Обзор операционной системы osFree (Open Source версии OS/2)

Date: ??-07-2011

Authors:

- Валерий “valerius” Седлецкий (_valerius@mail.ru),
- Юрий Прокушев (prokushev@freemail.ru),
- Вадим “Oxyd” Прилуцкий (<???>)

В данном докладе будут рассмотрены основные цели и задачи стоящие перед разработчиками операционной системы osFree, являющейся opensource версией популярной, в недалёком прошлом, системы IBM OS/2 Warp и совершён краткий экскурс в историю развития OS/2-like систем. Так-же, в общих чертах, будет показана архитектура системы, базирущейся на микроядре L4 Fiasco.

Вступление

Краткая история OS/2, альянс IBM/Microsoft

- В 1984-86 гг., в связи с выпуском фирмой Intel процессора i286 и недостаточной поддержкой операционной системой DOS возможностей машины, фирмы IBM и Microsoft взялись за реализацию новой ОС, которая должна была заменить DOS. Эта система впоследствии была названа OS/2, а перед этим носила рабочие названия сначала ADOS (Advanced DOS), затем CP/DOS (Control Program, название, закрепившееся за ядром OS/2). API ядра называлось CPAPI (Control Program API). По своим визуальным характеристикам (Look'n'feel с точки зрения пользователя) она наследовала командную строку DOS'a.
- Первая версия вышла в 1987 году, она была похожа на многозадачный DOS, могла запускать единственный сеанс совместимости с DOS и несколько полноэкранных сеансов командной строки OS/2. Тогдашним пользователям запомнился интерфейс CShell – программы, которая запускалась как менеджер сеансов и могла запускать сеансы DOS и OS/2 и переключаться между ними. Сейчас с подобной оболочкой – TShell тоже можно встретиться, она бывает очень полезна для загрузки с дискет и очень похожа на первоначальный CShell.
- Потом, когда появился процессор i386, между фирмами IBM и Microsoft стали возникать разногласия. Например, по поводу того, стоит ли продолжать разработку ОС для процессора i286. Процессоры i286 тогда были еще очень распространены, и OS/2 1.xx была единственной ОС, поддерживавшей эти процессоры. Поэтому IBM считала, что надо продолжать разработку OS/2 1.xx, а MS считала, что нет.
- Параллельно обе фирмы вели разработку 32-разрядной версии OS/2. В 1990 году появилась бета-версия 32-битной OS/2 2.0 от IBM, а MS продолжала разработку своей 32-битной версии OS/2 3.0 NT. Затем отношения обеих фирм полностью испортились. IBM продолжила в одиночку разработку OS/2 2.0, а Microsoft позже переименовала свою разработку в Windows NT 3.0, пригласив Дейва Катлера из DEC. С тех пор, WinNT наследует много идей из OS/2, смешав их с идеями DEC VMS. Фирма Microsoft разработала оригинальное ядро NT, поддерживающее подсистемы.
- Фирма же IBM в OS/2 2.0 выпускает оболочку рабочего стола WPS. Затем в версиях 2.xx было реализовано множество улучшений, включая 32-битный графический Engine и поддержку приложений Windows 3.1. В 1995 была выпущена OS/2 Warp 3.0, затем чуть

позже OS/2 Warp Connect, содержащая мощную поддержку сетей. 1996 г. был выпущен OS/2 4.0 Merlin с измененным пользовательским интерфейсом, поддержкой подмножества Win32 API под названием Open32, поддержкой Java машины от IBM – самая быстрая реализация Java-машины на платформе i386 (быстрее, чем Sun-овская).
- Также, в 1995-96 годах была выпущена OS/2 Warp Connect (PowerPC Edition) – часть проекта IBM Workplace OS – OS/2 personality для IBM-овского микроядра, представляющего собой модифицированное микроядро Mach. (Скопипастить интересные подробности из статьи на сайте Michal Necasek!) Также, кроме OS/2 personality, были разработаны DOS/MVM personality и AIX personality (?). Это был новый подход по уменьшению сложности ядра ОС, и с другой стороны, объединению в единую ОС поддержки приложений DOS, OS/2, Windows, AIX, AS/400 и других ОС. Параллельные API различных ОС предполагалось реализовать на основе микроядра и OS personalities, реализуемых как библиотеки и серверы в userspace. Этот подход альтернативен подходу в WinNT (параллель: NT subsystems – OS personalities). В отличие от ядра NT, драйвера, файловые системы и OS personalities работают в userspace. Это увеличивает изоляцию компонентов и устойчивость. IBM здесь использовала классический микроядерный подход.

