Ян Малаховски

Конспект лекций по курсу «Операционные системы»

version 3
compiled 23 мая 2011 г.

Оглавление

Введение

Документ, который вы читаете, представляет собой инновационный конспект по курсу «Операционные системы». Инновационность заключается в постоянном использовании буквы «ё», где надо и где нет (соответственно, в версии без буквы «ё» эта инновационность отсутствует всюду кроме этого введения).

Всё, что не касается технической стороны предметов, затрагиваемых в данном документе, является моей личной точкой зрения, которая не может считаться коль сколь-нибудь авторитетной в вопросах, в которых я не считаю себя экспертом (например, авторское право). Тем не менее, любые конструктивные замечания и исправления принимаются с радостью.

Соглашения

Определения и теоремы выделяются классически, используется сквозная нумерация внутри глав. Словосочетания, важности которых для данного текста не хватает, чтобы носить гордое название «Определение», выделены *курсивом*, точно также выделяются места с логическим ударением.

Определение 0.1. *Научная работа — работа в которой есть хотя бы одно доказательство.*

Теорема 0.2. *Этот текст — научная работа.*

Доказательство. Тут есть доказательство. □

Все приводимые примеры кода по возможности компилируются и оттестированы. В заголовках листингов указывается имя файла, в котором его нужно разместить, а в комментариях в конце листинга иногда указываются команды, которые необходимо ввести в оболочку, чтобы этот файл скомпилировать (слинковать, запустить, ...). Листинги без заголовков представляют собой, или псевдокод, или вывод каких-то программ, или примеры, которые невозможно запустить по каким-то причинам.

По отношению к структурному элементу многих языков программирования, обычно называемому «функцией», (из религиозных соображений) применяется термин «процедура» в не зависимости от того, имеется ли у этого элемента возвращаемое значение.

Все факты из данного документа настоятельно рекомендуется сверять с соответствующими мануалами, а все исходники изучать, компилировать и запускать, поскольку они могут содержать как случайные, так и не очень, ошибки.

Состав

В каждый момент времени этот документ не окончен, находится в разработке и в бета-версии. Секции и даже целые главы могут быть пропущены из-за моей лени, недостатка времени или слишком низкого приоритета (например, излагается очень простой материал, съедобную информацию по которому легко найти в сети).

Курс рассчитан на два семестра (а может быть даже и на три). В первом семестре проводится общее знакомство с системами типа UNIX-like без каких-либо серьёзных подробностей. Примерно половина

второго семестра посвящена трешу POSIX API, а вторая половина — каким-то более-менее общим (а потому хоть немного интересным) вещам. Третий семестр как бы рассчитан на advanced темы и мощных студентов и, если когда-нибудь и будет существовать, то, видимо, в качестве факультатива (семинаров, или чего-то такого).

В принципе, первый семестр — абсолютный треш и всё что в нём содержится можно самостоятельно изучить за неделю или две, ещё есть какие-то лабораторные работы, но они тупые, студентам лень их делать, а мне проверять. В каждую конкретную «интересную» тему второго семестра можно закопаться очень глубоко, однако это делать опасно, ибо можно обратно уже не вылезти. Про третий семестр ничего не знаю.

Часть I

Первый семестр

Тут когда-нибудь будут лекции по первому семестру. Или не будут.

Часть II

Второй семестр

Глава 1

Юристы и компьютеры

О юридических ужасах, подводящих к необходимости существования лицензий в этом страшном мире, свободном и открытом программном обеспечении, а также почему оно нужно, но не достаточно.

1.1 Юридические ужасы

Computer science и право различаются уже хотя бы в том, что законодателями, и в этой, и в той стране, в основном являются юристы, и, если без «операционных систем» в какой-то области можно обойтись, то без минимальных юридических знаний можно случайно совершить какое-нибудь серьёзное правонарушение и получить много проблем. Не менее прискорбно то, что законодатели обычно не знают теоретических основ computer science (теории вычислительной сложности, например), а потому содержание многих юридических документов, имеющих отношение к этой области знания, может вызывать недоумение. Это уже не говоря о том, что почти все юридические тексты написаны по таким правилам и таким языком, что очень часто их просто невозможно понять и адекватно толковать, не посвятив этому жизни.

