Развитие Linux: куда теперь?
Пути Linux неисповедимы
Дмитрий Царьков aka ddc, Четверг, 19 Январь 2006, 02:52

Preface

<P align=right>Снарки, в общем, безвредны. Но есть среди них.
(Тут оратор немного смутился.)
Есть и БУДЖУМЫ...' Булочник тихо поник
И без чувств на траву повалился.

Л. Кэррол, "Охота на "</P>
Так вот, как я уже писал выше, Linux в последнее время стремительно развивается. Растёт пользовательска база, ширится круг доступных приложений, увеличивается рыночная доля и появляются новые компании-поставщики решений Linux. Машина катится вперёд, причём делает это так стремительно, что непросто заметить, что именно находится впереди. А находиться там может многое.

Направлений в сущности два: направление MacOS X (далее иногда будет употребляться слово OSX'ификация), представляющее собой некое централизованно законченное рабочее окружение с чёткой направленностью на предельную визуализацию и наглядность пользовательски действий, и направление классической UNIX, ориентированноена маленькие "" выразительные утилиты, позволяющие легко комбинировать друг друга для быстрого достижения сложных результатов.

Тенденции развития Linux заключаются в постепенном смещении центра тяжести от последовательно реализации концепции UNIX (что было характерно, условно говоря, в 90 годы) к тотальной визуализации, которая характерна для многих новых приложений.

Но я считаю, что этот путь - это путь в никуда. Пытаясь идти вслед за Windows и MacOS X Linux обречена оставаться второй. Просто потому, что есть Apple и есть Microsoft, которые уже идут впереди, и первенство которых обеспечено тем объёмом ресурсов, которые тратятся на исследования в релевантных этому пути развития областях. И Linux, если пойдёт по их стопам, на их фоне всегда будет выглядеть дешёвой подделкой.

Напротив наследие UNIX предлагает совсем другой подход к работе. Причём этот подход отличается не тем, что кучка фанатов привыкла к нему и жить без него не может (вспоминаем MS DOS и Lexicon), а принципиально другой организацией работы, которая позволяет достичь тех же результатов, что и работа с мощнейшими приложениями Windows и MacOS X, при применении другого стиля работы.

[newpage]

Химеры, или Preface II

<P align=right>Л.И. Брежнев, открывая заседание политбюро:
В последнее время часто говорят, что вместо меня в машине возят чучело. Так вот, я ответственно заявляю, что это - неправда. Это чучело в машине и есть я!

Бородатый анекдот</P>
Из комментариев к первой редакции этого писания я сделал вывод, что читатель воспринимает мои мысли по поводу путей развития Linux как попытки назвать продукты Microsoft, WYSIWYG-приложения и т.д. плохими. Так вот, как Вы уже наверно поняли из эпиграфа, это не так.

Я действительно не люблю продукты Microsoft и WYSIWYG-редакторы, но я не хочу сказать, что это плохое ПО. Этот подход имеет свои плюсы и свои минусы, имеет потенциальных потребителей, рыночное обоснование существования и т.д., просто я действительно не понимаю, зачем это всё нужно переносить в Linux.

UNIX-way

<P align=right>Но где она живёт,
Вечная любовь -
Слепое знамя дураков?

"Агата ", песня "Вечная лю"</P>
Итак, что объединяет методологию работы в UNIX-подобных ОС?

1. единое файловое пространство;
2. маленькие "" утилиты;
3. взаимодействие программ;
4. стандарты и форматы;
5. функциональност в ущерб простоте в использовании;
6. изящество в решении задачи.

Чтобы не вдаваться в словоблудие, я решил попытаться раскрыть каждый из этих принципов в соответствующем "case study".

Единое файловое пространство

В UNIX есть один корень файловой системы, к которому всё монтируется. Любой файл файл файловой системы имеет выразительное название и выразительный путь, так что имея сведения о содержимом файла, можно с очень высокой точностью определить его название и местонахождение Файлами является всё; если что-то не является файлом, его нет в UNIX.

Такой подход в целом прослеживается во всех приложениях всех дистрибутивов UNIX. Но не все йогурты одинаково полезны! В качестве примера вредного йогурта возьмём монтирование CD в дистрибутиве SuSE.
[newpage]
Когда Вы вставляете диск в дисковод, система автоматически монтирует его, давая ему имя, соответствующее его заголовку. Не будем касаться такого интересного вопроса, как некорректные имена заголовков - людям, не живущим в условиях четырёх "" локалей этого не понять, остальные это знают и без меня. Куда интересней ситуация, когда всё работает

именно так, как задумывалось.

