ReactOS: Building a Free-Licensed Windows

Author: BYFIELD Bruce blog **Geodata:** Burnaby coding / hacking / free software / open-source

Image remixed by Horst JENS sources/rights: Succo, clker, openicons at Pixabay cc 0/public domain. Reactor OS Logo from Wikimedia.org: GPL

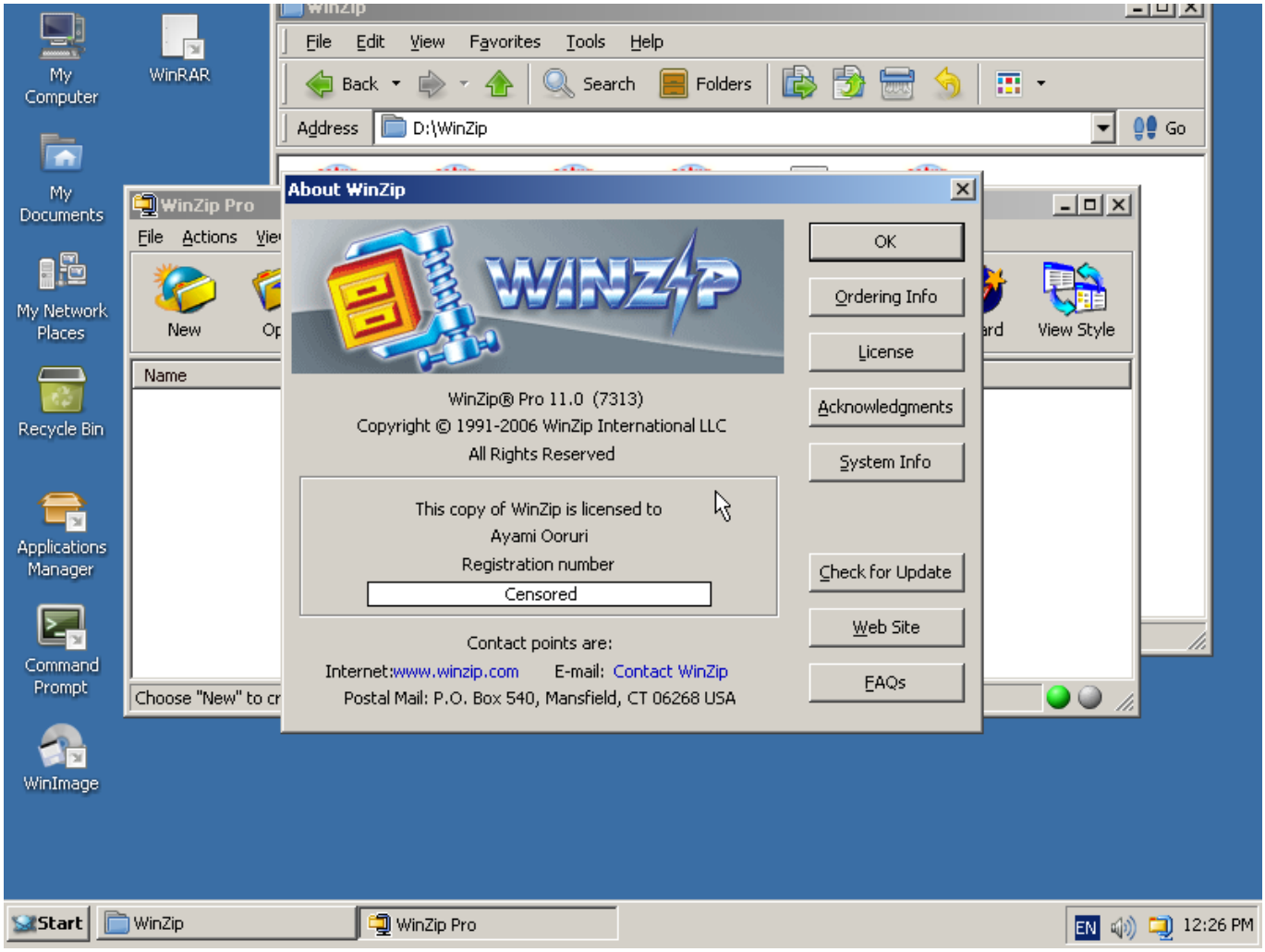
From dual-booting to **WINE**, free software has always struggled to provide a solution for running Windows applications. However, few of these efforts have been more ambitious than ReactOS, a free-licensed implementation of Windows. The project has been at work since 2006 and, in February 2016, ReactOS finally released its first alpha version, after a decade of difficult and necessarily cautious development.

According to developer Ziliang Guo, ReactOS grew from the failure of the FreeWin95 project, whose goal was a free-licensed implementation of Windows 95.

