

Демифологизация представлений о компьютере

Демифологизация --- не совсем точный термин. Так как в данном курсе не рассматриваются все детали, начиная с физических основ, то наличие в любом случае мифов неизбежно. Эта часть материала ориентирована на людей, освоивших Windows, возможно, прошедших курс повышения квалификации с использованием Windows, но не слушавших курсов по информационным технологиям. Основная цель заключается в том, чтобы убедить пользователя не применять бездумно при использовании ПСПО опыт, полученный при работе с Windows, отделить те представления, которые с точки зрения ПСПО являются верными, от ошибочных.

Вот некоторые распространенные заблуждения:

- Компьютер = ОС
- ПО = ОС
- GUI = ОС
- Иконки = файлы, программы; окна = программы
- Документы = их обработчики
- Мифологизация вирусной угрозы
- Необходимость постоянной "оптимизации" системы (дефрагментация, оптимизация реестра)
- Задача должна решаться специальной программой под задачу. Новая задача --- новая программа (взаимоднозначное соответствие множеств задач и программ)
- Документацию читать не надо, всё равно там ничего нет.
- Большая часть ПО --- контрафактное ПО
- ПО плохо работает потому, что оно краденое

В этом списке можно выделить две группы мифов различной природы:

- Мутные сущности, легенды о которых возникают в связи с тем, что в Windows эти вещи непрозрачны --- например, типы файлов, "оптимизаторы", firewall;
- Чёткие вещи, определения которых различаются в Windows и ПСПО. Легенды в таком случае возникают от того, что пользователю знаком лишь "путь Windows".

Рассмотрим подробнее эти и некоторые другие мифы.

В качестве вступления разоблачим утверждение "пользователи Windows тупые, пользователи Linux крутые". Это неправда. Неверно и утверждение "пользователи Linux грамотные, а пользователи Windows - нет". Некоторая корреляция имеется, и у неё есть свои предпосылки. Например, до последнего времени, UNIX-подобные системы были вотчиной сильно увлеченных компьютером людей. Нет ничего удивительного в том, что люди, пользующиеся Linux, в большинстве своем занимаются *изучением компьютера*. Миф же порождён основным отличием между Linux и Windows: в Linux живы традиции UNIX, структура самой операционной системы, как и структура информационного пространства ориентирована на изучение; Windows, в силу его несвободной природы, явно препятствует изучению и познанию на системном уровне, ставя труднопреодолимый барьер между пользователем и разработчиком.

Компьютер = ОС

Если человек не отделяет аппаратной части компьютера от программной, это означает, что он вообще незнаком с отраслью. Такой человек научился работать с программами, но не приобрел системных знаний. Это не столько миф, сколько недостаток знаний. В таком случае рекомендуется пройти курсы компьютерной грамотности на основе Linux.

Миф о Документации

Большинство документации к прикладным программам, устроенным по принципу "одна задача --- одна программа", предназначено не для того, чтобы пользователь получил информацию о том, как пользоваться программой, а для того чтобы сподвигнуть его сделать хоть что-нибудь. Например, там поясняются тривиальные значения пунктов меню, часто в повелительном наклонении: "Для того, чтобы переименовать файл, выберите пункт меню **Файл -> Переименовать**", и т. п. Эта документация ориентирована на преодоление компьютерной боязни. Поскольку авторы таких программ делают упор на интуитивную понятность, то их главной задачей становится уверение пользователя в том, что он поступает правильно. Очень показательны т. н. "помощники" -- привлекающие внимание пользователя виртуальные персонажи, основная задача которых -- подсказывать ему дальнейшие действия. Подобный подход плохо применим и к Linux как ОС, и к подавляющему большинству программного обеспечения под Linux.

В Linux даже в аналогичную документацию вкладывается информация, достаточная для изучения. Основная документация ОС также направлена на получение знаний. Это серьезное отличие. Оно ярко выражается, например, таким образом: в сообществе Linux не принято задавать вопросы, ответы на которые можно найти в документации; пользователи Windows же предпочитают подход наподобие "я не буду читать документацию, просто скажите, как решить эту конкретную задачу".

ПО = ОС

Убеждение в том, что всё, что работает на компьютере, является Windows возникает, если пользователь ни разу не устанавливал программное обеспечение. Если человек полностью переставляет систему, то он впадает в другую крайность --- начинает считать, что в ОС вообще ничего нет, для решения любой задачи надо искать специальные прикладные программы.

В Linux, с одной стороны, существует возможность продемонстрировать разделение ОС и приложений и показать, что ОС сама по себе очень небольшая. Но, с другой стороны, так как устанавливается дистрибутив, с его помощью можно сразу решать достаточно много задач, да и установка дополнительных приложений не представляет сложности для пользователя.

