2006-04-16 15:36:24 UTC
*Оригинальный материал: http://osnews.com/story.php?news_id=14353*
*Перевод: Павел Макаров (pavel.makarov@mail.ru)*

# ПОЧЕМУ Я ЛЮБЛЮ МИКРОЯДРА

Автор: Том Холверда (Thom Holwerda)

*Как раз между автомобильной аварией и Пасхой я узнал, что я должен написать статью в предвоскресную колонку. И так, вот он я: пожирающий несметное количество шоколадных яиц (а я никогда даже не любил шоколада), страстно желающий кофе (он для меня столь же важен, как дыхание) и одержимый идеей объяснить вам мою... правильно, зацикленность на микроядрах. Почему же я на самом деле люблю их? Почему же я на самом деле считаю, что микроядерный подход лучше, чем идея монолитных ядер? А вот почему.*

Сначала – небольшая поучительная история. Дебаты на тему «микроядра или монолитные ядра» отнюдь не новы. В 1992 году Линус Торвальдс (знаете его?) был вовлечён в знаменитый «большой базар» (я хотел бы , чтобы «базары» сегодня оставались бы такими же, как тогда) с Энди Таненбаумом, одним из авторов MINIX и книги о MINIX, сыгравшей главную роль в обучении Торвальдса написанию ядер. В то время, как Таненбаум уже проверил свои идеи на MINIX - микроядерной операционной системе, служившей учебным материалом для будущих разработчиков ядер, Линус был практически никто, только-только начав работать над Linux, монолитным ядром. Эта дискуссия возникала вновь и вновь в сообщениях на comp.os.minix, и это действительно очень интересное чтиво.

В любом случае, ясно, что всё, что я сейчас рассказал, не является «вторжением на чужую территорию»; давайте-ка лучше вернёмся к обсуждаемому предмету.

Микроядро (или μK) очень отличается от монолитного ядра тем, что в μK всё, что потенциально способно «обрушить» ядро, запускается вне пространства ядра, в виде отдельных процессов, известных как серверы. В MINIX3, например, каждый драйвер, за исключением часов (я не могу не удивляться: почему же эти чёртовы часы - исключение? Я собираюсь написать Таненбауму по этому поводу на днях), обитает в пространстве пользователя. Если один из этих драйверов «рухнет» из-за программной ошибки, то, к счастью, он не сможет «вырубить» систему. Вы можете просто перезапустить «рухнувший» драйвер и продолжить, как ни в чём ни бывало. Никаких тебе глобальных системных отказов, никакого простоя. А это заметно бодрит.

Это, естественно, особенно удобно при обновлении системы. Скажем, вы наконец-то устранили ошибку, «обрушивавшую» вышеупомянутый драйвер. Вы можете просто остановить старую версию драйвера и запустить модифицированную версию, даже не останавливая систему. Теоретически это означает работу абсолютно без остановок (unlimited uptime).

Это именно в теории, конечно же, поскольку остаётся всё-же часть системы, работающая в пространстве ядра, которая может содержать ошибки. Что касаемо

MINIX3, то её код содержит около 3800 строк кода; такого объёма вполне достаточно для существования ошибок. Однако Энди Таненбаум и его группа уверены, что эти 3800 строк кода могут быть вычищены от ошибок почти полностью, что позволило бы приблизиться ещё на один шаг к абсолютной безотказности (eternal uptime). Сравните это с монолитным ядром Linux 2.6.x, которое имеет приблизительно 6 миллионов строк кода, и которое надо очистить от ошибок (уж лучше уехать и провести время с семьёй на Пасху).

Другим преимуществом микроядра является его простота. В μK каждый драйвер, файловая система, функция и проч. является отдельным процессом в области пользователя. Это означает, что микроядра на этом самом локальном уровне являются относительно простыми и свободными от ошибок, что, по общему мнению, позволяет легче их поддерживать. И здесь мы сталкиваемся с дуализмом микроядра: чем легче поддерживать μK на локальном уровне, тем сложнее делать это на уровне глобальном.

