Оригинальный материал с сайта: http://citkit.ru/

MINIX 3. Общее обозрение

Алексей Федорчук

11 января 2007 г

Кто же из линуксоидов не знает старика MINIX'а? Эту миниатюрную учебно-показательную Unix-подобную ОС, созданную 20 лет назад профессором Эндрю Таненбаумом для вразумления студентов и приобщения их к идеалам UNIX на демократической PC-платформе и за символические, по тем временам, деньги, львиная доля которых приходилась на знаменитую книжку ее разработчика - "Операционные системы: разработка и реализация". Систему, которая вдохновила Линуса Торвальдса на сочинение своей терминальной программы, которой суждено было превратиться в Linux - ныне самый популярный представитель семейства свободных Unix-подобных операционок.

А ведь этим у большинства линуксоидов знания о MINIX и исчерпывались. Ну разве что еще многим известно было, что это - ОС с микроядерной архитектурой. И что по этому поводу на заре развития Linux между Эндрю и Линусом развернулась бурная дискуссия...

Не был исключением и автор этих строк - и его представления о MINIX исчерпывались приведенными выше сведениями. Более того, у меня никогда не возникало никакого желания с ней ознакомиться. И, боюсь, так и не возникло бы, если бы не два события. Первое - публикация на CITForum.ru обзора Сергея Кузнецова, посвященного статье Эндрю Таненбаума, Джоррита Хердера и Херберта Боса "Можем ли мы сделать операционные системы надежными и безопасными", а в дальнейшем - его же перевода более подробной и "техничной" работы тех же авторов - "Построение надежных операционных систем, допускающих наличие ненадежных драйверов устройств". Второе же событие - появление русскоязычного сайта http://www.minix3.ru, посвященного, как несложно догадаться, именно этой операционной системе. Точнее, сайт-то организовался раньше, но на его материалы я наткнулся случайно на исходе минувшего года.

Было еще и третье событие, сугубо личное - в первые дни нового года я на несколько дней потерял возможность делать что-либо, кроме как читать. А поскольку запас дефективов и прочего развлекалова был давно исчерпан, обратился к указанным источникам, внимательно перечитав их все. В результате меня охватило непреодолимое желание установить MINIX и поковыряться с ним. Почему - надеюсь, станет ясным из дальнейшего. А пока -

Немного истории

Операционная система MINIX была явлена миру ровно 20 лет назад - в январе 1987 года. Автор ее - Эндрю Таненбаум - был профессором Университета Врийе, Амстердам,

Нидерланды. И преподавал он в этом университете не что иное, как computer science, хотя и был по образованию физиком.

Надо сказать, что унивеситетское образование в этой области в 80-е годы прошлого века базировалось преимущественно, если не исключительно, на платформе Unix. Что создавало для студентов известные трудности. Во-первых, тогдашние версии Unix были проприетарными, и стоили немерянных, по нашим представлениям, денег. Во-вторых, они работали на еще более дорогих аппаратных платформах. Конечно, имелся еще и BSD Unix, который при определенных условиях можно было пользовать если не совсем бесплатно, то за много меньшую мзду. Но и он требовал дорогостоящих компьютеров. До экспансии Unix на демократическую PC-платформу оставались еще годы (попытки Microsoft с ее Xenix застолбить эту нишу потерпели фиаско).

Так вот, Таненбаум вел курс Unix, к которому написал собственный учебник - "Operating Systems: Design and Implementation". Но изучать Unix без системы - все равно, что обучаться музыке без инструмента. А с инструментом-то как раз и была напряженка. И ему не осталось ничего другого, как такой инструмент изготовить. Им-то и стала ОС MINIX (в дальнейшем получившая имя MINIX 1).

Это была маленькая и компактная операционка, работавшая на более-менее общедоступной тогда (у них) платформе - машинах с первым 32-разрядным процессором от Intel, 80386 (в дальнейшем эта платформа обычно называлась i386, и архитектура эта под все возрастающими номерами, а потом и различными именами существует и по сей день). Доступность MINIX усугублялась еще и тем, что ее можно было скомпилировать и в 16-битном варианте, и в этом качестве она становилась пригодной к использованию не только на PC-AT (80286), но даже, как говорят, на XT'шках, то есть на машинах с процессором 8086/8088.

