LINUX VOICE
TUTORIAL

LINUX 101: RUN WINDOWS APPLICATIONS WITH WINE

MIKE SAUNDERS

If you still depend on a few Windows programs, or you want to help newbies make the switch to Linux, Wine is mightily useful.

WHY DO THIS?

- Run legacy Windows apps without rebooting.
- Create multiple configurations for better compatibility.
- Help Windows users move over to Free Software.

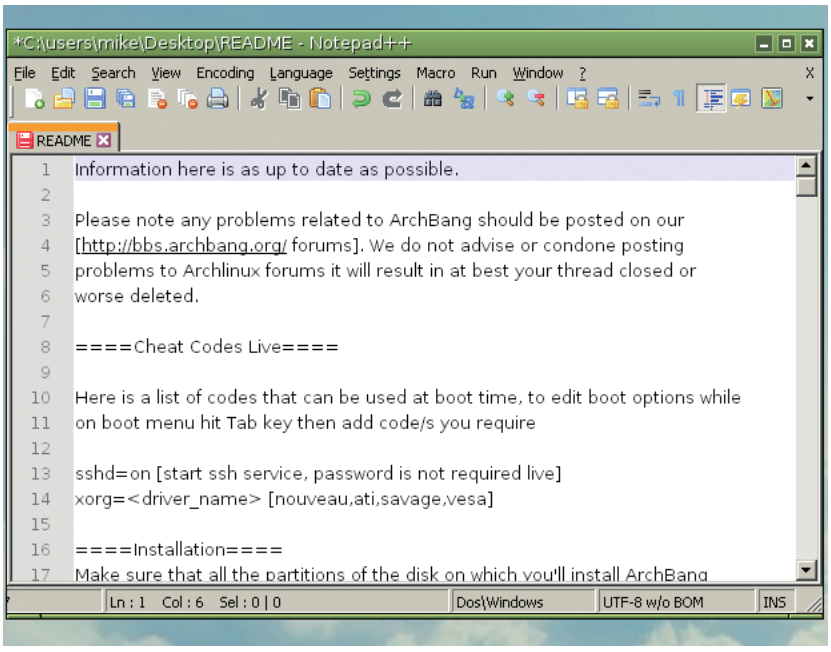
Question: what do you call a program that runs software designed for a different platform? An emulator, right? Well, the name *Wine* comes from "Wine Is Not an Emulator" – which is one of those recursive acronyms that are so loved in the FOSS world. But given that *Wine* lets you run Windows software on your Linux installation, why is it not an emulator? Essentially, *Wine* acts as a compatibility layer that translates Windows system calls to their Linux equivalents, and it doesn't actually emulate a complete Windows PC, with its CPU, graphics card and so forth.

Anyway, with that naming confusion out of the way, let's focus on the software itself. *Wine* is a godsend for many Linux users who've made the transition from Windows, but still need to run the occasional program from the latter operating system. It's completely free software – you don't need a licence from Microsoft to use it – and it's capable of running a wide range of programs. Not all of them, mind you, and very recent software can have problems. But some major applications like *Microsoft Office 2010* work well enough for daily use.

So if you're still dual-booting between Linux and Windows, and would rather spend more time in the former, here we'll show you how to use *Wine* and (hopefully) run your favourite Windows apps without rebooting. Or if you're a full-time Linux user and don't give a hoot about Windows, you can still use this guide when you're helping others make the transition to Linux, set up *Wine* for them and demonstrate the awesome power of free software.

1 GETTING STARTED

Wine is included in almost every major distro's repositories, so find it in your package manager or use your usual command-line tools to install it (eg **sudo apt-get install wine** on Ubuntu-based distros). We're using Arch Linux for this tutorial – but the commands are the same across other distros. If you want the latest and greatest version and are happy compiling software from its source code, you can get it from **www.winehq.org**. After you've got it installed, find a simple, standalone Windows program to test; in our case we're going to use the rather cool *Notepad++* text editor available from **http://notepad-plus-plus.org**. This program exists as a single .exe file and doesn't have a ton of complicated dependencies, so it's the perfect type of program to kick tires of a new *Wine* installation.



Here's our first app runing on *Wine* – *Notepad++*. It's a simple program and therefore has few compatibility issues with *Wine*.

