Open Source и OS/2

DATE: 2002-08-16 10:21:22

AUTHOR: Yuri Prokushev (prokushev@freemail.ru)

Введение

Идея Open Source OS/2 отнюдь не нова. Причина в этом лежит на поверхности и всем довольно хорошо известна. Это неопределенность политики IBM относительно OS/2. Постоянные намеки клиентам на мигрирование на другую платформу, с параллельным выпуском как новых версий операционной системы, так и обновление существующих, не вносят ясности.

На данный момент развитием клиентской версии OS/2 занимается фирма Serenity Systems. Будет развитие клиентской версии OS/2 (eComStation) успешным или нет - покажет время. Сейчас Serenity Systems внушает бОльшие надежды, чем IBM. Однако, доступ к исходному коду операционной системы позволил бы:

- расширять возможности системы силами не только одной компании, но и сторонними разработчиками (имеющими необходимые навыки и знания).
- значительно ускорить исправление ошибок и недочетов; и, естественно, иметь меньшую зависимость от ситуации на рынке.

Попытки получения исходного кода OS/2, в целом или частями, не увенчались успехом и не принесут результатов до тех пор, пока не будет виден какой-либо реальный результат работы сторонних разработчиков. Ярким примером может быть проект Odin, который получил в свое распоряжение библиотеку, используемую для переноса Netscape Navigator/Netscape Communicator на платформу OS/2. Другой яркий пример, это работа IBM над проектом Mozilla, а именно, версией для OS/2 - Warpzilla.

В данной статье приведены результаты попыток начала проекта Open Source OS/2, а также мое отношение к этим проектам. Также дана моя оценка перспективности данных проектов и их дееспособности.

Здесь я попробую выстроить систему, как я ее вижу и как она могла бы быть построена. Небезосновательно полагая, что система сама по себе, без приложений, никому не нужна, также рассмотрим ряд программ.

В качестве исторического введения. Проект FreeOS

Наиболее заметным и, вероятно, самым первым проектом Open Source OS/2 был проект FreeOS. Он начинался под эгидой (если я правильно помню) Daniel Caetano (автор PM Download Center). Последней попыткой представления данного проекта в интернете является сайт FreeOS. Однако, в рамках данного проекта не было написано ни одной строчки кода и все закончилось разговорами. С моей точки зрения, данный проект был обречен на провал по ряду причин. Это остутствие какой-либо четкой организации, отсутствия репозитория, где можно было бы держать результаты работы, и отсутствие какого-либо информационного ресурса в интернете, исключая некоторое появление в новостях и наличия списка рассылки. Проект просто

закончился, не начавшись. Как ни странно, люди зачастую предпочитают делать реальные вещи, а не заниматься проектированием и планированием, тем более, что проектировать-то на первой стадии особенно нечего, все и так уже существует.

Основа системы

Любая система начинается с ядра. На данный момент существует ряд ядер или же инструментарий для построения таковых. Ядра с закрытым кодом не рассматриваются по известным причинам. Наиболее известным на данным момент ядром является Linux. Монолитное, по устаревшей технологии, оно, тем не менее, завоевало широкую популярность, в основном благодаря системе GNU. Различные варианты GNU/Linux распространяются рядом компаний. Наряду с этим существует ядро GNU Hurd. Однако, оно так и не набрало оборотов. Кроме этих, наиболее известных ядер, существует еще ряд широко применяемых, в частности микроядро, на котором была в свое время построена OS/2 PPC. Все эти ядра обладают одним, но существенным недостатком: отсутствие драйверов устройств. Исторически сложилось так, что линейка операционных систем Windows обладает богатым набором драйверов и все производители выпускают со своим оборудованием драйвера именно для нее. В связи с этим, целесообразным является применение ядра, обеспечивающим совместимость с существующими драйверами. Единственным достигшим каких-либо успехов в этом направлении является ReactOS.

Проект ReactOS

Некоторые, возможно, удивятся. Причем здесь проект Open Source NT? А ответ очень простой. Помните NT 3.51? Она умела запускать программы OS/2 (в стандартной поставке - только консольные, но, как ни странно, Microsoft распространяла и Presentation Manager, как дополнительный компонент). А помните NT 4.0? В ней тоже имелась данная возможность. И дело не в том, что “NT базировалась на OS/2”. Дело в самой архитектуре NT. Но об этом позже. А сейчас о том, какое отношение ReactOS может иметь к Open Source OS/2? Да самое прямое, если хотите. Ядро. Со всеми вытекающими. Со всеми драйверами.

Попробую рассказать об архитектуре ReactOS, как я ее понимаю (и это понимание стоило листу рассылки ReactOS моему двухнедельному присутствию и надоеданию, вплоть до выяснения терминологии и доказывания одного и того же друг другу 🙂). Итак, руководствуясь ReactOS architecture 'for dummies' (dummies - это я 🙂), несколько слов об архитектуре (от железа к приложению).

