Роман Химов

Linux? Вам какой?

Критерием выбора дистрибутива для большинства пользователей становится обычно или программное наполнение, или система установки пакетов, или тот факт, что именно такая операционная система стоит у знакомого гуру. Но, в конце концов, решающим фактором при выборе системы становится ядро.

Что такое Linux? Внимательный читатель правильно (и наверняка с улыбкой) вспомнит, что на самом деле Linux — это ядро, и не более того. Важная, но все же лишь деталь в сложном механизме полноценной операционной системы. А уж дистрибутивы — это дело десятое. Нынешние пользователи дистрибутивов GNU/Linux редко задумываются о том, какое ядро установлено в их системе, а также что его можно и нужно обновлять, конфигурировать и компилировать. Это в целом хороший признак того, что Linux действительно честно отрабатывает слоган «Just works» в большинстве конфигураций, заботливо подобранных для пользователей разработчиками дистрибутивов. Однако не всех и не всегда удовлетворяют стандартные конфигурации ядер. Например, нынче стало модно компилировать ядра для PC-платформы вообще без поддержки шины ISA. Кто-то предъявляет повышенные требования к безопасности, и ему нужны самые последние исправления. Ну а кто-то хочет проникнуться тем самым духом fun, окружающим Linux, и опробовать какие-либо новшества, которых нет ни в одном дистрибутиве и которые доступны только тем, кто не боится скомпилировать ядро со своей (что на самом деле еще более страшно) конфигурацией. Поэтому начнем мы, собственно, с описания процесса сборки и сопутствующих ему действий.

Официальное ядро Linux

Во-первых, сразу же проясним, что значит «официальное ядро Linux». Сам Линус Торвальдс говорил, что официального ядра Linux не существует, есть лишь различные ветви, ведущиеся разными разработчиками; просто так сложилось, что ветвь основателя системы пользуется наибольшим доверием у пользователей. Что ж, это так, но именно его ветвь все-таки зачастую называется официальной, и именно от нее отталкиваются все параллельные. Официальное ядро Linux всегда доступно во всех разновидностях на http://kernel.org, но к этому мы еще вернемся, а сейчас перейдем к сборке.

Сборка

Итак, вы скачали около 40 Мбайт исходных кодов официального ядра актуальной версии. Что с ними делать? Нет никаких ограничений по тому, куда вы будете распаковывать ядро и где вы его будете конфигурировать и компилировать, но в целом его все-таки принято располагать в каталоге /usr/src/linux-uname -r. Принцип понятен, uname -r — команда, выдающая полную версию текущего ядра. Это значит, что ядро версии 2.6.12 будет логично поместить в каталог /usr/src/linux-2.6.12, поскольку многие сторонние скрипты-помощники ориентированы именно на такое расположение.

Теперь ядро необходимо сконфигурировать. Это делается командами make config, make menuconfig или make xconfig. Все они, по сути, идентичны, только представлены в разных обличиях. Первую команду вводить не рекомендуем, иначе через час-другой у вас покраснеют и остекленеют глаза, а окружающие подумают, что вы сходите с ума (проверено на личном опыте). Все эти неприятности могут случиться из-за того, что опций конфигурирования у ядра тьма тьмущая, а make config работает очень просто — последовательно требует у вас вразумительного ответа по каждой опции в виде «да/нет/модуль». Это долгий, требующий внимания процесс, поэтому лучше пользоваться make menuconfig и make xconfig. Первая программа состоит из текстовых меню, по которым можно гораздо проще ориентироваться и значительно быстрее сконфигурировать ядро, а последняя, соответственно, представляет собой полностью графический конфигуратор. По умолчанию он использует Qt, но можно набрать make gconfig, и конфигуратор будет собран для GTK. Полагаем, должно быть очевидно, что для этого потребуются заголовочные файлы Qt или GTK, впрочем, для ncurses (используемого в menuconfig) они тоже нужны (все конфигураторы собираются в системе при запуске).