Спад интереса к OS/2 со стороны IBM, OEM-контракт с Stardock, и затем с Mensys/Serenity. Разработка eComStation.

- После выпуска в 1999 году OS/2 4.5 Aurora, начался спад интереса к OS/2 со стороны IBM. Вскоре было сообщено, что разработка системы будет сворачиваться, но поддержка и обновления будут продолжаться до декабря 2006 года. Затем известная на рынке OS/2 приложений фирма Stardock Systems обратилась к IBM с предложением OEM-контракта на разработку клиентской версии OS/2 (от IBM была доступна только серверная версия Aurora). После отказа, последовавшего по неизвестным причинам, немного позже, в 2001 году, такое право было передано фирме Serenity Systems (не столь широко известной, как Stardock). Но у Serenity был влиятельный финансовый спонсор – голландская фирма Mensys, продающая софт в своем интернет-магазине.
- На текущий момент данная OEM-версия OS/2 продается под именем eComStation. Но мы считаем это полумерой. Фирмы Mensys и Serenity не имеют полных исходных кодов системы, в том числе, ядра и Presentation Manager, поэтому они вынуждены использовать их как “черный ящик”, лишь добавляя новые возможности, используя наследование в SOM и путем бинарных патчей. Тем не менее, Mensys и Serenity удалось добавить поддержку ACPI, и возможность загрузки с томов JFS, используя модульность ядра OS/2. Так что, потенциал еще не исчерпан, но он все же имеет свои пределы.

Петиции к IBM

- 1 ноября 2005 года сообщество энтузиастов OS/2 направило в IBM петицию с просьбой открыть исходные коды этой операционной системы. Несмотря на то, что петицию подписали 11 613 человек, IBM её проигнорировала. В ноябре 2007 года сообщество направило повторную петицию, которую подписали 3744 человека. IBM ответила отказом 16 января 2008, мотивируя своё решение техническими, юридическими и бизнес-причинами.
- Одной из причин отказа IBM в открытии исходников является то, что много исходного кода принадлежит третьим фирмам. Многие из этих фирм уже не существуют, а также, часть кода до сих пор принадлежит Microsoft. Но мы все же надеемся на частичный

успех, т.к есть компоненты ОС, полностью принадлежащие IBM. Также следует отметить, что по некоторым слухам, IBM утеряла много исходных текстов

Почему нужен проект Open Source OS/2

Положение дел с OS/2

Что нам нравится в OS/2 и хотелось бы сохранить. Почему нужен проект Opensource OS/2.

- OS/2 это операционная система для IBM PC, наследующая и развивающая все хорошее, что было в DOS и вобравшая лучшие технологии фирмы IBM. Ее можно по праву назвать первой по многим параметрам. – Это первая 32-разрядная система для IBM PC, реализующая все возможности процессоров i386.
- Поддержка MVDM (Multiple Virtual DOS Machines) делала OS/2 лучшей системой для разработки DOS программ а также лучшей системой для DOS игр (около 40 настроек DOS сеансов позволяли до последнего винтика настроить работу DOS приложений); в сеансе DOS запускалась даже специальная версия Windows 3.1, работающая как DPMI-клиент (вместо VCPI). Программы DOS и Win 3.1 хорошо интегрированы с родными приложениями OS/2.
- Под OS/2 впервые для IBM PC были реализованы IFS – Installable File Systems. Множество файловых систем было написано под OS/2, В том числе, такие замечательные как Netdrive, поддерживающий плагины, позволяющие писать ФС в userspace (она появилась задолго до появления аналогов в Linux и FreeBSD). Документация для разработки IFS является свободно доступной, и в отличие от IFS toolkit-а от MS, не надо платить много $$$, поэтому возможность создания пользовательских ФС оказалась очень востребованной.
- Под OS/2 IBM создала объектную модель SOM/DSOM на основе CORBA, позволяющую делать DLL с наследованием. Т.е., классы в одной DLL наследуют классы в другой DLL, причем программисту не нужно иметь исходный код базового класса. Также, класс, написанный на одном языке программирования, может наследовать класс на другом языке. Вдобавок к этому, для описания интерфейсов классов используется язык CORBA IDL, так что можно например, писать ООП программы на Си, а интерфейсы кодировать на IDL.
- На основе объектной модели SOM была разработана оболочка рабочего стола WPS (Workplace Shell), которая объектно ориентированна снизу доверху. Все, что видит пользователь на экране – окна, иконки, сам рабочий стол – есть объекты.
    - Все объекты принадлежат к определенному классу, классы образуют дерево наследования.
    - На полную катушку используются возможности, предоставляемые SOM.
    - Набор свойств и методов каждого объекта строго соответствует интерфейсу класса и месту, занимаемому в иерархии наследования. Т.е., например, есть базовый класс папки или файла, у него есть определенный набор свойств и методов. От класса “папка” могут быть унаследованы классы типа “рабочий стол”, “список минимизированных окон”, “шаблоны”, тулбары, содержащие другие объекты в виде кнопок, и др., которые полностью повторяют поведение базового класса “папка”, плюс добавляют новые свойства и особенности поведения.
- Фирма IBM добавила в OS/2 еще одну свою технологию, присутствующую во всех своих ОС, от PC до мейнфреймов – командный язык REXX.
    - Все, от командной строки до объектов рабочего стола и плагинов файловых систем