В этом разделе я постараюсь изложить своё видение необходимых человеку из computer science юридических терминов. Все права на упоминаемые в этом разделе торговые марки принадлежат их законным правообладателям.

Определение 1.1. *Интеллектуальная собственность — множество, включающее в себя* авторские и смежные права, торговые марки, патенты, *а также другие, не менее страшные, но менее известные юридические термины.*

Определение 1.2. *Торговая марка — некоторая строка символов, право использования которой в каких-то целях охраняется на государственном уровне. Является ли логотип торговой маркой мне не ясно.*

Например, строка «Coca-Cola» является торговой маркой. Считается, что торговые марки появились в средние века с той целью, чтобы один ремесленник мог выделить продукт своего производства из таких же продуктов, производимых другими ремесленниками, но так, что право на использование имени, которым данный ремесленник обозначал свои продукты, находилось под охраной государства. Тогда другого ремесленника, который делал бы неавторизованные данным ремесленником подделки, можно было бы как-то наказать. Таким образом, торговые марки были придуманы для защиты потребителей от потенциально некачественного товара.

Однако в современном мире торговые марки используются как раз для того, чтобы обманывать потребителей. Например, товарная марка «Coca-Cola» когда-то обозначала напиток, содержащий вещества из растения «соса», но скоро в США лавочку с использованием этих веществ прикрыли (объявив их наркотическими), однако потребителей до сих пор обманывают названием данной торговой марки напитка, в котором экстракта этого растения уже нет.

Не менее выгодным бизнесом оказалась продажа торговых марок «в аренду», при такой схеме, владельцу раскрученной торговой марки вообще можно ничего не делать, а только взымать «арендную плату» за использование строки символов. Например, «McDonald's» в разных странах представлен совершенно разными компаниями, «арендующими» эту торговую марку у оригинальной компании в США. Для защиты потребителей от такого обмана в некоторых странах были приняты законы, запрещающие передавать права на пользование торговой маркой без передачи технологий, используемых в оригинальном производстве. Это лучше чем совсем ничего, однако это не мешает «арендующему» производителю в другом государстве не соблюдать технических процессов, использовать другое сырьё и так далее.

Торговые марки обозначаются знаками «$^{\text{TM}}$» (trade mark) и «®» (registered trade mark), тонкостей разницы между этими двумя понятиями я не знаю.

Определение 1.3. *Авторство — констатация того факта, что кто-то сделал что-то.*

Например, Бетховен является автором своей девятой симфонии, и его авторство от самой симфонии оторвать никак нельзя, даже если бы Бетховен, в своё время (но по современным законам), и хотел бы продать авторство на эту симфонию, он бы этого сделать не смог.

Определение 1.4. *Научный приоритет — тоже самое, что и авторство, но для научных открытий.*

Пифагор является автором нескольких теорем, подобно тому как Бетховен является автором не только одной симфонии. Однако, когда речь идёт о научных открытиях, это почему-то называют не авторством, а научным приоритетом.

Определение 1.5. *Авторское право — права в отношении использования произведения, которыми наделяется его автор.*

Авторские права, в отличии от авторства, можно кому-то передать.

Определение 1.6. *Общественное достояние — то, куда попадают все произведения, авторские права на которые истекли.*

Вернёмся к Бетховену, почему-то живущему по современным законам. Пусть Бетховен сегодня сочинил свою девятую симфонию. Этот факт делает Бетховена автором симфонии, и, кроме того, наделяет его в отношении неё авторскими правами. Например, он, а после его смерти и его потомки в течении какого-то срока (в России это что-то типа семидесяти лет, если я не ошибаюсь), могут требовать денег за любое её использование. После истечения этого срока симфония перестаёт охраняться авторскими правами и переходит в общественное достояние. С произведениями в общественном достоянии кто угодно может делать что угодно.