Hardware Abstraction Layer (HAL) (hal.dll). Это что-то связанное с драйверами и в данном контексте не представляет интереса. А вот далее идут **Executive Subsystems**, в некоторой литературе упоминаемые как native executive. Это семь основных подсистем (реестр,

менеджер ввода/вывода, менеджер памяти и пр. - ntoskrnl.exe). И дополнительные подсистемы (терминология видимо сменилась после моих докапываний, о которых далее):

- Расширения Win32 (ввод пользователя, графика, обмен сообщениями и пр.) (win32k.sys)
- Расширения POSIX (сигналы, виртуальные терминалы, эмуляция Linux и пр.) (psxk.sys)
- Расширения OS/2 (это мое добавление, возможно, что никаких расширений и не потребуется)
- Прочие расширения

И последняя часть, это драйвера:

- Class Drivers
- Device Drivers (поверх классов устройств)
- Filesystems (поверх устройств)

А теперь о самом вкусном и представляющим практический интерес именно для Open Souce OS/2. Это подсистемы. Яркий пример подсистемы из Windows NT, это подсистема Win32 (win32ss). А вот запуск программ OS/2 выполнялся как раз с помощью подсистемы OS/2 (os2ss)! На самом деле, поверх ядра (ReactOS или NT) можно построить подсистему для любого приложения. И реально заставить их работать вместе, разделяя один рабочий стол, буфер обмена и прочая и прочая.

Давайте рассмотрим планируемые подсистемы ReactOS:

- Session Manager (system-wide debug manager, startup and shutdown manager) (smss.exe)
- Local Security Authority (LSASS) (lsass.exe)
- Win32
- POSIX+
- OS2SS
- DOS
- Java

Первое и второе не представляет большого интереса в данном контексте, а вот остальное представляет большой интерес. Подсистема Win32 дает возможность работы приложений Win32. Не требуется никаких Odin, VirualPC, bochs и прочего. Подсистема POSIX дает возможность использовать программы стандарта POSIX, что включает простой перенос программ с систем *nix. Это не требует огромных затрат на разработку систем типа emx, posix/2, libemu/libext. Да в них и вообще нет тогда смысла. Подсистема OS/2 дает, ну сами понимаете, что :). Да и остальное тоже.

Здесь может возникнуть закономерное возмущение. А накой мне нужна винда? Я хочу любимый WPS, любимый WarpCenter (LaunchPad, XWorkplace, подставить по вкусу), а не урезанный explorer? Логично. Но никто и не требует использовать программы подсистемы Win32 как менеджер рабочего стола! Запускайте WPS! С точки зрения системы без разницы, запускать explorer или wps. Это те же приложения. Весь вопрос только в корректной организации подсистем. Так, например, начиная с NT 4.0 практически вся графическая подсистема Win32 вынесена в Win32 Extensions (win32k.sys), что, конечно, влияет на стабильность системы в целом (а, скажем, драйвера PM не влияют? Очень даже как влияют!), но позволяет использовать единый менеджер окон (а значит и разделять окна на одном рабочем столе), единый буфер обмена и прочая и прочая. Т.е. если подсистемы используют win32k, то они вполне могут сосуществовать на одном рабочем столе.

Как и было обещано, об изменении терминологии. Win32k классически считается частью подсистемы win32, что, в принципе, не далеко от правды, так как основано это расширение как раз на частях подсистемы win32. Но это уже часть ядра, не зависящая ни коим образом от win32ss, а следовательно, может свободно использоваться прочими подсистемами. В смущение вводит еще и тот факт, что имя у расширения win32k. Т.е. часть win32 в ядре, но уже не часть win32ss. Поэтому данные части рассматриваются как расширение **ядра** для поддержки подсистем(ы).

Теоретически, подсистемы не должны пересекаться друг с другом, кроме как на уровне ядра. Практически это можно сделать (так, например, сейчас все подсистемы ReactOS работают через программы-терминалы win32, так как еще отсутствет поддержка управления окнами), но это необходимо пресекать, если необходима независимость подсистем друг от друга и возможность свободной компоновки подсистем.

Что над ядром?

А над ядром строятся приложения. Достаточно наличия необходимого API и двоичной совместимости (в нашем случае понимание формата LX) для запуска любого существующего приложения. Текущее API OS/2 достаточно документировано, а многие недокументированные функции уже известны. Обеспечив на уровне подсистемы OS/2 совместимость по API, хотя бы консольного, можно переходить непосредственно к работающим программам. Зачастую в командном режиме запускаются утилиты командной строки. Но до этого нужны средства разработки.