и сетевого стека, имеет интерфейсы API для скриптов на языке REXX.
    - Мы любим REXX за простоту, мощность и расширяемость.
    - Расширения REXX пишутся как DLL специального вида.
- Ядро OS/2 это очень высокопроизводительное ядро, написанное на 40% на ассемблере.
    - Оно реализует большинство “фишек” архитектуры 386 процессора, позволяет на равных правах исполняться 16- и 32-битным OS/2 программам (не путать 16-битный защищенный режим. который используется в данном случае, с реальным 16-битным режимом DOS).
    - Оно является монолитным ядром, работающим в Ring0 и позволяющим подключать внешние драйверы и файловые системы.
    - Одной из первых, OS/2 в конце 1980-х гг поддерживала вытесняющую многозадачность и многонитевость.
        - Многонитевость в OS/2 одна из лучших на платформе i386.
        - Она позволяет создавать сложные многонитевые программы, состоящие из порядка тысячи нитей
            - например, сервер тренажеров для буровиков, разработанный Иосифом Шраго – по его словам, OS/2 это единственная система из доступных, позволяющая создавать подобные приложения. Windows и Linux этот тест не прошли.
            - Аналогичная история с эмулятором ЕС-ЭВМ АПК “Букет” - коммерческий продукт позволяющий достаточно полно эмулировать ЕС-овское железо на PC. По словам разработчика Вадима Урусова максимальная производительность у эмулятора написанного под OS/2, хотя существуют версии под WindowsNT-based и Linux.
        - Можно еще очень долго перечислять то, за что мы любим OS/2. Она нетребовательна к ресурсам, может работать на очень слабых машинах; имеет наиболее интуитивный и логичный пользовательский интерфейс. Но фирма IBM больше не хочет развивать ее, не желая отдавать исходные коды ОС сообществу.

osFree

как все начиналось

1. FreeOS - Первая попытка.
    - Наиболее заметным и, вероятно, самым первым проектом Open Source OS/2 был проект FreeOS. Он начался в 1999-м году под эгидой Daniel Caetano (автор PM Download Center). Последней попыткой представления данного проекта в интернете был сайт www.freeos.cjb.net. (На даный момент сайт недоступен.)
    - Однако, в рамках данного проекта не было написано ни одной строчки кода и все закончилось разговорами.
    - Данный проект был обречен на провал по ряду причин.
        - Это отсутствие 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.

