An Overview on the Future of Windows Compatibility

THE REACTOS PROJECT

Speaker Info

- Alex Ionescu
- Lead Kernel Developer for ReactOS Project. Have been working on the project for almost 3 years.
- Student in Montreal, Quebec, Canada.

Outline

- About the Project
  - Description
  - Motivation and Goals
  - Current Status
- ReactOS Architecture
  - Kernel
  - Native + Subsystems
  - User (Win32)
- The Importance of ReactOS on FOSS
  - For Users
  - For Businesses
  - For Developers
- Roadmap for 2007
- Call for Action

Implementation, Motivation, Goals, and Current Status

About ReactOS

Description

- ReactOS is an operating system written from scratch.
- It is an NT-based kernel and closely follows NT architecture.
- NT is a 32-bit Windows-family OS written in the early 90ies by Microsoft and constantly updated by new releases. Windows 2000, XP, 2003, Vista are different versions of NT.
- ReactOS targets Windows XP/2003 (NT 5.1/5.2).
- ReactOS has been in development for 10 years, some code is based on NT 4 architecture, while some APIs support extensions added by NT 6 (Vista).
- ReactOS includes the kernel, Win32 libraries, system libraries and drivers, base applications, system components, subsystem support and window manager.
- ReactOS excludes anything not part of an NT installation.

License and Shared Code

- ReactOS is GPL 2.0 licensed, but it includes 3rd-party code under its respective compatible licenses.
- 3rd party code includes:
  - Wine makes up the bulk of ReactOS's Win32 Libraries, which are mostly left untouched.
  - Freetype provides font rendering support for the window manager.
  - libxml, libpng, bzlib, adns provide support for specialized Win32 libraries and applications.
  - MESA provides software OpenGL rendering.
- In return, ReactOS code has been used by:
  - Captive, for NTFS write support in Linux.
  - Some patches went upstream into Wine.
  - NDISWrapper.
  - LinuxBIOS support for booting NT.

More on Cooperation

- Due to the large size of our codebase and Win32 requirements, several patches and improvements have been made by our developers:
  - MinGW DDK Headers.
  - Patches to MinGW GCC and Binutils.
  - Patches to QEmu and KQEmu.
- As we begin supporting more Win32 specialized APIs, more libraries will probably by imported, especially for security APIs and GDI+ support.
- However, we will never include whole applications or APIs not present in Windows.

Motivation

- NT 5.2 provides a rich and extensible architecture with a highly scalable and optimized set of components.
- A secure and reliable OS, written for C2 security level certification, and updated to B1 for Vista.
- Also a great learning opportunity and research material for students and academia.
- Runs 95%+ of all computer software and drivers, most pervasive consumer OS on the planet.
- But…

Motivation

- Plagued by bad design decisions made early-on in 16-bit Windows 9x history but kept for compatibility.
- Plagued by a myriad of hacks to support badly written applications and drivers from 3rd party developers.
- Plagued by bad design decisions still being made to maintain corporate agenda (DRM, Driver Signing, etc).
- Plagued by bugs in bundled software (Internet Explorer/Windows Media Player/Outlook Express) and bad security decisions (users run as Administrators, etc) which undermined architectural security and reliabilty.
- Closed source, costly, poorly documented in regards to system architecture and undocumented functionality (compared to competing FOSS operating systems).
- Most extensibility features kept undocumented and not open to 3rd party modification.

Goals

- ReactOS aims to offer all of the features and performance of NT without all the hacks, restrictive design decisions and license restrictions.
- It aims to offer no-cost Windows compatibility at a level no other solution can.
- It aims to document the undocumented, and to provide binary-compatible components which would be used to provide extensibility.
- Great teaching platform for academia. UNIX/Linux are good to learn from, but NT does some interesting and different things that deserve the same attention.
- Will not include applications such as IE, OE or WMP. Users will be encouraged to install FireFox, ThunderBird, OpenOffice, Mplayer, etc.

Current Status

- Large parts of the kernel are now fully compatible with Windows 2003 SP1: Executive, Kernel Core (Scheduling, Dispatching, Interrupts, etc), HAL, Local Procedure Call, Process and Thread Management, and most of I/O support (except PnP).
- Other parts are totally foreign compared to NT design, notable the Cache Controller, Configuration Manager (Registry backend) and Memory Manager.
- Win32 application support largely depends on two components:
  - Win32k – Kernel-mode GUI Server, analogous to X.
  - Win32 libraries (gdi32, user32, kernel32, advapi32) – Taken from Wine.
- Some Win32 functionality depends on kernel behavior.

Current Status

- User-mode applications are currently supported in a limited fashion – specific apps targeted: Firefox, Thunderbird, OpenOffice, etc.
- 90% of application compatibility problems are due to Win32K or subtle kernel bugs.
- Theoretically, ReactOS should run at least what Wine runs.
- Drivers have not been fully tested, but several problems exist:
  - Some drivers use hacks that depend on offsets, memory locations and variables specific to NT.
  - Other drivers depend on subtle PnP implementation differences (synchronous vs asynchronous behavior).
  - File-system support is bad due to deep Memory Manager/Cache Controller differences.
- On the other hand, the Windows 2003 USB Stack works nearly perfectly out of the box, as does, for example, the Named Pipe File System driver.
- NVidia video drivers hack deep into kernel variables, so they need binary patching.