Проект OpenWatcom

Прекращение поддержки VisualAge линейки для OS/2 является значительной утратой для OS/2. Единственный полноценный инструментарий под OS/2 прекратил свое существование. Существующий на данный момент GNU Compiler Collection более удобен для переноса *nix программ, а не для разработки родных приложений. К счастью существует проект OpenWatcom. На данный момент это версия 0.8.x. Неплохая альтернатива для разработчика. Координируется фиромой SciTech, которая широко известна своим пакетом видеодрайверов. OpenWatcom претендует на роль ведущего компилятора OS/2.

Среда разработки OpenWatcom принята как основная для проекта osFree, что, несомненно, лучше, чем GNU C для данной задачи.

Проект osFree

Все помнят недавний шум вокруг osFree. Собранный из исходных текстов, “утекших” из IBM, дистрибутив вызвал море обсуждений у всех осевиков.

Дистрибутив до сих пор периодически всплывает то на одном, то на другом сервере, но факт остается фактом. Дистрибутив незаконный. Те, кто подписывался на лист рассылки osFree помнят обсуждение о легальности кода и то. что единственным представителем со стороны osFree team был John Martin, наиболее часто фигурирующий как JMA. Факт то, что он отказался от дальнейшей поддержи нелегального дистрибутива и предложил все-таки разрабатывать Open Source OS/2. Так что проект osFree теперь не имеет отношения к osFree Technical Preview и, по сути дела, ведется на полностью легальной основе.

Однако, проект снова ушел в разговорную стадию. Планирование, конечно, представляет интерес, да и писать код, имея четкий план в голове и на бумаге на порядок проще. Но кроме разговора должно присутствовать и дело.

И наконец, появились маленькие кусочки кода и появился репозиторий.

Пока он относительно молодой и содержит немного утилит. Это, на момент моего последнего просмотра, ansi, chkdsk, recover, sysinstx плюс начало формирования структуры репозитория. Согласитесь, это уже прогресс по сравнению с FreeOS. Уже есть надежда, что проект наберет силу. Если на данной стадии не обновить информацию о проекте в новостях, не сообщить об открытии cvs и не поддерживать желающих поучаствовать, то все опять может успешно заглохнуть.

На данный момент osFree самый перспективный из проектов и который может в итоге дать хотябы набор утилит командной строки и некоторых приложений PM. На большее пока расчитывать рано.

Проект ReginaREXX

Одним из любимых инструментов пользователя OS/2 является REXX. Интегрированный в систему (по сути дела - библиотека) он используется всем и вся. Так нужна ли замена? Нужна и уже давно существует. ReginaREXX может быть легко использован, причем с малыми затратами.

А что с PM?

Интересный вопрос. На данный момент самое приятное лежит над PM. Это SOM, WPS. Без них мало кто мыслит себе OS/2. Как реализовать? В рамках подсистемы OS/2 ReactOS. Используя уже существующий Win32k. Проект практически аналогичен Odin, но в обратную сторону. Да и API поменьше будет.

SOM. С SOM все сложнее. По слухам, существет SOM3, с открытым исходным кодом. Также существует еще ряд аналогичных технологий. Существующие SOM-классы более или менее документированы. Т.е. нужно создать опять же API. Учитывая то, что SOM является реализацией CORBA, логично использовать существующие Open Source реализации данной технологии.

С WPS уже сложнее. Аналогов нигде не существует и вряд ли будет. Придется руководствоваться тем же Toolkit. Вполне возможно использование XWorkplace как основу для многих классов.

А остальное?

А остального как бы и не существует. Есть Open Scripting Architecture, нечто вроде Windows Scripting. Есть OpenDoc, который так и остался в начале. Есть MMOS/2, которую никто полноценно не использует. Сложно назвать что-то еще. Хотя это JFS. Так что можно более или менее успешно применить FreeJFS в качестве ReactOS IFS.

Визуальные средства разработки. Проект OpenSibyl

Прекращение разработки SpeedSoft Sibyl могла бы стать непоправимой утратой. Потому как это единственный аналог Delphi для OS/2 и можно с уверенностью утверждать, что RAD среды способствуют появлению программ и утилит. Где-то в 1999 году были открыты промежуточные исходные тексты Sibyl для проекта Medigo, который планировал разработку RAD на основе Free Pascal Compiler для GNU/Linux. По всем данным им так и не удалось откомпилировать данные исходные тексты (что очень странно, ибо у меня на их сборку ушло не более часа). Проект Medigo закончился не начавшись, но исходные тексты остались. В последующем я не раз встречал в списках рассылки вопросы о компиляции данных исходников и никто не добился результата.

Netlabs.org анонсировала открытие проекта OpenSibyl и был создан список рассылки. Проект вызвал большой интерес, но… Снова влияла неповоротливость осевиков. Проект очень медленно набирает обороты. Однако, на данный момент имеется CVS и список рассылки, что необходимо, но недостаточно. Так как я непосредственно участвую в данном проекте (и, по сути дела, являюсь единствееным разработчиком), то могу более или менее осветить ситуацию.