FreeWin95 got basically nowhere because the people involved got bogged down in technical discussions about how to implement the OS, with no one willing to actually do the actual coding.

As a result of this lack of progress, Jason Filby and David Welch created a new project to create a free version of Windows NT. Jeff Know, another project member, suggest the name ReactOS because the effort was a reaction to Microsoft's monopoly on the desktop.

Blast from the past: WinZip running on ReactOS



Reverse Engineering and Documentation

From the start, ReactOS had difficulties. When the project started, most Windows compilers were **proprietary**, and the few free-licensed ones, painfully lacking. Guo remembers project member Casper Hornstrup as the developer chiefly responsible for MinGW (Minimalist GNU for Windows), a major step in developing the necessary tools. Under these circumstances, even the ability to boot ReactOS was a major milestone in development. As Guo remarks,

Actually booting an operating system is a more involved task than most lay-people would think.

Another early problem was a lack of documentation for the internal architecture of Windows NT. For instance, lack of information on kernel level APIs made compatibility with NT drivers difficult. Similarly, internal interdependencies had to hacked, which led to and increased technical debt — that is, a gradual increase in problems that slowed progress.

When the team was able to move onto implementing missing functionality, there was often a need to go back into existing components and remove the hacks, which could end up in something of a rabbit's hole of hacks holding together other hacks, and the entire thing collapsing in unfortunate ways.

Guo explains.

Today, the situation is somewhat improved. *"There is plenty of available documentation about Microsoft Windows,"* says project coordinator Aleksey Bragin, including the MSDN site, as well as books such Windows Internals, Inside Microsoft Windows, and Windows Graphics Programming. This information can be used freely, but sometimes contains errors, and undocumented parts remain.

Another advantage has been WINE's compatibility layer for the Win 32API, which ReactOS could reuse. But *"Of course the team still needed to implement the Win32 subsystem proper that would power those APIs,"* says Guo.

Consequently, despite the occasional advantage, to move forward, ReactOS continues to rely on **reverse engineering** by third parties. At other times, project members resorted to blackboxing, systematically testing Windows' responses to different types of input in the hopes of understanding what was happening internally.

Unfortunately, reverse engineering is often a legal mine field.

There has always been an aswareness on the part of the project that Microsoft might view us as a strategic threat and seek to shut us down

says Guo. To avoid any potential problems, ReactOS has always been careful to make certain that all contributions were not derived from restricted Microsoft code. For example, code created through reverse engineering, and then converting Assembly into C code is not accepted, but blackboxing is.

In fact, according to Bragin, ReactOS prefers to avoid reverse engineering whenever possible. When reverse engineering is unavoidable, contributors are expected to apply 2.1 of the GNU Coding Standards, and avoid refering to any proprietary source code — especially Windows' — for or during work on React OS. Instead, all code must be as different possible from the original.

Even so, reverse engineering remains a gray area. Guo recalls that, at one point, project developers argued over differing interpretations of what constitutes untainted code. The dispute caused several developers to quit, and resulted in an internal audit of the project's code. Fortunately, no tainted code was found, and the project was eventually able to continue.

The Next Challenges

The release of alpha version 0.4 comes after several years of crowdsourcing with limited success. The new release is available as a Live CD, which installs easily in VirtualBox, where — naturally enough — it should be registered as Windows NT. The operating system boots in a matter of seconds, perhaps because it emulates the systems of twenty years ago on modern hardware. It runs a variety of software, including Winzip and free software such as LibreOffice, as well as a few older or simpler games.

ReactOS running its version of Solitaire.

However, challenges remain. As always, qualified developers are rare:

The number of people who are NT kernel experts who don't work for Microsoft already are few and ar between.

Guo notes. Supporting current hardware can be difficult as well, although Guo adds that variable hardware standards are a problem that he euphemistically describes as being *"a charlie foxtrot of epic proportions that not even Microsoft had a trouble-free experience with."*