2. osFree TPE - Dirty license.
    - В начале 2000-х из фирмы IBM “утекли” исходные коды ядра OS/2. на основе которых в 2002-м году был выпущен дистрибутив получивший название osFree TPE - Technology Preview Edition.
        - Данный дистрибутив вызвал море обсуждений у всего OS/2 community тех лет.
        - 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 тот, что он отказался от дальнейшей поддержки нелегального дистрибутива и предложил все-таки разрабатывать Open Source OS/2.
3. Проект osFree.
    - В 2001-м году, в результате истории с osFree TPE JMA был начат проект полностью Open Source OS/2 под названием osFree. (Чет с датами нестыковка. Пунктом выше 2002-ой год)
    - перерывы в разработке
        - С тех пор группа волонтеров начала разработку osFree в открытых исходниках. Они начали с утилит командной строки, таких как format, chkdsk, label, bldlevel etc. Следует отметить среди них Cristiano Guadagnino из Италии, Bartosz Tomasik из Польши и Michal Necasek из Чехии (?).
        - В 2002 JMA отстраняется от проекта, передав дальшейшее развитие Юрию Прокушеву. Затем, в связи с небольшой поддержкой проекта среди общественности, разработка затихает. Юрий Прокушев практически в одиночку собирает исходники в единую систему сборки, что-то фиксит, поддерживает веб-сайт, публикует статьи… Но разработка ведется очень медленно и притока новых сил нет. Авторы коммандлайновых утилит потихоньку “остывают” и перестают что-либо писать. С тех пор состав проекта полностью сменяется. Люди, выпускавшие osFree TPE, перестают участвовать в разработке.
        - дискуссия о выборе ядра
            - В 2002 году Юрий Прокушев на сайте ecomstation.ru публикует две статьи на тему “Open source и OS/2”. Он предлагал тогда дистанцироваться от выбора ядра и сосредоточиться на юзерленде. В первой статье он сделал обзор опенсоурсного софта под OS/2 и попросил помощи у сообщества. Вторая статья предлагала в качестве ядра выбрать ядро NT, а точнее, его Open Source аналога – ядро ReactOS, с возможностью опционального выкидывания Win32 подсистемы и замены dominant subsystem на OS/2. Обе статьи вызвали активную дискуссию в комментариях к статье, но дело так и не сдвинулось с мертвой точки.
            - Попытка использования ядра NT вызвала неприятие общественности. В результате, на www.osfree.org был вывешен опрос, какое ядро предпочтительно для реализации Open Source OS/2? Предлагались варианты ядер L4, Linux, Mach, ReactOS, Other. Этот опрос довольно долго висел на сайте, пока к концу 2005 г на сайт не наткнулся Samuel Falvo из США, знакомый с разработкой под L4. Он нам разрекламировал преимущества этого микроядра (насколько я помню, чет было не так. Обследование различных ядер на возможность реализации OS/2 API делал Посохов, если я правильно помню.).

- Прежде всего:
  - это его высокопроизводительность и универсальность и поддержка драйверов в userspace.
  - L4 это микроядро 2-го поколения, содержащее множество оптимизаций и архитектурно улучшенное, по сравнению с предшественниками.
    - Это включает в себя оптимизацию межнитевого обмена сообщениями и переключения контекста процессов.
    - Управление памяти было вынесено в userspace, и могло быть реализовано в произвольном виде. (для “плоской” модели памяти)
    - Операции над страницами памяти осуществлялись в реальном времени в userspace и можно было делать map/unmap страницы из обычных непривилегированных программ.
    - Это было безопасно, т.к. программы могли манипулировать только со страницами, отображенными в их собственное адресное пространство.
    - Обработка прерываний также была реализована в userspace. При возникновении прерывания, ядро направляло обработчику прерывания в userspace, сообщение от имени виртуального (несуществующего) треда. Таким образом делается виртуализация обработки прерываний. Тред получает сообщения о прерываниях, а ядро автоматически после этого делает EOI (сигнал End Of Interrupt – конца обработки прерывания). Механизм передачи сообщений микроядра достаточно высокопроизводительный для того чтобы сообщения посылались сразу по мере прихода сигнала прерывания.
- Samuel сначала начал писать реализацию механизма загрузки микроядра, (Прокушев первоначально отрицательно относился к GNU GRUB, используемому обычно для загрузки L4 – он хотел более традиционную для OS/2 последовательность загрузки, поэтому Samuel взялся за boot sequence). В качестве первой ФС им была выбрана довольно экзотическая LEAN FS, взятая из FreeDOS-32. Затем Samuel выпускает утилиту format для LEAN FS, это было довольно трудоемко и он этим довольно долго занимался. Затем Samuel куда-то пропадает, и новостей от него не слышно. Потом приходит новость, что он получил продвижение по службе, и больше мы его не видели.

Затем до конца 2006 года следует еще один большой перерыв в разработке osFree.

