Август 2006 г.
Автор: Павел Макаров (pavel.makarov@mail.ru)

Простота – сущность философии Unix

Философия проекта «10000 lines of code» (10kloc)

«Всё гениальное просто» - всю свою жизнь я на практике убеждаюсь в справедливости услышанного ещё в детстве утверждения. Добавлю от себя, что, как показывает вся известная нам история человечества, все истинно гениальные творения: идеи, проекты, изделия, художественные произведения и проч. - создавались личностями, а не коллективами. Не следует, естественно, путать творческое созидание с изготовлением изделия – Санкт-Петербург, например, строили тысячи неизвестных строителей, однако **создавали** единицы: Растрелли, Росси, Кваренги, Стасов, Воронихин, Бенуа...

Только ленивый не упомянет в связи с обсуждаемой темой великого Леонардо да Винчи, Эйнштейна, Пушкина или Чайковского. А ведь были ещё Эдисон, Яблочков, братья Черепановы, Эйфель, Менделеев и многие тысячи известных (и не очень) представителей homo sapiens - «человека разумного», творения которых, безусловно, повлияли на жизнь каждого из нас.

Среди наших современников, имеющих дело с «computer science», также предостаточно людей талантливых, а то и действительно гениальных (хотя это пусть потомки решают). Steve Jobs, Brian W. Kernighan и Dennis M. Ritchie, Bill Gates (да-да, по-своему очень талантливый человек), Richard Stallman, Seymour Cray, В.М. Глушков, П.Г. Кузнецов и С.П. Никаноров, Б.А. Бабаян... Их произведения заслужили признание у миллионов пользователей (поэтому среди специалистов они имеют статус именно произведения, а не изделия или продукта) и являются *de facto* своего рода «отраслевыми эталонами».

У всех этих людей есть одна общая особенность: всё, с чего они начинали, было сделано **собственными руками**; «изюминкой» **всегда** являлась оригинальная идея, а не конкретная её реализация, которая со временем стараниями «сподвижников» зачастую превращалась во что-то огромное, внушающее почти священный трепет... но не до конца и не совсем правильно работающее (некогда «вылизывать» - конкуренты не ждут!), и потому требующее постоянной доработки и сопровождения в эксплуатации. А ведь: «...какая была идея!»

В неявном виде мы наблюдаем классический результат: всё гениальное просто (и один человек может достаточно быстро изготовить *это* своими руками). И наоборот: всё, что один человек не в состоянии понять, охватить разумом и изготовить в одиночку (или с *небольшой* группой единомышленников), скорее всего, далеко от совершенства. Как следствие, все такие изделия будет очень непросто изготавливать серийно, что достаточно быстро приведёт к их «безвременной кончине» вследствие появления лучших (более надёжных, удобных в эксплуатации, компактных и менее дорогих) инженерных решений.

Естественно, говоря о простых и компактных решениях, я не имею в виду только логически или физически небольшие объекты (не больше 10 тысяч строк кода, не тяжелее 10 кг, не длиннее 1 метра, не больше 100 тысяч транзисторов на кристалле и т.п.). При таком формальном подходе любой технически сложный объект из нашей жизни (корабли, самолёты, АЭС... список можно продолжить до бесконечности) можно легко заклеймить как некачественное, а то и потенциально опасное изделие. На самом деле это справедливо далеко не всегда: те из сложных объектов, которые легко поддаются декомпозиции на большое число действительно небольших подсистем с ясными, полностью и корректно формализованными правилами их объединения и взаимодействия, также по сути являются логически простыми и имеют право на существование в качестве корректного технического решения.