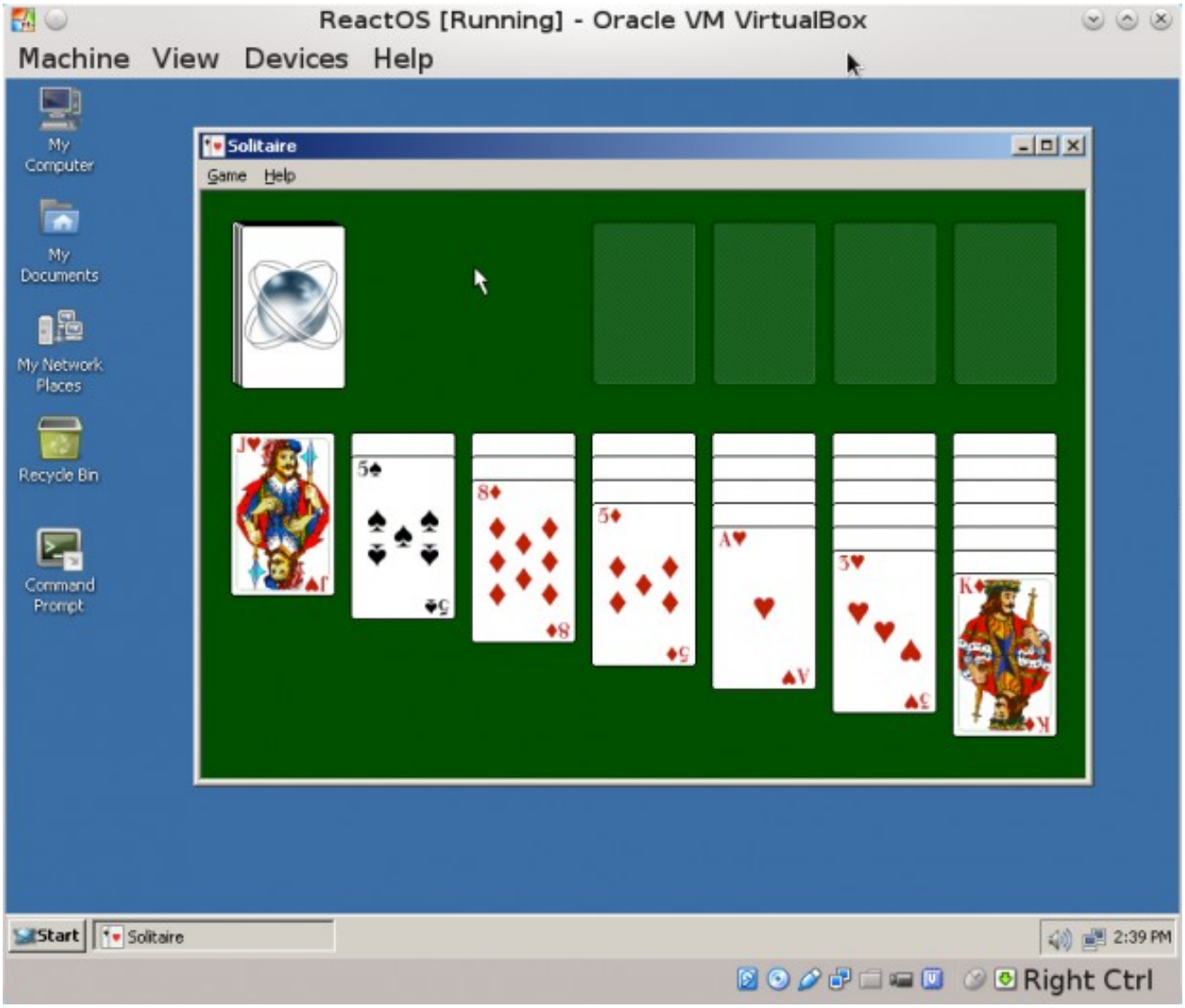
Another problem is that, after some extensive work on usability in recent releases, in Guo's words,

Most of the low hanging fruit has been plucked, with the consequence that major, user-visible changes are far fewer that in the past.

What remains are some fundamental problems. For example, asked about priorities, Bragin responds that

The most important gap is stability of the system, related to memory management and caching. There are other problems, of course, but having a really stable kernel would be a great success for the project.

Another pressing problem is support for DirectX for games. However, ReactOS hopes to piggyback on WINE's development of a DirectX wrapper over OpenGL. Bragin writes that game support remains



important because their use of memory, the file system and networking make them *"a good test for the OS itself."*

Still, the first alpha release remains a significant milestone. ReactOS is already advanced enough that Bragin uses it for teaching about operating systems.

But I really want to see a moment when ReactOS is able to substitute for Windows in some use case. That moment will definitely happen within the next five years

he predicts.

The Drive Towards General Release

As often happens with free software interacting with proprietary development, ReactOS would appear to be perpetually behind — reacting, if you pardon the pun — to Microsoft's constant changes to its API. However, that situation may be less gloomy than it first appears. For one thing, so long as the project continues its blackbox practices, Microsoft's apparently friendlier attitude towards free software these days might mean fewer legal dangers.

Even more importantly, ReactOS has no interest in reproducing every feature of each Windows release.

We try to produce good features, for example UI theming allows us to switch between various themes. However, not everything that Microsoft adds is good. For example, there is no catch up play for the Metro Interface, or for any other new APIs until they gain enough popularity. If someone really wants it, she could do a Windows 8 Metro Theme. However, I am not sure there are many who would love such an

interface. I just don't see that much added value in Windows 10 or Windows 8.