Распространялась она исключительно как сопроводительный материал к упомянутому выше учебнику. Весь комплект, по свидетельству Линуса Торвальдса, стоил 169 долларов при заказе по почте. Что на самом деле не так дорого: помнится, в те годы на Западе, только-только переставшем загнивать, ни одно специализированное книжное издание не стоило дешевле 100 баксов (сужу по геологическим монографиям). Так что фактически основная, если не вся, затратная часть для пользователя приходилась на книжку, да и дискеты были не так дешевы, тогда как ОС как таковая могла рассматриваться в качестве бесплатного приложения. И, во всяком случае, это было несоизмеримо дешевле тех тысяч долларов, в которые обходилась лицензия на любой из существовавших тогда проприетарных Unix'ов. Требовавших, к тому же, сущей безделицы в виде соответствующей рабочей станции в несколько десятков тысяч.

Разумеется, ОС MINIX распространялась в сопровождении исходных текстов, предназначенных для изучения и потрошения. Необходимость в котором возникла очень скоро.

Дело в том, что, предназначенная исключительно для учебных целей, ОС MINIX в принципе не была приспособлена для выполнения каких-либо реальных задач. Однако шаловливые студенческие (и не только) руки так и чесались прикрутить ее к чему-либо пригодному к использованию. В результате система очень быстро обросла всякого рода патчами, из которых главным был патч от австралийца Брюса Эванса. После наложения этих патчей система становилась способной выступать как платформа разработчика. Именно на такой патченой системе Линус Торвальдс и начал создавать свою терминальную программу.

Однако сама по себе MINIX по прежнему распространялась исключительно в первозданном виде - как чисто учебная система, и лишь в сопровождении книги (или, напротив, сопровождая книгу). То есть, будучи открытой, она не была свободной ни в понимании лицензии BSD, ни, тем более, GPL. Ибо права на MINIX принадлежали издательству Prentice-Hall, выпустившему учебник Таненбаума. В сущности, правовой статус MINIX был точно таким же, как и обычной книги. Что, однако не мешало тому, что на протяжении 10, а то и более, лет по ней учились поколения студентов.

Тем не менее, шли годы, и MINIX даже в качестве учебной системы стала устаревать. Что особенно проявлялось в связи со стандартизацией как проприетарных, так и свободных Unix-систем, обусловленной следованию стандартам POSIX, первая редакция которых была принята в 1988 году. MINIX же, напомню, был обнародован в самом начале 1987 года, а разработан еще раньше.

В результате, после принятия уже второй редакции POSIX (1996 год) возникла необходимость в доработке MINIX. Что и было сделано в 1997 году, одновременно со вторым же изданием учебника, написанным Таненбаумом в соавторстве с Альбертом Вудхаллом. В качестве приложения к нему шла версия MINIX 2. Плюс к приведению ее в соответствие с действующими стандартами, она была значительно усовершенствована в плане многозадачности, поддержки защищенного режима и так далее.. Но чисто учебный ее характер сохранился в неприкосновенности, как и условия распространения.

Следует заметить, что в промежутке между MINIX 1 и MINIX 2 в декабре 1992 года была выпущена еще и версия MINIX 1.5 (именно после нее первая и получила свое полное имя). Главной ее особенностью была поддержка архитектур, отличных от Intel: Motorola 680XX (на них базировались тогдашние Macintosh'и, а также ныне забытые Amiga и Atari), SPARC (Sun SparcStation 1), National Semiconductor NS32032 и вроде еще каких-то, мне не известных. Однако к 1998 году они отмерли, да и при жизни их обладатели в очередь за MINIX не становились. Так что MINIX 2 опять поддерживала исключительно платформу Intel, хотя возможность 16-битной компиляции (и, соответственно, работы на доисторических уже XT'шках и AT'шках) сохранилась.

Пока MINIX 2 продолжала свое, вполне успешное, существование в качестве "песочницы" для начинающих юниксоидов, кардинал лелеял коварные замыслы. А именно: Таненбаум задумал превратить MINIX в полноценную операционную систему, в которой были бы реализованы его представления о том, какой должна быть современная ОС. А заодно - сделать ее свободной в полном понимании этого слова. Напомню, что "несвобода" предыдущих версий объяснялась не особой жадностью профессора, а тем, что они были написаны как приложение к книге и автоматически обретали тот же правовой статус, что и она.

