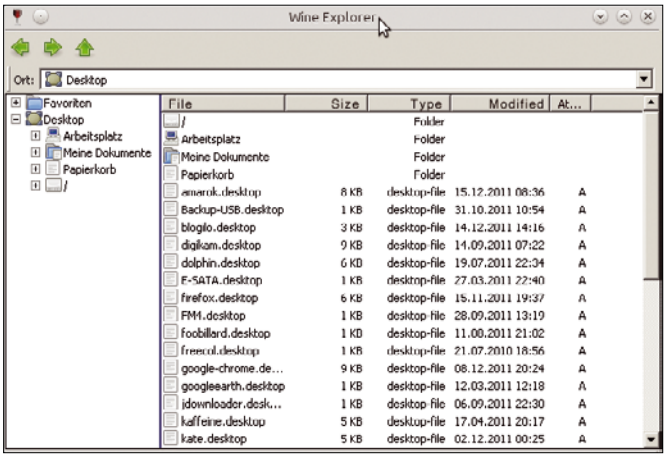

Figure 3: The built-in Wine Explorer is perfect for testing a fresh installation.

admin intervention, Wine only shows CD-ROM and DVD drives if they are mounted on the Linux system.

A parallel exploration of the prefix directory with Linux tools quickly reveals what is essentially a self-explanatory directory structure: `~/name_of_prefix/drive_c` maps the "booted" hard disk of a typical Windows installation; `~/name_of_prefix/dosdrives` lists the drives created, along with their drive letter assignments.

Regedit

Users launch the Wine Regedit tool (Figure 4) with the following command:

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

The command edits the system's prefix-specific Registry. The tool stores the values that you inspected or edited values in three files below `~/name_of_prefix/`: `system.reg`, `user.reg`, and `userdef.reg`. Even if the temptation is great, hard-core shell users should steer clear of these files and use Regedit. The Wine wiki provides a handy overview of the Regedit boot options [4].

From now on, the nerve center for any further configuration of the prefix is the graphical Wine tool `winecfg`, which users can launch by typing

```
env WINEPREFIX= ~/name_of_prefix winecfg
```

On the one hand, this GUI provides a convenient approach to configuring Wine's behavior, and on the other hand, it also launches or installs applications. That said, if you want full control over the installation process, there is no alternative to the command-line programs.

A Matter of Taste: Native or Built-in

The wine installation puts a considerable number of libraries (DLLs) on the target machine. Most of these libraries were re-engineered by the Wine developers, which often removes the need to own a Windows license. Because these DLLs cover many system calls, many applications will work without much assistance. If something fails, it is often helpful to check out the `winecfg` application's *Libraries* tab, which lets you change the order Wine loads Windows libraries.

You can expect headaches after a Wine installation if the supplied libraries fail to provide essential functions for a specific application. In this case, it is necessary to integrate and use the native DLLs provided by the Microsoft Windows installation instead of the DLLs that come with Wine. The `winecfg` tool's *Libraries* tab lets you specify native Microsoft libraries.

Here, one needs to use the Microsoft `cfgmgr32` DLL, which the admin will have to dig out from a Windows instance and plant in the right position in the Wine directory structure. This means the file from the Windows `c:\windows\system32\cfg-`

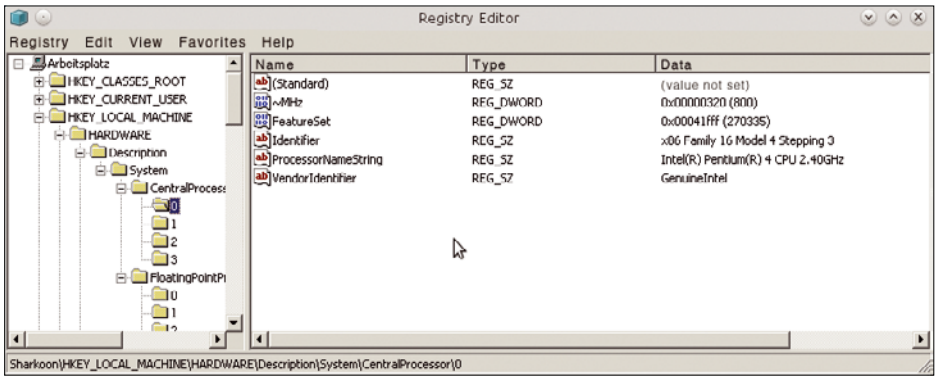

Figure 4: Wine comes with its own Registry Editor.