Давать советы по конфигурации — дело неблагодарное, так что лучше будет найти информацию по этой теме в Сети и просто уяснить разницу между статической компиляцией, компиляцией модуля и некомпилированием модуля или функции вообще. Вся конфигурация ядра будет сохранена в один-единственный файл config в каталоге с ядром. Его полезно куда-нибудь скопировать, чтобы иметь впоследствии стартовую точку для любых дальнейших конфигураций: обычно его кладут в каталог /boot рядом с уже скомпилированным ядром в виде файла «config-версия-ядра». Кстати, позже, при обновлении ядра и наличии отлаженной конфигурации, вписать эту самую конфигурацию в новое ядро очень легко — копируете тот самый config и набираете «make oldconfig». Вас спросят только о новых опциях (если они будут), и вы будете готовы к компиляции.

Компиляция тривиальна — make, и все дела! В случае с веткой 2.4 чуть сложнее — make bzImage и make modules. Далее можно созерцать процесс компиляции. В нем красиво показывается сборка ядра, иногда некрасиво проскакивают многочисленные «warning» (в последних версиях их всего около 200 штук в конфигурации «yes» для всего, однако большей их части можно не придавать значения). По окончании этого процесса стоит установить модули ядра (если, конечно, они присутствуют в вашей конфигурации) по команде make modules_install. Все модули при этом аккуратно поместятся в директории /lib/modules/. Наконец, следует установить само ядро с помощью make install или же скопировать его вместе с файлом System.map (его необходимо переименовать в «System.map-версия-ядра») вручную из директории arch/i386/boot/bzImage в /boot.

После этого конфигурируем загрузчик. Эта процедура зависит от того, что вы используете в вашей системе — GRUB или LILO, и ее мы описывать не будем (читайте — man grub, man lilo). Мы настоятельно рекомендуем GRUB, поскольку постоянно переустанавливать LILO просто-напросто неудобно, да и параметры командной строки у GRUB довольно мощные.

Хочется сказать пару слов о самих патчах. Как правило, это один файл, упакованный чем-нибудь наподобие bzip2, применяющийся с помощью команды patch -p1 в каталоге исходников. Обычно легко сделать что-то подобное:

bzcat bla-bla.patch.bz2 | patch -p1

После такого некраткого ликбеза мы переходим к самому интересному — рассмотрению конкретных версий ядра Linux. Так уж получается, что этот обзор во многом является парадом разработчиков Linux, поскольку у каждой ветки есть главный разработчик, и он же, как правило, является основным, если и не кодером, то, по крайней мере, «приемщиком» патчей для своей ветки.

2.6.x

Конечно же, логично начать с самого главного, а конкретно — с ветки Линуса Торвальдса. Мы не случайно говорили о том, что это всего лишь одна из веток ядра, и «официального» ядра Linux как такового не существует. Тем не менее все ориентируются на ветку Линуса, и она определенно главная в ядре, поэтому ее принято называть «официальной». Про нее, пожалуй, много не скажешь, кроме разве того, что ветка 2.6.x значительно отличается по модели разработки от 2.4.x — в ней допускаются значительно большие изменения без старта нестабильной ветки 2.7. Это способствует ее быстрому развитию и обновлению, а также позволяет производить полезные изменения, что очень хорошо, но в то же время, по общему признанию, ветка 2.6 в целом не такая стабильная, какой была 2.4. Конечно же, эти релизы можно найти на http://kernel.org

2.4.x

Несмотря на то что ветка 2.4.x уже давно не является основной, она по-прежнему поддерживается. Более того, некоторые дистрибутивы (к примеру, Slackware 10.1 и многие Live CD) используют в качестве основы именно ее. Поэтому нелишним будет сказать, что эту ветку нынче ведет Марсело Тосатти, и в последнее время объем патчей для нее все уменьшается. В основном туда переносятся исправления обнаруженных ошибок, однако иногда проскакивают обновления драйверов. Марсело также выпускает -pre-версии и -rc-версии следующих релизов. Между ними есть одно важное различие: -pre — это ранние версии, еще не до конца протестированные, и в дальнейшем в них могут вноситься некие серьезные изменения. Версии -rc уже значительно стабильнее. Более того, начиная с первой версии идет чистая стабилизация ядра, то есть принимаются только исправления различных ошибок. Как правило, последний кандидат в релиз после некоторого тестирования становится финальным релизом без каких-либо изменений в коде.