Вот вставил я диск с фильмом "Апокалипсис сегодня" в дисковод, и появился у меня в дереве каталогов такой путь: "/mnt/APOCALYPSE_NOW". И каков итог? Нарушена концепция файловой системы UNIX: название каталога не соответствует его назначению, да ещё и постоянно меняется, так что несложный скриптик для работы с содержимым диска, который отлично работал бы в любом менее экстравагантномдистрибутиве без изменений веками, должен редактироваться каждый раз или неоправданно усложняться.

Вывод: попытка сделать пользователю Windows "как " обратилась в истязание ортодоксальных UNIX'истов и вообще poweruser'ов.

Маленькие "" утилиты

Программы в UNIX умеют справляться только с какой-то узкой задачей. Т.е. текстовый редактор умеет редактировать текст, но не умеет редактировать картинки. Чем меньше возможностей даёт программа, тем она лучше. Чем меньше строк кода в этой программе, тем она качественней. Чем меньше программ дублируют функционал друг друга, тем совершенней система.

И тут стоит сравнить GStreamer с одной стороны и xine-lib с другой. Что представляют собой эти программы? Это типичные конвертеры: на входе принимается некий файл, который переводится на выходе в формат RAW audio и отправляется в звуковую подсистему. Но КАК они это делают! xine-lib - это монолит, который будет заниматься сразу всем: и звук, и видео, причём во всех форматах и сразу. Напротив, GStreamer полностью разделён: есть библиотека GStreamer, которая вообще не обрабатывает какие бы то ни было аудио- или видеоданные, и есть куча gst-plugin-*, каждый из которых занимается только своим делом, каждый из которых развивается независимо от остальных и не зависит от проблем остальных. Фактически GStreamer представляет собой стандартный интерфейс к кодекам.

Вывод: Xine-lib является очень неплохим программным продуктом, но GStreamer является более правильным ПО, и потому достоин большего.

[newpage]

Взаимодействие программ

Когда требуется совершить сложную операцию, пользователь UNIX задействует сразу много программ. При этом система упрощает жизнь пользователю, позволяя использовать pipes - механизм непосредственно передачи информации с выхода на вход.

Примером правильного поведения здесь можно назвать почтовый клиент mutt, вся деятельность которого сводится к работе с архивом почтовых сообщений и автоматизации взаимодействия любимого текстового редактора пользователя, его любимой программы отправки почты, любимой программы получения почты, программой шифрования и т.п., так что собственная функциональност программы сводится к взаимодействию с пользователем.

Примеров нарушений можно привести бесконечное множество. В качестве самого показательного мы возьмём OpenOffice.org, который решает все вопросы внутренними средствами, из-за чего пользователям приходится запоминать специфические подстановочные знаки.

Вообще, свои мелочные счёты с окружающим миром сводят почти все. Так, для меня всегда было загадкой, зачем авторы браузеров так старательно задают внешний вид своих детищ? Пользователь всегда лучше знает, какие сочетания цветов, шрифты и пиктограммы ему нравятся больше, так зачем над ним издеваться, заставляя искать и скачивать подходящую к оформлению его оконного интерфейса тему (которая ещё может и не подойти в связи с несовместимость между разными версиями программ). Красивыми решениями в этом смысле можно назвать Konqueror (в случае KDE) и Dillo (в случае GTK+), которые не навязывают пользователю собственное понимание красоты, а просто используют стандартные элементы управления и стили.

Но пользовательски характеристики - это только одна сторона медали. А вторая заключается в том, что код имеет свойство давать сбои. И эти сбои надо вычищать. И уж точно меньше сбоев будет давать код, который уже несколько лет интенсивно вычищается. И это лишний раз говорит о том, что надо внимательней относиться к чужим достижениям.

Вывод: повторное использование кода является неотъемлемой частью эффективной модели программировани.

Стандарты и форматы

Из механизмов взаимодействие программ вытекает ещё одно требование: чтобы две программы взаимодействоваи, они должны понимать друг друга. А чтобы любая программа понимала любую другую программу, все данные на входе и выходе должны быть приведены в предельно простой и стандартный вид.