Приятно в связи с вышеизложенным отметить, что в мире больших и сложных систем гигантомания последних десятилетий также начала уступать дорогу здравому смыслу. Нетрудно заметить, что на смену гигантским ядерным подводным ракетоносцам (типа американского Trident или нашей «Золотой рыбки» проекта 661) приходят небольшие дизельные (!) подводные лодки четвёртого поколения (дизель-электрическая подводная лодка "Санкт-Петербург" проекта 677); всё меньше иллюзий питает человечество в отношении сверхбольших аэробусов типа A-380 или Boeing-787 (самыми безопасными, дешёвыми и потому популярными оказываются небольшие ближне- и среднемагистральные самолёты вместимостью от 20÷50 до 100÷150 пассажиров, а также скоростные поезда типа нашего ЭР-200, французского TGV или японского N-700 Синкансэн); ядерная энергетика тоже склоняется в пользу относительно небольших реакторов (либо традиционного типа, хорошо зарекомендовавших себя на атомоходах всех видов, либо реакторов нового поколения на быстрых нейтронах). Не трудно найти примеры и из других областей науки и техники. Так что, всё возвращается «на круги своя».

И словно в подтверждение вышеизложенного, я недавно обнаружил в Сети сайт (*http://wmii.de/*), автор которого, Ансельмом Гарбэ (Anselm R. Garbe), придерживается аналогичной точки зрения, хотя и рассуждает исключительно о проблемах разработки программного обеспечения. Мне импонирует ясность и краткость исповедуемой автором философии, которую он свёл к лозунгу: «10000 lines of code» (или кратко: «10kloc»). В связи с этим ниже предлагается перевод соответствующей страницы сайта.

Философия проекта «10000 lines of code» (10kloc)

Проект «10kloc» предназначен для программных проектов, размер которых не превышает 10 тысяч строк исходного текста. Мы убеждены, что большая часть ПО, размер которого превышает этот максимум, просто раздута и в корне ошибочна (seriously wrong). Мы также уверены, что не существует человека, который ***полностью*** *разбирается в программных комплексах, состоящих из более, чем из 10 тысяч строк кода.*

Многие дилетанты «от open source» гордятся тем, что они достигли большого размера «исходников», поскольку они верят, что, чем больше строк кода они написали, тем большего прогресса они достигли. А чем большего прогресса они достигли, тем более опытными они стали. Это просто заблуждение.

Превышение этого максимума в 10 тысяч строк кода часто является как раз показателем плохого качества кода и практически полного отсутствия заботы о качестве кода со

стороны создателя программы. Обычно большое количество строк кода идёт «рука об руку» с архитектурой, не имеющей ничего общего с первоначальным замыслом. Существуют дюжины известных примеров, подтверждающих это утверждение, например, версия GNU Unix userland.

Есть разные причины этого явления. Большинство дилетантов на самом деле не уделяют должного внимания качеству кода. Поэтому, если они берут что-либо работающее, что, как им кажется, решает проблему, то они просто «подсаживаются» на это. Если такая разновидность разработки ПО используется при разработке одного и того же исходного кода на протяжении всего его жизненного цикла, мы в результате останемся с большим количеством кода, абсолютно извращённой структурой кода и с испорченным проектом системы. И всё это - благодаря недостаточной ясности и целостности концепции проекта на стадии разработки.

Сложность кода есть первопричина раздутого, трудного в использовании и абсолютно несостоятельного ПО. При наличии сложного кода задачи решаются далеко не оптимально, драгоценные ресурсы бесконечно расходуются понапрасну, производительность падает вплоть до нуля, а уязвимости становятся обычным делом. Единственный выход – остановить проект вообще и заново переписать его с нуля.

Плохие новости: переписывание ПО для улучшения качества кода случается крайне редко, поскольку дилетанты «от open source» гордятся большим размером кода. Они думают, что они понимают всю сложность кода и потому нет нужды переписывать его. Они считают себя крутыми, понимая, что все остальные не могут даже и мечтать о том, чтобы «врубиться в тему». Для таких «специалистов» сложное ПО – просто мечта.

Но все оригинальные идеи просты. Оригинальное ПО также является простым. Простота – это сущность философии Unix. Чем больше строк кода вы удалили, тем большего прогресса вы достигли. Так же, как и тот факт, что чем больше строк кода исчезает из вашего ПО, тем более опытным вы становитесь.

