Почему вам больше не стоит писать собственное ядро

Эммануэль Марти (Emmanuel Marty), Понедельник, 27 Декабрь 2004, 20:40

Вы не знаете, что тратите свое время впустую

Я подумывал написать продолжения статей на эту тему, но вместо этого приказом суда был сослан в Школу Заголовков Статей на два года. За это время и мир, и восприятие мною различных вещей, не только заголовков статей, развились. Хотя я и надеюсь, что кто-нибудь где-нибудь счел мои советы достаточно полезными для того, чтобы начать писать ядро, я осознал, в то же время, что радость делать это с нуля, для любых целей и применений, уже фактически запрещена.

Я надеюсь, что успел поймать вас до того, как вы успели сжечь все доказательства того, что вы когда-либо загружали Intel System Programming Manual и теперь нервно смотрите в окно, ожидая появления Полиции Кода, которая пошлет вас в центр реабилитации кодеров. Расслабьтесь, написание своего ядра не запрещено на самом деле и вам не надо переходить на написание систем бух. учета на C#. Многие люди с чувством долга, серьезными программистским навыками и свободным временем работают над такими проектами прямо сейчас, в то время как вы читаете эту статью, хотя на самом деле этому уже нет цели. Некоторые проекты, вроде Syllable и SkyOS продвинулись достаточно далеко к пригодному к использованию стабильному окружению и не устают поражать читателей OSNews своими возможностями. Ирония такого мнения на OSNews не ограничена только мной. Я полагаю, что любителям ядер стоит пропустить эту статью точно также, как это сделал бы я двумя годами ранее. Нелегко прощаться с редкими умениями, которым вы учились долгое время. Как выздоравливающий ядроголик, я бы очень хотел, чтобы мне показали, что я неправ, сильной демонстрацией в один-два удара, и вовлекли бы в участие в проекте любительской ОС.

Честно говоря, я не сдерживаю себя. Ваши усилия напрасны, если, конечно, единственной вашей целью не является замещение Роберта Зелени (Robert Szeleney), как самого уважаемого разработчика неудавшихся ОС. Мне необходимо пояснить, как я медленно пришел к такому выводу и почему это хорошее достижение. В игру вступают один кусочек информации и две пересекающиеся истории.

Как вы можете научиться через постоянные щелчки по носу

Сначала кусочек информации. Я стал менеджером не по Гарвардской MBA, а из опыта, так что я полагаю, что все, кто прошел через какое-либо обучение бизнесу прочитают оставшуюся часть статьи и скажут "да, и что?". Что ж, пффф! вам - это звук высовывающегося языка. Меня не учили позиционировани продуктов в школе, или тому, как фокусироваться на том, что ты действительно делаешь хорошо. Я учился приходя от покупателей с окровавленным носом.

Переходим к первой истории. В прошлом году я серьезно раздумывал о присоединении к проекту Syllable. Насколько я знаю, это единственный проект настольной ОС, который одновременно и удобен в использовании, и имеет открытые исходники и где я бы действительно мог внести значительный вклад, в отличие от Linux, где индивидуальный вклад - всего лишь капля в море. У ядра Syllable есть серьезные недостатки. Вы можете

ощутить на себе при использовании ее круговой (round-robin) диспетчер и неработающую виртуальную память. Нехватка стабильных примитивов или четкой документации процессов имеет неприятные последствия, например, сервер приложений не закрывает окон приложения, когда оно обрушивается. Я думал, что я смогу помочь решить эту проблему вместо того, чтобы критиковать из кресла. Я проделал серьезную работу по замещению базового ядра чем-то, что сможет поддерживать оставшуюся часть Syllable поверх себя, но было бы конкурентноспособно с Linux по производительноти.