Примером соблюдения правила может служить то, что классические текстовые редакторы и утилиты для работы с текстом используют один и тот же формат - "plain text" - текст, читаемый в любом редакторе.
[newpage]
Это парвило стало нарушаться в связи с массовым заготовлением Linux для офисной работы. Де факто стандартом документа является бинарный формат DOC, используемый в Microsoft Word, в котором содержатся файлы пользователей. Не намного лучше и родные форматы KOffice и AbiWord.

Но есть претензии и к родным форматам OpenOffice.org... Как известно, файлы SXW и ODT представляют собой набор сжатых XML файлов. Здесь меня более всего беспокоит слово "". Понятно, зачем это делалось: экономия места и объединение файлов, из которых состоит документ, в один файл. Однако в результате мы получаем формат, который не может быть прочитан на произвольном компьютере: у пользователя должна быть утилита unzip, которая далеко не всегда входит в дистрибутивы Linux, да и сквозь теги, используемые в OpenOffice, продираться не слишком просто.

Вывод: если есть возможность сохранять данные так, что пользователь может прочесть их, пропустив файл через пейджер, эту возможность необходимо использовать.

Функциональност в ущерб простоте в использовании

Если уж утилита из UNIX взялась выполнять определённую работу, то она будет выполнять её настолько близко к пожеланиям пользователя, насколько это возможно. Т.е. UNIX программа скорее предоставит 2000 опций командной строки, чем сделает невозможным какой-либо приём. При этом делается упор на вменяемость пользователя, который должен быть способен выбрать из возможностей программы нужные именно ему и использовать их.

Примером предельно правильного приложения можно назвать EMACS - специальная среда (framework) для редактирования текста, которая обладает столькими функциями, что едва ли хотя бы один из разработчиков или пользователей знает их все. Здесь и подсветка синтаксиса, и интеграция с утилитами проверки кода, отладки, работы с CVS и т.п.

В качестве примера на несоблюдение этого принципа приведём типичное приложение среды GNOME. Боясь напугать неопытного пользователя, авторы GNOME с каждым релизом этого окружения всё уменьшают и уменьшают возможности по настройке. К примеру, мы лишились возможности независимо устанавливать фоновые рисунки для рабочих столов. При этом разработчики всё продолжают и продолжают ограничивать пользователя в свободе выбора. Почему-то многие разработчики сконцентрировалсь на начинающих пользователях, считая, что всё в порядке. И не было бы ничего плохого в заботе о новичке, если бы эта забота тихой сапой не превратилась в парадигму пользовательскоо интерфейса изрядной доли программ. К сожалению, авторы не осознают последствий своих действий: система, ориентированная на новичков, культивирует дилетантство.

Вывод: функциональност не должна ограничиваться пригодностью для начинающего пользователя.

[newpage]

Изящество в решении задачи

За всё время существование человечества были выработаны две методики выполнения задачи, которые можно было бы условно назвать "изящное " (т.е. решение, которое является структурно согласованным, концептуально проработанным и академически правильным) и "решение, которое " (т.е. минимально достаточное для выполнение задачи). Так вот, мир UNIX всеми силами отбивается от второго подхода.

Здесь "героем дня без " выступает конгломерат утилит от Mozilla Foundation. Так был раньше набор программ Mozilla Suit, который, несмотря на полную неспособность работать так, как это должна делать программа в UNIX, представлял собой неплохой пакет, состоявший из движка обработки HTML, браузера, почтовой программы, WYSIWYG HTML-редактора, IRC-клиента и календаря. Этот монстр представлял себе ком в горле любой UNIX-подобной системы, поскольку нарушал все правила: случайным образом генерирующиеся имена

постоянных файлов, чёткая неспособность взаимодействоваь с внешним миром иначе чем по-разному откликаться на специфические ключи вызова, собственный формат почтового ящика, неполная поддержка CSS при наличии собственных CSS-свойств и достаточно ограниченная функциональност делали это приложением камнем на шее сообщества UNIX'истов. Однако этот пакет имел очень красивую внутреннюю структуру: не было двух частей, отвечающих за одно действие, наличествовала модульность, выражавшаяся во взаимоотношениях частей программы между собой и с плагинами. Но коварный удар в спину испортил идиллическую картину.