GUI = OS

С точки зрения Windows это отчасти правда. В Windows интерфейсом управления системой является либо ABI (Application Binary Interface), либо приложения основанные на ABI, но имеющие графический интерфейс. В Linux много различных средств управления системой, как, впрочем, возможностей общения с системой, и этот факт помогает пользователю понять, что ОС --- гибкое понятие.

Задача = программа

Миф состоит в убеждении, что если для задачи нет специальной программы с красивым графическим интерфейсом, то решить задачу невозможно. У этой проблемы есть две стороны.

Во-первых, привязанность к конкретному графическому интерфейсу. Например, человек использует Photoshop, и настолько привыкает, что малейшее изменение интерфейса, даже не в ущерб функциональности, переносит болезненно. При этом привыкание зачастую вызывают первые купленные, выбранные из соображений популярности, а не качества, продукты. В Linux для решения задач существует много методов, и еще больше их реализаций, с разнообразными интерфейсами. Например, пользователь Photoshop

спрашивает "а есть у вас в GIMP групповые операции?". Правильным ответом будет: "для подобной задачи есть ImageMagick".

Во-вторых, убежденность в том, что если для задачи нет специального приложения, то задача неразрешима. Пользователь Linux привыкает использовать суперпозиции инструментов. Windows-пользователи, обычно, ею не пользуются, в Linux же это весьма распространенный подход. Суперпозиция --- это мощный инструмент, которым не стоит пренебрегать.

Иконки = файлы, программы. Окна = программы

Суть этого заблуждения в том, что неопытный человек не отделяет графическую среду от операционной системы. При использовании Linux, в котором существуют различные графические среды, это заблуждение быстро развеивается. Организация рабочего стола в Windows провоцирует распространенную ошибку отождествление объекта, которым можно манипулировать на рабочем столе и программы/файла, к которым можно получить доступ через этот объект. Для развенчания мифа можно, например, создать файл в каталоге ~/Desktop, продемонстрировав, что он появился на рабочем столе. Полезно изучить содержимое .desktop-файла.

Отождествление окон и программ выявляет недостаток знаний о теории вычислительных систем --- процессах, межпроцессном взаимодействии. Для прояснения ситуации можно поэкспериментировать с процессами, сигналами, и т. п.

Слияние файлов и их обработчиков

Обычно означает недостаток опыта у пользователя. Даже в Windows пользователь достаточно быстро осознает, что файл --- это одно, его обработчик --- это другое, обработчики можно менять, и всё это сильно зависит от такой не очень понятной вещи, как тип файла. Причина возникновения мифа в том, что Windows определяет тип файла по расширению файла (символы после последней точки в имени), и по умолчанию не показывает расширение у известных системе типов файлов. В Linux расширения обычно показываются (да и вообще не имеют смысла, ибо определение типа файла происходит по-другому --- а именно, по начальным байтам файла, см. `man file`), отсутствует строгая типизация, естественно, в графической среде настраиваются ассоциации типов файлов и обрабатывающих их приложений. Для развенчания мифа можно показать редактор ассоциаций KDE, продемонстрировать работу утилиты `file`. Фактически, следует доходчиво объяснить, что такое ассоциирование.

Что такое операционная система

Управление ресурсами компьютера

Одно из определений операционной системы (ОС) гласит: операционная система --- это средство управления ресурсами компьютера. Можно выделить три уровня разделения ресурсов, на каждом из которых распределение и использование ресурсов компьютера происходит по-своему.

Однозадачный режим

Начальный уровень, на котором не нужны никакие дополнительные действия для обеспечения доступности ресурсов компьютера --- когда у нас есть ровно одна программа, которая полностью распоряжается всеми ресурсами компьютера, пока она не закончит свою работу. Такая система называется однозадачной, и то, насколько разумно она распоряжается аппаратными ресурсами, может повлиять только на работу единственной запущенной программы. Если программа написана плохо, например, пожирает память, то в конце концов она просто перестаёт работать, потому что оперативной памяти ей не хватит. Хорошо написанная программа должна успешно распоряжаться процессором, всей оперативной памятью и всеми внешними (по отношению к процессору) устройствами.

Много ли нужно программного обеспечения, чтобы позволить этой задаче эффективно использовать память и устройства и нужно ли создавать какое-то отдельное программное обеспечение для поддержки этого процесса? Возможный ответ --- "нет", так как запущенная программа, как правило, содержит в себе все необходимое ПО. Типичный пример -- это старинная игровая консоль (SEGA, SNES), использующая картриджи. Каждый картридж --- отдельная программа, загружающаяся и работающая по принципу однозадачной системы.