1. Начало разработки FreeLDR
   - В 2005 году после ухода Samuel Falvo к нам присоединяется Sascha Schmidt из Германии. Он написал парсер файла config.sys и вместе с Юрием Прокушевым продолжает работу над последовательностью загрузки. Они выделяют из GNU GRUB подпрограммы для чтения ФС и обработки протокола multiboot и комбинируют их с исходниками Freeldr (http://www.edm2.com/0705/freeldr/freeldr.html) – проекта Open Source замены загрузчика os2ldr, написанного Дэвидом Циммерли в 1999 году.
   - Sascha и Юрий тогда не смогли заставить работать эту комбинацию – они плохо

были знакомы с ассемблером.
- В ноябре 2006 года в проект приходит Валерий Седлецкий. Он был немного знаком с ассемблером и вскоре, наша версия freeldr, содержащая оригинальный код freeldr и подпрограммы из GRUB'a, портированные на 16-битный Си, заработала. Она успешно грузила микроядро L4Ka::Pistachio. Дальше было решено за нашей разработкой сохранить название FreeLDR, т.к. оно согласовалось с названием проекта osFree, являлось Free software и наследовало часть кода оригинального freeldr.
  - (Замечание. Это название совпадает с одноименным загрузчиком из ReactOS по чистой случайности. Просто, в OS/2 загрузчик называется os2ldr, а в NT ntldr, и вполне естественно для обеих проектов было название freeldr).
- Вскоре было решено отказаться от 16-битных подпрограмм доступа к ФС при помощи microfsd и портировать добавочный код из GRUB'a – 32-разрядные ФС из GRUB и сделать вокруг них 16-битные обертки реального режима для совместимости с OS/2-шными microfsd. Параллельно могли использоваться и 32-битный API защищенного режима, и 16-битный реального. 32-Битные ФС были отделены от бутлоадера, и сделана возможность раздельной компиляции загрузчика и ФС. В отличие от GRUB-а, ФС просто прописывались в конфигурационном файле. Далее было решено делать свой собственный загрузчик на основе GRUB и совмещать логику загрузки по протоколу multiboot и OS/2-like boot sequence. ФС-бэкенды, взятые из GRUB, грузились специальным менеджером бэкендов, который носил название pre-loader. Pre-loader грузит ФС-бэкенд, пытается смонтировать ФС и дальше работает через него, опционально делая 16-битный враппер, служа 16-битным microfsd, совместимым с OS/2-шным. Он также должен делать прозрачную декомпрессию и парсинг исполняемых файлов на лету (но последнее пока не реализовано).
- Кроме пре-лоадера есть также 32-битный multiboot loader, аналогичный GRUB-у. Но он абстрагирован от структуры ФС, форматов компрессии, исполняемых форматов и т.п. Кроме того, мультибут-лоадер может опционально заменяться на стандартный os2ldr, и тогда pre-loader для него служит microfsd.
- Для FreeLDR нами был разработан ФС-бэкенд для файловой системы HPFS и спортирован бэкенд для NTFS. Загрузка ядра Linux и chainloading были отделены от multiboot-загрузчика и реализованы в виде отдельных multiboot-ядер.
- Также, было реализовано специальное multiboot-ядро, служащее для загрузки классического ядра OS/2 и на его основе была реализована технология загрузки OS/2 с CD/флешки/HDD/флоппи/etc и технология, альтернативная memdisk и загрузочный CD с OS/2 на ее основе.

2. Старт реализации OS/2 API
   - с конца 2006 до конца 2008 мы занимались в основном, загрузчиком и связанными технологиями, а также, новой системой сборки исходников.
   - Затем в проект приходит Sven Eric Rosen из Швеции. Он написал загрузчик для исполняемых файлов формата LX.
   - Затем Юрий Прокушев на его основе добился загрузки простейшего EXE, грузящего единственную API-функцию для вызова сообщения из msg.dll и вызывающий его. Затем он сделал OS/2 API сервер, запускавшийся под тремя системами – OS/2, Windows и Linux. Он грузил маленький EXE, который писал на экран строку “I'm really small!”.
   - Затем тестовый пример постепенно усложнялся. Прокушев портировал этот сервер под L4 и добился запуска там. При этом уже было реализовано несколько API-функций.
   - Затем он попытался сделать минимальный интерактив – сделал минишелл, который