Результатом явился анонс новой операционки, MINIX 3, который состоялся 24 октября 2005 года. Это была не просто следующая по номеру версия, а именно новая операционная система, почему цифру "3" здесь следует рассматривать как часть ее имени собственного. Таненбаум неоднократно подчеркивал, что сходство ее с предшественниками - лишь в первом компоненте названия, а различие между MINIX 1/2 и MINIX 3 не меньше, чем между Windows 3.1 и Windows XP. Обособленность от предшественников подчеркивалась и тем, что отныне MINIX 3 будет распространяться не как довесок к книге, а совершенно самостоятельно, под лицензией BSD.

На протяжении последующих месяцев периодически выходили новые версии системы - динамику их можно проследить на официальном сайте проекта, http://www.minix3.org/, в

разделе новостей. Последней на момент сочинения этих строк (январь 2007 года) является версия 1.2a.

Надо заметить, что новые версии Таненбаум создает не в одиночку: в проекте на постоянной основе участвуют: его соавтор по второму изданию учебника Альберт Вудхалл, сотрудники Университета Врийе Йоррит Эрдер, Герберт Бос, Бен Грас, Филип Хомбург и другие. На сегодняшний день полный список участников можно видеть здесь. Приятно наблюдать в этом списке нашего соотечественника, Романа Игнатова. Вероятно, как и любой открытый проект, MINIX 3 не обходится также без содействия волонтеров, официально в проекте не числящихся.

Впрочем, тут заканчивается история и начинается разговор

О микроядрах

Итак, какова же должна быть операционная система в представлении Таненбаума и его соратников? Это практически исчерпывающе описано в ряде их работ, русские переводы которых, как уже упоминалось, можно найти на http://citforum.ru и http://www.minix3.ru (список русскоязычных источников по MINIX 3 будет приведен в заключительной части этой заметки). Поэтому позволю себе остановиться на устройстве MINIX 3 лишь вкратце.

Как я уже говорил в самом начале заметки, то, что MINIX с самого своего рождения представлял собой микроядерную ОС, знает каждый школьник-линуксоид. А вот что это такое, микроядерность?

Для начала зададимся вопросом, а что же такое ядро вообще? Традиционно ядро определяется как программа, обеспечивающая взаимодействие всего остального системного и прикладного софта с аппаратной частью компьютера, и распределение его ресурсов между приложениями. В соответствие с этим ядро функционирует в отдельной области памяти, которая так и называется — пространством ядра. Память же, отводимая под все остальные программы, именуется пользовательским пространством; протекающим там процессам доступ в пространство ядра закрыт, и как-либо влиять на ядро, в том числе и негативно (вследствие ошибок в программе или злоумышления), они не могут. Но все процессы внутри пространства ядра взаимодействуют друг с другом, и ошибка в одном из компонентов может повлечь за собой тяжкие последствия, вплоть до краха системы.

Понятие аппаратуры компьютера, однако, оказывается двояким. С одной стороны, это те узлы, без которых машина в принципе не может функционировать - процессор и память. Эта сторона взаимодействия охватывается такими службами ядра, как обработчик прерываний, средства запуска и останова процессов, планировщик задач, механизм межпроцессного взаимодействия.

С другой стороны, в понятие аппаратной части включаются и внешние, по отношению к системе процессор/память, устройства, от видеокарт и жестких дисков до принтеров, сканеров, сетевых адаптеров и многого, многого другого.

Более того, у этой аппаратной медали есть еще и третья сторона - сервисы доступа к файловым системам, сетевым протоколам и так далее. Они представляют собой связующее звено между собственно внешними устройствами и пользовательскими программами. Например, сервисы доступа к файловым системам обеспечивают возможность взаимодействия между дисковыми устройствами, несущими файловые системы, и

обращающимися к ним приложениями.

Ядра, обеспечивающие все три рассмотренные функции, именуются монолитными. И они вполне успешно функционировали до тех пор, пока внешних устройств и сервисов было мало. Однако со временем количество и тех, и других стало расти, как снежный ком. Вспомним, сколько на наших глазах появилось только критически важных устройств, таких, как дисковые интерфейсы или файловые системы.

В результате ядра стали катастрофически увеличиваться в размерах. Что влекло за собой а) непроизводительные расходы ресурсов, в первую очередь памяти, и б) рост числа ошибок, обусловленный огромными объемами ядерного кода, недоступного восприятию человека.

С первой болезнью научились бороться посредством модульного подхода: многие части ядра, обеспечивающие работу отдельных внешних устройств (т.н. драйверы устройств) или ядерных сервисов (в Linux их подчас также называют драйверами, например, драйвер файловой системы "имя рек"), могут не встраиваться жестко в исполняемый файл - образ ядра, а подключаться к нему в виде внешних модулей. И ныне все Unix-подобные системы, такие, как Linux или BSD, содержат в себе модульно-монолитные ядра.