`mgr32.dll` directory ended up in the Wine prefix below *~/name_of_prefix*`/drive_c/windows/system32`.

Cozy Coexistence

This change leaves the following entry in the `user.reg` file and a corresponding entry in the Registry,

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

enabling Wine to emulate the behavior that Microsoft introduced in Windows XP, known as side-by-side (SxS) [5]. The `c:\windows\WinSxS` directory contains several versions of the same system library side-by-side, broken down by subdirectories. Windows applications can request clearly specified versions from the system.

Installation routines quite often make use of this ability. If an application install fails and no special information is available on the reasons for the failure, replacing this directory with the equivalent directory from a Windows installation can be useful, at least for the duration of the install.

However, special caution is advised: System libraries are quickly replaced, entire directories and tools are expeditiously modified, and tools such as Winetricks [6] dump a plethora of free Microsoft files into the Wine environment. This behavior is not always comprehensible; therefore, the Wine developers often warn users: If you expect help from the community, you should refrain from "enhancing" your installation with too many DLLs [7]. Installing a Windows application is something users should tackle without in-depth manipulation of the system.

Launching Applications

Use the following command to launch the typical Windows Setup program for a custom application or game from a DVD drive mapped to the drive letter `D` at the command line:

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

Wine also offers other mechanisms for starting programs. The official syntax, according to the Wine wiki, is:

```
env WINEPREFIX=~/name_of_prefix wine
   start 'D:\Setup.exe'
```

Or you can use the Unix convention for the pathname:

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

The second variant is recommended, especially if the installation routine or the program does not implement individual paths properly [8].

If the Setup program completes without canceling, the chances are good that the installation has worked. In this case, Wine works just like Windows would: If the installation calls for an entry on the desktop with a (linked) icon, Wine provides it, and an entry appears in the Linux desktop menu for Wine; the installed programs are located below this menu.

Gather Information

If the installation fails or the Windows application will not launch, the following are useful troubleshooting techniques – preferably consulted before you start initial testing:

- Check to see whether the application is entered in the Compatibility Database by the commercial Wine developers, CodeWeavers [9].
- Look for an answer on the web. Many attempts – successes and failures – are documented online but not registered with CodeWeavers.
- Install the program on a native Windows instance for the sake of comparison.
- Use the special Wine Debugger [10].

Working with a parallel Windows instance requires some preparation because the aim is to capture the details of the installation process and the way the application is called. Your tool of choice should log the installation, telling you which directories, files, and Registry entries the installation routine creates, and what libraries it adds to the system. Even if you can achieve this with a modicum of manual work, software such as the free TrackWinstall (Figure 5) [11] offers the same service and some convenience on top.

It is ideal to have a tool that, after monitoring an installation process, creates the directories and files, the new Registry en-

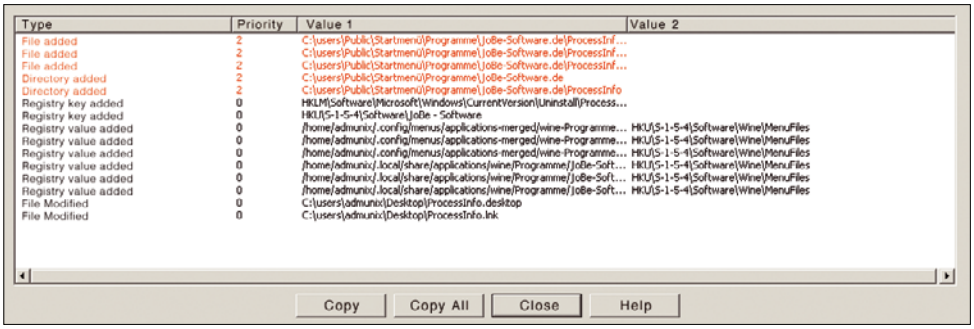

Figure 5: The results list shows all the changes found on a Windows installation by TrackWinstall.

tries, and the relevant system libraries and lets you easily export them directly into a plain vanilla Wine environment. If you then avoid reinstalling, you can launch the application via the commands or icons described above.

