Павел Макаров (makarov@minix3.ru)

ВТОРАЯ ЧАСТЬ «МАРЛЕЗОНСКОГО БАЛЕТА»

Дискуссия Таненбаума и Торвальдса: часть II

> *- Если бы сейчас была дискуссия, - начала женщина, волнуясь и загораясь румянцем, - я бы доказала Петру Александровичу...*
> *- Виноват, вы не сию минуту хотите открыть эту дискуссию? - вежливо спросил Филипп Филиппович.*
>
> *М.А. Булгаков «Собачье сердце»*

Я люблю диспуты и дискуссии. Ибо, как известно, именно в спорах и рождается истина. Правда, при одном условии – если стороны действительно стремятся не только довести свою точку зрения до собеседника, но и как минимум услышать друг друга. А вот с этим, как показывает жизнь, сложнее.

Видимо, именно эта причина стала поводом для появления открытого обращения Эндрю Таненбаума, которое я не преминул перевести на русский язык по причине излишней, чуть ли не болезненной (на мой личный взгляд) популярности в нашем отечестве ссылок на знаменитую дискуссию по поводу микро- и монолитных ядер операционных систем. Тем более, что самые ярые участники русскоязычной части «диспута», как показывает практика, даже и не пытались познакомиться с оригинальным материалом.

Попытаюсь со своей стороны этим переводом упростить задачу членам «дискуссионного клуба». И, по возможности, доставить несколько приятных минут тем, кто действительно интересуется проблемой, ибо Таненбаум как никто умеет (что большая редкость, увы, по нынешним временам) кратко и ясно излагать свои мысли. А этому у Мастера не грех и поучиться.

Оригинальный материал находится по адресу: http://www.cs.vu.nl/~ast/reliable-os/

Предисловие

Всё выглядит так, будто бы вернулись времена дебатов по поводу микроядра. Прежде, чем перейти к собственно технической аргументации, я хочу сказать пару вещей. Многие люди говорили или подразумевали, что Линус и я являемся врагами или вроде того. Это всеобщее заблуждение. Я встречал его только однажды. Он – парень «приятный во всех отношениях» и очень умный. Мы с ним можем быть несогласны в некоторых технических вопросах, но это отнюдь не делает нас врагами. Не следует путать несогласие по поводу идей и личную вражду. Я ничего не имею против Линуса и отношусь с большим уважением к тому, что он совершил.

На тот маловероятный случай, если кто-нибудь пропустил это событие, сообщаю: пару лет назад Microsoft заплатила парню по имени Кен Браун (Ken Brown) за написание книги, в которой говорилось, что Линус выкрал код Linux из моей операционной системы MINIX 1. Я очень энергично опровергал

это обвинение с тем, чтобы не дать запятнать доброе имя Линуса. Я могу быть не совсем согласен с конструкцией Linux, но это его детище, а не моё, и я был очень недоволен, когда Браун сказал, будто бы он (Линус) скопировал его (Linux) с моей ОС.

Прежде, чем перейти к деталям, я хочу также заметить, что мой интерес по существу лежит отнюдь не в области микроядер. Мой интерес – в построении очень надёжных (и безопасных) операционных систем, и я думаю, что использование микроядер является хорошим способом для достижения этой цели. Теперь излагаю подробнее.

Изрекающие уста

Считается, что на здании Военно-воздушной Академии в Колорадо начертан такой девиз: «***Be sure brain is in gear before engaging mouth***» (*что, видимо, лучше интерпретировать как аналог отечественного: «Думай, прежде чем сказать», а не переводить почти дословно: «Убедись, что врубил мозги прежде, чем разевать рот», ибо грубовато-прямолинейная форма выражения своих мыслей уместна скорее в среде доблестных американских военных, нежели в посвящённой техническим проблемам академической публикации - прим. переводчика*). Я не знаю, правда это или нет, но думаю, что в любом случае идея эта хороша.

С годами на форумах, таких, как Slashdot, не прекращаются дебаты о том, что микроядра медленные, как трудно программировать микроядра, что они не используются в коммерческих ОС и много всякой другой чепухи. Фактически вся эта корреспонденция исходит от людей, которые даже представления не имеют о том, что такое микроядро и что оно может делать. Я думаю, что уровень дискуссии здорово бы повысился, если б люди, отправляющие эту корреспонденцию, сначала бы попробовали поработать с микроядерной ОС и только потом бы отправляли письма вроде «Я поработал с микроядерной ОС и сразу же обнаружил X, Y и Z». Верилось бы лучше.