Внешние модули могут подгружаться в память при загрузке, а также часто и в работающей системе, по необходимости, нередко они обладают и способностью выгружаться, когда эта необходимость пропадет. Однако в любом случае модули функционируют в пространстве ядра, так что их введение проблемы общей устойчивости не решает: криво написанный драйвер устройства все равно сохраняет способность обрушить систему. А поскольку, как уже говорилось, сложность кода ядра растут, возрастает и вероятность появления ошибок, в том числе и критичных для работы системы.

Для решения проблемы устойчивости ядра и были придуманы микроядра. Идея их - в том, чтобы, по словам Таненбаума, "вынести ядро за пределы ядра". То есть оставить в ядре только средства управления базовой аппаратурой, а драйверы устройств и сервисы выделить в отдельные программы, функционирующие в пользовательском пространстве памяти. При необходимости они обращаются к функциям ядра через специальные процедуры, но влиять на него каким-либо образом не могут. Такой подход приводит к повышению надежности системы, но снижает производительность, поскольку требует дополнительных накладных расходов.

Собственно, идея микроядра появилась очень давно, чуть ли не одновременно с самим Unix. Однако долгое время производительность машин не позволяла эффективно использовать микроядра в составе практически применяемых систем. Тем не менее, за прошедшие годы различных их реализаций было написано много. Время от времени то или иное микроядро пропагандировалось как основа оперционной системы будущего, но удачных реализаций цельных микроядерных систем оказалось довольно мало.

Среди удачных решений на базе микроядра наибольшей известностью пользуется QNX. Правда, это система проприетарная, закрытая, и потому об ее устройстве известно немного.

Из свободных микроядерных реализаций наибольшую известность приобрело микроядро Mach, разрабатывавшееся вплоть до второй половины 90-х годов университетами - сначала Карнеги-Меллона, а затем штата Юта. Различные его версии в разное время составляли основу законченных систем, таких, как BSD Mach, NeXT, MacOS X. Все они имели в своем составе микроядро Mach, поверх которого запускалась драйверно-сервисная часть от ядра

BSD, собранная в виде отдельного модуля.

Впрочем, из всех перечисленных систем только BSD Mach можно назвать по настоящему микроядерной, так как у него BSD-окружение ядра функционировало в пользовательском пространстве. И у NeXT, и у MacOS X BSD-окружение запускалось в том же пространстве ядра, так что микроядерными их можно назвать только по имени.

Наконец, начиная с 1987 года и по настоящее время существует MINIX 1/2, основанная на собственной реализации микроядра, послужившее прототипом для микроядра нашей героини. Так что пора поговорить

Собственно о MINIX 3

Для начала заметим, что Таненбаум, по его же словам, не ставил самоцелью создание именно микроядерной операционки: задачей его команды было просто построение надежной и безопасной операционки. Другое дело, что решение обеих проблем было найдено именно в микроядерной архитектуре. Причем в виде, кардинально отличающемся от всех предлагавшися ранее реализаций.

Отличие первое - микроядро MINIX 3 самое микроядреное микроядро в мире. Из него вычищено все, кроме перечисленных ранее компонентов, таких, как обработчик прерываний, средства запуска и останова процессов, планировщик задач, механизм межпроцессного взаимодействия; правда, почему-то в ядро включен и один из сервисов - сервис часов. В результате все это хозяйство укладывается менее чем в 4000 строк кода - и только оно исполняется в пространстве ядра.

Второе отличие - драйверная и сервисная части, вычлененные из ядра, разделены между собой. В результате образуется знаменитая четырехслойная модель-метафора, в основании которой лежит ядро, надстраиваемое "драйверным слоем", которое, в свою очередь, перекрывается "сервисным слоем" и венчается "слоем пользовательских программ".

Кроме того - и это третье отличие, - каждый драйвер и каждый сервис представляет собой отдельный процесс в пользовательском пространстве, аналогично обычным пользовательским приложениям. В результате ни один драйвер и ни один сервис, как бы криво они не были написаны, не в состоянии обрушить всю систему - точно также, как в любом Unix'e это не могут сделать обычные пользовательские приложения. Не могут повлиять они и на соседние процессы, так как напрямую взаимодействовать они не могут, а вынуждены при необходимости обращаться к ядру.