Теперь Mozilla Foundation продвигает три собственных продукта (давайте сделаем вид, что NVu не имеет ни малейшего отношения к продуктам Mozilla Foundation): браузер Firefox, почтовый клиент Thunderbird и календарь Sunbird. Прелесть ситуации заключается в том, что разделённый набор (а рассматривать как набор это всё всё равно стоит, ибо поклонники Mozilla Suit предпочитают осколки пакета иным приложениям; добавляя корректности в сравнение Mozilla Suit очень активно отмежёвывалась от образа монолита: каждое из приложений можно было не устанавливать, так что функционально Mozilla Suit с лёгкостью ограничивалась до своих последователей) не просто не стал лучше, а только ухудшился. Браузер Mozilla Firefox потерял половину функций Mozilla Navigator а совокупный размер трёх частей программы стал превосходить суммарный размер исходного пакета. Но не зря я описываю это всё в секции, посвящённой посвящённой изящности решения: новый пакет перешёл из категории отлично структурированнго и качественного ПО в качественное ПО с колоссальной брешью на уровне проектирования. Как иначе можно охарактеризоват тот факт, что одновременно установленные Mozilla Firefox, Mozilla Thunderbird и Mozilla Sunbird используют три различных версии Gecko (кстати говоря, не пересекающиеся функционально) и три различных интерпретатора языка плагинов?

Вывод: перед написанием кода надо думать.

[newpage]

Кроме того

<P align=right>Любое дело можно сделать тремя способами: правильно, неправильно и так, как это делают в армии.

Офицерский анекдот</P>

Два слова о WYSIWYG

Традиционная парадигма редактирования текста под UNIX предполагает использование языков разметки. Так, само редактирование осуществляется посредством написания текста и нанесения на него разметки, соответствующей желаемой структуре и желаемому оформлению, после чего результат достигается компиляцией полученного файла. Однако пользователям Windows с детства близка и знаком обратная парадигма - WYSIWYG. Этот стиль редактирования текста предполагает редактирование текста, который компилируется в процессе ввода; действия разметки же осуществляются посредством объектной работы с текстом: пользователь выделяет фрагмент текста и меняет его атрибут.

Принцип WYSIWYG не прижился в классических UNIX'ах потому, что снижал роль

пользователя в оформлении текста: сама программа определяла, каков режим атрибутов вводимого текста, как обрабатывать текст с множественными изменениями атрибутов в небольшом врагменте (попробуйте угадать, как трактует Microsoft Word последовательно одинаковое изменение одного и того же атрибута у двух соседних фрагментов текста!) и т.п. Между тем, новые пользователи Linux, перешедшие из Windows, привыкли иметь дело именно с WYSIWYG-редакторами, и теперь испытывают в них потребность.

Для оценки этой ситуации следует постараться понять, почему именно так сложилось, что в мире Windows оказался популярен подход WYSIWYG, а в UNIX - текст с разметкой. В UNIX такой выбор был продиктован исключительно озвученной выше концепцией: именно текст с разметкой позволял использовать один единственный любимый текстовый редактор для любого вида текстов, при этом максимально контролируя результат и используя максимум возможностей среды - ведь работа с "гладкими " в UNIX'ах оказалась особенно эффективна в силу особенностей механизмов взаимодействия программ. В Windows таких строгих конвенций не было, но если UNIX проросла на домашние компьютеры с высокопроизводиельных компьютеров, пользователями которых были технические пользователи, имевшие потребность в расчётах, а значит обладавшие техническим складом ума; Windows же проросла из DOS, обитавшей на IBM PC, т.е. компьютеров для широкого круга не-технически мыслящих пользователей. Этим людям всегда хотелось оказаться минимально вовлечёнными в работу на компьютере, а потому программное обеспечения для них было расчитано на минимальную мнемоническую нагрузку, достигавшуюся за счёт меньшего функционала и большей наглядности. При этом, если в UNIX'ах подход к работе "решение, которое " карался, то в Windows он наоборот культивировался, поскольку обратное поведение выдавливало пользователя из-за компьютера, чем достигался предельно негативный для всех эффект. И поэтому в среде пользователей Windows до сих пор (и вопреки всем попыткам насаждения цивилизации) считается нормальной практикой форматирование текста посредством повторяющихся пробелов и/или разрывов строки. Кроме того пользователями WYSIWYG редакторов практикуются многократные изменения небольшого количества атрибутов на ограниченном участке текста с целью побочным эффектом таких действий достичь нужного форматирования; эта практика приводит к тому, что однажды сохранённый текст открывается не в том виде, в котором был закрыт, поскольку некоторые действия, проделанные в предыдущий сеанс работы с этим текстом, при новом открытии не были повторены, и их побочный эффект не оказал влияние на форматирование документа.
[newpage]
Казалось бы, всё перечисленное выше является исключительно проблемой соответствующих редакторов, но это не так. Пользователь, завлечённый в лоно WYSIWYG-редактора ошибочно воспринимает его наглядность как простоту, не замечая времени, потраченного на приведение документа в нужный вид и отладки результата. Такой пользователь будет считать, что WYSIWYG-редактор точнее отвечает его требованиям, чем редактор, обеспечивающий комфортное редактирование текста с разметкой, и в результате пользователь сделает неверный выбор. А статистическое множество таких пользователей (а приток пользовательско массы происходит именно за счёт них) оказывает влияние на концепцию ПО в целом, так что система постепенно теряет свои свойства, и приложения типа Mozilla Firefox и OpenOffice.org воспринимаются не как катастрофически монстры, а как правильные, удобные и перспективные программы.