2.4.x — это одна из основных веток, так что за ней тоже следует идти на http://kernel.org

2.4.x-hf

Также стоит сказать сразу про единственное, на наш взгляд, заслуживающее внимания ответвление 2.4 — патчи -hf (hot-

fix) от Вилли Торрью. Хотя называть их ответвлением не совсем корректно, все патчи, появляющиеся в -hf , попадают в следующий релиз 2.4.x. В этой ветке собираются исправления серьезных ошибок и безопасности, которые с ее помощью становятся доступны раньше, чем выход следующего релиза 2.4.x. Следовательно, это однозначно полезные патчи. Нелишним будет убедиться в их наличии для того ядра, которое вы захотите поставить. Патчи находятся по адресу: http://linux.exosec.net/kernel/2.4-hf/.

2.6.x.y

Теперь вернемся к 2.6 и обозначим еще одну довольно новую, но тоже очень важную ветку. Как я уже говорил, 2.6 в целом не так стабильна, как 2.4, и Линус решил, было, изменить схему наименования релизов, для того чтобы их больше тестировали и, соответственно, чтобы ветка была более стабильной. От изначальной идеи Линуса отговорили (в подробности вдаваться не будем, но, на наш взгляд, справедливо), но предложили создать дополнительную ветвь -stable, которую мы теперь знаем как 2.6.x.y. Линус согласился, однако спросил, где же найти таких «suckers», которые будут ее вести? Но они нашлись: Грег Кроа-Хартман и Крис Райт добровольно заполучили это почетное звание и, судя по ветке 2.6.11.y, делают свою работу очень хорошо.

В ветке -stable появляются только те патчи, которые помогают решать конкретные проблемы пользователей. В ней нет никаких обновлений, новой функциональности, и все изменения, помимо исправления какой-либо конкретной ошибки, должны удовлетворять еще одному важному требованию — простоте. Фактически это тот же «must have», что и -hf для ветки 2.4.

2.6.x-pre, 2.6.x-rc

Помимо этого стоит упомянуть -pre-, а также -rc-версии ветки Линуса. Как уже упоминалось выше, Марсело Тосатти строго разграничивает -pre и -rc, однако этого нет в ветке Линуса. В последнее время версии -pre вообще стали появляться довольно редко, а кандидаты в релиз начали реально стабилизироваться только к третьей-четвертой версии. В общем, получается, что сегодня -rc стоит использовать разве что в тестовых целях, а также тогда, когда они становятся действительно надежными. Этот момент определяет сам Линус, и на правильных сайтах об этом всегда сообщается. Версии хороши, если требуется функциональность, появившаяся совсем недавно, или какое-либо исправление драйвера, к тому же, конечно, их достаточно удобно тестировать.

2.6.x{-pre,-rc,}-git

Также, что касается официальной ветки, сейчас доступны еще и версии -git. Они накладываются на -pre или -rc (а если таковых еще не было, то на последнее официальное ядро) и содержат в себе снимок разрабатываемого Линусом ядра. Git — название нового инструмента управления исходниками (точнее, основы этого инструмента — хранилища) для Linux, пришедшего на смену BitKeeper. Git изначально был написан Линусом Торвальдсом, а затем в значительной степени доработан и облагорожен Петром Баудисом. Новые версии -git автоматом появляются каждый день, так что их очень удобно тестировать (особенно выяснять, какой патч вызвал проблему), однако использовать на постоянной основе не стоит, если, конечно, на это нет особых причин.

Кстати говоря, нынче очень многие ветки ведутся с помощью git, и особо заинтересованным будет полезно почитать документ Джеффа Гаржика о работе с git: http://lkml.org/lkml/2005/5/26/11.

Эту ветку Линуса, равно как и 2.6.x.y, а также релизы -rc и -pre, тоже можно найти на http://kernel.org. Долго искать не придется — ссылка на эту ветку расположена прямо на первой странице сайта.

2.6.x-mm

От относительной стабильности давайте перейдем в полную нестабильность — ветку Эндрю Мортона -mm. Она включает в себя все возможные патчи, которые только существуют в природе. Конечно, не совсем все, но, по замечанию Линуса, она содержит «wild and wacky patches» — там они «варятся», тестируются и потом плавно переходят в основную ветку Линуса. Сегодня очень многое попадает к Линусу именно через фильтр -mm. Посему ветка экспериментальная (запросто может не скомпилироваться, хотя с i386 такое случается редко). Тестировать ее удобно, но на постоянное использование должны быть веские причины.

Впрочем, одна уже поддержка файловой системы Reiser4 и технологии FUSE (подробности см. в специальной врезке) — вполне достаточные причины.