If this approach fails, a more in-depth inspection of the application's activity in the parallel Windows installation is then necessary. The first step is to start the application and check the system libraries it loads when launched. Simple tools are fine for this task; many of these tools are available as freeware or as demo versions for limited free use. DLL Show [12] is a veteran already, but it still provides good service and reliably reveals applications and associated libraries loaded in the operating system's memory space.

A similar service is provided by Process Info [13], which is free for 30 days of use (Figure 6). The limitations of this temporary version compared with the full version will be irrelevant to most Linux users, especially because the tool can be installed and executed in a Wine environment.

On the basis of the list of libraries the monitored software loads, admins can gain an initial impression of what they need to pay attention to. The output in the launch console might already reveal whether a Wine library fails to offer an application all the relevant functions and thus has to be replaced with a native version. In any case, loading the Wine Debugger when starting the application should help.

Cuvée

A scenario that Windows administrators refer to as *DLL Hell* clearly demonstrates that continuously replacing libraries is an approach with limitations – not all versions of the system libraries will cooperate nicely. When you reach this point, any action you take requires skill, research, luck, and – last but not least – patience.

To access debug information easily, pass in the `WINEDEBUG` environment variable at launch time. The value `+relay` in Wine debug mode outputs all function calls and descriptions of the associated libraries on the console. This gives the admin some initial information as to which function call is causing Wine to fail and what DLL is responsible for it.

If you redirect standard output into a logfile, you can then analyze the startup process in your own good time:

```
env WINEPREFIX=~/name_of_prefix WINEDEBUG=+relay wine ➥
   application.exe&>debug.log
```

```
File  Edit  View  Terminal  Help
0010:Ret  KERNEL32.CloseHandle() retval=00000001 ret=68235463
0010:Call KERNEL32.SetEvent(00000030) ret=6822c713
0014:Ret  KERNEL32.CloseHandle() retval=00000001 ret=68235463
0014:Call ntdll.RtlFreeHeap(00110000,00000000,00117db0) ret=68222276
0014:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=68222276
0014:Call ntdll.RtlFreeHeap(00110000,00000000,00000000) ret=68222276
0014:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=68222276
0014:Call ntdll.RtlFreeHeap(00110000,00000000,00000000) ret=68235a01
0014:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=68235a01
0014:Call ntdll.RtlFreeHeap(00110000,00000000,00118138) ret=681cfd67
0014:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=681cfd67
0014:Call ntdll.RtlDeleteResource(001180b4) ret=6821e9f8
0010:Ret  KERNEL32.SetEvent() retval=00000001 ret=6822c713
000f:Ret  KERNEL32.WaitForSingleObject() retval=00000000 ret=6822c0aa
000f:Call KERNEL32.ReleaseMutex(0000002c) ret=6822c0b8
0014:Ret  ntdll.RtlDeleteResource() retval=00000000 ret=6821e9f8
0014:Call ntdll.RtlFreeHeap(00110000,00000000,00118090) ret=6821ea18
0014:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=6821ea18
0014:Call ntdll.RtlFreeHeap(00110000,00000000,00118168) ret=681cfd67
0014:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=681cfd67
0014:Call ntdll.RtlDeleteResource(0011823c) ret=6821e9f8
000f:Ret  KERNEL32.ReleaseMutex() retval=00000001 ret=6822c0b8
000f:Call ntdll.RtlFreeHeap(00110000,00000000,00117b18) ret=68222276
000f:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=68222276
000f:Call ntdll.RtlDeleteCriticalSection(00117ae8) ret=6822cef6
000f:Ret  ntdll.RtlDeleteCriticalSection() retval=00000000 ret=6822cef6
000f:Call KERNEL32.CloseHandle(0000002c) ret=6822cf04
000f:Ret  KERNEL32.CloseHandle() retval=00000001 ret=6822cf04
000f:Call KERNEL32.CloseHandle(00000030) ret=6822cf12
000f:Ret  KERNEL32.CloseHandle() retval=00000001 ret=6822cf12
000f:Call ntdll.RtlFreeHeap(00110000,00000000,00117ad0) ret=6822cf43
000f:Ret  ntdll.RtlFreeHeap() retval=00000001 ret=6822cf43
000f:Ret  PE_DLL (proc=0x6823ac10,module=0x681f0000 L"rpcrt4.dll",reason=PROCESS_DETACH,res=0x1) retval
```

Figure 7: Launching a Wine application with the `WINEDEBUG=+relay` environment variable set generates a huge volume of output.