Go to the Downloads section, then grab the "minimalist package" and save it to your home directory. This is in 7-zip format, so install the tool to extract that in your distro's package manager (it should be provided in the package **p7zip**). Then open a terminal and enter:

```
7z x -onpp npp.6.6.9.bin.minimalist.7z
cd npp
```

Here we're extracting the download into a new **npp** directory – if there's a newer version of *Notepad++* by the time you read this, change the version number accordingly. We then switch into the directory, and if you enter **ls**, you'll see that there's a file there called **notepad++.exe**. Let's run it!

```
wine notepad++.exe
```

You may be prompted to install Mono and Gecko packages; these aren't important for now, so just click Cancel in the dialog boxes that appear. And after a few moments, you'll see *Notepad++*, a Windows program, in all its glory on your Linux desktop. Not bad – it's as simple as that!

CrossOver: the commercial alternative

If you're looking for improved compatibility, easier installation and technical support, *CrossOver* (**www.codeweavers.com**) is worth a look. This is a commercial version of *Wine* with various extras, and is especially useful if you want to run Microsoft Office (XP to 2010), *Adobe Photoshop* and several triple-A games like *World of Warcraft*. *CrossOver* includes a tool called *CrossTie*, which lets you install applications straight from the web with just a couple of clicks, and it also uses a bottles system to stop different configurations from overwriting one another. It also has some tweaks to integrate more smoothly with KDE and Gnome.

Pricing starts from €32; this gets you one month of email support and upgrades (when new versions are released). For €48 you get the whole package, which includes one year of email support and upgrades, along with a "phone support incident" – ie you can speak to one of the devs on the phone if you're having serious trouble. (Subsequent calls cost €16.95 each.) *CrossOver* employs *Wine* developers and contributes code back to the main tree, so if you find *Wine* really useful and want to support its development financially, buying *CrossOver* is a good idea. Alternatively, you can donate directly at **www.winehq.org/donate**.

Well, for small programs it's simple; we'll get to the more complicated setups later. For now, try exploring the program. As mentioned, *Wine* translates every Windows system and library call that the program makes into a Linux equivalent. So if you save a file from *Notepad++*, the program calls Windows' file saving routine, *Wine* intercepts it, and forwards on the request to the Linux equivalent. When *Notepad++* wants to draw something on the screen, it makes requests to Windows libraries – and *Wine* has its own versions of these, which then talk to the X server on Linux. It's very cool technology.

Two worlds collide

Of course, *Wine* can't magically make some of the differences between Linux and Windows disappear. Go to File > Open in *Notepad++*, for instance, and select My Computer from the "Look in" list. You'll see C:, D: and Z: drives – but they make no sense in the Linux world. Well, *Wine* maps them to different locations in your filesystem. The Z: drive is mapped to the root directory (**/**), which is the base of everything in a Linux/Unix system. So you can use that drive to go to your home directory in **/home** and access your personal files – or use the My Documents shortcut.

But where does C: point to? If you click into it, you'll see some familiar folders from a Windows installation: Program files, windows and so forth. These were created when you first ran *Wine*, so let's explore them in more depth. They're located in **.wine/drive_c** in your home directory, so close *Notepad++* and switch into that directory like so:

```
cd ~/.wine/drive_c
```

Enter **ls** and you'll see those folders again. Switch into the windows directory with **cd windows** and run ls again – this time, you'll notice some common tools like *Regedit*. Basically, *Wine* has created a very minimal Windows installation in your home directory, comprised of fully open source software, of course. So you can run the included tools like so:

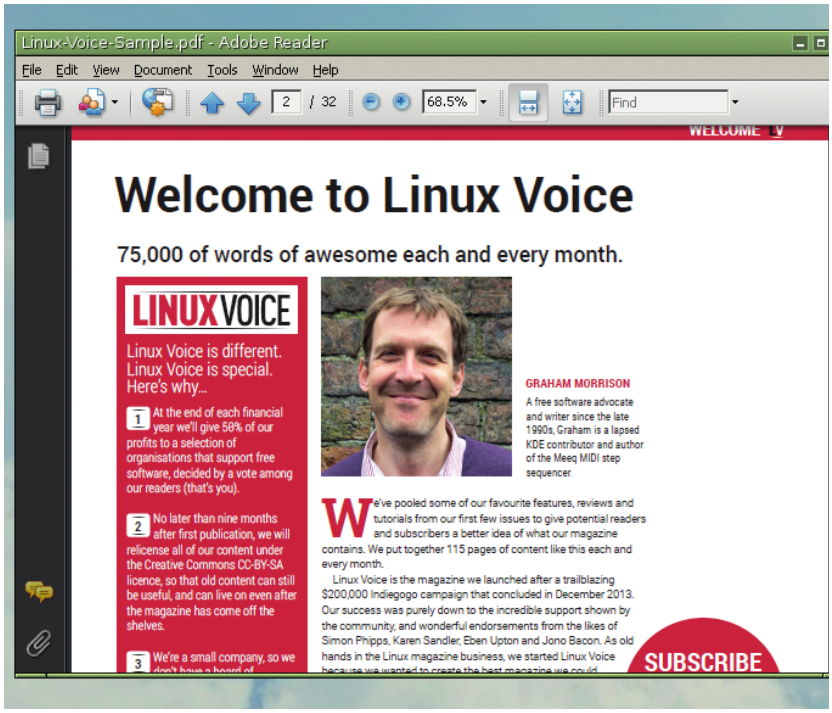
```
wine regedit
```

(Note that you can omit the **.exe**.) This looks just like the real Windows registry editor, but go to Help > About and you'll see that it's a tool written by the *Wine* developers. Back in the terminal, if you head into the **windows/system32** directory (and **syswow64** on 64-bit installations), you'll see a bunch of DLLs. These are *Wine*'s own implementations of core Windows libraries – and again, they're fully open source.

Now, they're not always as feature-complete as the original Windows versions, so in some cases you can copy DLLs from a real Windows installation into this directory, to improve compatibility with certain programs. The only ones you must never overwrite are **kernel32.dll**, **gdi32.dll**, **user32.dll**, and **ntdll.dll** – you can only use the *Wine* versions of these.

As an aside, the ReactOS project (**www.reactos.org**), which aims to create an open source Windows-compatible operating system, uses many *Wine* DLLs. The underlying structure of ReactOS is very different to Linux and Unix, as it aims to be compatible with Windows drives as well as software, but there's a decent amount of code-flow between ReactOS and *Wine*. We've been following ReactOS for many years and it's making slow but steady progress – what Microsoft's lawyers think about it, though, remains to be seen...

> **"Wine translates every system call that an application makes into a Linux equivalent."**



Adobe has dropped support for *Reader* on Linux, but thanks to *Wine*, you can get some Windows versions running.

2 INSTALLING SOFTWARE AND CUSTOM SETUPS

So far, we've just tested a simple standalone **.exe** program. But what if you want to install something more complicated, like a program that extracts and installs its own files? Let's try *Adobe Reader*, given that it's no longer supported on Linux. The first thing to do before installing any program is to check its compatibility ratings, so in this case we go to **https://appdb.winehq.org** and enter "Adobe Reader" in the search box in the top-right. When the results come up, we click on the "WineHQ – Adobe Reader" link.

Here we can see that different versions of the program have different ratings: gold means that a program works almost flawlessly, whereas silver and bronze mean that the program is usable, albeit with some glitches or other issues. (These compatibility ratings are provided by the community, and some programs have only been tested with older *Wine* releases, so it's possible that compatibility has improved in the meantime.)

Adobe Reader 9.x is rated as silver, so click it and then, on the following page, the first "Free Download" link on the left. Save the **AdbeRdr90_en_US.exe** file to your home directory, fire up a terminal, and enter:

```
wine AdbeRdr90_en_US.exe
```

This isn't the program itself, but rather its installer – so follow the prompts, and don't worry if the installer gets confused and crashes right at the end. (This is a common occurrence in *Wine*, when installers don't quite understand that they're not running in an original Windows environment, but usually it's not a problem.)

Now, like in regular Windows, *Adobe Reader* has been installed in a **Program Files** directory. In this case, you'll find it in **~/.wine/drive_c/Program Files (x86)/Adobe/Reader 9.0/Reader**. If you **cd** into that directory and enter **ls**, you'll see **AcroRd32.exe** – that's the program you want to run with *Wine*. So give it a go, and try opening some PDFs – by and large, it does its job without major bugs. You can now create a launcher on your desktop that runs the program with this command (note the use of quote marks to get round spaces in directory names):

```
wine ~/.wine/drive_c/"Program Files (x86)"/Adobe/"Reader 9.0"/Reader/AcroRd32.exe
```

Another useful launcher to create is "wine explorer", which starts up a file manager, so you can then browse into Program Files (x86) yourself and launch programs by double-clicking on them.