Например, автор этой статьи использует эту ветку постоянно — как раз ради Reiser4 и FUSE, примеряя к своему будущему LFS (Linux from Scratch). Также в ней сегодня можно обнаружить массу других интересных вещей, но в основном они касаются удобств и новшеств для самих разработчиков (а как хорошо работать с inotify!). В целом эту ветку можно порекомендовать отчаянным гикам, не боящимся проблем, но желающим держаться действительно на острие прогресса, а также что-либо потестировать.

Патчи этой ветки доступны в двух видах: цельный «мегапатч» — один файл, где все лежит вместе; или россыпь патчей, входящих в -mm, в одном архиве. Последний удобен для индивидуального тестирования и/или применения. То есть, если вам нужна поддержка Reiser4, но не хочется экспериментировать с остальными -mm-патчами — пожалуйста, выкачивайте их либо отдельно (доступно для каждого релиза в отдельном каталоге) и применяйте, либо берите весь архив и доставайте необходимое. Скачать патчи можно по адресу http://kernel.org/ patchtypes/mm.html.

2.6.x-mm-jedi

Кстати говоря, к ветке Эндрю Мортона существует еще один набор патчей -jedi. Его ведет Френк Дэнис, и, как ни странно, для принципиально нестабильного -mm это нечто вроде -stable для ветки Линуса: здесь собираются исправления различных ошибок, которые, как это часто бывает, появляются в самый момент релиза. Обычно имеет смысл посмотреть, есть ли пат-

чи -jedi для последнего релиза -mm, если вы хотите его опробовать. Заодно получите устрашающее название версии ядра. Например, не так давно пришлось работать с ядром 2.6.12-rc2-mm2-jedi1.

Архив патчей находится на сервере ftp://ftp.c9x.org/pub/linux-kernel.

Подсистемы 2.6.x

Идем далее и переходим уже к значительно менее популярным веткам. В первую очередь надо отметить многочисленные ветки разработчиков каких-либо подсистем Linux. Это ACPI, IEEE1394, I2C, USB, IA64 и многие другие. Соответствующие патчи, очевидно, будут касаться конкретных подсистем и иногда будут необходимы. Не раз приходилось видеть сообщения о том, что какое-нибудь железо работает только при наличии последнего патча, например IEEE1394, но эти патчи довольно скоро перейдут в ветку Линуса, к тому же постоянно подбираются Эндрю Мортоном в свою ветку.

Основной «точкой сбора» подсистем на сегодня, наверное, можно назвать http://kernel.org/git/, однако некоторые подсистемы ведутся отдельно, например SCSI: www.parisc-linux.org/cgi-bin/gitweb.pl.

2.6.x-ck

Переходим к менее «официальным» веткам. Тут сразу же стоит упомянуть австралийского доктора-анестезиолога Кона Коливаса и его ветку -ck. В нее включены патчи к диспетчерам процессора и ввода/вывода, кроме того, обещается улучшенная отзывчивость системы. Ветка сама по себе стабильная (и, кстати, включает в себя патчи 2.6.x.y целиком), несмотря на то, что сам Кон рекомендует употребить немного коньяка перед загрузкой такого ядра. Это ядро я также постоянно использую для другой своей машины (старенький Celeron 488 + 128 Мбайт памяти, далее фигурирует как assam), планирую перевести на него еще одну машину, как только доберусь) и чисто субъективно с этим ядром при параллельной компиляции меньше дергается музыка. Патчи можно найти по адресу http://ck.kolivas.org/patches/2.6

2.6.x-ac

Стоит отметить ветку Алана Кокса — -ac. В нее входят исправления ошибок плюс дополнения, в основном, касающиеся драйверов. Особенно заметны в ней, пожалуй, патчи подсистемы IDE — у Алана есть серьезные претензии к сегодняшнему ведущему разработчику IDE, Бартоломею Золнеркевичу. Это ядро также знакомо пользователям Fedora Core и RHEL, так как по умолчанию там используется именно оно. Веб-сайт — http://kernel.org/patchtypes/ac.html. Также ветку 2.6.x-ac можно найти на kernel.org, правда, иногда kernel.org запаздывает.