умел всего несколько команд: “cd”, “help” и старт внешней программы. На этом был небольшой затык и интерактив никак не хотел работать.
- Затем за дело взялся Валерий Седлецкий. Он временно закомментировал в коде лишний функционал, оставив только команду help и показ приглашения командной строки.
- После нескольких дней отладки интерактивный шелл заработал! Он использовал всего три API функции – для записи на stdout, чтения из stdin и выхода из программы. Шелл показывал командную строку, реагировал на команду help и нажатие Enter.
- Следует заметить, что мы используем не голое микроядро L4/Fiasco, а также набор сервисов к нему под названием l4env. В отличие от другого микроядра L4Ka::Pistachio, которое мы хотели использовать вначале, здесь очень большой набор сервисов.
    1. Это и библиотека thread для управления тредами,
    2. и библиотека semaphore для создания event семафоров
    3. и библиотека lock для мьютексов,
    4. и сервер консоли l4con, предоставляющий девять графических консолей с возможностью прокрутки назад и поддерживающий как графические, так и текстовые программы.
    5. Это и менеджер физической памяти dm_phys,
    6. и сервер loader, умеющий грузить исполняемые файлы формата ELF с поддержкой разделяемых библиотек,
    7. и провайдеры файлов,
    8. а также порт библиотеки языка Си uCLibc и многое другое.
    9. Также, существует подсистема l4vfs для поддержки иерархического namespace, в том числе, файлов и других объектов, такие как пайпы, сокеты или ключи регистри. Существует центральный name_server, который поддерживает корень Namespace, в который монтируются прочие серверы, а также поддерживающий разрешение имен в пары (Server_id, Local_object_id), т.е. аналог inode для файла. После такого разрешения имен клиент обращается напрямую к файловому серверу и получает доступ к файлу. Подсистема l4vfs поддерживает как обычные файлы, так и терминалы, служащие для реализации stdin/stdout/stderr для программ. Сервер терминалов прикрепляется к виртуальной консоли l4con и запись/чтение файла терминала работает как запись stdio потоков. Также, в сервере simple_file_server реализована простейшая работа с файлами, загруженными в память GRUB-ом в виде модулей. На данный момент доступна работа только с такими файлами.

Затем в разработке OS/2 Server'a следует перерыв на полгода и работа с FreeLDR.

- текущее состояние
    - На данный момент времени реализованы порядка 50-ти функций API ядра OS/2 (CP/DOS). Всего ядро OS/2 экспортирует около 250-300 32-битных функций и примерно столько же, или чуть меньше, 16-битных. Многие функции дублируются в 32-битном и в 16-битном варианте, т.е., ядро содержит по два ординала для многих функций. Дело в том, что OS/2 славится своей совместимостью со старым софтом, и пользователи OS/2 этим очень дорожат.
    - Реально чисто 16-битных программ осталось очень мало, но среди них есть полезные программы, например командный процессор cmd.exe и хекс-вьюер hiew (но для них, как и многим другим, имеется достойная замена – командный

процессор 4os2.exe и вьюер бинарных файлов biew, оба Opensource, между прочим). Поэтому несмотря на здоровый консерватизм пользователей OS/2, поддержку чисто 16-битных программ следует считать не столь важной, тем более, если мы хотим, чтобы наша ОС работала на x86-64, от 16-битных программ все же следует отказаться.

- То, что надо отказаться от чисто 16-битных программ, это верно, но и среди 32-битных программ много таких, которые содержат маленькие 16-битные кусочки – thunk'и в 16-битные API, т.е., переходники на ассемблере, служащие для преобразования FLAT указателей в FAR16 указатели для вызовов 16-битных API. Дело в том, что многие 16-битные API до сих пор не имеют 32-битных аналогов, и тем не менее, они до сих пор востребованы. – Например, Kbd/Mou/Vio API соответственно, Keyboard/Mouse/Video API, т.е., консольное API до сих пор 16-разрядное. Фирма IBM сделала эти API 32-битными в OS/2 Warp (PowerPC Edition), но в обычной Intel версии все осталось по-старому.). Все эти API до сих пор нужны, и широко используются.
- Для osFree мы поступили аналогично OS/2 для PowerPC – частично реализовали 32-битное консольное API – оно аналогично 16-битному, и отличается разрядностью аргументов и 32-битной конвенцией вызова – thunk-и больше не нужны.
- Т.к. одной из целей проекта osFree является бинарная совместимость с OS/2 (Intel), то нам необходимо каким-то образом грузить программы, содержащие 16-битные фрагменты.
- Нами было решено отказаться от чисто 16-битных программ, но полностью поддерживать 32-битные программы с thunk-ами. Для этого предполагается на этапе загрузки исполняемого файла в память “выкусывать” 16-битные фрагменты на лету и вызовы 16-битных API подменять на вызовы их 32-битных аналогов.
- Вместо всех 16-битных функций в “ядре” osFree реализуются только 32-битные API. В результате, программа, загруженная в память, чисто 32-битная и дальше, в принципе, ее можно в таком виде сохранить на диск (в каком-то смысле, аналог JIT-компиляции в Java).