As you'd expect, *Wine* is a hugely configurable piece of software, but fortunately there's a fairly good GUI tool that lets you tweak settings without fiddling around inside config files. Enter **winecfg** and a small Windows utility pops up with various tabs. The most important of these is Applications: here you can select **.exe** files in your *Wine* installation, and then choose the Windows version that *Wine* should emulate for them. So if you know that a certain program works best in Windows XP, or Windows 7, you can select it at the bottom. In general, *Wine*'s compatibility is more complete when it comes to older Windows versions.

Another useful tab here is Libraries. Here you can choose whether *Wine* should override its inbuilt libraries with native ones, as mentioned earlier. Under the Graphics tab you can also customise the screen resolution, which helps if certain programs are being displayed incorrectly.

Bottle it!

Things can get complicated when you've customised your *Wine* installation for one particular program, and it's running beautifully, but then you install another program that needs different settings, library overrides, and so forth. It's a colossal pain to keep switching options manually, but thankfully, *Wine* has a solution for this called prefixes (aka *Wine* bottles). This lets you create and maintain separate *Wine* installations for your programs – albeit with a bit of extra disk space usage.

LV PRO TIP

It's important to note that Wine programs can access your Linux system like any other app. They're not sandboxed or restricted. Of course, the inbuilt security mechanisms of Unix and Linux should stop a malicious app from completely hosing your system, but if you want to be ultra secure, use Wine on Linux inside a virtual machine!

LV PRO TIP

Got a program that's supplied as an **.msi** file? You can install these using the **msiexec** tool along with the **/i** flag – for instance, **msiexec /i filename.msi**. This tool is provided as part of a standard *Wine* installation.

Running DOS software with DOSBox

If you've got some really old programs that you'd like to get running again, from the days when MS-DOS ruled the (business) world, you're also in luck. FOSS platforms have had great DOS compatibility for many years thanks to *DOSEMU*, but that program hasn't seen many updates recently, and can be fiddly to set up. A better alternative can be found in *DOSBox* – and it's available in all major distro's repositories.

Once you have it installed, you just need to point it at a directory containing your DOS program(s). In this example we've got *Frontier Elite II* in a directory called **frontier**, so we just run:

```
dosbox frontier
```

When *DOSBox* starts, it maps its emulated C: drive to the directory you specified. So if you enter DIR now, you'll see the files inside the **frontier** directory – including **frontier.bat**, which you can use to run it. *DOSBox* will often grab the mouse cursor for itself, so to free it, press Ctrl+F10 at the same time.

To configure *DOSBox*, run it on its own (without a directory) and then enter **config -wc dosbox.conf** in the emulated DOS session. Type **exit** to leave, and you'll see **dosbox.conf** in your current directory. You can now edit this to tweak settings, such as full-screen mode, mouse sensitivity, and how quickly it should run (look at the cycles option). If you need any help, you'll find plenty of it on the wiki at **www.dosbox.com/wiki**.

Elite: Dangerous should be out by the time you read this, but we'll never stop loving *Frontier*.

The key to this is the **WINEPREFIX** environment variable. Say you want to install program **fooapp.exe** into a new *Wine* installation, and not **.wine** in your home directory. You would run this command:

```
env WINEPREFIX=~/.wine_fooapp wine fooapp.exe
```

A new *Wine* installation, with fresh settings and libraries, will be created in **.wine_fooapp** in your home directory. Once the app is installed, you can run its executable as usual, but make sure to keep the **env WINEPREFIX=~/.wine_fooapp** part otherwise it all gets messy. Essentially, you can make as many *Wine* prefixes as you like (at the cost of 35MB each time), but always make sure you're pointing **WINEPREFIX** to the appropriate place, otherwise one installation can overwrite settings from another.

To run **winecfg** for a particular prefix, you also need to specify the environment variable:

```
env WINEPREFIX=~/.wine_fooapp winecfg
```

If you want to completely remove a program and its prefix, just remove the directory.

Another useful environment variable is **WINEARCH**. If you're running a 64-bit distro, *Wine* will start in 64-bit mode by default; if this leads to problems with your programs, you can change this by using **env WINEARCH=win32** before your commands.