Kernel, Native and User Mode – Subsystem Paradigm

ReactOS Architecture

NT Kernel Mode Design

- Kernel-mode NT is implemented by one large module (ntoskrnl), a hardware abstraction layer (HAL), and a set of loadable kernel modules (drivers and other kernel libraries).
- The kernel is written in portable C, and ports exist (or once existed) for MIPS, Alpha, AXP64, Sparc, i810, IA64, x86, x86-64, PowerPC. Some low-level parts are written in assembly for each architecture.
- Therefore a HAL is required to manage hardware-specific implementations: interrupts, processor initialization, DMA, PCI/ISA bus access, timers, etc.
- Some drivers are specialized for a type of hardware (PCI driver, ATAPI driver, IDE driver, etc).
- Other drivers are generic modules, or file systems (Partition Manager, Volume Manager, NTFS, Mailslot Driver, Kernel Debugger Library, etc).

ReactOS Kernel Mode Design

- Nearly identical to NT.
- Current tree has \ntoskrnl directory implementing the kernel itself. CONFIG_SMP switch exists, mimicking NT's NT_UP, to build an SMP-compatible kernel (due to some tiny differences in dispatching code related to synchronization – especially spinlocks).
- Also have a \hal directory, with \halx86 and \halxbox. \halx86 also has UP vs SMP specific code (spinlocks and interrupts are implemented in the HAL). Main difference here is that SMP machines have APICs, UP machines have a PIC, usually.
- No ACPI support yet – HAL uses PIC and legacy timer hardware.
- Currently the only port being worked on is PowerPC.

Kernel Design Differences

- As mentioned earlier, the Cc, Cm APIs are incompatible, even at the application level. This is a show stopper for many drivers.
- Mm APIs are more compatible, but internally implemented differently. This is currently hurting performance and stability.
- PnP Manager communicates with drivers through IRPs (I/O Request Packets) and offers notifications/events/operations in a specific order – ReactOS doesn't have this fully compatible yet.
- NT Boot Loader sets up paged mode, IDT, GDT, TSS and low-level system structures. ReactOS boot loader works in protected mode only, and the kernel is responsible for system structures. This is currently being addressed.

Native Design - Subsystems

- Microsoft wasn't originally sure of the target API that the NT platform would support; original plan was OS/2.
- This later changed to Win32; designers needed to find a way to support this change, as well as implement POSIX compatibility.
- NT Subsystem model was created, similar to BSD's architecture.
- NT exposes system calls (Native APIs) that are at the core of the kernel.
- One Session Manager process, written in "native" mode, loads subsystem processes.
- Subsystem processes register non-native applications, such as Win32 or POSIX, which come with their own DLLs.
- These DLLs wrap the Native APIs, and sometimes need to call the subsystem process for functionality not accessible/available through Native APIs.
- Win32 supports consoles, NT doesn't -> subsystem process handles this.

Native Design - Subsystems

- Some functionality cannot be emulated by subsystem processes in user-mode alone -> a kernel-mode subsystem server is further required.
- Case in point: NT doesn't natively support the concept of a GUI, so Win32K acts as the subsystem server for gdi32 and user32. Kernel32 on the other hand can wrap native APIs (except for Console API).
- The only DLL NT comes with is ntdll, which has the system call entrypoints for all the Native APIs, as well as a Runtime Library, with over 600 Rtl* APIs for various operations (similar to the C runtime library/standard library).
- NTDLL also contains the loader for all PE files, but it doesn't know how to register a non-native application with its subsystem server.
- Kernel32 does this, by looking at the subsystem header in the PE file, which identifies it as POSIX, OS/2, WinCE, WinNT, etc.

ReactOS Native Design

- Closely follows NT but allows more extendibility.
- Session Manager allows custom-made subsystems, and doesn't have any hardcoded assumptions like NT's
  - SkyOS subsystem was made to run SkyOS applications.
- Win32 subsystem server (Win32K) is present, as well as the client (csrss), however Win32K APIs are not compatible (not required to).
- CSRSS communication with Win32 DLLs is done differently, but this is also not really required from a compatibility standpoint.

Win32 User Mode Design on NT

- Base API located in kernel32. Handles subsystem-part of loading applications and registering with subsystem client.
- Console API is also located in kernel32, talks with subsystem client for functionality.
- Graphical functionality is located in user32 and gdi32, which talk directly with the subsystem server through system calls.
- Other DLLs wrap more complex functionality directly in user-mode, or by calling kernel32 (advapi32, netapi32, winsock, etc).