Унификация доступа

Однако даже при однозадачной системе обычно неразумно всю задачу организации доступа к ресурсам возлагать на прикладную программу, поскольку это сильно затрудняет их разработку.

Как минимум, хотелось бы унифицировать способ использования различных аппаратных ресурсов, например внешних устройств. Для решения этой задачи необходима некая прослойка между программой, которая пользуется ресурсами компьютера, и самими ресурсами. Эта прослойка будет скрывать от программы подробности того, как именно этими ресурсами воспользоваться. Рассмотрим это на примере графической платы: мы поменяли видеокарту, вместе с ней поменяли прослойку, которая обеспечивает доступ по стандартному (и не зависящему от конкретной карты) протоколу к ресурсам этой видеокарты, и прикладные программы не "заметили" изменений, хотя в действительности по шине PCI передаются уже принципиально другие данные.

Таким образом, даже на уровне однозадачной системы встает вопрос унификации доступа к ресурсам. Идея в том, чтобы ПО не принимало во внимание особенности аппаратного обеспечения, чтобы за это отвечала указанная прослойка. Зависящая от конкретной аппаратуры часть этой прослойки называется драйвером.

Многозадачный режим. Разделение доступа

Как только выясняется, что ресурсами компьютера пользуются несколько различных программ (назовём этот режим многозадачным), сразу возникает вопрос -- к каким ресурсам

какие программы имеют доступ? Простейший вариант: пусть программы выполняются по одной друг за другом -- сначала отработает первая, затем вторая и т.д. Даже в этом случае, если они работают друг за другом и не знают о существовании друг друга, возникает вопрос организации дисциплины доступа к таким внешним ресурсам, которые существуют в течение их работы, например, к накопителю данных.

Разумеется, если программы работают по очереди, нет необходимости защищать, скажем, оперативную память, но общее для всех хранилище данных на диске хорошо бы организовать таким образом, чтобы одна программа знала о том, где хранит свои данные другая программа, для того, чтобы избегать коллизий. Если это файловая система, как в Linux, неплохо, если бы программа работала с именами файлов, и любая операция с некоторым файлом должна быть заложена в программу сознательно.

Еще более очевидна необходимость разделения ресурсов, когда мы имеем многозадачную систему с параллелизмом или псевдопараллелизмом (что вообще-то почти одно и то же с точки зрения разделения ресурсов). Параллелизм возможен при наличии нескольких процессоров, одновременно выполняющих разные программы; псевдопараллелизм -- это ситуация, когда эти программы в действительности работают последовательно, но для пользователя создается впечатление, что они работают параллельно, и происходит это так: один квант времени работает одна программа, потом она приостанавливается, следующий квант времени работает другая программа, следующая, потом снова начинает работать уже первая программа, и все снова. Выполняющуюся (загруженную в память) программу называют процессом.

Псевдопараллелизм --- наиболее распространенный вариант, его еще называют вытесняющей многозадачностью. Программы не обязательно выполняются в строгой последовательности, возможно (как в случае, когда приложение ожидает завершения операции ввода-вывода) изменение порядка их работы. В целом задача планирования процессов достаточно сложна, например 2007-й год прошёл под флагом революции в области планировщиков процессов в UNIX-системах (как в Linux, так в BSD-подобных), этой теме был посвящен отдельный семинар.

Таким образом, в многозадачных системах все ресурсы компьютера --- процессорное время, оперативная память, внешние устройства -- должны быть разделены между всеми работающими программами. Например, в случае псевдопараллелизма нужно создать дисциплину распределения доступа к процессору (например, мы можем захотеть, чтобы "деж" процессорного времени был "справедливым" --- каждой программе поровну). Другим примером будет разделение оперативной памяти: нужно определить дисциплину распределения памяти, а так же организовать доступ к ОП таким образом, чтобы одна программа не могла случайно испортить содержимое участка памяти другой программы, поскольку эти программы не обязаны знать о том, что они сосуществуют вместе на одном компьютере. Третьим примером будет разделение принтера или сканера --- пока принтер полностью не завершил обработку одной заявки на печать, он не может начать обработку следующей.

Таким образом, все ресурсы, которые используют прикладные программы, должны быть явно обозначены, и должны существовать договоренности об их использовании, чтобы ситуация, когда одна программа модифицирует какой-то ресурс, принадлежащий другой программе, была исключительно штатной и контролируемой. Совокупность этих правил называют механизмом разделения ресурсов.