Законодательство некоторых стран (но не России) позволяет автору самостоятельно отказаться от своих авторских прав и передать своё произведение в общественное достояние.

Авторские права, по большому счёту, означают полный контроль над тем, кем и как используется созданное произведение в течении строка до попадания произведения в общественное достояние. Например, Бетховен мог бы разрешить записывать и продавать диски с его симфонией какой-то одной компании, при этом запретив ей распространять свою симфонию, например, в Африке. Кроме того, Бетховен имел бы, например, право опубликовать симфонию под своим именем, под псевдонимом или вообще анонимно. Когда речь идёт о правах на копирование и воспроизведение произведения — это называется *имущественным авторским правом*, или *копирайтом* (copyright). В других случаях — *неимущественным авторским правом*.

Определение 1.7. *Смежные права — права, которыми наделяется исполнитель произведения, если исполнение требует техники, оборудования, таланта, мастерства, и тому подобного.*

Например, смежными называются права, которыми обладает оркестр, исполнивший девятую симфонию Бетховена, и звукозаписывающая компания, выпустившая этот концерт на диске.

Определение 1.8. *Патент — исключительное право на использование результатов изобретения.*

Патенты были изобретены в качестве метода борьбы с секретами производства, уходящими в могилу вместе с мастерами, их использующими. Идея заключалась в том, чтобы дать ремесленникам способ, при помощи которого можно было бы раскрыть миру секреты своего производства, но так, чтобы без разрешения автора или его потомков в течении какого-то срока (что-то около двадцати лет, если я не ошибаюсь) никто не смог бы эти секреты использовать.

Проблема заключается в том, что если секрет действительно очень важный, то выгоднее его держать при себе и вообще не патентовать, зато запатентовав что-нибудь, что используют или будут использовать все, можно ничего не делать и красиво жить. Таким образом в современном мире патенты превратились в средство шантажа и метод монополизации целых отраслей производств. Например, в индустрии производства ПО монополия длиной в двадцать лет на какой-то тип софтварных продуктов практически равносильна монополии навсегда.

Основная логика, прослеживающаяся в различиях между авторскими правами и патентным правом, заключается в том, что патентное право предоставляет исключительные права первооткрывателям каких-то фактов или конструкций (не смотря на то, что, потенциально, до любого открытия могут дойти два и более независимых учёных, исключительное право на использование предоставляется первому), а авторские права защищают произведения, которые (почти достоверно) не могут быть повторены двумя независимыми авторами (не смотря на утверждение про миллион обезьян за печатными машинками, маловероятно, что одну и туже «Войну и мир», почти одновременно напишут два независимых писателя).

Тем не менее, существуют области знаний которые вообще не поддаются охране авторскими правами и патентами, например, математические результаты (хотя сам текст, содержащий математический результат может быть защищён авторскими правами в качестве литературного произведения). С другой стороны, во многих странах программы можно защитить авторскими правами (тем самым исходники приравнены к литературным произведениями), а в некоторых странах можно даже запатентовать программные алгоритмы.

Однако заметим, что теория вычислительной сложности утверждает, что любой алгоритм представим в виде программы для универсальной машины Тьюринга, а любую программу на любом языке программирования можно преобразовать в эквивалентную программу для машины Тьюринга. Любая программа для машины Тьюринга представима в виде строки символов, которую можно интерпретировать как запись, например, натурального числа (потенциально очень большого, но всё же). Таким образом, задача поиска программы, обладающей требуемыми свойствами, сводится к поиску какого-то натурального числа (а на самом деле даже ряда чисел). Возникает вопрос: «Можно ли запатентовать натуральные числа и не является ли программа, а заодно и алгоритм, математическим результатом?»

В общем, не всё так гладко, как хотелось бы. Почти всё, что изложено в этом разделе было почёрпнуто из:

- `http://gorod.tomsk.ru/index-1248681368.php`;
- `http://www.groklaw.net/article.php?story=20091111151305785`;
- Гражданский Кодекс РФ, часть IV.