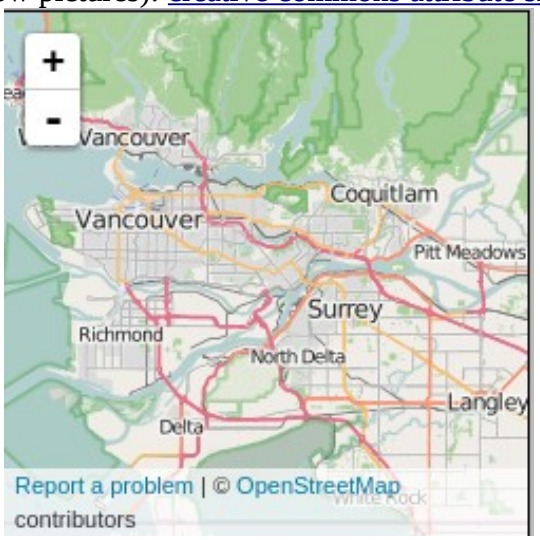
says Bragin.

Whatever happens, with its alpha release, ReactOS appears to be gaining momentum. Although its goal remains challenging, after a decade of work, its prospect for a general release seems to be in sight at last, if only distantly.

This article was first published on 2016-03-03 by Bruce BYFIELD on OCSMag, the Open Content & Software Magazine under CC-BY-SA 4.0 license.

Name: Bruce BYFIELD
Contact: @bbyfield, bbyfield@axion.net
Shorturl: http://goo.gl/XbRb9E
Hashtag: `#reactos`
Fork / improve: on Github
Extras:: Author's Blog, original article

Location:



about the author: I am a freelance writer and editor. Mainly, I specialize in free and open source software (FOSS). Software reviews, community news, business news, legal events -- if it's about FOSS, I write about it. I also write about management and careers, and, in my spare time, about Northwest Coast Art.
To date, I have published over 1200 articles, mostly online on such sites as Datamation, IT Manager's Journal, Linux.com, Linux Developer Network, Linux Journal, LinuxPlanet, LWN, NewsForge, Techwr-l, and Wazi. Other places I have published include The New Internationalist, Linux Pro Magazine and The Linux Journal. My article "11 Tips for Moving to OpenOffice.org" was the cover story for the March 2004 Linux Journal.
You can see some comments about my writing on my Kudos and Abuse pages. **Donate author**: No donation address given yet.

---

technical terms explained by Wikipedia:

WINE: (recursive acronym for *Wine Is Not an Emulator*) is a free and open source compatibility layer software application that aims to allow applications designed for Microsoft Windows to run on Unix-like

operating systems. Wine also provides a software library, known as Winelib, against which developers can compile Windows applications to help port them to Unix-like systems.
It duplicates functions of Windows by providing alternative implementations of the DLLs that Windows programs call, and a process to substitute for the Windows NT kernel. This method of duplication differs from other methods that might also be considered emulation, where Windows programs run in a virtual machine. Wine is predominantly written using black-box testing reverse-engineering, to avoid copyright issues,
The name Wine initially was an abbreviation for Windows emulator. Its meaning later shifted to the recursive acronym, Wine is not an emulator in order to differentiate the software from CPU emulators. While the name sometimes appears in the forms WINE and wine, the project developers have agreed to standardize on the form Wine. Read full wikipedia article...

Proprietary Software, non-free software (in the sense of missing freedoms), or closed-source software is software that fails to meet the criteria for free or open-source software. Although definitions vary in scope, any software which places restrictions on use, analysis, modification, or distribution (unchanged or modified) can be termed proprietary.
A related, but distinct categorization in the software industry is commercial software, which refers to software produced for sale, but without necessarily meaning it is closed-source.Read the full Wikipedia article...

Reverse engineering, also called **back engineering**, is the processes of extracting knowledge or design information from anything man-made and re-producing it or reproducing anything based on the extracted information. The process often involves disassembling something (a mechanical device, electronic component, computer program, or biological, chemical, or organic matter) and analyzing its components and workings in detail.

The reasons and goals for obtaining such information vary widely from everyday or socially beneficial actions, to criminal actions, depending upon the situation. Often no intellectual property rights are breached, such as when a person or business cannot recollect how something was done, or what something does, and needs to reverse engineer it to work it out for themselves. Reverse engineering is also beneficial in crime prevention, where suspected malware is reverse engineered to understand what it does, and how to detect and remove it, and to allow computers and devices to work together ("interoperate") and to allow saved files on obsolete systems to be used in newer systems. By contrast, reverse engineering can also be used to "crack" software and media to remove their copy protection, or to create a (possibly improved) copy or even a knockoff; this is usually the goal of a competitor. Read the full Wikipedia article...