Наконец, четвертое, и, пожалуй, главное: сервер реинкарнаций. Это процесс, выступающий родительским по отношению к процессам всех драйверов и сервисов. Которые он запускает при старте, а в дальнейшем отслеживает состояние. Если процесс какого-либо драйвера или сервиса в силу неких причин самопроизвольно "умирает", он запускает его вновь. Если один из драйверов или сервисов начинает вести себя "нехорошо", сервер реинкарнаций в состоянии убить соответствующий ему процесс и тут же запустить его заново, обеспечивая, таким образом прозрачное для пользователя самовосстановление системы при отказе почти любого драйвера устройства или системной службы.

Очевидно, что такая, достаточно сложная, схема взаимодействия драйверов и системных служб не может не привести к некоторой потере производительности по сравнению с обычными системами, где драйверы и сервисы сосуществуют в едином пространстве

памяти, вне зависимости от того, пользовательском или "ядерном" (а подчас еще и вместе с ядром). То есть - неизбежно должны вызвать снижение быстродействия системы. В каких масштабах? Исследования команды Таненбаума дают ответ на этот вопрос, и к нему я еще вернусь. Но сначала сделаю маленькое отсутпление на тему, что такое быстродействие операционной системы вообще и с чем его едят, сиречь, меряют.

Когда обсуждается проблема быстродействия любых ОС, первым делом обычно вспоминают о скорости загрузки. Вероятно, потому, что ее проще всего измерить: достаточно посидеть с секундомером перед несколькими машинами с разными операционками или дистрибутивами, чтобы потом уверенно утверждать о их сравнительном быстродействии.

Однако имеет ли скорость загрузки системы к быстродействию ее при реальной работе? Отнюдь. Достаточно вспомнить, что MS DOS 3.3 на IBM PC/XT грузилась быстрее, чем любой Linux на любом Athlon64. Строго говоря, как сказал бы Сергей Образцов, измерение скорости загрузки даже к скорости загрузки никакого отношения не имеет :) Потому как зависит скорость загрузки в первую очередь от количества подгружаемых модулей и стартовых сервисов. Так что измерение ее на самом деле меряет радиус кривизны рук пользователя, меру его лени или, напротив, количество свободного времени, которогое он способен выделить на доведение системы до ума. Но никак не какие-либо программно- или аппаратно-зависимые показатели.

Не лучше и с тестами на реальных приложениях под разными ОС. Например, со столь любимым сравнением скорости обработки запросов web-сервером под Linux и FreeBSD,на основании чего делается вывод о превосходстве одной оперционки над другой. Кстати, и не помню даже, кого над кем, да это и не важно. Ротому что чразу же возникает закономерный вопрос: а что меряется в этом случае? Сравнительное быстродействие ОС? Или все-таки реализаций Apache и, например, MySQL под ту и другую систему?

В общем, отдав в свое (не такое уж давнее) время дань увлечению всякого рода тестированием (это занятие было бы точнее классифицировать как пузометрию или ... ну, то, что в этнографической литературе называют "сравнение мужей"), я пришел к стойкому убеждению, что в большинстве случаев это либо измерение аршином с точностью до ангстрема, либо, по изящному выражению Таненбаума, сравнение яблок с апельсинами.

И тем не менее, методика тестирования, предложенная командой Таненбаума, производит впечатление. Во-первых, она (команда) поставила себе целью оценить влияние на быстродействие системы одного-единственного фактора: выноса драйверов за пределы ядра и перемещения их в пользовательское пространство. И потому тестирование быстродействия MINIX 3 проводилось... на MINIX 2 :) Каким образом? Очень просто: в качестве сравнительных объектов использовались каноническая MINIX 2, с одной стороны, и она же, пересобранная с удалением из ядра драйверов устройств и еще некоторыми модификациями, что фактически превратило ее в MINIX 3 (именно так, в соответствие с исходной работой, они и будут именоваться далее).

Во-вторых, в качестве тестов выполнялись процедуры, скорость которых действительно зависит от ОС исключительно или очень существенно: время исполнения системных вызовов, скорость чтения из файла и записи в файл, а также чтения непосредственно из блочного устройства (винчестера). Тесты с реальными приложениями тоже проводились - но предельно простыми (в смысле - мало подверженными посторонним по отношению к ОС влияниям): пересборка образа системы и набора контрольных тестов POSIX (что это такое - будет рассмотрено в следующей заметке), а также обработка текстового файла утилитами

типа sed, grep и др.