1.2 Лицензии, GNU, Creative Commons и другие

Лицо, обладающее копирайтом на какой-то объект, может этим своим копирайтом с кем-нибудь поделиться. Например, автор программы может предоставить кому-то конкретно (или вообще всем) право безвозмездно ей пользоваться. Если предоставляется право использования и модификации исходных кодов, то автор может потребовать, например, чтобы ему был предоставлен доступ к результатам модификаций.

Определение 1.9. *Лицензия — правила, в соответствии с которыми третье лицо может использовать объект, охраняемый авторскими правами.*

Идея тут такая: по умолчанию никто кроме автора не может пользоваться его произведением, а при помощи лицензии автор устанавливает правила, выполняя которые третьи лица могут использовать его произведение. Лицо, нарушающее условие лицензии, по которой распространяется объект, теряет заодно и все права, предоставляемые ему этой лицензией.

Если вы вдруг помните хоть что-то из истории UNIX-подобных операционных систем, то вас не удивит тот факт, что в 1983 году Ричард Столлман уходит из MIT для того, чтобы заняться разработкой своей свободной операционной системы GNU, поскольку по трудовому договору авторские права на всё, сделанное за время работы в MIT, принадлежат MIT. Основными результатами, полученными в ходе разработки GNU, являются лицензия *GNU GPL* (GNU General Public License), набор компиляторов (самый известный из которых — *GCC*) и набор стандартных юзерспейс-утилит, соответствующих стандарту POSIX. И компиляторы, и утилиты распространяются по лицензии GPL.

Идеология, продвигаемая Столлманом и *FSF* (Free Software Foundation — организация, «выросшая» из GNU, сейчас GNU является одним из проектов FSF), основана на понятии *свободного программного обеспечения*, содержащего четыре основных свободы.

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

В английском языке с термином «свободный» обнаруживаются неожиданные проблемы, которые можно выразить известной фразой «Free as in Freedom or free as in beer» (мой вольный перевод: «Свободный как Свобода, или свободный как халявное пиво»). В поисках лучшего синонима Эрик Реймонд придумал термин «*открытое программное обеспечение*», который сегодня используется намного чаще. Однако в за этот термин яро ухватились крупные корпорации, распространяющие плоды своей деятельности в виде общедоступных исходных кодов, но под лицензиями, допускающими только некоторые из четырёх вышеназванных свобод. Другими словами, по факту, сегодня любое свободное ПО является открытым, но не любое открытое — свободным. В том числе и поэтому, Столлман регулярно выступает на разных конференциях и пишет статьи о том, что не стоит путать эти два термина.

Лицензии, предоставляющие четыре вышеупомянутых свободы всем пользователям, в противоположность копирайту, принято называть *копилефтом* (copyleft).

Первая версия GNU General Public License создавалась в качестве единой лицензии для всего свободного ПО и во многом объединяла общие части лицензий, использовавшихся при распространении программ того времени. Однако кроме даров пользователям ПО, состоящих из четырёх свобод, GPL требует, чтобы всё, использующее код, распространяемый под GPL, также распространялось под GPL. Именно поэтому некоторые важные в крупных корпорациях лица в своё время называли GPL «вирусом» индустрии разработки ПО.

Поскольку любая лицензия является юридическим текстом, а, как известно, юридические тексты очень часто содержат не достаточно формально определённые понятия, то как только кто-то находит дыру в возможности трактовки того или иного куска существующей лицензии, так сразу в эту дыру начинают ломиться все кому не лень. Потому FSF иногда выпускает новые версии своих лицензий, поддерживая их в актуальном состоянии, и закрывая «юридические баги».

Вместе со второй версией GPL на свет появилась лицензия *GNU LGPL* (GNU Library General Public License, позже переименованная в GNU Lesser General Public License), разрешающая использовать код программы без модификации, вместе с кодом, не распространяющимся под LGPL, однако в случае модификации LGPL-кода результаты также должны распространяться под LGPL. Как ни странно, но первой версии LGPL сразу была назначена вторая версия, дабы подчеркнуть её связь с GPLv2.