Win32 User Mode Design on ROS

- Can’t use Wine for kernel32. Our implementation needs to call ntdll, which then performs the system calls, while Wine implements the APIs directly in the DLL.
- Same principle applies for GDI and USER32 libraries. NT design dictates going through kernel-mode (Win32K) – Wine does this inline.
- Other DLLs are directly shared from Wine with minimal changes to support our headers and build system.
- DLLs are supposed to work and be binary compatible with Windows.

Who we help, who helps us, and how to increase help

ReactOS and FOSS

ReactOS for Home Users

- Provides a solution for Linux users who still keep a Windows machine around for “that one program”.
- Also for specialized, outdated or even recent hardware which is only supported through Windows drivers.
- Completely eliminates the need to use Windows for the cases above.
- Also useful as a replacement platform for less tech-savvy users: parents, young children, etc. Allows to keep the same UI and application support, but without the hassles of the typical Linux installation – which isn’t fully ready for desktop use when it comes to less adept users.

ReactOS for Corporate Users

- Allows companies to migrate without support or training costs, since the UI and functionality is the same.
- Sometimes employees keep a Windows machine around for thin/smart-client applications.
- Can run in-house proprietary apps which have never been ported, and maybe whose source code is even lost.
- ReactOS will support SMB, Active Directory, IIS and other Windows Networking components or protocols.

ReactOS for Developers

- Allows FOSS developers to target the Windows platform easier:
  - MSDN documentation is sometimes incomplete, wrong, or inexistent for undocumented functionality.
  - Some bugs can occur in deep call stacks full of Windows DLL code, so finding bugs can be a pain.
  - Undocumented flags, behavior and quirks are hard to detect.
- Most corporations employ some degree of reverse engineering Windows binaries to better understand behavior – especially true in kernel-mode driver development.
- FOSS projects have less ability to do this, as well as an increased chance of legal repercussions.
- ReactOS solves these problems by providing GPLed implementations of these APIs, binaries, drivers and kernel.

ReactOS for Developers

- By being able to see the source for the entire system, developers can better understand race conditions, optimizations, side-effects and other low-level repercussions of their code.
- Debugging a crash in a DLL can now be done with source-level debugging.
- Access to a team of experts always willing to help (#reactos or the Mailing List), bugzilla.
- Due to bugs in ReactOS, this may even cause your application to exhibit bugs in untested conditions.

ReactOS and the Community

- We increase awareness to FOSS in general, by showing even hardcore Windows developers the benefits of having access to source code.
- By not bundling proprietary applications, we increase awareness of projects such as Firefox, Mplayer, Samba which are recommended for installation or can even be bundled as part of distributions.
- As mentioned earlier, projects such as Qemu, MinGW/GCC, BinUtils get a lot of input from us since we aggressively use these applications/toolkits.
- Libraries such as libpng, zlib, MESA also get exposed by default through DLLs which wrap their functionality, resulting in a fully open OS composed of many other project's work.

ReactOS and the Community

- ReactOS couldn't exist without FOSS; apart from the libraries and the people behind it, other FOSS projects make our management tick:
  - SVN provides our source repository software.
  - CIA provides notifications by mail and IRC.
  - BuildBot is the central authority for regression testing.
  - Piper-mail handles the mailing lists.
  - Our website and other servers are all LAMP.s

ReactOS and Freenode

- ReactOS would not exist without freenode!
- Our team is composed of less then twenty people, with almost one person per country, sometimes a bit more.
- Development is thus completely decentralized, and victim to time zones, language barriers, etc.
- IRC and freenode solve this problem:
  - All development is coordinated through #reactos.
  - Developers are almost always on, logging the channel for any news.
  - Private message and memos can be sent.
  - Users can look at op or voice status to look for help.
- Mailing list is used in cases where the developer is not on IRC or when global discussion is needed.

Anticipated progress and usability

Roadmap and Call to Action

2007 Roadmap

- We’re hoping that the kernel will be entirely complete and compatible for 90% of drivers out there, by year’s end.
- Also hoping for Base/Core APIs to be fully working and regression tested.
- 2007 will feature heavy work on Memory Manager, Cache Controller, FS support, PnP Support, Win32K, DirectX, Sound support and Windows Networking.
- Largest focus will be on stability and usability however; 2006 suffered from too many rewrites and changes without proper testing and regression checks.

Call to Action

- ReactOS needs help!
  - Financial: Fundraising campaign is now in effect, hoping to raise 4000 Euros.
  - Manpower: Desperately require more developers familiar with Windows development, either Win32 or kernel mode.
  - Testers: Regressions need to be caught sooner and faster; automated systems aren't always perfect.
  - Awareness: Project is very popular in some European countries (notably Germany) but has poor penetration in US/Canada. We could reach out and help a lot of people!

Questions, comments

- http://www.reactos.org
- ReactOS IRC Channel on Freenode – #reactos
- ReactOS Mailing List – ros-dev
- My blog: http://www.alex-ionescu.com
- Email: alex.ionescu@reactos.org