Разделение доступа к ресурсам

Как следует из проблем создания систем с параллельным выполнением программ, задача управления ресурсами оказывается связанной с задачей разделения доступа к ресурсам.

Поэтому второе определение операционной системы гласит: операционная система --- это средство разделения ресурсов компьютера между прикладными программами.

Разделение центрального процессора

В персональных машинах чаще всего используется всего лишь один центральный процессор. Ресурсы ЦП включают в себя не только время его работы, но и объекты, связанные с хранением данных (регистры процессора). Это значит, что когда мы организуем псевдопараллелизм, мы должны предусмотреть не только ситуацию разбиения по времени (некоторое время работает одна, а некоторое --- другая задача), но и механизм сохранения тех элементов памяти, которые не дублируются для каждой программы, от одной программы, на то время, пока работает другая программа. К таким элементам относятся прежде всего регистры центрального процессора.

Всё состояние системы (в первую очередь --- центрального процессора), которое необходимо сохранить, чтобы в будущем продолжить выполнение процесса, называется контекстом процесса.

Это что значит, что в случае псевдо-параллелизма мы имеем один процессор, набор оперативной памяти, а в очереди сидят процессы (процесс-1, процесс-2 и так далее). С каждым созданным процессом связан контекст процесса, сохранённый в той же оперативной памяти. Когда процесс дожидается своей доли процессорного времени, то перед его выполнением, контекст подгружается, в частности, восстанавливаются значения регистров процессора. Затем процесс некоторое время (обычно порядка миллисекунд) выполняется, после чего останавливается и контекст на момент прекращения выполнения сохраняется обратно. Затем подгружается контекст следующего по очереди процесса, он выполняется, и так далее. Даже такая с виду простая штука, как разделение ресурса под названием "центральный процессор" уже приводит к довольно сложным механизмам разделения доступа к этому ресурсу: квантование выполнения и сохранение контекста.

Разделение оперативной памяти

Тут ситуация чуть более простая, если в ней не разбираться, и чуть более сложная, если в ней разбираться досконально. #Никому не надо рассказывать слишком подробно, что на некотором уровне абстракции оперативная память представляет собой последовательность ячеек с номерами от 0 до самого большого значения, и сколько у нас воткнуто в плату памяти, столько и будет. Однако, в действительности всё значительно сложнее.

Существует несколько видов адресов памяти. В системах, которые ведут свое происхождение от однозадачных систем и не обеспечивают надлежащее разделение памяти между задачами (типичная такая система -- DOS, а также, с оговорками, Windows 3.*), основной являются физические адреса памяти --- одна программа занимает адреса от 1000 до 100000, другая от 120000 до 202000 байт, и так далее.

Если при этом одной программе по причине ошибки приходило в голову исписать нулями не свою, а "чужую" память (начав, к пример, в своей, а в результате ошибки убежав за границу), то тогда она убивала всю оперативную память, доступную операционной системе, и компьютер "зависал". Типичная ситуация в DOS: написал программу на языке Си с ошибкой, запустил, в итоге перегрузил компьютер. В младших версиях Windows наблюдалось то же самое, причём отдельные рецидивы были вплоть до Windows 98 (но не в линейке полноценных ОС Windows NT).

Операционные системы, которые обеспечивают нормальное разделение памяти, пользуются аппаратным страничным механизмом разделения памяти, котому уже около 30-ти лет, а в недорогих компьютерах он появился вместе с процессором Intel i386 почти четверть века назад. Устроен этот механизм следующим образом. Вся оперативная память делится на

страницы некоторого фиксированного размера (обычно по 4096 байт). Пусть запускаются два процесса в псевдо-параллельном режиме, причём никто из них не ведёт обмен с внешними устройствами, а жаждет лишь выполняться на процессоре. Для каждого создается контекст процесса, набор регистров, который нужно хранить, и все остальное, потом каждый из них запрашивает себе память, которая ему выделяется. Куски памяти, запрошенные разными процессами, с физической точки зрения перемешаны совершенно произвольно. Но реальных адресов с 0 до полного объема памяти ни один процесс не наблюдает. У него происходит аппаратно поддерживаемое со стороны процессора преобразование виртуального адреса, с которым они работают, в адрес физический.

При этом все страницы памяти, принадлежащие процессу, складываются в некоторую т.н. виртуальную память с непрерывной адресацией (допустим, объёмом 100 Мб), которую, собственно, и "видит" этот процесс. В таких случаях в понятие контекста процесса включается указанное преобразование, поскольку для каждого процесса оно своё. Когда процесс обращается к памяти, он видит только ту память, которую ему выдали, от нулевого адреса и до конца запрошенной памяти (видит он даже до физически максимального адреса). Доступ же в память напрямую, по произвольному физическому адресу, процессу запрещён.