Простейший способ попытаться сделать это – загрузить себе MINIX 3 и поработать с ней. Она свободна и имеет открытый код (под лицензией BSD), так что, получив образ CD, вы можете записать его на CD-ROM, загрузиться прямо с него и войти в систему как root. Однако для того, чтобы сделать что-нибудь полезное, вам лучше выделить логический раздел на жёстком диске (1ГБ будет достаточно) и установить MINIX 3 в него. Только сначала распечатайте и прочитайте, пожалуйста, руководство по установке. Сама установка займёт около 10 минут. Затем установите с компакт-диска все пакеты, как описано в руководстве по установке. Теперь запустите X-сервер и вы сможете получить некоторый реальный опыт работы с микроядерной ОС. Попробуйте пересобрать ОС целиком, как описано в руководстве. Полная сборка - ядро плюс драйверы и все серверы в режиме пользователя (всего 125 компиляций) - займёт 5-10 секунд.

Примите, пожалуйста, к сведению, что MINIX 3 отнюдь **не есть** та самая ОС MINIX вашего дедушки. MINIX 1 была написана как учебное пособие; она по-прежнему широко используется в этом качестве в университетах. Эл Вудхалл (Al Woodhull) и я даже написали учебник о ней. ОС

MINIX 3, естественно, основана на ней, но она (MINIX 3) изначально создавалась как высоконадёжная, самовосстанавливающаяся и не раздутая операционная система, принципиально применимая в таких проектах, как $100 ноутбук для детей из стран третьего мира и, возможно, во встраиваемых системах. MINIX 1 и MINIX 3 соотносятся так же, как Windows 3.1 и Windows XP: совпадает только первая часть имён. Вот почему стоит попробовать поработать с MINIX 3, даже если вы использовали MINIX 1 во время учёбы в колледже; вы будете приятно удивлены. MINIX 3 - это хоть и очень маленькая, но полнофункциональная Unix-совместимая операционная система с X, bash, pdksh, zsh, cc, gcc, perl, python, awk, emacs, vi, pine, ssh, ftp, инструментальными средствами GNU (GNU tools) и с более, чем 400 других программ. Всё это построено на крошечном микроядре и доступно прямо сейчас.

Так что, **пожалуйста**, не нужно больше комментариев типа «Если Таненбаум думает, что микроядра такие крутые, так что же он не сделает микроядерную ОС?» Он сделал. (На самом деле, сделали один из его студентов и двое из его программистов). Она, конечно же, пока не такая совершенная и зрелая, как Linux или BSD, но она однозначно доказывает, что создание надёжного, самовосстанавливающегося мультисерверного клона UNIX в пространстве пользователя, построенного на основе маленького и лёгкого для понимания микроядра, возможно. Только не путайте недостаточную зрелость конкретной ОС (мы работаем над ней чуть более года втроём) с различными аспектами применения микроядер в целом. Мы можем и будем со временем расширять функциональность MINIX 3, а также переносить в неё много прикладного ПО (и в этом ваша помощь очень желательна). С конца октября 2005 года, когда стартовал проект MINIX 3, уже зарегистрировано более 400,000 посещений сайта проекта. Попробуйте MINIX 3 сами и посмотрите, что получится.

Публикация

Недавно мой аспирант Йоррит Хердер, мой коллега Герберт Бос и я написали статью, озаглавленную «Можем ли мы делать операционные системы надёжными и безопасными?» и послали её в IEEE Computer magazine, крупнейший журнал компьютерного сообщества IEEE. Она была принята и опубликована в майском выпуске 2006 года. В этой статье мы утверждаем, что для большинства пользователей надёжность ОС более важна, чем её производительность, и обсуждаем 4 современных исследовательских проекта, направленных на улучшение надёжности операционных систем. Три из них используют микроядра. IEEE разместил нашу публикацию на своём интернет-сайте. Затем кто-то сделал ссылку на неё на Slashdot, возобновив, таким образом, древнюю дискуссию о микроядрах и монолитных системах. И хотя я с удовольствием сознаюсь в соавторстве этой публикации, но я совсем не ожидал, что она заново перезапустит дискуссию на тему «Linux отжил своё».

