Апрель 2006

Оригинальный материал: http://www.usenix.com/publications/login/2006-04/openpdfs/musings.pdf

Перевод: Павел Макаров (makarov@minix3.ru)

РАЗМЫШЛЕНИЯ

Редакционная статья в журнале ;login:

Рик Фэрроу (Rik Farrow), rik@spirit.com

Программирование – это искусство. Когда разные люди пытаются выполнить одни и те же задачи, их программы в общем случае будут совсем разными. И если их тщательно проанализировать, то окажется, что некоторые программы будут выделяться прекрасным кодом, в то время как остальные легко могут быть отнесены к разряду безобразных.

Я изучал программирование в колледже, имея дело с мучительно неудобными в работе перфокартами и мэйнфреймами. Единственная опечатка означала многочасовое (а иногда и до следующего дня) ожидание для того, чтобы увидеть результаты исправления. Несмотря на мою веру в то, что использование перфокарт стимулирует некоторую дисциплину и аккуратность в кодировании, это так и не смогло зародить во мне настоящего понимания того, как писать хорошие программы. Я мог писать (и действительно писал) добротный и функционально насыщенный код, но в нём, естественно, отсутствовала та элегантность, которую я мог время от времени наблюдать в программах, написанных другими людьми.

Топорно сделанный код, однако, задевает не только эстетические чувства. Однажды я был озадачен написанием программы форматирования текстов с требованиями, аналогичными найденным в книге Кернигана и Плогера «Программные средства» (Kernighan and Plauger's «Software tools»). Я не знал в то время (1979 год) о существовании этой книги и «пахал что есть силы». Закончив, я написал программу, которая работала правильно, беря размеченный текст, форматируя его и генерируя оглавление, причём всё это на компьютере, использующем для хранения файлов два 8-дюймовых гибких диска. Компиляция 16-страничной программы занимала около 15 минут, да и использование готовой программы, написанной на PL/Z (версия PL/I от компании Zilog), также отнимало много времени.

После того, как я уволился из той компании, я узнал, что человеку, сменившему меня, было дано то же самое задание, но с использованием BASIC. Его версия программы работала *в 3 раза быстрее*, поскольку BASIC имел гораздо лучшие подпрограммы обработки строк, чем PL/Z. Я знал, что моя программа была местами неэффективна, и если бы я переписал эти фрагменты на ассемблере, она, вероятно, стала бы столь же быстрой. Но я был сбит с толку.

Заглядывая глубже

Компьютеры, на которых я учился, по сравнению с сегодняшними выглядят, как электро-механические калькуляторы. Мэйнфрейм, который я использовал в колледже, заполнял собой всю комнату, требовал интенсивного охлаждения и для ввода-вывода (чтения перфокарт, записи их на ленту и печати) на самом деле использовал другие компьютеры. Шум от вентиляторов системы охлаждения стоял невероятный, но компьютеры работали так медленно, что на мигающих на панели управления лампочках на самом деле можно было увидеть адреса памяти, к которым осуществлялся доступ. На системах, подобных этим, каждую инструкцию в программе приходилось принимать во внимание.

Когда мы сегодня пишем программы, нам легко может показаться, что элегантность и эффективность программ более не важна. Вместо этого наши могучие компьютеры могут вводить нас в заблуждение, заставляя думать, что всё прекрасно работает. Часто проблемы не всплывают до момента, когда программа выходит на стадию производства и «падает» под реальными нагрузками, или содержит ошибку безопасности, превращающую код в лазейку в системе безопасности.

Для этого выпуска я искал программистов, которые желали бы написать о своём искусстве. Я обрадовался тому, что Брайан Керниган (Brian Kernighan) захотел поделиться своим опытом преподавания современного программирования (advanced programming). Статья Брайана объясняет, как он использует тестирование для поддержки AWK и использует аналогичное

тестирование в своих классах. Я даже обратил внимание на собственное удивление по поводу того, что я бы сам стал более «крутым» программистом, если бы изучил тестирование как дисциплину, которую Брайан прививает своим студентам сегодня.

Колонка Perl Дэвида Бланк-Эдельмана (David Blank-Edelman) также начинается с обсуждения тестирования в Perl. Различные модули Perl обеспечивают инфраструктуру (framework), которая прекрасно (или не очень) может быть использована для помощи в компоновке пакетов, которые, в свою очередь, могут быть протестированы ещё до инсталляции.