Из скольких страниц состоит выделенная процессу память и какие у них физические адреса --- процессор не волнует, это проблема операционной системы. Пусть у нас всего 1 Гб памяти, процесс заказал 100 Мб, он 100 Мб получил, а эти 100 Мб собраны из 25600-ти страниц памяти по 4 Кб каждая.

Аналогичная ситуация происходит со вторым процессом программой, который так же видит только свою память, строго от нуля, чужую память он не видит в принципе. Таким образом, при использовании виртуальной памяти программа не может случайно испортить чужую память. По предварительной договорённости можно обратиться в вышестоящей организации и сказать: у меня есть две программы, которые используют общий сегмент памяти, разреши им пользоваться, потому что так проще данные передавать: одна запишет, другая возьмёт. Операционная система ответит: хорошо, тогда вот вам еще один сегмент, который будет подключен в адресное пространство обоих процессов (со своими адресами). Когда один процесс в него пишет, другой процесс сразу видит изменения. Но, в отличие от ситуации, когда программа расписала нулями всю чужую память, эта ситуация --- контролируемая. Вы написали программы так, что они заказали себе один и тот же сегмент, и по Вашей воле производят там изменения.

Ещё одним достоинством страничной организации памяти и виртуальных адресов является возможность записать малоиспользуемые процессом таблицы памяти на диск, а при обращении к ним считать их обратно в память, причём все это совершенно незаметно для процесса. Так устроена подкачка памяти с диска ("своп").

Разделение внешних устройств

Если система многозадачная, то разделения требуют вообще все ресурсы, не только процессор как время и как недублируемая память, и не только оперативная память как объект желаний процессов. Разделения требуют, например, звуковая карта, т.к. если несколько программ пытаются играть звук, это, во-первых, отвратительно слышно, во-вторых, если не организован процесс разделения, то первая программа может получить доступ к ресурсу, а все остальные могут его и не получить. Если операционная система предоставляет некоторый интерфейс к звуковой карте, это значит, что либо в каждый момент времени по отношению к звуковой карте она "однозадачная": первая по порядку запуска программа получает полный доступ, а всем остальным не хватило, т.е. имеет место монополярный доступ, либо она как-то решает проблему разделения звуковой карты. Для решения этой проблемы система может поддерживать несколько виртуальных звуковых карт, а каждой процесс будет работать со своей виртуальной картой. Затем система организует наложение

друг на друга звуков от нескольких процессов и передачу результатов физической звуковой карте. То же самое касается видеокарты, периферийных устройств и так далее, причём в случае принтера "наложение" недопустимо, а в случае видеокарты происходит сплошь и рядом.

Ограничение доступа

Задачи операционной системы не ограничиваются обеспечением доступа к ресурсам и их разделением между программами. В случае многозадачной системы постоянно возникает ограничить ряд процессов таким образом, чтобы они не смогли получить доступ к каким-либо ресурсам в принципе. Пример: допустим я храню пароли в текстовом файле и не хочу, чтобы их мог подсмотреть процесс, запущенный любым другим пользователем и вообще любой процесс, которому я не доверяю.

В результате решения этой проблемы мы приходим к тому, что система не просто многозадачная, но и задачи бывают разные с точки зрения уровня доступа. Про одни задачи известно, что им можно иметь доступ к некоторому ресурсу, а про другие известно, что нельзя. В простейшем случае задачи делятся на группы, принадлежащие разным пользователям, причём это могут быть как фиктивные системные пользователи, так и пользователи, связанные с пользователями-людьми. Одной группе задач пользователь разрешает иметь доступ к некоторому ресурсу, а другим группам --- не разрешает. И мы переходим от системы многозадачной к системе многозадачной и многопользовательской.

Если в случае многозадачной системы все задачи была с равными правами и главной целью системы было обеспечить невозможность случайного доступа к чужим ресурсам, то в случае многопользовательской системы разные задачи могут иметь разные права, и основной целью будет недопущение не просто случайного, а несанкционированного доступа к ресурсу. Если один пользователь не хочет, чтобы задачи другого пользователя имели доступ к ресурсам первого пользователя, то он это должен уметь запретить. Это задача называется ограничение доступа к ресурсам.

Что такое операционная система

Мы получаем три группы задач, которые должна решать операционная система:

- унификация доступа;
- разделение доступа;
- ограничение доступа.

