

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Российский Государственный Социальный Университет»**

Специальность – Информационные системы (по отраслям)

Дисциплина «**Основы алгоритмизации и программирования**»

«**Платформа Wine, принцип работы**»

**Выполнил:**

студент 2 курса

группы ИН-К-0-Д-2020-2-11,

Чайковский Н. О.

**Проверил:**

преподаватель

Ерпелев А. В.

Введение

Моментом зарождения самой идеи эмуляции другой операционной среды можно считать 1993г., тогда Microsoft успешно продвигала Windows как будущее персональных компьютеров. IBM тоже хотели получить свою часть рынка, и сторили большие надежды на свою OS/2, но уже тогда они признали, что поддержка приложений Windows будет необходима, и внедрили её в свою ОС.

Годом ранее, в сентябре 1992г. Sun Microsystems начала разработку продукта под названием Wabi, и в 1993 продемонстрировала его на Solaris Developers Conference.

Продукт позволял пользователям Solaris x86 и Solaris 2.2 для SPARC запускать приложения разработанные для Windows из коробки.

В то время и другие решения позволяли запускать приложения Windows, но они требовали установку DOS и Windows. Уникальность Wabi была в том что она позволяла напрямую транслировать оконные вызовы в X Windows вызовы. При этом эмуляция остального x86 кода позволяла в некоторых случаях работать приложениям быстрее.

В то время пользователи Linux уже собирались в достаточно дружное сообщество, и они начали дискуссии на тему подобного подхода как у Wabi, но на Linux системах. В то же время шансы на то, что Wabi портируют были близки к нулю.

Название Wine было быстро согласовано, и в числе первых разработчиков были одни из первых разработчиков ядра Linux Eric Youngdale и David Metcalfe, Alexandre Julliard (до сих пор лидирует разработкой Wine), Miguel de Icaza (разработчик GNOME) и Bob Amstadt который руководил разработкой.

Начать было решено с запуска 16-bit бинарных файлов, прогресс шёл быстро и спустя полгода уже было возможно запустить Solitaire.

В ноябре 1993 так же был совершен первый порт Wine, который позволял запускать эмулятор на NetBSD. В то время Боб сказал, что с текущей скоростью разработки они выпустят первый релиз через полгода-год. Иронично что эти "полгода-год" продлились десятилетие.

В первые годы было очень много изменений в разработке Wine. Оконные вызовы были переписаны на прямые вызовы Xlib, Microsoft начала выпуск 32-битного кода и добавление нового функционала в их операционную систему. Уже нельзя было просто загружать код и исполнять его. Нужна была более тесная интеграция с системой, на которой непосредственно запускается Wine.

В то время архитектура Wine полагалась на общее пространство между приложениями, но быстро выяснилось, что это достаточно плохое решение. Разделение пространства нужно было что бы улучшить безопасность и проблему общих библиотек, которые пытались получить доступ к одному и тому же пространству. Работа началась в начале двухтысячных и продолжается по сей день.

Alexandre Juliard на первой Wineconf выделил следующие важные контрольные точки разработки Wine в то время:

* Май 1995: начало поддержки Win32 архитектуры
* Июль 1995: переписывание кода на поддержку autoconf
* Июнь 1996: успешный запуск Word и Excel
* Ноябрь 1997: запуск сайта winehq.com

В начале Wine была лицензирована под BSD-like лицензией, в 1999 руководитель разработки отметил, что такая лицензия несовместима с GPL лицензией. Что потенциально означает что много библиотек открытого исходного кода было бы невозможно добавить в проект. В ходе дискуссии они пришли к решению изменить лицензию на LGPL.

Изменение лицензии стало переломным моментом в разработке, обновления стали выходить чаще. В 2006 вышла Wine 0.9.0 которая была официальной бетой, и Wine 1.0 окончательно получила релиз в середине 2008.

Разработка продолжается и сейчас с обновлениями от разработчиков каждые две недели, написано боле 1.4 миллиона строк C кода, больше 700 людей тем или иным образом вложили часть себя в проект.

Основная часть

Внутренняя реализация

Главной целью Wine ставит реализацию тех Windows API которые необходимы для запуска приложений.

Большая часть программ на Windows состоят из DLL файлов, эти файлы содержат очень много оберточных подпрограмм для системных вызовов ядра. Типичная программа вызывает некоторые Windows DLL файлы, которые в свою очередь вызывают gdi/user32 библиотеки, а те уже используют kernel32.dll который отвечает за связь с ядром при помощи системных вызовов.

Слой системных вызовов приватный слой, т.е к нему если и существует документация, то только внутренняя, доступная разработчикам Microsoft. Кроме этого, есть некоторые программные интерфейсы, которые реализованы как сервисы, и запускаются в отдельных процессах. Общение с такими сервисами из публичной части реализуется через RPC.

Wine в свою очередь реализует Windows ABI (application binary interface) полностью в пространстве пользователя, а не в качестве модуля ядра. При этом архитектура файлов и сервисов повторяется. Сервисы, которые обычно предоставляются ядром, вместо этого отдает демон wineserver, целью которого является реализация базовой функциональности Windows и интеграфия с X Window System.

Хоть Wineserver и реализует некоторые аспекты ядры Windows, исходя из архитектуры Wine невозможно использовать некоторые нативные драйверы Windows, из-за особенностей архитектуры Wine.