Линус после этого ответил и это выглядит так, будто бы мы затеяли небольшую новую дискуссию. Ну что же, хорошо, но только просьба ко всем: давайте обсуждать именно технические моменты.

Аргументация Линуса

Основная позиция Линуса заключается в том, что микроядра требуют распределённых алгоритмов и вообще это очень трудно (*самый мягкий вариант перевода фрагмента* they are nasty – *прим переводчика*). Я согласен с тем, что распределённые алгоритмы – та ещё штуковина, хотя совместно с Маартеном ван Стином (Maarten van Steen) я написал книгу на эту тему. За последние десять лет я также разработал и создал две распределённые операционные системы, Amoeba (для LAN) and Globe (для WAN). Проблема с распределёнными алгоритмами заключается в отсутствии общей синхронизации процессов по времени (lack of a common time reference) вкупе с вероятностью потери сообщений и неопределённостью в том, заблокирован ли удалённый процесс или он просто проистекает слишком медленно. Однако ни одна их этих проблем не имеет никакого отношения к микроядерным ОС, работающим на одной единственной машине. Так что, хоть я и соглашаюсь с Линусом в том, что распределённые алгоритмы очень непросты, однако всё это не имеет никакого отношения к данной дискуссии.

Наоборот, хотя в нашей ОС большая часть её компонентов в пространстве пользователя и является драйверами, однако с серверами они взаимодействуют очень просто. Все драйверы символьных устройств (они читают и записывают потоки байтов) работают по практически такому же протоколу, как и все драйверы устройств с блочным доступом (которые читают и записывают блоки). Количество серверов в пространстве пользователя довольно мало: файловый сервер, сервер процессов, сетевой сервер, сервер реинкарнации, хранилище данных (data store) и ещё немного. Каждый из них имеет чётко определённую задачу и чётко определённую процедуру взаимодействия с остальными частями системы. Хранилище данных, например, отвечает за службу рассылки по списку и, соответственно, подписки не неё (publish/subscribe service), обеспечивая при необходимости слабую связь между серверами. Количество серверов, вероятно, не сильно увеличится и в будущем. Так что сложность нашей операционной системы вполне постижима и контролируема (manageable). И это отнюдь не умозрительное заключение. В конце концов, мы её уже создали. Установите MINIX 3 и потестируйте её самостоятельно.

Линус также утверждал, что разделяемые структуры данных – это хорошо. Здесь мы с ним во мнениях расходимся. Если вы когда-либо слушали курс по операционным системам, то без сомнения помните, как много времени в курсе (и места в учебнике) уделялось семафорам и синхронизации взаимодействующих процессов. Если два или более процессов могут получить доступ к одной и той же структуре данных, то вам следует быть очень-очень аккуратным, чтобы не «подвесить» себя самого. Организовать такую процедуру доступа к общему ресурсу чрезвычайно трудно даже со всеми семафорами, мониторами, мьютексами и прочими ухищрениями.

Моя точка зрения заключается в том, что следует избегать разделяемые структуры данных везде, где только можно. Системы должны строиться из наименьших модулей, полностью скрывающих свои внутренние структуры данных от кого бы то ни было. Они (модули) должны иметь чётко определённые «тонкие» интерфейсы, которыми остальные модули могут

пользоваться по ходу дела. Это как раз то главное, что декларируется в объектно-ориентированном подходе к программированию: закрытие внешнего доступа к данным (hiding information), а не совместное (sharing) их использование. Я думаю, что именно закрытие внешнего доступа к данным в соответствии с подходом Дэвида Парнаса (Dave Parnas - *разработчик концепции модульного проектирования, лежащей в основе современного объектно-ориентированного программирования – прим. переводчика*) – вот действительно хорошая идея. Это означает, что до тех пор, пока вы сохраняете неизменным межмодульный интерфейс, вы можете свободно менять внутренние структуры данных, алгоритмы функционирования и саму конструкцию любого модуля без влияния на корректность работы системы в целом. Об этом говорится в любом курсе по проектированию программного обеспечения. Линус же фактически утверждает, что работы в области объектно-ориентированного программирования последние 20 лет шли по неверному пути. Я с этим не согласен.