Фактически, это определение операционной системы: операционная система --- это программное обеспечение, которое обеспечивает выполнение трёх указанных задач для того аппаратного обеспечения, на котором эта ОС запущена. Не надо также забывать о том, что наша задача -- повысить удобство работы с компьютером. Ради чего были сделаны операционные системы? Ради того, чтобы человек, который создаёт прикладные программы или их запускает не занимался решением указанных задач самостоятельно и вручную.

Вообще говоря, нет полностью удовлетворительного определения ОС. Более того, если уйти в историю, мы поймём, что пользуемся этим словом неправильно. Operating system -- это не ПО, это систематическое (system) руководство по использованию (operating) компьютера, совокупность методов. Как правило, эти методы программируются и получается программное обеспечение, но смысл термина изначально был именно такой.

Примеры

Из приведённого определения выкинуты подробности отдельных существующих

представителей операционных систем. Хотелось бы избежать ситуации, когда под операционной системой понимается нечто с кнопкой "Пуск" в нижнем левом углу экрана. Во-первых, никакого экрана может вообще не быть, поскольку нет графических ресурсов у данной аппаратной платформы (ближайший пример --- домашний ADSL-модем или маршрутизатор). Во-вторых, может быть экран есть, но нету графической кнопки для нажатия на нее мышкой, т.к. может случиться, что эта аппаратная платформа управляется без помощи графических инструментов, а исключительно подачей ей команд, а всякая графика -- прикладная, а не интерфейсная (пример: интерфейс управления самолётом).

Кроме того, термин "ОС" применялся и к сущностям, которые трудно отнести к операционным системам. Приведём пример DOS, всё ещё знакомый многим. Почему DOS -- это не ОС? DOS -- это, дословно, дисковая операционная система, если кто помнит. Многозадачная (в памяти могло быть несколько программ, но без квантования времени и вытеснения задач), а так же однопользовательская. DOS не умеет обеспечивать защиту памяти, со стороны ОС не предоставляются возможности обеспечить даже защиту от случайного, непреднамеренного доступа. Не предоставляет нормальных способов разделения таких ресурсов как видео, аудио, или даже последовательный порт. Для решения этих задач каждый разработчик должен был написать специальную программу-драйвер или, что еще хуже, вписывать в свою программу работу непосредственно с регистрами видеокарты, только после чего всё и начинает работать.

Единственные инструменты по унификации и разделению ресурсов, которые предоставляет DOS пользователю, это файловая система, но она и в те времена вызывала уже много нареканий. Таким образом, DOS содержит только зачатки функций операционной системы. В настоящее время существуют специализированные операционные системы, которые, несмотря на своё название, не выполняют всех поставленных нами задач, поскольку являются компромиссным вариантом и пытаются достигнуть, например, наименьшего времени отклика системы. Все современные ОС общего назначения, в том числе линейка Windows NT, GNU/Linux и BSD-системы вместе с Mac OS X на все поставленные здесь вопросы отвечают полностью утвердительно.

Демифологизация: продолжение

Контрафактное ПО

Для прояснения этого мифа рекомендуется рассмотреть свободное и несвободное ПО (см. первую лекцию школы), при этом, возможно, имеет смысл обратить внимание на свободное программное обеспечение в качестве основного примера. Основные легенды:

1. Ни одну программу копировать нельзя.
2. Программное обеспечение работает плохо потому, что оно контрафактное.
3. Так как свободное программное обеспечение бесплатно, то оно производится неквалифицированными энтузиастами, и как следствие, ничего не готово к использованию прямо сейчас. В этом случае надо показать процесс разработки, надо сказать, что действительно есть много незаконченных проектов, но, если судить по качеству, готовности и так далее, ситуация схожа с ситуацией с несвободным программным обеспечением, но модель распространения и использования --- другая.

Обновление системы

Существуют два мнения:

- Операционную систему Windows не надо обновлять, поскольку это может нарушить её работоспособность. В это есть смысл, так как установлено много сторонних программ, и они могут пострадать в результате обновления всей системы.
- Система без обновления обязательно имеет проблемы с безопасностью. Это не легенда, а правда. В случае с GNU/Linux критических дыр очень мало, одна в несколько лет, и хороший дистрибутив обновляется через несколько часов после появления такой ошибки.

Установка и обновление программ

Если Вы устанавливаете программу в дистрибутив или хотите обновить уже установленную программу, всё зависит от того, что это за более новая программа. Если это критические обновления, то они наверняка будут в updates. Причем эта программа будет совместима с предыдущей версией. Если же ставится версия, не проверенная на совместимость с предыдущей, то это вина того, кто ее установил. В случае, если Вы берёте новый непроверенный пакет из хранилища, лучше брать src.rpm и собирать его в своём окружении. Если это удалось, значит, всё хорошо, если не собрался, то правильно сделали, что не ставили.