Теперь вторая история. Я создал компанию, которая разрабатывает, продает и продвигает компонентную программную архитектуру для потребительскихэлектронных продуктов. Компания существует с 1998 года и продукт начался с того ПО, которое мне нравится писать - с операционной системы. По существу, в нем были реализованы две противоположныевозможности. Первая - компонентная операционная система для потребительскихэлектронных продуктов, первая в своем роде, которая позволяет вам изменять любые правила системы, такие как диспетчер, управление памятью или питанием самостоятельно. Вторая - компонентная модель, которая просто переносит хорошо известные преимущества Corba, DCOM или .NET, такие как упрощенное повторное использование кода и легкая изоляция, в мир потребительскойэлектроники. На данный момент повторное использование является огромной проблемой для индустрии, поскольку производители, по сути, перестали быть компаниями, производящими аппаратуру вроде видео-камер или аналоговых телевизоров, у которых было немного специализированого ПО, а стали компаниям, производящими программные решения, производящими DVD-RW и цифровые телевизоры, которым необходимы серьезные объемы специализированого ПО. Мы действительно старались, очень здорово старались продвинуть свою операционную систему покупателям.
[newpage]
Если они хотели пользоваться компонентной моделью и преимуществами повторного использования, им также нужна была ОС - берите или уходите. Что же, они уходили, постоянно, на японском, корейском, немецком, французском, английском и других языках. Упс. Мы сваливали вину на множество причин, но потом все равно пришли к очевидным фактам. Во-первых, интерфейс ядра настольной, серверной или встраиваемой ОС был отполирован за время, чтобы стать более удобной для программировани машиной, нежели набор инструкций. Концептуально это стандартный интерфейс. Точно также, как у вас есть наборы инструкций Intel или ARM, которые поддерживают сходные операции, у вас есть API POSIX, Windows и µITRON для управления памятью и семафорами, но они сходны концептуально. Если вы предлагаете радикально отличающиеся примитивы, прикладному программисту они бесполезны по причине портируемости другого кода или их знаний.

Во-вторых, для успеха платформе нужны приложения и сообщество разработчиков. Когда вы начинаете писать ядро с нуля, у вас нет ни того, ни другого. Вы можете скопировать существующий подход и API для того, чтобы получить приложения и разработчиков, но это уже сделано. Linux и BSD свободны и подходят для практически любых целей.

В-третьих, новая ОС и повторное использование кода противоречат друг другу. По какой-то причине мы не можем доказать нашим покупателям, что они смогут повторно использовать гораздо большее количество кода, чем сегодня, и смогут лучше делать свое ПО, выбросив весь свой код и переписав его под нашу ОС. У наших клиентов достаточно наглости, чтобы отказаться от инвестирования миллионов долларов в переписывание того ПО, что у них уже

есть.

В-четвертых, вы можете быть лучшими в мире или даже очень-очень хороши только в одном. Все понимают, что это правда для небольшой или средней команды, и я верю в то, что этот принцип применим к любой организации, даже с размерами Microsoft или деньгами Google. Microsoft производит множество продуктов, все они - трата баснословных денег на разработку и поддержку, кроме операционной системы и тесно связанных с ней основных приложений. Менеджеры Google не будут трогать ничего, что четко не связано с поиском. У ветеранов менеджмента для этого даже есть термин: концепция ежа. Ежи на самом деле довольно тупые животные, но они, в отличие от многих, чувствуют, что они знают только одну вещь и сворачиваются в клубок колючек, когда их атакуют.

В-пятых, когда вы что-либо проектируете у вас нет норм работы. А вашим пользователям, в не зависимости от того, будут ли они платить за коммерческий продукт или будут использовать свободный/открытый код, наплевать на то, что вы потратили тысячи часов на действительно тяжкий труд, чтобы сделать то, что вы им предлагаете. Syllable и SkyOS действительно несут в себе изящные технические решения, но они менее функциональны, чем Linux и не привлекательны для обычных массовых пользователей. Linux'у до сих пор еще надо выяснить как быть лучше, чем Windows на домашних машинах, чтобы завоевать пользователей.

Вооружившись такой проверкой реальности, моя компания репозиционироваа свой продукт от операционной системы четко на компонентную модель и повторное использование кода. По сути это означало портирование нашей ОС поверх популярных платформ в японской потребительскойэлектронике, на тот момент это Linux и TRON, и дальнейшее удаление того, что уже не нужно. И так уж случилось, что покупатели начали нам звонить, чтобы заказать наш продукт. Сфокусировавшис на нашей компонентной модели мы затем смогли улучшить ее возможности и сегодня, я думаю, что это действительно стоящий продукт. Оглядываясь назад, я бы сказал, что попытка переманить клиентов на новую ОС была бы только затягиванием петли.