И если уж однажды вы решили, что каждый модуль должен держать свои грязные лапки подальше от структур данных других модулей, то следующим логическим шагом будет помещение каждого модуля в отдельное адресное пространство и установка аппаратного диспетчера памяти (MMU) для проведения этого правила в жизнь. Применительно к операционной системе вы берёте микроядро и набор процессов из пространства пользователя, взаимодействующих с использованием механизма сообщений и чётко определённых интерфейсов и протоколов. Делается это для большей ясности и лучшей ремонтопригодности всей конструкции ОС. Естественно, Линус рассуждает на основе его опыта работы с монолитным ядром и, вероятно, на меньшей степени его знакомства с микроядрами и распределёнными системами. Мой же собственный опыт основан на конструировании, разработке и реализации мною самим множества таких операционных систем. Отсюда у нас с ним разные точки зрения на то, что является трудным, а что – нет.

Для знакомства с другой точкой зрения на проблематику надёжных операционных систем посмотрите работу Джонатана Шапиро (Jonathan Shapiro), озаглавленную Debunking Linus's Latest (*что переводится примерно как «Развенчание последних утверждений Линуса» - прим. переводчика*).

Годятся ли микроядра для реальной жизни?

Если кратко, то: да. На Slashdot существует бесконечное количество комментариев примерно следующего содержания: «Если микроядра такие крутые, то почему они не используются повсеместно?» На самом деле они используются. Кроме MINIX 3, существуют следующие микроядерные ОС:

- QNX
- Integrity
- PikeOS
- Symbian
- L4Linux
- Singularity
- K42
- Mac OS X

- HURD
- Coyotos

QNX широко используется в реальных коммерческих системах. Например, Cisco использует её в старшей модели маршрутизатора, а Cisco, смею вас заверить, **ОЧЕНЬ** заботится о производительности.

На военном и аэрокосмическом рынках, где надёжность является наиболее критичным параметром, одной из самых «продвинутых» операционных систем является микроядерная ОС Integrity, разработанная компанией Green Hill.

PikeOS - это пример ещё одной микроядерной системы реального времени, широко используемой в оборонных, аэрокосмических, автомобильных и промышленных системах.

Следующей популярной микроядерной ОС, используемой прежде всего в сотовых телефонах, является Symbian. Однако она, строго говоря, не чисто микроядерная, а представляет собой некий гибрид с драйверами в ядре, но с файловой системой и с модулями поддержки сетевых взаимодействий и телефонии, расположенными в пространстве пользователя.

Я мог бы продолжать и продолжать, но и так уже ясно, что для приложений, где надёжность и безопасность являются критически важными, разработчики неоднократно выбирали микроядерные ОС. И хотя лично Линус может не интересоваться встраиваемыми системами реального времени (где производительность, надёжность и безопасность являются параметрами первостепенной важности), этот рынок огромен и множество действующих на нём компаний считают, что использование микроядер является способом обеспечения и производительности, и надёжности, и безопасности.

Бросив взгляд на мир персональных компьютеров, мы обнаружим там ОС L4Linux, которая была написана группой Германа Хартига (Hermann Härtig) из Технического Университета Дрездена (Technical University of Dresden - TUD). Эта ОС запускает весь Linux в пространстве пользователя поверх микроядра L4 с потерей всего лишь пары процентов производительности. Однако использование микроядра позволило людям из TUD построить на базе микроядра L4 новые системы, такие, как DROPS (система реального времени) и NIZZA (безопасность), имея при этом для обеспечения новых возможностей доступ ко всей функциональности Linux без его модификации. При таком подходе они могут экспериментировать с новой функциональностью, но при этом по-прежнему имеют возможность запускать уже существующие программы. Для исследований в области операционных систем микроядро L4 используют и другие группы, такие, как NICTA (National ICT Australia – *Австралийский национальный центр исследований в области информационных и коммуникационных технологий – прим. переводчика*) с проектом Wombat - паравиртуализованного Linux, спроектированного для поддержки работы существующих приложений на встроенных системах. Ещё одной ОС на базе L4 является TUD-OS и таких систем на самом деле много.