KDE

Редкая тема может похвастаться такой популярностью как обсуждение K Desktop Environment. Внесу свою лепту и я.

Почему-то бытует мнение, что KDE - это куча монолитного кода с претензиями на звание Windows II. Это не так. Более того, это строго противоположно действительност.

Давайте посмотрим, что такое KDE. Фактически это рабочее окружение представляет собой набор базовых библиотек (kdelibs), система организации взаимодействия процессов (DCOP), несколько back-end'ов (aRTS, KHTML, KATE и т.д.) и куча front-end'ов (то, что мы видим в меню). При этом мы видим, что с помощью DCOP front-end'ы легко связываются с back-end'ами, сокращая количество кода и затраченного времени. Т.е. мы видим классическую схему организации тесно интегрированнойсреды с открытым входом - DCOP легко подхватит любое приложение, которое способно его запросить. При этом гармонично соблюдаются все принципы UNIX:

1. Единое файловое пространство не нарушается KDE. Названия постоянных файлов и путей к ним логичны и понятны, настройки, исполняемые файлы и прочие данные программ находятся в самых очевидных местах.
2. Маленькие "" утилиты фактически являются способом существования KDE. Здесь нет "швейцарских армейских ", все программы делают именно то, что следует из их названия.
3. Взаимодействие программ также на высоте. Чтобы не быть голословным, приведу пример. В KDE есть целая масса редакторов: KEdit, KWrite, Kate, Quanta. К этому списку можно добавить внешнее ПО - KDevelop и Kile. Так вот, функция редактирования текста во всех них отдана компоненту "Embeded Advanced Text Editor". С другой стороны, Konqueror является одновременно front-end к библотекам KDE, отвечающим за файловую систему, к aRTS, к EATE, к KHTML, к Konsole и к библиотекам графики. И, что особенно важно, всё это имеет единообразную настройку, как в области оформления, так и в области используемости. И, что самое главное, приложения KDE открыты не только для других приложений KDE, но и для всего внешнего мира: к примеру aRTS может использоваться как звуковой синтезатор и теми приложениями, которые и не в курсе существования KDE. Именно это обстоятельство позволяет нам расставить в разные углы такие комплексы как полностью самодостаточныйпакет Mozilla Suit и настолько же открытое окружение KDE.
4. Стандарты и форматы в KDE выдержаны на высоте: все форматы открыты и предельно просты для восприятия; более того, все настройки, которыми оперирует KDE, содержатся во вполне стандартных для UNIX-подобных систем файлах конфигурации в формате "OPTION=VALUE", причём конфигурационны файлы разбиты таким образом, чтобы пользователь при желании мог быстро найти нужный файл конфигурации и отредактироватьего вручную.
5. Вопросы соотношения функциональност и простоты решены очень специфически. К примеру, пользователю предлагается три текстовых редактора на выбор: KEdit, KWrite и Kate, причём все три являются фактически EATE с разной степенью настраиваемостии функциональност, так что функциональност предоставлена в полном объёме, тогда как уровень доступности выбирается пользователем. Правда вызывает нарекания пакет kdemultimedia, который позволяет более-менее детально настраивать опции воспроизведения. Однако в проект KDE входят такие программы как KMPlayer и amaroK, которые могут быть соотнесены с Noatun и JuK соответственно так же, как Kate с KWrite.

В свою очередь минималистом выступает Kaboodle, который вообще ничего не позволяет кроме банального воспроизведения.