С полными результатами тестирования можно ознакомиться в работе "Построение надежных операционных систем, допускающих наличие ненадежных драйверов устройств". Я же хочу привести тут некие синтезированные из этих тестов данные - средние для каждой перечисленной группы тестов, на которых авторы не акцентировали внимание, хотя они блестяще подтверждают их итоговый вывод.

Итак, далее - отношение результов MINIX 3 к таковым MINIX 2, усредненное для каждой группы тестов. Поскольку единицы измерения временные, белые (MINIX 3) везде начинают и проигрывают. Но - смотрим, где и насколько.

Группа тестов	MINIX 3/MINIX 2
Системные вызовы	1,12
Чтение из файла	1,08
Запись в файл	1,09
Чтение из BD	1,09
Приложения	1,06

Можно видеть, что наибольшее отставание MINIX 3 проявляется в чисто "ядерно-драйверном" тесте исполнения системных вызовов (12%), тогда как в тесте приложений оно снижается до 6%. Я достаточно занимался пузометрией, чтобы утверждать со всей ответственностью: при реальной работе разницы между MINIX 2 и MINIX 3 не будет никакой.

Вот, пожалуй, и все, что я хотел сказать о MINIX 3 до того, как перейти к описанию процедуры знакомства с нею, что будет предметом следующей заметки. Но сначала -

Источники информации

Разумеется, основным источником информации о MINIX 3 является официальный сайт проекта - http://www.minix3.org/. Кроме того, что с него можно скачать саму систему, там находится и весьма представительная документация - разумеется, на английском, но и ссылки на все имеющиеся переводы на другие языки там присутствуют.

Впрочем, за русскоязычными источниками информации лучше обратиться на сайт http://www.minix3.ru, ведомый Романом Игнатовым (официальным разработчиком MINIX) и Павлом Макаровым, перу которого принадлежат как переводы документации с официального сайта, так и оригинальные материалы на эту тему. Из официальных документов наиболее важны следующие:

- Руководство по установке (http://www.minix3.ru/docs/setup_manual_rus.pdf)
- FAQ (http://www.minix3.ru/docs/faq.pdf)
- Введение в дисковые разделы (http://www.minix3.ru/docs/partitions.tgz)
- Обмен данными с другими ОС (http://www.minix3.ru/docs/datxchng.pdf)

Наиболее важные материалы:

- Эндрю С. Таненбаум, Введение в MINIX 3 (http://www.minix3.ru/articles/introduction_minix3.pdf)
- Эндрю С. Таненбаум, Йоррит Н. Эрдер, Герберт Бос, Можем ли мы делать

операционные системы надёжными и безопасными? (http://www.minix3.ru/articles/OS_Reliable_Secure.pdf)
- Эндрю С. Таненбаум, Йоррит Н. Эрдер, Герберт Бос, Бен Грас, Филип Хомбург, Модульное системное программирование в MINIX 3 (http://www.minix3.ru/articles/MSPinM3.pdf)
- Э.Таненбаум, ВТОРАЯ ЧАСТЬ «МАРЛЕЗОНСКОГО БАЛЕТА» (http://www.minix3.ru/articles/balet.pdf)
- Р.Игнатов, MINIX 3 - реинкарнация? (http://www.minix3.ru/articles/Ignatov_minix_reincarnation.pdf)
- П.Макаров, Дороги, которые мы выбираем... (http://www.minix3.ru/articles/Makarov_Roads.pdf)

Материалы по MINIX 3 на сайте http://citforum.ru:

- Сергей Кузнецов, Обзор статьи Эндрю Таненбаума, Джоррита Хердера и Херберта Боса "Можем ли мы сделать операционные системы надежными и безопасными" (http://citforum.ru/operating_systems/microkernel_tanenbaum)
- Эндрю С. Таненбаум, Йоррит Н. Эрдер, Герберт Бос, Построение надежных операционных систем, допускающих наличие ненадежных драйверов устройств (http://citforum.ru/operating_systems/reliable_os/)

Некоторое количество сведений о MINIX 3 можно почерпнуть из русской Википедии - http://ru.wikibooks.org/wiki/Minix.

В Сети можно нагуглить еще немало ссылок по теме MINIX 3, но в основном они повторяют то, что имеется на http://www.minix3.ru.

Ну а обсудить MINIX 3 можно в специальной теме форума http://posix.ru.