1. перспективы
   - Описание различных personality-neutral серверов и серверов OS/2 personality.
     - L4 система (для случая L4/Fiasco с l4env) грузится загрузчиком GRUB или любым аналогичным, multiboot-совместимым загрузчиком.
       - GRUB загружает вторичный загрузчик bootstrap, служащий multiboot-ядром, и набор модулей.
       - Первым модулем идет микроядро fiasco,
       - вторым sigma0 – корневой пейджер, который в начале старта системы владеет всей физпамятью, и потом отдает память другим серверам по запросу.
       - Третий модуль – roottask – корневой сервер, который отвечает за дальнейшую инициализацию системы и запускает остальные серверы.
       - Далее идут дополнительные модули (запускаемые roottask).
     - У roottask есть набор параметров командной строки. Например, “task modname 'simple_file_server' attached 44 modules” означает, что после модуля 'simple_file_server' идут 44 модуля, которые не запускаются roottask-ом, а обрабатываются сервером 'simple_file_server'. В данном случае, 'simple_file_server' их экспортирует как файловую систему.
     - Параметр 'task modname 'loader' allow_cli' в данном случае означает право для сервера loader исполнять инструкцию cli, это необходимо для того, чтобы roottask мог отдать ему доступ к I/O портам, которые тот

запросит. В свою очередь, loader дает возможность запускаемым через него программам получить порты ввода-вывода. Например, для l4linux они нужны, а также для run, чтобы получить возможность перезагружать систему. (Иначе, перезагрузка через кнопку '^' в run не работает).

- Сервер log представляет собой сервер логов. Он собирает отладочные сообщения от всех других серверов, и отправляет их в компорт, и их можно просматривать через любой эмулятор терминала. Также, имеется возможность для ограниченного числа поддерживаемых сетевых карточек, подключаться к логсерверу через клиент telnet. Кроме того, существует logcon – версия log сервера, дублирующая все сообщения на экран.
- Events сервер предназначен для рассылки нотификаций о различных событиях (типа завершения программы) серверам, подписанным на эту рассылку. Например, task сервер (simple_ts) при получении такой нотификации деаллоцирует task number для завершившейся задачи, и делает его доступным для использования другой задачей.
- Names сервер является сервером службы имен. Каждый сервер в системе, предоставляющий какой-либо сервис другим программам, регистрирует на names имя сервиса. Тогда любой клиент, сделав запрос по этому имени у names, может получить thread ID нити сервера, предоставляющей сервис, и уже дальше обращаться к этой нити.
- Dm_phys это сервер физической памяти. Он забирает память у sigma0 и отдает ее по запросу другим серверам, реализуя для этого абстракцию dataspace.
- L4io предоставляет интерфейс другим серверам к портам ввода-вывода и прерываниям, а также PCI configuration space
- Rtc это сервер часов реального времени. Используется для получения и установки времени системных часов. Например, нужен для l4linux.
- L4con это сервер консоли. Реализует функциональность нескольких графических консолей с возможностью прокрутки назад.
- Simple_ts это task server. В L4 задачи, которые могут быть запущены, представляют ограниченный ресурс. При запуске задачи на task server'е аллоцируется task ID, при завершении он возвращается в пул доступных tasks.
- Дальше идут серверы l4vfs. L4vfs это система поддержки иерархического пространства имен, для единого именования объектов типа файлов, терминалов, устройств, пайпов, ключей регистри и т.п. в виде единого дерева имен
    - Name_server это главный сервер l4vfs, который реализует корневое namespace. Он позволяет монтироваться другим серверам в корневое namespace в виде ветки. Также служит для разрешения pathnames в пары (Volume_ID, Local_Object_ID).
    - Vioterm это переименованный term_con – сервер, реализующий терминалы, поддерживающие ANSI последовательности, и доступные как специальные файлы. Позволяет ввести поддержку stdin/stdout/stderr.
    - Simple_file_server это простейший файловый сервер, который предоставляет доступ к модулям, загруженным GRUB-ом в память, в виде обычных файлов.
    - Fstab это простейшая программка, монтирующая на name_server