6. Ну что может быть изящней KDE? Куча мелких back-end'ов, ловко привязываемых гибкой и конфигурируемойсистемой связывания к front-end'ам, настройка которой облегчена до предела благодаря единообразному управлению. При этом сохраняется единообразие пользовательскоо опыта, что само по себе похвально.

Также хотелось бы отметить, что даже такие 'UNIX'истские задачи, как WYSIWYG-редактирование, выполняются в приложения KOffice предельно изящно, обеспечивая всё то же единообразие пользовательскоо опыта, экономию кода и систематичность Т.е. едва ли какое-либо графическое окружение может похвастаться близостью к концепции UNIX, достигнутой KDE.

[newpage]

Обратно KDE, или True Linux GUI

По интернету блуждает статья Виктора Вагнера True UNIX GUI. Статья очень красивая и правильная, во многом я с ней согласен, хотя кое где хотелось бы поспорить. Так или иначе, автор этой статьи со ссылкой на неё осудил мой подход к оценке KDE. Эта заметка заставила меня призадуматься, поскольку согласиться с автором я не могу, но и не признать его концептуальной правоты я также не могу. Хотелось бы только отметить, что в этом эссе я говорю о Linux, т.е. об операционной системе, "стоящей на плечах " (И. Ньютон). Ограниченность раскрытия идей UNIX была заложена в Linux изначально. Также сложилось и с GUI в Linux, хотя в этой части ограниченность проявилась наиболее ярко.

Что уж поделаешь, Linux не совершенна. И KDE - гармоничная часть этого несовершенства.

А если всё же?..

<P align=right>- Ну чего ты прёшь на эти грабли? Ведь опять же по лбу рукоятью получишь!
- А вдруг!

Анекдот</P>

За кем гонимся?

Прежде чем пуститься в погоню за OSX'ывостью, надо всё же осознать, в чём она заключается.

Для многих это просто красивые скриншоты. Поставил бело-металлическую тему с полосками и кнопочками а-ля разноцветные стекляшки - и 'OSX'ился. Но говорить серьёзно об этом уровне OSX'ификации просто смешно.

Как мы все знаем, MacOS X - это жирная прослойка поверх операционной системы Apple Darwin. При этом, будучи полноценной UNIX'подобной ОС Darwin выполняет роль HAL и теоретически вполне заменима, а пользователь MacOS X не знает и не догадывается о существовании какой-то там Darwin. Всё их взаимодействие с компьютером происходит на

уровне графического интерфейса.

[newpage]

В чём же достоинства этого интерфейса, которые сделали MacOS X такой популярной?

1. Однородность пользовательскоо опыта. Пользователь, открыв совершенно новую для него программу, знает, что делает каждая кнопка.
2. Наглядность пользовательскоо опыта. Совершая действие пользователь сразу видит все результаты: файл был тут, а стал там, буковка была красная, а стала серая. Действия, не поддающиеся визуализации, либо недоступны пользователю, либо не попадаются ему на глаза.
3. Единообразие и монолитность, тотальная стандартность. В MacOS X есть один официальный GUI, на каждую функцию есть соответствующий канонический API, так что любой программист может обеспечить себе ситуацию, что его программа вообще не будет иметь зависимостей вне собственных ресурсов MacOS X.
4. Эргономика и красота. Это - основной принцип организации взаимодействия программы с пользователем. Этому принципу может приноситься в жертву даже функциональност программы.

Соответственно, говоря о желании развивать Linux в сторону MacOS X, мы имеем в виду перенос этих правил на Linux. Но на этом пути нам встречаются едва ли преодолимые препятствия.

Непроектировани

Проще всего обстоят дела с ориентацией на эргономику и красоту. Здесь вопрос решается автоматически, поскольку процент дизайнеров среди разработчиков ПО для GNU/Linux минимален, т.е. заниматься этими вопросами попросту некому. Да, дизайн приложения будет функционален и очевиден, но пробег мыши по окну окажется всё равно выше, чем у аналога от Apple.