Нравится ли мне работать над тем, что мы продаем сейчас? Вместе с дюжиной программистов, что были связаны с разработкой ядра, я считал, что рынок выбрал для нас самую неинтересную работу. Я не мог использовать свой опыт программировани ядер. Но потом, я осознал, что в сравнении с написанием кода для ядра, все остальное проще. Это как тренироваться для теннисного матча с грузом, а потом убирать его перед большой игрой.

Навыки, полученные вами при разработке ядра, точность, строгость, тестирование, тщательная отладка могут использоваться с девятикратной эффективностью при работе над чем-либо выше ядра. Вам только надо найти, что действительно вас интересует и поставить цель своей работы.

Похоже, что моя компания не единственная из тех, у кого было такое прозрение. Tao Group продавала, угадайте что - операционную систему для потребительскойэлектроники. Возможно вы слышали о них как о части саги об Amiga. В их арсенале была архитектура виртуального процессора, которая позволяла писать портируемый, но написанный вручную ассемблерный код. Каким-то образом им удалось заставить это работать. Когда деньги закончились, их ОС как продукт была отстрелена и теперь они очень успешно продают одну свою хорошую идею. Их архитектура VP (Virtual Processor) позволяет очень эффективно

разрабатывать графические приложения, портируемые между всеми CE (Consumer Electronics) устройствами. JVM, реализованная ими поверх собственной архитектуры VP, мощна. Они нашли одну свою хорошую идею и спрятали все остальное под ковер.

OK, вторая история была немного длиннее, чем первая. Если от меня ничего не будет слышно в следующие два года, присылайте мне гуманитарную помощь в Школу Написания Статей. А теперь обе истории как скоростные поезда идут настречу друг другу и сталкиваются.
[newpage]
Как вы можете сделать этот мир лучше

Прозрение насчет нашего продукта и отсутствие цели для еще одной ОС убили мое желание поработать над Syllable. В конце концов я и не стал этого делать. В проекте Syllable собралась небольшая группа талантливых людей и я верю в то, что они абсолютно правы в своих попытках исправить недостаток интеграции настольных приложений, от которого страдают большинство дистрибутивов Linux. Сегодняшние исследования пользователями настольного Linux лучше всего описываются как глупые. Хорошо интегрированноеядро и настольный API, с подходом управляемого кода в RAD, вроде того, что предлагает Mono, даст пользователю более однородную среду. Однако, в свете прозрения, я считаю, что команда Syllable серьезно ошибается в методах решения этой проблемы. Они хотят быть лучшими в такой интегрированнойнастольной среде, но также имея хорошее ядро, драйвера и то, что еще есть у вас.

Давайте забудем о типичных ядрах, которые до сих пор сидят в "стадии ". Большинство проектов ядер начинаются с идей вроде "давайте создадим ОС вокруг вот этой новой штуки в файловой системе". Потом, ночь за ночью, небольшая, но растущая команда программистов дублирует диспетчеризацию, виртуальную память, API в стиле POSIX, реестр Windows, драйвера устройств для сетевых карт ISA и стек USB. Через два года штуковина файловой системы реализована. Вам стоит похвалить разработчиков проекта за то, что они продвинулись так далеко, но затем выясняется, что эта штучка не так полезна, как она казалась, и проект переквалифицирутся в ОС общего назначения, коих множество. Хуже, если она действительно полезна, но никто никогда не сможет ощутить ее прелести поскольку любимое приложение пользователя не работает в этой странной ОС. Еще хуже для проекта, если кто-либо украдет идею и выпустит ее для Linux с бОльшим успехом. В любом случае, множество усилий были потрачены на дубляж, в то время как не было сделано ничего нового для мира ОС, или пользователей, ну знаете, тех самых, которые иногда бросают нам косточки, чтобы мы могли позволить себе продолжать наслаждаться программированим.