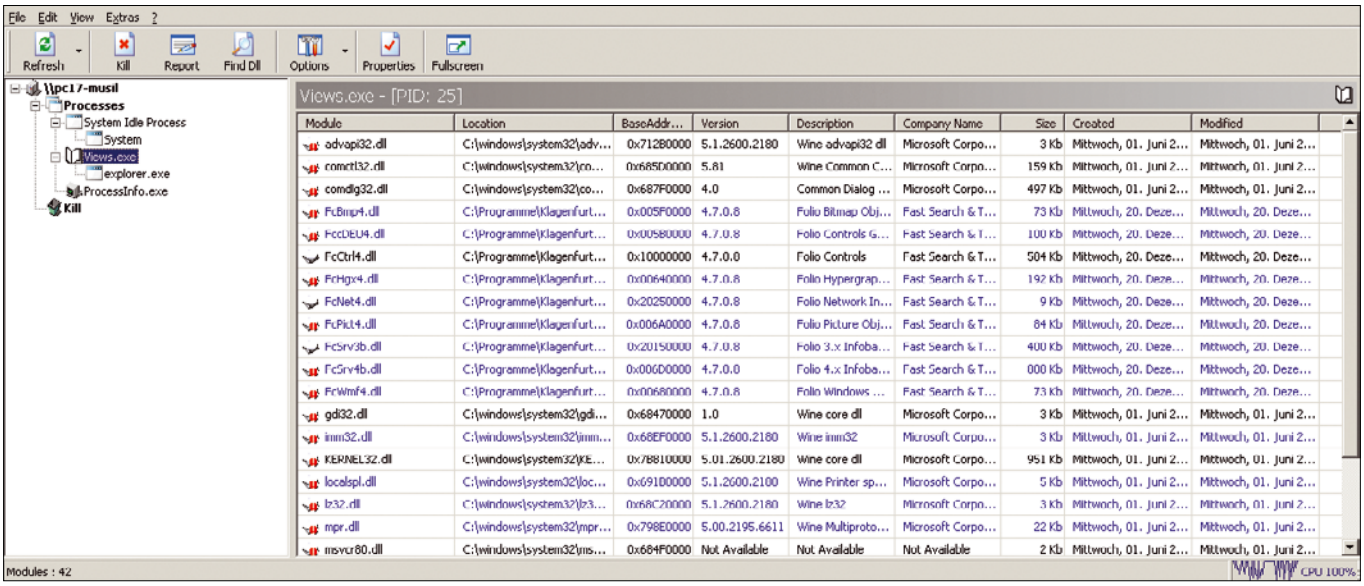

Figure 6: Process Info lists the DLLs linked to a Windows executable.

The `+all` parameter for the environment variable `WINEDEBUG` (Figure 7) makes the debugger far more talkative:

```
env WINEPREFIX=~/name_of_prefix ↵
  WINEDEBUG=+relay wine application.exe&>↵
  start_application.log
```

Wine then routes all the information accessible in debug mode to standard output. This significantly delays the application start, but once you gain a little practice, this approach can give you a far more in-depth impression of the implemented mechanisms and the pitfalls, which you can then try to circumvent.

In addition to setting the `WINEDEBUG` environmental variable, you can also directly launch an application with the built-in Wine debugger:

```
env WINEPREFIX=~/name_of_prefix ↵
  winedbg application.exe
```

```
File  Edit  View  Terminal  Help
The commands currently are:
  help                                  quit
  break [*<addr>]                       watch *<addr>
  delete break bpnum                    disable bpnum
  enable bpnum                          condition <bpnum> [<expr>]
  finish                                cont [N]
  step [N]                              next [N]
  stepi [N]                             nexti [N]
  x <addr>                              print <expr>
  display <expr>                        undisplay <disnum>
  local display <expr>                  delete display <disnum>
  enable display <disnum>               disable display <disnum>
  bt [<tid>|all]                        frame <n>
  up                                    down
  list <lines>                          disassemble [<addr>][,<addr>]
  show dir                              dir <path>
  set <reg> = <expr>                    set *<addr> = <expr>
  pass                                  whatis
  info (see 'help info' for options)
The 'x' command accepts repeat counts and formats (including 'i') in the
same way that gdb does.

The following are examples of legal expressions:
 $eax     $eax+0x3   0x1000   ($eip + 256)  *$eax   *($esp + 3)
 Also, a nm format symbol table can be read from a file using the
 symbolfile command.

Wine-dbg>
```