Библиотеки

Wine позволяет загружать как Windows DLL так и Unix shared object файлы для использования программами Windows. Встроенная реализация самых базовых Windows DLL таких как: NTDLL, KERNEL32, GDI32, USER32, используют shared object метод, так как они используют функции непосредственно в операционной системе. Библиотеки уровнем выше, например WineD3D обычно используют DLL формат. В большинстве случаев пользователь может конфигурировать Wine что бы настроить какие DLL файлы будут загружаться в нативной реализации Windows, а какие в реализации Wine.

Графика

Игры, в отличии от большинства офисного ПО используют комплексную GPU-ускоренную графику. Что бы запускать такие приложения Wine необходимо транслировать инструкции по отрисовке напрямую в ОС хоста. Основная проблема в том, чтобы сделать так, что бы хост это понял.

* Wine реализует такие API для отрисовки графики как:
* DirectX 11.2 для OpenGL
* DirectX 12 реализация для Vulkan API
* В случае Mac OS Wine транслирует сигналы из Vulkan API в Metal API
* XAudio (низкоуровневое API для звука) реализовано с помощью библиотеки FAudio, так же.

С помощью реализации XInput and Raw Input (API для получения пользовательского ввода) через shared object и SDL (Simple DirectMedia Layer) Wine поддерживает работу с игровыми контроллерами. Так же существует поддержка Direct2D

Основные силы разработчиков Wine уходят на реализацию Direct3D через WineD3D, который представляет собой слой для перевода Direct3D и DirectDraw вызовов в OpenGL. Реализация работает хорошо или плохо в зависимости от игры, но с помощью Wine есть возможность запуска достаточно массивных игр, например Overwatch или Dota 2.

Кроме того, использование WineD3D DLL позволяет запускать на более старых процессорах игры под новые версии DirectX.

Так же существует проект vkd3d который реализует Direct3D бэкэнд для Vulkan API.

Детали функциональности

Разработчики Direct3D части Wine продолжают работу, например внедряя пиксельные шейдеры, чтобы улучшить поддержку игр. Кроме того, Wine может использовать нативные библиотеки Windows, но для их использования нужна и лицензия Windows.

Так же Wine включает в себя собственную реализацию таких приложений как notepad, wordpad, control, internet explorer, explorer

Существует так же Wine Application Database (AppDB), библиотека приложений и игр, которые работают на Wine или без конфигурации (из коробки), или с конфигурацией, которая так же предоставляется в этой библиотеки. Так же есть описания в случае того почему какая-то из программ может не работать, и собираются ли это исправлять.

Список из первых 10 "Платиновых" (продукты, которые работают прекрасно без конфигурации) продуктов из этой библиотеки:

* World of warcraft
* StarCraft 1
* Fallout 3
* The sims
* Gothic 3
* Company of Heroes Obsolote
* TES 5: Skyrim
* .NET Framework 3.5
* System Shock 2
* Diablo 3

Обратная совместимость

Wine гарантирует хорошую обратную совместимость со старыми приложениями Windows, даже включая программы написанные для Windows 3.1x. Однако, поддержка приложений Windows 1x и Windows 2.x была убрана в версии 1.3.12. Но существует способ запускать такие приложения используя функцию Wine Windows 2.0 используя установленный MS-DOS.

Обратная совместимость Wine на голову превосходит таковую у Windows. Ведь Windows обычно «убивает» приложения, которые официально не поддерживаются, а Wine в свою очередь предоставляет "Compatibility Mode" который позволяет запускать даже 16 битные программы в 64 битном окружении.

64 битные приложения

Первая версия поддержки 64 битных приложений была добавлена в Wine 1.1.10 в Декабре 2008 г., а стабильной её начали считать только в Апреле 2019 года. Так же существует две отдельных версии приложения Wine. Где Wine64 поддерживает окружение способное запускать только приложения x86-64.

В то же время Wine выпустил стабильную версию поддержки WoW64. Которая позволяла 32 битным приложениям запускаться в 64 битном окружении.

MS-DOS

Ранние версии Microsoft Windows запускались поверх MS-DOS, и некоторые программы зависели от реализаций MS-DOS. Wine не имеет хорошей поддержки этой системы, но с версии 1.3.12 пытается запускать MS-DOS программы в окружении DOSBox, если таковая доступна.

Приложения Microsoft

Wine реализует свою имплементации JScript и VBscript для поддержки Internet Explorer и .NET Framework. Однако версии Internet Explorer ниже восьмой не признаются официально рабочими "из коробки". На момент написания этого реферата последние версии Google Chrome и Edge признаются рабочими практически без дополнительной конфигурации.

Вывод

Как можно увидеть необходимость разработки подобных прослоек совместимости (вспоминаем что Wine нельзя назвать эмулятором), возникла практически в тот момент, когда разные ОС предоставляли разный API наружу. При этом пользователям одной ОС ну очень хотелось использовать приложения, которые были написаны для другой.

Над Wine была проведена огромная работа, и основная причина её объемности - закрытый исходный код, а также отсутствие документации к большинству библиотек Windows. Поэтому разработчикам очень часто приходится опираться лишь на свой опыт и практики reverse-engineering. Однако, сейчас Wine позволяет практически стереть грань между двумя ОС с небольшими ремарками.