Диомидис Спинеллис (Diomidis Spinellis) написал об эффектах многих уровней производительности (effects of the many levels of performance), обнаруженных в современной компьютерной памяти. Размер памяти, доступный для запуска программ с максимальной производительностью на современных процессорах оказывается крошечным, и каждый дополнительный уровень (*видимо, речь идёт об уровне памяти – прим. переводчика*) обеспечивает меньшую производительность. Диомидис объясняет, как работают различные уровни, предлагает подсказки (hints) для улучшения производительности в критических областях и завершает анализом соотношения «цена/производительность» для памяти, что, несомненно, будет достаточным для инициализации некоей дискуссии.

Вы также обнаружите и другие статьи на тему программирования. Чаос Голубитски (Chaos Golubitsky) пишет о cflow, инструменте, который она использовала при анализе безопасности серверов IMAP в её работе, представленной на LISA '05. Люк Канье (Luke Kanies) объясняет, почему он выбрал Ruby для своей реализации Pappet. Если вы наслышаны о Ruby и сомневаетесь в том, надо ли изучать его, вам стоит почитать статью Люка.

Ник Стоутон (Nick Stoughton) рассказывает о своей работе в нескольких комитетах по стандартизации, работе, которая оказывает реальное влияние как на программирование, так и на open source. Если вас волнуют эти две темы, вам следует почитать доклад Ника.

Тщетные грёзы

Пока я был занят разглагольствованиями о необходимости создания новой операционной системы, Эндрю Таненбаум и его студенты были заняты написанием MINIX 3. Я точно не знаю, как долго я писал о необходимости маленького ядра, которому можно доверять, и которое может запускать сервисы на исполнение без привилегий, в их собственных защищённых областях памяти, но MINIX 3 уже делает это.

Энди писал MINIX как средство для изучения операционных систем давно, когда лучшей из ОС была UNIX, операционная система, которая уже переросла легко понимаемый размер исходных текстов и была обременена авторскими правами и юристами компании AT&T. Хотя мы сегодня и имеем операционные системы с открытым кодом, такие, как Linux и семейство BSD, эти ОС с годами сильно растут в размерах и усложняются. MINIX 3 сумела совершить невозможное, став операционной системой следующего поколения постижимого размера.

Под «следующим поколением» я подразумеваю тот факт, что MINIX является микроядерной как по конструкции, так и по философии. Только ядро запускается в привилегированном режиме, а все остальные сервисы, включая диспетчеризацию процессов, файловые системы, работу в сети и все драйверы устройств, запускаются в своих собственных индивидуальных адресных пространствах. Одно только перемещение драйверов устройств из ядра в их собственное адресное пространство приводит к тому, что они могут отказывать без «обрушения» ядра. Это также означает, что системный код, включая драйверы устройств, может тестироваться без перезагрузки системы и неисправные драйверы (и серверы) могут быть перезапущены.

И хотя MINIX 3 не собирается заменять вашу настольную ОС сегодня, она уже является хорошим кандидатом для встраиваемых систем, для которых критичны устойчивость, надёжность и малый объём памяти. Возможно, и ваш сотовый телефон когда-нибудь будет работать на MINIX 3.

Что, нет раздела «Безопасность»?

Кстати: в этом выпуске *;login:* нет раздела «Безопасность». Есть две статьи системных администраторов, Дэвида Мэлоуни (David Malone), пишущего детективный рассказ о загадочном наплыве HTTP-запросов, и Рэндольфа Лэнгли (Randolph Langley), повествующего о созданном им программном обеспечении для лучшей регистрации в sudo.

В этом месяце у нас появились две новые колонки. Хисон Чак (Heison Chak) будет писать о VoIP, создавая в этой колонке фон для грядущих статей, которые будут помогать системным администраторам совершенствоваться в поддержке и применении VoIP в своих сетях. Роберт Феррелл (Robert Ferrell) взялся за юмористический раздел, развлекая нас вместе с /dev/random.

В конце находятся краткие обзоры LISA '05, WORLDS '05, и FAST '05. Вы можете удивиться, почему это декабрьские обзоры не появились до апреля, но если вы взглянете на график публикаций *;login:*, вы сможете увидеть, что ни одна из этих конференций не завершилась до момента подготовки материалов для февральского выпуска. Я, естественно, неоднократно прочитал все эти обзоры, и я призываю вас познакомиться с тем, что было представлено на конференциях, которые вы не посетили.

И, наконец, у нас есть «Мнение» Тома Хэйнса (Tom Haynes). Том пишет, что он до такой степени напрягся насчёт OpenSolaris, что он немедленно должен был с этим что-нибудь сделать. И он сделал.