Размышляя о настольных или серверных операционных системах я не могу придумать ни одной новой идеи, которую нельзя было бы реализовать как часть существующей ОС. Имея исходники проверенных временем свободных ядер и учитывая предыдущее заявление о том, что поверх этого можно построить базовый интерфейс ядра, не существует оправданий отбрасыванию экспериментов по реализации новой идеи как части существующего ядра или в пользовательско пространстве. Например, я думаю, что работа драйверов устройств и других модулей как отдельных задач, даже на уровне ядра, и асинхронная работа с обращениями к ядру только через сообщения достойна уважения. Это нехилый и вызывающий проект, а для конечных пользователей он означает то, что они смогут выгружать или перезагружать модули ядра без каких-либо шансов на неудачу, так как вы не можете оставить потоки работающими, если код выгружается. Такая система также будет хорошо масштабироватьс

на SMP, гораздо лучше чем блокировки тут и там. Возможно вы уже захотели написать ядро вокруг хотя бы этой идеи, но нет оправданий для отказа создания патча к Linux, который сделает это. Это будет огромная работа по модификации и тестированию драйверов устройств, но это гораздо меньшая работа, чем если бы вы начали с нуля. Ее можно делать постепенно, всегда поставляя ОС, которая работает, хотя не все драйвера устройств и модули ядра еще пользуются преимуществами новой идеи.

И представьте себе, проект DragonFlyBSD занимается как раз этим, хотя они начали с кода FreeBSD. Я немножко лучше знаю развитие встраиваемых операционных системам, нежели серверных или настольных. Их пользователи, разработчики ПО, которые работают на гигантов потребительскойэлектроники действительно умны. Когда продаешь им продукты этот народ упрям, но благодарен. Они хорошо поняли, что их покупатели, вы и я, покупаем практически тот же самый записывающий привод DVD каждый год и не думаем о неясных возможностях, которые может дать им особая ОС, так что они стандартизироваи платформы ОС. Раньше это были коммерческие RTOS продукты или свободные спецификации вроде TRON, а теперь индустрия довольно серьезно мигрирует на Linux как на наименьший общий знаменатель для своих продуктов. Даже если ваш проект открытый и свободный, если это просто "тоже-" ядро, которое гонится за кормовыми огнями Linux, никого это не заинтересует. Возьмите Linux и реализуйте одну свою хорошую идею в нем, и ей заинтересуется гораздо большее число людей.

Я не верю, что существует такой рынок, хоть в виде платящих пользователей, хоть в виде людей, которые действительно будут использовать ваш код как повседневное окружение, для которого еще необходимо написание ядра с нуля. Это удручает когда вы изучили искусство сооружения своих очередей ожидания, диспетчера и семафоров так, что все действительно эффективно и надежно. Любая область человеческих знаний со временем обобщается и это хорошо для всех, а экспертам стоит поучиться применять свои экспертные оценки на более высоком уровне проблем.

Хорошая новость в том, что теперь операционная система не ограничивается ядром и интерфейсом open-read-write-close. Поразительно малое количество университетскихисследований идет над концепциями расширений существующих ОС для быстрейших разработок приложений более высокого качества. Почти все подходы делаются частными компаниями или нефинансируемым открытыми проектами, и как уже скромно показала моя компания, это область для множества новых, действительно интересных проектов, над которыми стоит работать.

Как у разработчиков ядер, ваши навыки на порядок выше, чем у других в деле написания и поставки кода для закрытия дыр, которые вы обнаруживаете где-либо в операционной системе. Если вы перенесете свои цели на его добавление а не дублирование существующего проверенного временем ПО, я даже не представляю насколько лучше станет программная среда для всех нас.

Об авторе:
Эммануэль Марти (Emmanuel Marty), основатель и главный технический директор (CTO) NexWave Solutions, которая первой представила коммерчески доступную компонентную архитектуру для потребительскойэлектроники, используемую ведущими производителями. Работает с компьютерами с возраста в 10 лет. Сейчас ему 28 и он живет во Франции, в

Монпелье со своей женой и двумя дочерьми-близняшками.