Microsoft тоже интересуется микроядрами. Она как никто другой отчётливо понимает все проблемы сопровождения монолитных ядер. Windows NT 3.1 была нерешительной попыткой создания микроядерной системы, однако попытка эта была реализована не совсем корректно. Да и производительность ОС на оборудовании начала 90^{x} годов также была недостаточно хороша, поэтому Microsoft на время отказалась от этой идеи. Но недавно она (Microsoft) попыталась сделать это снова, но уже на новом оборудовании, что привело к появлению ОС Singularity. Сегодня, как я вижу, многие полагают, что если Microsoft делала это, то она явно сошла с ума. Но руководившие проектом Singularity Гален Хант (Galen Hunt) и Джим Ларус (Jim Larus) – очень умные ребята и они-то как раз хорошо понимают, какую мешанину на самом деле представляет собой ОС Windows и насколько компании Microsoft необходим совершенно новый подход для её дальнейшего развития. Даже работающие над Vista люди видят, что у них имеются серьёзные проблемы. Поэтому-то они и переносят драйверы в пространство пользователя, то есть делают именно то, что я и рекомендую.

Около 10 лет назад компания IBM начала «с нуля» разрабатывать новую высокопроизводительную ОС для своих очень больших заказчиков. Объявленной целью проекта было перемещение системной функциональности из ядра в серверы и прикладные программы аналогично тому, как это делается в микроядре. Эта система, называемая K42, в настоящее время развёрнута в DoE (*Department of Energy – Министерство энергетики США – прим. переводчика*) и где-то ещё.

Mac OS X также является разновидностью микроядерной ОС. Изнутри она представляет собой Berkeley UNIX, функционирующий поверх модифицированной версии микроядра Mach. Однако, поскольку всё это запускается в режиме ядра (для того, чтобы выбрать всю возможную производительность), Mac OS X не является истинно микроядерной. Тем не менее, поскольку Университет Карнеги-Меллона (Carnegie Mellon University) уже много лет назад запускал Berkeley UNIX в пространстве пользователя поверх микроядра Mach, то это, вероятно, могло бы быть сделано снова, хотя и с небольшой потерей производительности, как в случае с L4Linux. В частности, существует проект Darbat по портированию кода Apple BSD (Darwin) на L4 с тем, чтобы сделать её истинно микроядерной ОС.

GNU HURD, несмотря на громкие обещания и скромные деяния, тем не менее, существует и также построена на микроядре. Даже на двух. Первая версия была построена на Mach, а вторая – на L4. Третья версия будет, вероятно, построена на ещё одном микроядре, Coyotos. HURD была разработана Ричардом Столлменом (Richard Stallman), автором emacs, gcc и множества других широко используемых программ, а также создателем лицензии GPL и лауреатом престижной награды MacArthur 'Genius' Award.

Ещё одна разрабатываемая микроядерная система называется Coyotos и является преемником EROS. Хотя проект Coyotos акцентирует внимание больше на безопасности, чем на надёжности, тем не менее разработчики пытаются обеспечить обе эти характеристики ОС в связи с очевидностью того, что раздутые ядра могут привести к проблемам на обоих фронтах.

И это я ещё не касался гипервизоров (hypervisors), таких, как Xen и Trango, которые хотя и очень сильно отличаются от микроядер, но тоже имеют крошечный код, работающий в режиме ядра, что, по моему глубочайшему убеждению, и является ключом к построению надёжных и безопасных систем.

Несмотря на то, что MINIX 3, QNX, Integrity, PikeOS, Symbian, L4Linux, Singularity, K42, HURD, Coyotos и другие ОС представляют собой достаточно «разношёрстную компанию», достаточно очевидно, что я не одинок в попытке использовать микроядра. И если вы удивлены тем, что микроядра всё ещё не используются широко, то я отвечу, что это просто большая инерция системы. Почему Linux или Mac OS X до сих пор не заменили Windows? По той же причине: большая инерция системы. Большинство автомобилей в Бразилии могут работать на местном топливе – этаноле, поэтому Бразилия использует относительно немного нефти для автомобильного топлива. Так почему же США не делают этого и не уменьшают свою зависимость от нестабильного Ближнего Востока? По той же причине: большая инерция системы. Подвигнуть людей на изменения очень тяжело, даже в случае вполне очевидной практической выгоды.