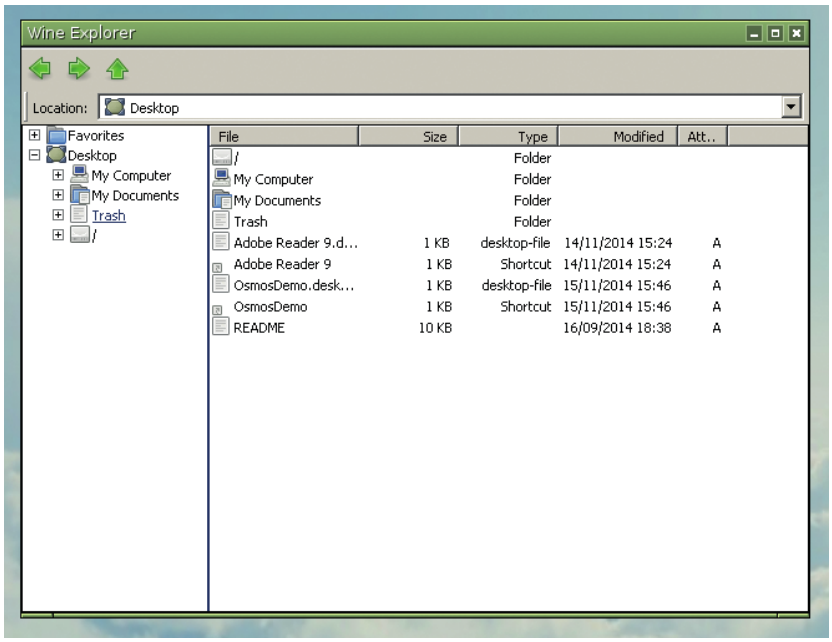
Tricks up the sleeve

Finally, we want to give a mention to *Winetricks* (**www.winetricks.org**), a very handy little script that assists you in installing various programs and games. Many distros include it in their package repositories – if you can't find it, just grab it from the website (the Installing page explains how to do it step-by-step). You'll also need some kind of utility for displaying dialog boxes, such as *Zenity* or *kdialog* from KDE.

When you run *Winetricks*, you'll be prompted to install an application, benchmark or game. Try installing an app: you'll see that many of them can be downloaded automatically (check the Media column), but in some cases, such as with Microsoft Office, you'll need the original CD or DVD. As a test, try installing the *AbiWord* word processor: *Winetricks* will download the setup.exe file and run it in *Wine*.

After the installation, *Winetricks* will return to its original menu, but you'll see a new item: "Select AbiWord". Click on this and then OK, and another menu will appear so that you can configure the installation. You can access the usual **winecfg** tool in this way, or also fine-tune options via the Change settings item. Note the titlebar here – *Winetricks* has installed *AbiWord* into its own prefix, in **~/.local/share/wineprefixes/abiword/**. So if you **cd** into that directory in a terminal, you'll see the usual **drive_c/Program Files (x86)** subdirectory underneath it, and then *AbiWord* under that. (The launcher, **AbiWord.exe**, is inside the **bin** directory.)

Winetricks also provides access to a large list of games and demos, many of which are great for testing the performance of *Wine*. Just remember that they can swallow up your disk space quickly, so it's a good idea to remove the prefixes when you're done with them!



Wine Explorer is a simple file manager that you can use to browse files and launch programs.

The future of Wine

Wine's development dates back to the mid-90s, so it's one of the longest-running projects in the Free Software world. After two decades of development, though, why are there still compatibility problems with so many programs? Well, part of the problem is that Windows is a moving target. When *Wine* started, its goal was to provide compatibility with Win32 – in other words, the APIs used on Windows 95, 98 and NT. But since then, we've seen many more releases of Windows, and *Wine* developers keep trying to chase the latest APIs.

Some would argue that the *Wine* team should set a very specific goal: compatibility with Windows XP, for instance, and forget about Vista, 7 and 8. This could make sense in positioning *Wine* as a solution for legacy applications, but other users and developers want to use *Wine* to play the latest games and run recent versions of *Office*. As most *Wine* developers are hacking on the code out of a labour of love, nobody can force them to limit the compatibility to a specific Windows release.

And then, trying to be compatible with Windows APIs is an adventure in itself. Many APIs are undocumented or don't behave as expected, so it's not just about following a spec like POSIX. After all, it's in Microsoft's interests that a compatible OS from a third party isn't developed. Sure, the Redmond giant has been more friendly with the FOSS community recently, but we don't expect it to suddenly get behind the *Wine* project or open up reams of specifications to help its development.

Mike is a recursive acronym and stands for "Mike ist kein Emulator". Blame his parents.

LV PRO TIP

Spaces in file and directory names are common in the Windows world, but they're a royal pain in the rear on the Linux command line. You can use escape characters (backslashes) to get around them if you're a long-time Linuxer, but for new users it's best to just use quotes. So if you need to **cd** into the **Program Files (x86)** directory, enter **cd "Program Files (x86)"**.