Логика этого утверждения относительно проста и понятна. Я бы предпочёл привести здесь свою собственную аналогию, если бы не тот факт, что CTO (видимо, Chief Technical Officer - технический директор проекта TUNES, см. http://tunes.org/wiki/Main_Page - *примечание переводчика*) уже предложил лучшую для объяснения этого противоречия между локальной и глобальной сложностью:

«Возьмите большой и тяжёлый кусок мяса, нарубите его на маленькие кусочки, заверните каждый из этих кусочков в гигиенические пластиковые пакеты и соедините эти пакеты верёвками; несмотря на то, что каждый кусочек гораздо меньше исходного куска мяса, суммарно конструкция будет тяжелее куска мяса на вес пластиковых пакетов и верёвок, причём в обратной пропорции к размеру кусочка (иными словами, чем больше кто-либо гордится локальной простотой, достигнутой в его микроядре, тем больше глобальной сложности он на самом деле добавит по отношению к аналогичной конструкции без микроядра)». (источник: http://tunes.org/wiki/Microkernel)

Не правда ли, это всё хорошо объясняет? И вот теперь эта глобальная сложность приводит меня ко второму основному недостатку μK: непроизводительным издержкам. Поскольку все эти процессы изолированы и должны взаимодействовать друг с другом более длинными путями, μK неминуемо будет медленнее в смысле производительности, чем эквивалентное монолитное ядро. Вот почему Линус выбрал монолитное ядро для Linux: в начале 90$^{x}$ компьютеры на самом деле не были столь мощными, поэтому любой возможный способ снижения непроизводительных издержек приветствовался обеими руками.

Однако, я думаю, что ситуация сильно изменилась с тех пор. Монолитность была логичной 16 лет назад, но на современных мощных компьютерах с процессорами, работающими большую часть времени, как 300-долларовые пылесосы, преимущества μK просто перевешивают его неизбежные, незначительные (хотя с пользовательской точки зрения, вероятно, огромные) прелести производительности.

---

Всё вышеизложенное является технической, то есть более объективной, стороной дискуссии. Однако, существует также и более субъективная сторона дела. Я предпочитаю эту точку зрения в случаях, когда приложение или устройство делает что-то одно, но делает это хорошо. Это та самая причина, по которой я предпочитаю в качестве музыкального центра иметь набор компонентов класса Hi-Fi, а не моноблочные конструкции, это та самая причина, по которой я предпочитаю GameCube, а не Xbox/PS2, это та самая причина, по которой я предпочитаю обычный телевизор плюс отдельный DVD-проигрыватель, а не компьютерный медиа-центр, это та самая причина, по которой мне не нравится идея смешения музыкальных стилей (зачем смешивать хип-хоп с роком, если тебе и так нравится каждый из них?), это та самая причина, по которой я предпочитаю механическую коробку передач автоматической (у нас в Голландии мы говорим «автоматы только для женщин»; не обижайтесь там, в США, где автоматические коробки более популярны; если ты носишься по ту сторону Атлантики, тебе приходится покрывать существенно большие расстояния), это та самая причина, по которой я предпочитаю простой обед дорогому 4-фазному обеду в классном ресторане (конечный результат тот же самый: подзарядка жизненно важным питательным веществом), это та самая причина, по которой я предпочитаю обычный Мартини Бьянко причудливым коктейлям (опять-таки, конечный результата тот же самый, но я оставлю это на ваше усмотрение), это та самая причина, по которой я предпочитаю чёрный кофе сливкам и сахару в кофе... и так далее, и так далее, и так далее.

---

Всё это подводит меня к одной мысли. Помните, я упомянул автомобильную аварию? Почитав сопроводительную переписку, вы могли бы понять, как это получилось из-за ошибки со стороны другого водителя («тонкий намёк» автора на печальные последствия ошибок в драйверах: слово «driver» по-английски означает в том числе и «водитель» - *примечание переводчика*).

Вот теперь и скажите: случилась бы та авария, если бы человеческий мозг работал как μK?