Сейчас стоит следующий ряд проблем:

- Обновление существующих и разработка новых частей RTL Free Pascal/2
- Перенос библиотеки классов(SPCC) и IDE (SVDE) под новую RTL
- Исправление ряда критических ошибок в SVDE (запуск компилятора, чтение проекта)

- БОльшая совместимость с Delphi/Kilyx.

На данный момент основное внимание уделено получению полновесной и минимально необходимой rtl для fpc/2 и запуск компилятора fpc/2, вместо Speed Pascal/2. В принципе, при активном участии 2-3-х человек это можно осуществить в довольно короткие сроки.

Более сложная задача состоит в переносе библиотеки классов под Free Pascal. Из-за ряда различий в реализации классов (в частности, с заполнением VMT) перенос SPCC значительно усложнился, чем предполагалось в начале. Поэтому в планах скорое прекращение поддержи компилятора Speed Pascal и полный переход на Free Pascal.

Приложения Internet. Проект Mozilla

Вышедший недавно релиз Mozilla 1.0 показал просто отличные результаты. Это уже не тот Netscape, что был раньше. И хотя уже не раз были крики о просто дикой тормознутости браузера, то Mozilla 1.0 сильно порадовал. Большинство обзоров по Mozilla отмечают значительное ускорение по сравнению с 0.x серией, но забывают об очень многом. При всей своей первоначальной направленности на Internet Mozilla является отличным набором технологий. Копнув поглубже многие заметят, что это, в принципе, конструктор. Здесь вы найдете такие широко используемые возможности Internet Explorer, как embedding (встраивание браузера в другие приложения), скины (причем на порядок мощнее). А по вопросам поддержки стандартов можно вообще умлочать. HTML, XML (xhtml, mathml, svg и пр.), ECMAscript и многое другое показывают значительно лучшие результаты, чем IE.

Также представляет интерес ряд прочих технологий, в частности XPCOM. Но общий интерес в том, что Mozilla дает мощный инструмент для создания приложений, как для Интернет, так и не для Интернет, которые легко могут быть перенесены с одной платформы на другую.

Офисные приложения. Проект OpenOffice

Наверняка все знают о печальной судьбе StarOffice. Получив права на StarOffice Sun недолго думая похоронила версию для OS/2, выдвинув в качестве аргументов несусветную глупость, что мы, мол, не смогли скомпилировать его в VAC 4. Но они открыли исходные тексты StarOffice и мир получил в свое распоряжение OpenOffice. Но до OS/2 версии так и не добрались. В списке портов числятся разработчики OS/2, но патчей так и не поступило, в то время как OpenOffice 1.0 уже вышел. Объединив antiword/2 и OpenOffice можно получить неплохую поддержку форматов MSWord, что является, как мне кажется, одной из основных задач.

Выводы

В данной статье не рассматривалась возможность построения Open Source OS/2 на базе GNU/Linux. В частности из-за того, что это инструментальная среда, по сути большой компилятор, и построение абсолютно чуждой по идеалогии среди на базе системы GNU не имеет смысла. Также не рассматривались вопросы о модифицированной OS/2, что естественно, необходимо. Была попытка показать существующие проекты и возможность их развиния.

На данный момент имеются два действующих проекта по реализации базовых компонентов системы. Это osFree, который, с моей точки зрения, не ставит целью разработать ядро, а разработать полновесные аналоги каждой части OS/2 (утилит командной строки, PM, SOM, WPS, и многое другое). С этой точки зрения он полезен для получения Open Source аналогов подсистем OS/2 и их расширенных версий.

Второй проект, это ReactOS. Сейчас работой над подсистемой OS/2 занимается Robert K. (к сожалению, не знаю фамилии, вероятно Koepferl). Но он не желает поддерживать 16-битный API, который широко используется консольными приложениями. ReactOS может послужить ядром и источником подсистем Java, Posix, Dos, Win16, Win32, что снимает необходимость в разработке их для OS/2, а позволяет напрямую их использовать. Плюс отсутствие проблемы драйверов как таковой, пока существует NT.

В принципе, каждый волен принять посильное участие как в первом, так и во втором проектах. Вопрос только в отсутствии двойной работы. Чтобы одна команда не работала над тем же, что и вторая. Но это, ИМХО, решаемо. Написание утилит вроде more, cmp, tree, bldlevel и пр. под силу практически любому. Написав одну утилиту вы сделаете небольшой, но полезный шаг в становлении Open Source OS/2.

- **Замечание**: Данная статья была впервые опубликована здесь: Open source и OS/2, часть 1
- Комментарии к статье: comments