Figure 8: Wine Debugger supported commands.

The Wine debugger gives experienced testers the familiar debugging capabilities from within the Wine system library environment. Admins thus have an extensive range of methods for launching applications in a controlled manner.

The similarities to other debuggers are not coincidental; the commands implemented by the Wine debugger (Figure 8) are a subset of the GNU Project debugger (GDB). The developer website on Wine debugging [10] has a good overview of a systematic approach to an unknown problem that occurs when trying to run a Windows application in a Wine environment.

CodeWeavers – The Master Vintners

If all this debugging and testing has not helped you get an application on Linux and Wine running, you still have a couple of other options. With the motto "Little guys trying to change the computing world for other little guys," CodeWeavers [14] drives the integration of Windows applications in Wine. Although the products CrossOver and CrossOver Games are commercial and closed source, they walk many users – and especially gamers – through the process of talking Microsoft applications into running on Linux. See the article on CrossOver that appears elsewhere in this issue.

Also useful, with the added benefit of being free and open source, is the PlayOnLinux project (POL) [15], which is included with most distributions and comes with its own graphical front end for Wine. The long list of supported software comprises games and programs such as AutoCAD, CATIA, Microsoft Office, iTunes, Google's SketchUp, and Adobe's Photoshop and Dreamweaver. If the application in which you are interested is not listed in the PlayOnLinux GUI, searching for a `.POL` file in the online repository could help. Incidentally, PlayOnLinux also uses Wine prefixes, which can prevent unwanted interaction with other games and programs.

Windows to New Worlds

Combinations such as Wine, Linux, and VirtualBox allow for a complete and platform-independent open-source solution that lets you lock non-Linux applications into a container for permanent keeping. Templates containing everything necessary for operations can easily be launched, moved, or simply installed by copying. And if this isn't enough, you can reroute the output from the virtual instances using protocols such as VNC, RDP, or NX, thus giving ancient Windows applications multiuser and networking capabilities. ■■■

INFO

[1] Wine headquarters: *http://wiki.winehq.org*

[2] Description of Wineboot: *http://wiki.winehq.org/wineboot*

[3] Wine DIB engine: *http://wiki.winehq.org/DIBEngine*

[4] Description of Regedit: *http://wiki.winehq.org/regedit*

[5] Microsoft's description of side-by-side assemblies (SxS): *http://msdn.microsoft.com/en-us/library/aa376307.aspx*

[6] Winetricks: *http://www.winetricks.org*

[7] Details on Winetricks from the Wine wiki: *http://wiki.winehq.org/winetricks*

[8] Command-line options in Wine: *http://wiki.winehq.org/FAQ#head-3b297df7a5411abe2b8d37fead01a2b8edc21619*

[9] CodeWeavers Crossover Compatibility Center (CCCC): *http://www.codeweavers.com/compatibility*

[10] Wine debugger how-to: *http://www.winehq.org/docs/winedev-guide/wine-debugger*

[11] TrackWinstall: *ftp://ftp.heise.de/pub/ct/ctsi/trackwinstall_111.zip*

[12] DLL Show: *http://www.gregorybraun.com/DLLShow.html*

[13] Process Info: *http://www.jobe-software.de/en/index_en.htm*

[14] CodeWeavers: *http://www.codeweavers.com*

[15] PlayOnLinux: *http://www.playonlinux.com*

[16] Jele, H. *Robert Musil und das Matroschka-Prinzip*: *http://wwwu.uni-klu.ac.at/hjele/publikationen/container_ka_2013/20111114_matroschka_rmi.pdf* (in German)