- Эшелонирование источников ПО ОС Linux:
 - Дистрибутив, в нём достаточно много программ, чтобы не приходилось идти дальше.
 - Репозиторий, там есть ещё больше программ.
 - Интернет (стороннее ПО). Это уже более рискованно, даже более рискованно, чем установка проверенного ПО в ОС Windows, кроме случая, когда программа собиралась для той ОС той версии, которую вы используете.

В рамках обычных пользовательских потребностей все это, в целом, очень просто, так как довольно мало свободного программного обеспечения не входит в репозиторий.

Вирусы

О вирусах существует несколько мифов:

1. В 2000х годах в рамках рекламы ОС Windows и, соответственно, антирекламы ОС Linux Microsoft, стараясь завоевать серверный рынок, утверждали, что ОС Linux разные, а ОС Windows всегда одинаковый. Это действительно так, и одним из следствий этого является то, что программа, эксплуатирующая одну уязвимость некоего программного продукта в одном дистрибутиве, скорее всего, не будет работать в другом дистрибутиве или даже в том же дистрибутиве через несколько недель или даже дней.
2. Обычный пользователь в ОС Linux выполняет всего две операции над всей системой с правами администратора: установка ПО и некоторые редактирования конфигурационных файлов. Таким образом, помимо Вашей воли нарушить работу системы нельзя никак. Если каким-то образом случится так, что проэксплуатируется ещё не закрытая уязвимость в какой-то программе и будет запущен вредоносный код, то он скорее всего не сможет сделать ничего плохого с системой, потому что каждый демон, отвечающий за некоторый сервис, запускается от лица специального пользователя, сильно поражённого в правах. А если он запускается с правами суперпользователя, то в изолированном окружении.
3. Вы скачали и установили ПО, содержащее зловредный код. В каком случае это может быть? У вас задача, решения которой нет в хранилище, но есть некий неизвестный производитель программ, который выпускает нужный продукт. Для начала, стоит проанализировать задачу --- зачастую её можно будет решить при помощи находящегося в репозитории программного обеспечения. Если же программа стороннего производителя настолько уникальна, что без неё обойтись никак нельзя, можно, тем не менее, предпринять некоторые шаги по сохранению безопасности системы. Как-то, запускать программу в изолированном окружении, запускать программу от лица специального поражённого в правах пользователя, если это Windows-программа --- настроить окружение wine таким образом, что эта программа не сможет навредить нигде, кроме заданного каталога.
4. Не последнюю роль играет также и то, что ОС Linux - нераспространенная и трудновзламываемая ОС. То, что называют "взломами" для ОС Linux, в большинстве случаев есть уязвимости в некотором программном обеспечении. И если эти уязвимости представляют серьёзную угрозу безопасности, то они будут устранены весьма быстро --- иногда, в течение нескольких часов.

Тем не менее антивирус для ОС Linux есть --- это ClamAV. Под ОС Linux есть и Антивирус Касперского, и Dr. Web, однако они не всегда корректно работают: это несвободные программные продукты, а у авторов нет возможности адаптировать свои программы *для всех* дистрибутивов Linux *каждый раз*, как они обновляются. Назначение антивируса а Linux -- проверить почту и файлы при передаче их пользователям Windows.

На сегодня главное достоинство антивируса -- это не количество знакомых ему вирусов, а оперативность обновления базы данных, по которым определяются *новые* вирусы. Чем раньше новый вирус станет известен, тем больше шансов не "пропустить" вирусную атаку. ClamAV здесь показывает неплохие результаты, находясь, по некоторым сведениям, в тройке лучших по оперативности добавления информации о новых вирусах.

Firewall

Существует убеждение, что в системе должна быть сущность под названием Firewall, которая постоянно задает вопросы вроде "А хотите ли Вы, чтобы к вам присоединились?" или "А хотите ли Вы, чтобы какая-то программа куда-то присоединилась?". Откуда такое представление? Во-первых, человеку абсолютно бесполезно объяснять, что такое стэк протоколов TCP/IP, во-вторых, в ОС Windows много входящих портов и служб, в которых есть проблемы с безопасностью, оставлены открытыми, и в этой ОС необходимость наличия такой сущности очевидна. С другой стороны, это связано с предыдущим пунктом ---

существует масса ПО вирусного типа, которое устанавливается без санкции пользователя, и делает нечто незаметное. И Firewall играет роль некоего сторожа, который это отслеживает, и, если что-то кажется ему подозрительным, он требует разрешения пользователя на это действие.