Другими словами: удалённый код есть отлаженный код и он, естественно, совершенно не использует память и процессорное время; так что даже лучше, если бы вы его не писали вообще.

Не правда ли – «не в бровь, а в глаз»? Краткость, оказывается – сестра не только таланта, но и надёжности, отказоустойчивости, безопасности, равно как и множества других достоинств любого IT-изделия: быстродействия, компактности, низкой себестоимости и др.

А поскольку сущностями философии Unix (и не только Unix, добавлю я от себя, а всех на самом деле оригинальных и эффективных инженерных решений) являются именно ясность и простота, то не следует ли поставить этот критерий во главу угла при оценке, в частности, **любого** открытого проекта из мира *nix?

Мир Windows лучше не трогать, поскольку правила в нём определяются пресловутой «коммерческой

целесообразностью». Согласно ей программный продукт может быть сколь угодно большим, с неполной функциональностью, содержащим множество ошибок, но должен хоть как-нибудь работать и быть обязательно выброшен на рынок в срок. А уж потом... Как сказал классик - «...там у них другие мерки, чуть что не так – сожрут живьём». А ведь ещё великий Генри Форд говаривал: «Преобладающая забота о деньгах, а не о работе, влечёт за собою боязнь неудачи; эта боязнь ***тормозит правильный подход к делу****, вызывает страх перед конкуренцией, заставляет опасаться изменения методов производства, опасаться каждого шага, вносящего изменение в положение дел.» Так что, оставим в покое «мир чистогана» - всё лучшее людьми всё равно было сделано не благодаря, а вопреки законам его функционирования.*

Блестящим примером, подтверждающим действенность философии Unix, является работа профессора Таненбаума и его коллег – операционная система Minix 3. Имея оригинальную микроядерную архитектуру и, соответственно, фантастически малый размер ядра (около 4000 строк кода), эта ОС, изначально строившаяся по принципу необходимого минимума компонентов ядра, достигла удивительной компактности за счёт тщательной отработки её идеологии и алгоритмов функционирования, а также благодаря многократной оптимизации открытого исходного кода ОС. Достаточно сказать, что процесс её превращения из концептуальной модели (Minix версий 1 и 2) в пригодную для практического применения ОС (Minix 3), включающий как работы строго научного, концептуального характера, так и отработку программной реализации, занял почти два десятилетия!

Интересно отметить, что Эндрю Таненбаум – человек, блестяще «владеющий предметом», досконально знакомый со всеми мало-мальски известными ОС, прекрасно понимающий всю функциональную сложность операционной системы как универсального программного «фундамента» современного компьютера и не испытывающий вроде бы особых финансовых затруднений с реализацией своих идей (поскольку является одним из лучших в мире специалистов в этой области, под разработки которого любая серьёзная компания легко даст необходимые средства), тем не менее, в качестве **оптимального** со всех точек зрения варианта универсальной ОС остановился на компактной микроядерной архитектуре. Впрочем, о причинах своего выбора он с коллегами очень подробно написал в книге «Operating Systems Design and Implementation»; краткое изложение этих причин можно также найти в статье «Можем ли мы делать ОС надёжными и безопасными?» на русском сайте www.minix3.ru, посвящённом этой операционной системе.

Итак, вывод: не стоит ли, развивая серьёзные (а правильнее будет сказать - фундаментальные) **современные** проекты в области операционных систем и соответствующих пользовательских приложений, возвратиться к истокам самой философии Unix и взять за основу сформулированный Ансельмом Гарбэ принцип «10kloc»?

Ей-Богу, это не только определённая гарантия качества программного кода, но и прекрасная возможность для каждого из нас отточить собственное мастерство, научиться создавать **истинные произведения инженерного искусства** (каковым, без сомнения, является ОС Minix 3) взамен заполонивших Сеть типовых поделок дилетантов «от open source». Так пусть же и Minix 3, и создаваемое для неё прикладное ПО станут отраслевыми эталонами качества для всего мира open source.