Третья версия GPL (а заодно и LGPL) была опубликована в 2007-ом году и стала запрещать *тивоизацию* (tivoization — аппаратное ограничение модификации прошивок устройств, получаемых с использованием свободного ПО, например, ограничение на замену прошивок в мобильных телефонах, основанных на ядре Linux), использование *DRM* (Digital Rights Management — программно-аппаратные средства защиты авторских прав на произведения в цифровом формате), передачу прав на использование патентов только некоторым пользователям кода (чтобы одних пользователей любить, а других — судить) и так далее. GPLv3 вызывала много споров о необходимости столь строгих ограничений ещё до публикации окончательной версии, и некоторые проекты (например, ядро Linux) менять лицензию не стали. Также в 2007-ом году появилась *GNU AGPL* (GNU Affero General Public License), требующая доступ к исходным кодам ПО, с которыми программа, распространяемая под AGPL, общается по сети.

Однако лицензий GNU на все случаи жизни не хватает. Например, в мире операционных систем семейства BSD используются всевозможные модификации оригинальной *BSD License* (Berkley Software Distribution), фактически разрешающей любое использование оригинального кода, кроме перелицензирования. Также достаточно часто можно встретить *MIT License* (она же *X11 License*), требующую распространения текста самой лицензии со всеми модифицированными версиями кода. Многие литературные произведения, а также всякие другие произведения искусства, распространяются по лицензиям из семейства *Creative Commons*. В общем, всех не перечислить. Сводную табличку с разными лицензиями и их свойствами можно наблюдать в Википедии по адресу `http://en.wikipedia.org/wiki/Comparison_of_free_software_licenses`. Там же можно обнаружить, что некоторые не-GNU лицензии признаются FSF «свободными», а некоторые — признаются *OSI* (Open Source Initiative — ещё одна организация, эксплуатирующая термин «открытое ПО», подобно тому как, FSF эксплуатирует термин «свободное ПО») «открытыми», при чём пересечение этих множеств не совпадает ни с одним из них.

Но, конечно, никто не может запретить автору распространять один и тот же код под несколькими лицензиями, например, часто можно встретить комбинации из GPL и BSD, а также разных версий GPL («GPLv2 or any higher»).

1.3 Почему свободности недостаточно

Как ни странно, но кроме религиозных, экономических и философских соображений на тему «почему всё должно быть свободным», существуют и вполне конструктивные измышления на эту тему «почему даже всё свободное не решает всех проблем».

Теорема 1.10. *Можно верить только тому программному обеспечению, которое написано лично вами.*

Thomphson compiler hack. Пусть у нас есть компилятор A и доступ к его исходным кодам (назовём их A'). Модифицируем A' в исходный код компилятора E (E') так, что E:

- распознаёт, когда ему на вход подаётся что-то похожее на A' и модифицирует необходимые части дерева разбора A', так, чтобы из него получалось дерево разбора E';
- распознаёт, когда ему на вход подаётся исходный код утилиты login (sshd, telnet, всё что угодно) и добавляет в неё, например, возможность логина в любого пользователя, если в качестве пароля написать «itisagooddaytodie».

После чего

- компилируем E' при помощи A, получаем исполняемый E;
- компилируем A' при помощи E, но получаем E;
- кладём в дистрибутив бинарный компилятор E и исходники A'.

Теперь, если кто-то вздумает скомпилировать утилиту login (или что мы там хотим испортить), то в результате система будет иметь backdoor, который не будет отображён ни в одних исходных кодах. Более того, даже при компиляции новой версии исходных кодов A', без какого-либо вмешательства со стороны автора E', с большой вероятностью получится новая версия компилятора E, со всеми новыми возможностями A, но с «интересными особенностями» при компиляции утилиты login. □

Заметим, что, в особо извращённом случае, даже дизассемблирование компилятора E может не спасти, потому как E может добавлять «цензуру» в исходники ядра, отвечающие за чтение бинарника E, или в работу с файловыми дескрипторами, через которые будет проходить дизассемблированный код E. Подобные же ухищрения, реализованные в аппаратуре, на практике вообще не обнаружить.