2.6.x-tiny

Существует еще одна интересная ветка -tiny. Она нацелена на уменьшение занимаемого ядром дискового пространства и, самое главное, памяти. Также в ней содержатся некоторые функции, полезные для ограниченных в ресурсах конфигураций, таких как встраиваемые системы, старые 386-е и наладонные компьютеры. На наш взгляд, это прекрасный хакерский патч, которому место в официальной ветке (хотя бы частично, что касается структуры).

Он открывает множество недоступных параметров конфигурации (хотя, конечно, никто не мешает править их в исходниках и без этого патча, но надо знать, где и как): поддержка AIO (асинхронный ввод/вывод), файловой системы sysfs (исключительно полезная вещь), системного вызова SYSENTER (аппаратная особенность процессоров Pentium II и старше, значительно ускоряющая вызов функций ядра), ptrace (контроль выполнения потока), утилиты dnotify (извещение пользователей об изменениях в файловой системе), vm86 (читайте — эмуляция DOS). А ведь если выключить поддержку режима vm86 для процессоров IA32, то можно сэкономить целых 6 кбайт памяти!!! Еще в нем же представлено несколько инструментов, интересных разработчикам ядра. Мэтт Мэколл, который, собственно, и ведет эту ветку, утверждает, что минимальное ядро -tiny способно запуститься даже на машине с 2 Мбайт памяти.

Многие из открывающихся параметров экспериментальны, так что играться с ними надо аккуратнее, но у меня это ядро работало на машине assam, и делало это вполне неплохо. Чего-либо сверхъестественного не замечено, но, полагаю, для слабых компьютеров оно может быть полезно (кстати,

Перспективные технологии

Новые веяния — уже сегодня

Reiser4 (http://namesys.com) не просто с легкостью выигрывает множество тестов производительности, это еще и очень интересная файловая система с точки зрения архитектуры. Можно даже сказать, что это не файловая система, а система хранения объектов — разница огромная. На деле Reiser4 лишь организует взаимодействие плагинов, которые и представляют каким-либо образом данные, преобразовывают их в те или иные объекты. Даже сам файл, как структура и каталог — всего лишь плагины в Reiser4. В подробности вдаваться не буду, так как все это заслуживает отдельной статьи.

Скажем еще и про FUSE (http://fuse.sourceforge.net/). Эта технология реализует поддержку файловых систем в пользовательском пространстве и позволяет пользователю без проблем монтировать. Звучит, возможно, сложно, но на самом деле все довольно просто — файловая система, какая бы она ни была, в FUSE работает в пользовательском пространстве, то есть для ее работы не нужно накладывать патчи на ядро, монтировать от root или что-либо подобное. Именно поэтому с помощью FUSE сейчас реализуются самые разные и самые смелые проекты, касающееся ФС, например SSHFS — файловая система SSH. Не поняли, как это? А знаете про sftp? Вот это — то же самое, только все файлы вы будете видеть в своем привычном пространстве имен, где вам удобно. Точно так же можно работать с архивами.tar.gz и .tar.bz2 или с FTP-соединениями, а также шифровать файлы с помощью EncFS. Монтируем куда удобно и получаем доступ. Удобно? Конечно.

такое ядро по умолчанию собирается с опцией -Os GCC). Данная ветка представлена в Интернете на странице http://selenic.com/tiny.

2.6.x-kj

Упомянем также ветку -kj (kernel janitors — дворники ядра), она интересна разве что как средство смягчения приступов альтруизма и хороша для начинающих хакеров ядра. Собственно, она и создается начинающими хакерами под руководством более опытных товарищей. В ней происходит вычищение ядра от всевозможных остатков старых API, проводятся некие мелкие оптимизации, исправляются очевидные ошибки и прочее в том же духе. Как уже говорилось, она создается начинающими хакерами — понятно, что они лишь изучают различные подсистемы/API ядра и по этому поводу очень часто просматривают исходники, а попутно делают и что-то полезное. Приветствуется всяческое тестирование этой ветки, но обычно внутреннее тестирование в ней хорошее, да и патчи не очень серьезные, поэтому изменения из нее довольно быстро попадают в ветку -mm, а затем и к Линусу — в основное ядро.

Подробную информацию об этой ветке можно найти на сайте http://janitor.kernelnewbies.org.