Что я пытаюсь доказать?

На самом деле и MINIX 3, и моё исследование посвящены вовсе **НЕ** микроядрам, а построению высоконадёжных, самовосстанавливающихся операционных систем. Я буду считать, что эта моя работа закончена, только тогда, когда никто больше не будет производить персональные компьютеры с кнопкой RESET. У телевизоров этой кнопки нет. У музыкальных центров - тоже нет. И у автомобилей нет этой кнопки. Они битком набиты разным программным обеспечением, однако не нуждаются в таких кнопках. Компьютерам же кнопка RESET нужна только потому, что их программное обеспечение «падает» слишком часто. Я знаю, конечно, что компьютерное ПО отличается от автомобильного, но люди-то просто хотят использовать и то, и другое и совершенно не хотят выслушивать лекции о том, почему они могут рассчитывать на нормально функционирующие автомобили и не могут рассчитывать на нормально функционирующие компьютеры. Я хочу построить операционную систему, которая будет иметь характерный временной интервал между отказами, намного больший, чем время жизни компьютера, так чтобы средний пользователь никогда бы и не познакомился с отказом. У MINIX 3 есть много специфических свойств, обеспечивающих её высокую надёжность. Несмотря на то, что мы пока не реализовали все эти особенности (например, виртуальная память пока только в плане разработок на конец этого года), я думаю, что высокая надёжность – это сегодня самый большой вызов разработчикам операционных систем. Обычный пользователь не должен заботиться по поводу обновлений ПО или выжимания последних капель производительности из компьютера, однако должен беспокоиться о том, чтобы его компьютер безупречно работал все 100% времени и никогда не отказывал. Поинтересуйтесь, например, мнением вашей бабушки по этому поводу.

Чем же на самом деле микроядра в этом смысле могут быть полезны? А тем, что они делают возможным создание самовосстанавливающихся систем.

Это и есть то, что меня заботит, и то, чему посвящена моя работа. Перемещение большей части ОС в группу пользовательских процессов – по одному для каждого драйвера и различных серверов – отнюдь не уменьшает количество ошибок в коде, однако существенно уменьшает возможность любой ошибки вызвать серьёзное повреждение, и к тому же сокращает размер надёжной части кода (trusted computing base). В нашей конструкции в случае отказа большинства драйверов сервер реинкарнации может перезапустить их свежую копию, а в дополнение может сохранить образ ядра «умершего» драйвера в целях отладки, зафиксировать это событие и послать e-mail администратору или разработчику и т.п. При этом система продолжает работать и только в очень крайнем случае может быть аккуратно остановлена без снижения эффективности работы или потери данных. Некоторые компоненты, такие, как собственно сервер реинкарнации, файловый сервер и сервер процессов, являются критичными и их неисправность, естественно, приведёт к отказу системы. Но нет никаких причин позволять «обрушивать» систему неисправным аудио-, принтерным или сканерным драйверам. Они просто должны быть перезапущены и работа продолжится. Это и есть наша цель: создание систем, которые могут обнаруживать свои собственные отказы и восстанавливаться после них. Вы легко можете сделать это в микроядерных системах. Сделать это в системах с монолитным ядром гораздо сложнее, хотя исследователи из Университета Вашингтона (University of Washington) выполнили неплохую работу с Nooks, да и группа из Университета Карлсруэ (University of Karlsruhe) также сделала интересную работу на основе технологии виртуальных машин.

Домашнее задание

Прежде, чем разглагольствовать о том, что микроядра могут делать, а что – не могут, лучше поработайте с MINIX 3 и станьте информированным оратором. Это повысит вашу компетентность. Для того, чтобы получше познакомиться с конструкцией MINIX 3, почитайте статью в майском номере IEEE Computer, эту статью по модульному программированию, которая только что появилась на страницах USENIX ;login, или этот технический отчёт (*русский перевод всех трёх публикаций см. на сайте http://www.minix3.ru – прим. переводчика*).

А если вы давно уже сделали это, то отдельное вам за это спасибо.

Эндрю Таненбаум, 12 мая 2006 г.