набор других серверов в виде веток.
    - Fprov_proxy это спец. прокси, предоставляющий доступ к файлам, доступным через l4vfs, как dataspaces.
    - Т.е. это специальный враппер для доступа к файлам через другой интерфейс.
    - Loader это runtime загрузчик исполняемых файлов формата ELF. Он парсит исполняемый файл и загружает его, и все необходимые для него shared libs.
- Дальше идут серверы собственно OS/2 personality.
    - Os2fs это файловый сервер. Реализует файловые API и врапперы для l4vfs для поддержки семантики файловой системы OS/2. Т.е., вводятся буквы дисков, файловый доступ к пайпам в ветке \pipe\.., девайсам в \dev\.. и т.п. Также, (пока не реализовано, но планируется) os2fs должен в будущем поддерживать файловые системы стандарта IFS в виде плагинов-DLL формата LX. В качестве первой файловой системы предполагается спортировать ramfs.ifs – IFS-based ramdisk. В отличие от обычных Intel OS/2 IFS, IFS в osFree предполагаются как userspace библиотеки.
    - Os2exec сервер это сервер OS/2 personality, предназначенный для загрузки исполняемых файлов и DLL различных форматов. Также реализуются различные API для работы с исполняемыми модулями. Поддержка исполняемых форматов отделена от самогО сервера, и выделена в отдельную библиотеку. Т.е., есть например, lx.ixf для поддержки исполняемых файлов формата LX. Предполагается написать аналогичный драйвер elf.ixf для формата ELF – аналогично OS/2 Warp (PowerPC edition), будет ализована поддержка EXE и DLL формата ELF. Т.е., как это было сделано в IBM, предполагается сделать поддержку специальных секций .imports для поддержки импортов, .exports для поддержки экспортов и .resource для поддержки загружаемых ресурсов в DLL файлах. Т.е., формат ELF достаточно расширяем и универсален, что можно сделать поддержку не только shared objects, как в UNIX-based системах, но и обычных библиотек с стандартной DLL семантикой. Формат ELF нам может пригодиться также как средство поддержки исполняемых файлов для платформ, отличных от i386. Как известно, формат LX заточен под платформу Intel и хотя он хорошо оптимизирован под ее возможности, но для других платформ его использование может быть проблематичным. Формат ELF напротив, абстрагирован от платформы, поддерживает как 32-битный, так и 64-битный вариант, и на семантику секций ELF-файла не накладывается никаких ограничений. Можно вводить и свои специальные типы секций.
    - Os2srv это главный сервер OS/2 personality. Он координирует запуск других серверов OS/2 personality, при старте системы обрабатывает файл config.sys и стартует остальные серверы. При старте задаются пути поиска программ и библиотек, запускаются другие серверы через оператор runserver=, затем запускаются OS/2 программы (не серверы) через run= (асинхронно) и call= (синхронно) и, наконец, запускается программа, прописанная как protshell=.
        - Как и в обычной OS/2, protshell это программа, запускающая

оболочку, стартующую пользовательский интерфейс системы. В нашем случае это программа minicmd.exe. Это минишелл, запускающий командную строку. Поддерживается запуск внешних программ, а также навигация по файловой системе – команды cd и dir. Работает простейшая подсказка командами help/?. Задачей minicmd.exe является минимальный тестовый комплект функций пользовательской программы для тестирования и демонстрации работающих API, а также запуск других внешних тестовых программ. По мере совершенствования серверов и добавления новых API в minicmd и другие тестовые программы добавляются новые возможности для их тестирования.

- Os2app это не сервер, а специальное приложение l4env, которое представляет собой своеобразный враппер для OS/2 приложения. Каждое OS/2 приложение запускается в контексте своей копии os2app. При запуске minicmd.exe, например, стартуется os2app, которому передаются параметры командной строки: '–stdin /dev/vc0 –stdout /dev/vc0 –stderr /dev/vc0 c:\minicmd.exe'. – Таким образом, в комстроке передается, к какой консоли цепляться в качестве stdin/stdout/stderr и какое OS/2 приложение стартовать. Os2app содержит стартап l4env приложения и добавочный код инициализации, нужный для OS/2 приложения. Он подготавливает нужную раскладку адресного пространства, характерную для OS/2 приложений: адреса от 0 до 0x10000 (64 KB) зарезервированы, код приложения начинается с адреса 0x10000, адреса от 0x10000 до 0x04000000 (64 MB) зарезервированы под приватную область приложения. Адреса от 0xc0000000 (3 GB) и выше зарезервированы под микроядро. Ниже 3 GB идет шаренная область, в нее грузятся библиотеки DLL. Затем os2app грузит приложение и все нужные DLL, подготавливает регистры и делает jmp на точку входа в приложение.

- Open source и OS/2, часть 1
- Open source и OS/2, часть 2