2.6.x-RT

Еще одним из интереснейших на сегодняшний день патчей является расширение реального времени для Linux от Инго Молнара. Этот патч содержит изменения, позволяющие использовать Linux в системах мягкого и жесткого реального времени (конечно, с некоторыми допущениями, все же основа Linux и все драйверы совсем не реального времени, и с этим приходится считаться), что в первую очередь ценно для различных встроенных применений, но также уже используется и на других системах. Например, с этими патчами работают те, кто профессионально связан со звуком (пользователям Jack настоятельно рекомендуется), потому что латентность, а точнее непредсказуемость, в обычном ядре Linux сильно мешает нормальной работе.

На сегодняшний день часть патчей из этой ветки уже интегрирована в официальное ядро — именно оттуда растет вытесняемое ядро (опция сборки ядра — CONFIG_PREEMPT) в текущих версиях ядра, а также огромное количество исправлений для многопроцессорных машин: введение вытесняемого ядра позволило обнаружить массу неявных гонок и блокировок в ядре. Для конечного пользователя в целом этот патч может быть интересен, поскольку снижение латентности и стабильность интересна всем, даже несмотря на то, что большая часть этих изменений не будет видна глазу. Я также пробовал это ядро на машине assam и не могу сказать, что обнаружил какую-то большую разницу в скорости реакции. Правда, с аудио- и видеоконтентом мне работать не приходится, и оценить все его прелести довольно трудно. Поэтому пришлось вернуть на место -ck с включенным CONFIG_PREEMPT.

Сейчас ведутся дискуссии о том, стоит ли включать этот патч в официальное ядро, и, вполне возможно, он войдет через некоторое время в ветвь Эндрю Мортона.

Патчи можно скачать на сайте http://redhat.com/~mingo/realtime-preempt.

2.6.x-vm

Название -vm не совсем официальное, однако именно оно было предложено для ветки, ведомой Риком ван Риелом. Появилась эта ветвь с подачи Ника Пиггина, который заметил, что используемый ныне в Linux алгоритм замещения страниц виртуальной памяти — весьма популярный LRU, уже порядком устарел и на современных конфигурациях работает далеко не оптимально. Соответственно, Рик взялся и уже реализовал замену LRU — алгоритм CART. На сегодня в этой ветке еще не было официальных релизов, но предварительные версии доступны по адресу www.surriel.com/patches/nonresident. Wiki-страничка ветки: http://wiki.linux-mm.org/wiki/AdvancedPageReplacement. Мне тем временем остается только заметить, что по еще не выясненной причине я этот патч не опробовал — серьезное упущение, буду наверстывать. Патч, конечно, экспериментальный, так что будьте осторожны.

Полагаю, впрочем, что после стабилизации этот патч довольно быстро войдет в -mm и далее в ветку Линуса.

Stay tuned

На самом деле веток ядра Linux существует гораздо больше, а уж количество патчей к ядру, витающих по просторам Сети, вовсе не поддается исчислению, но это уже совсем частные случаи. Оставаться в курсе выхода новых версий разных веток Linux можно подписавшись на LKML (Linux Kernel Mailing List). Для этого достаточно отправить письмо с темой «subscribe linux-kernel» по адресу majordomo@vger.kernel.org, однако надо быть готовым к трафику порядка 300 писем в день. Можно еще заглядывать на http://kerneltrap.org (там английский язык и периодически пропускаются релизы, хотя есть масса различной интересной информации), либо обратиться к странице http://osrc.info.

Ну а если вам стало интересно, что еще происходит вокруг и внутри ядра (а это иногда просто необходимо, чтобы понять, зачем существует тот или иной патч, что означает тот или иной новый пункт в конфигурации и какое новое оборудование поддерживается или будет поддерживаться), стоит начать изучение с сайта http://kernelnewbies.org/. Знакомьтесь с Linux!

На прощание можно порекомендовать поэкспериментировать с ядрами — получите массу удовольствия, узнаете много нового, сможете поддерживать ядро в актуальном состоянии (а это значит, что количество дырок в ядре будет стремиться к минимуму). Особенно же порекомендую тестировать различные -rc- и -pre-версии — этого сегодняшнему ядру действительно не хватает, размеры патчей все растут, и управляться с ними разработчикам все сложнее, так что вспоминайте дух братства свободного сообщества, тестируйте и не стесняйтесь писать разработчикам о проблемах. Файл MAINTAINERS в корне дерева исходников Linux предназначен именно для таких случаев. Успехов в ядерном синтезе! |