Модель разработки

Нравится Вам это или нет (а лично мне очень нравится), модель разработки GNU/Linux не предусматриваетникаких механизмов централизации разработки. Так мы имеем несколько ветвей разработки Linux, а проект GNU, хоть остальные компоненты системы и разрабатываются в его рамках, ощутимого влияния на пользовательско ПО не оказывает, так что каждый разработчик делает то, что хочет, при этом используя интересные ему инструменты и библиотеки. В GNU/Linux нет той решающей силы, которая сможет сконцентрироват вокруг себя всю разработку, зато есть много API, имеющих примерно одинаковую популярность и одинаково плотные ряды сторонников. Такое положение полностью исключает как однородность пользовательскоо опыта в системе, так и монолитность конечного результата.

Более того, если даже отклониться немного от курса и заглянуть в мир BSD, где безоговорочным лидером выступает FreeBSD, то и здесь речь идёт о монолитности на уровне базовой системы. но как только установка самых базовых программ завершена, пользователь оказывается посреди всего того многообразия, которым его встречает классическая

UNIX-подобная ОС.

[newpage]

Мотивация

Разработчик MacOS X работает за деньги, которые платит ему Apple. Он получает от Apple задачу, выполняет её и дважды в месяц получает зарплату. Вся его личная инициатива находится строго в корпоративной канве (или генеральной линии Партии, кому так ближе), так что при написании ПО он руководствуется в первые три очереди предписаниями руководства, а в четвёртую (если успеет и не забудет) собственным мнением. Именно этим и объясняется тот факт, что кнопка поиска не может иметь значок в виде бинокля потому, что в других приложениях она уже выполнена в виде лупы. А ведь именно это представляет собой ядро психовизуальноймодели пользователя продуктов Apple или Microsoft.

Психовизуальная модель работы разработчика приложения для Linux в большинстве случаев очень сильно отличается от модели восприятия, присущей пользователю наглядных интерфейсов. Ведь это программист, который пишет приложение для себя, а потому вкладывает в него недружелюбность к незамутнённому инструкцией "", а свои навыки, свои потребности и свой стиль мышления. И продукт получается соответствующий.

А поскольку проекты таких фрилансеров обычно выполняют роль базы для коммерческих разработок, возможность корпоративного давления со стороны Novell, Red Hat, Xandros и т.д. (в особенности под давлением фактора разобщённости) снижается. Действительно, кто будет рубить сук, на котором сидит?

[b]Противодействие/b]

Но ответ сторонников OSX'ификации на эти невзгоды уже созрел: стандартизаторыво всю пытаются крестить окружающих в свою веру. Вспомним LSB, навязывавший дистрибутивам RPM как стандарт пакетирования. Этот путь может привести к снижению издержек от преодоления перечисленных выше препятствий, и дорога OSX'ификаторов окажется открыта.

[newpage]

А делать-то что?

Всё очень просто. Вот несколько простых шагов:

1. Прочитать произведения Эрика Реймонда (Eric S. Raymond), который очень подробно описывает идеологию UNIX.
2. Если понравилось, попробовать месяц пожить в консоли. Пускай там нельзя выполнить всех задач, зато такой опыт позволит научиться стилю мышления UNIX, который в консоли укрепился глубже.
3. Если и это понравилось, просто перейти на использования ПО, разработанного в традиции UNIX.

И самое важное: если всё это не понравилось, оглянитесь вокруг! Уже на подходе Haiku -

наследник BeOS и открытый конкурент MacOS X. Активно разрабатывается ReactOS - открытая NT-совместимая система. Вполне функциональна Syllable. Может быть, именно они являются тем, что Вам нужно?

Послесловие

Особое значение выбора пути для GNU/Linux заключается в том, что роль как UNIX, так и FOSS в ближайшие годы тесно переплетена именно с этой торговой маркой. Ведь именно ОС GNU/Linux первой из свободного ПО стала приносить деньги разработчикам, именно она заняла подавляющее большинство x86-совместимых компьютеров, не занятых Windows. Именно она ушла на Mainframe'ы, вытесняя проприетарные UNIX'ы и Windows.

Да, у UNIX'истов есть запасные варианты: семейство BSD, GNU/Hurd, полузабытая, но не ставшая от того менее прекрасной Plan9... Есть свои надежды и у FOSS - впереди ReactOS и Haiku, да и Microsoft уже подготовила лицензии, признаваемые FSF свободными. Т.е. жизнь, конечно, найдёт выход. Вопрос в том, когда и какой кровью этот выход будет найден.

Однако провал станет тактическим провалом и FOSS, и UNIX, который откинет их на много лет назад, а успех GNU/Linux в канве OSX фактически похоронит последние мечты о UNIX, чего лично мне бы очень не хотелось.