Программы подобного класса в ОС Linux есть, например, Suse personal firewall. Но необходимость подобного ПО очень и очень сомнительна, так как нет необходимости контролировать входящие и исходящие подключения --- по умолчанию нет ни одной службы, которой можно подключиться снаружи. И если администратор что-то открыл, то он что-то открыл, сознавая это, и это его дело. Используется же firewall, который с точки зрения системы выглядит иначе: это таблица правил, указывающая, каким образом поступать с какими пакетами в различных сетевых соединениях на различных уровнях сетевого взаимодействия. В ОС Linux таким firewall-ом является iptables.

Оптимизация

Наличие оптимизаторов --- явный признак того, что систему можно привести в состояние, требующее оптимизации.

- Оптимизация системы.
 - В ОС Linux существует только одна возможность заполнить ненужными пакетами систему: если устанавливаемая программа зависит от других программ, то они не будут автоматически удалены при удалении самой программы. Таким образом, единственный способ внести некоторый беспорядок в систему --- постоянно что-то ставить и удалять, после чего останется некоторое число пакетов, которые не используются. Но их вполне можно найти и удалить штатными средствами.
 - Что же касается ОС Windows, то в ней практически невозможно составить гармоничный список того ПО, которое ставится на компьютер. Это усугубляется реестром, который является базой данных, содержащей информацию обо всех настройках системы. Реестр практически не документирован и слишком велик, чтобы держать все настройки в голове. Он очень часто приходит в замусоренное состояние и крайне неудобен в использовании человеком. В ОС Linux есть каталог /etc, в котором хранятся конфигурационные файлы. Конфигурационные файлы принадлежат пакету, при удалении установленного пакета они либо удаляются, либо откладываются с другим именем. Таким образом, невозможно перепутать действующий конфигурационный файл с отложенным.
- Оптимизация диска.
 - Файловые системы в ПСПО (ext3) устроены так, что оптимизация им не требуется. Платится за это порогом оптимизации --- 10% - т.е. они всегда неоптимальны, но не более, чем на 10%. Она устроена следующим образом: файл расположен в пределах одной группы цилиндров (цилиндр --- это элемент диска на уровне более низком, чем файловая система), внутри которой дефрагментация не требуется (фрагментация просто-напросто отсутствует). Когда система пишет файл на диск, она пытается записать его так, чтобы он попал в одну группу цилиндров. Если файл не влезает в группу цилиндров, это значит, что просто не хватает места, и она записывает остаток файла в другие группы, по возможности соседние. Если же удалить часть файлов и поработать с остальными, то фрагментация, точнее, её отсутствие, восстановится. Кроме того, небольшая часть диска всегда свободна - её может заполнить только root.

Администратор

Во всех POSIX-системах, в частности, ОС Linux, в частности, ПСПО, есть 2 категории пользователей. В первой --- root (не администратор, как в ОС Windows), во второй -- все остальные. На суперпользователя (root-a) не распространяются никакие ограничения по правам доступа (но при включении неких дополнительных мер безопасности это может быть и не так). В результате, он имеет права на запись к большинству файлов в файловой системе. У пользователя есть доступ только к домашнему каталогу. ОС Linux организована так, что пользователю не нужно никуда, кроме как в это каталог, что-либо записывать. Работа с ПО происходит от лица обычного пользователя всегда, даже если это служба. Обычному пользователю не рекомендуется выполнять действия от лица суперпользователя, не зная, к чему они приводят. От лица root-a запускается две операции: настройка сети, графики, системных служб и установка/удаление программ (Alterator и Synaptic). Но и то, и другое происходит не в результате работы самой программы с правами суперпользователя сразу, а запуском специальных программ, которые эти права получают. При попытке запустить графическое окружение от лица суперпользователя будет показано предупреждение.

Отдельно надо упомянуть об администраторах служб: администратор принтерного сервера, БД и прочее. Для этого зачастую не нужно получать права суперпользователя, всего лишь состоять в особой группе, а даже если нужно, то всё сводится к изменению конфигурационных файлов или запуску специальных приложений, что можно эффективно ограничить (например, при помощи программы sudo).

Не следует путать root-a и администратора Windows. Администратор с точки зрения ОС Linux это не регистрационное имя, а человек. Он работает большую часть времени с правами обычного пользователя и изредка совершает некоторые действия от лица суперпользователя (собственно говоря, он вообще имеет возможность их совершать), и обычно эти действия связаны с настройкой системных служб.