Программное обеспечение

Программное обеспечение – наряду с аппаратными средствами, важнейшая составляющая информационных технологий, включающая компьютерные программы и данные, предназначенные для решения определённого круга задач и хранящиеся на машинных носителях. Программное обеспечение представляет собой либо данные для использования в других программах, либо алгоритм, реализованный в виде последовательности инструкций для процессора.

В компьютерном жаргоне часто используется слово «софт» от английского software, которое в этом смысле впервые применил в статье American Mathematical Monthly математик из Принстонского университета Джон Тьюки (John W. Tukey) в 1958 г.

В области вычислительной техники и программирования **программное обеспечение** – это совокупность всей информации, данных и программ, которые обрабатываются компьютерными системами.

Классификация ПО

1. По способу распространения (доставки, оплаты, ограничения в использовании):
    1) Commercial Software,
    2) Freeware,
    3) Shareware,
    4) Abandonware,
    5) Adware,
    6) Free Software,
    7) Careware...
2. По назначению ПО разделяется на
    1) системное,
    2) прикладное
    3) инструментальное.

Рассмотрим каждый класс по-отдельности.

По способу распространения (доставки, оплаты, ограничения в использовании)

1.1

Коммерческое программное обеспечение (англ. **commercial software**) – программное обеспечение, созданное коммерческой организацией с целью получения прибыли от его использования другими, например путем продажи экземпляров.

Коммерческое и свободное программное обеспечение.

Множество людей ошибочно считают коммерческое и свободное противоположностями. Различия между этими двумя видами не столь критичны, как может показаться на первый взгляд. Ведь свободным ПО считается с того момента, как автор предоставляет права на свободную модификацию, распространение и извлечение прибыли со своего продукта. Из этого следует, что свободные программы вполне могут быть и коммерческими продуктами. Хорошими примерами коммерческих программ относящихся к разряду свободных могут служить компилятор GNU ADA или многие операционные системы на основе GNU/Linux.

Противоположностью свободного программного обеспечения является собственническое программное обеспечение, которое также может быть как коммерческим, так и бесплатным (freeware).

Техническая поддержка продукта

Наиболее важной особенностью коммерческих программных продуктов является поддержка крупных компаний, прямо заинтересованных в распространении своего детища. Многие организации предоставляют исключительно платную поддержку своим продуктам, такой подход, как правило, используют организации, предоставляющие открытые исходные коды. Для продуктов, распространяющихся на

коммерческой основе, действуют обычно бесплатные службы поддержки призванные увеличить уровень доверия у клиентов и потенциальных покупателей.

Сроки добавления изменений

Далеко не всегда, но, как правило, сроки изменений критически важных в коммерческих продуктах значительно меньше, чем у некоммерческих проектов. Это связано с тем, что над коммерческим продуктом работают целые группы разработчиков и эта работа является их основным занятием. Разработчикам-любителям, как правило, приходится искать дополнительные способы заработка и это уменьшает время затрачиваемое на дополнения и изменения программ.

Спектр выполняемых задач

Так как основным движущим фактором создания коммерческого ПО является получение прибыли, то коммерческие программные продукты первыми заполняют свободные ниши и предлагают варианты решения задач сразу по мере обнаружения вакуума в каком–либо секторе рынка.

Программы «на заказ»

Отдельный вид коммерческих программ, когда работа по их написанию оплачивается непосредственно заказчиком. Такие программы зачастую лишены всех преимуществ коммерческих продуктов, так как имеют ограниченный бюджет, но более адаптированы к требованиям заказчика, чем аналоги.

1.2

Бесплатное программное обеспечение (**Freeware**, от англ. free – «бесплатный» и software – «программное обеспечение») – это программное обеспечение, которое может бесплатно распространяться всяким желающим.

В отличие от свободного программного обеспечения (англ. free software) freeware может распространяться без исходных текстов и содержать ограничения на коммерческое использование, модификацию и т.д.

В отличие от Shareware не предполагает никакой платы разработчику и, соответственно, никаких дополнительных услуг, улучшенных версий и т.п. за эту плату.

Программы, раздаваемые разработчиком бесплатно, но не имеющие публичной лицензии на распространение, такие как Adobe Flash Player, относятся к проприетарному ПО, и, по-видимому, не являются freeware, хотя имеется различное словоупотребление.

Free Software Foundation и проект GNU рассматривают freeware как разновидность несвободного программного обеспечения.

1.3

Shareware – это тип программного обеспечения, обусловленный особенностями распространения таких программ. В русском языке этот термин интерпретируется как «условно-бесплатное программное обеспечение».

Исторически, слово обозначало программы, свободно распространявшиеся третьими лицами (например, через BBS (Bulletin Board System, электронная доска объявлений) или FidoNet, в сборниках программ на компакт-дисках) и содержавшие в себе просьбу заплатить деньги автору программы. Однако с течением времени значение изменилось и, говоря о Shareware, все чаще стали иметь в виду не свободное распространение (которого в наше время может и не быть – программа может быть доступна только с сайта производителя) а приемы и уловки, используемые авторами shareware-программ с целью побудить пользователя заплатить за уже полученную им бесплатно программу.

Сейчас Shareware чаще всего понимают также тип, способ или метод распространения проприетарного ПО на рынке (то есть на пути к конечному пользователю), при котором испытателю предлагается ограниченная по возможностям (неполнофункциональная или демонстрационная версия), сроку действия (триал версия, от англ. trial – пробный) или версия с встроенным раздражающим блокиратором-напоминанием (NAG) о необходимости оплаты использования программы. В лицензии также может быть оговорен запрет на коммерческое или профессиональное (не тестовое) её использование. Иногда программа спустя некоторое время (например, 30 дней) прекращает запускаться. Иногда становится недоступной часть функций. Иногда функциональность остаётся в полном объёме, но пользователю время от времени показывается напоминание о том, что он пользуется незарегистрированной версией.

Основной способ распространения shareware-программ – это каталоги программного обеспечения или реклама в поисковых системах.

1.4

Abandonware – (англ. abandon – покинуть, оставить; и software – программное обеспечение) программное обеспечение (операционная система, текстовый процессор, компьютерная игра или медиафайл), которое больше не выставляется на продажу компанией-производителем, и от которого производитель больше не получает доходов.

Этот термин не имеет юридического значения, и многое abandonware не является общественным достоянием, и по законам большинства государств не может легально распространяться без разрешения правообладателя. Однако часто правообладатель не занимается преследованием их самовольных распространителей, поскольку они не являются ему конкурентами и не наносят материального ущерба, превышающего расходы на преследование. Во многих случаях принадлежность прав на abandonware-программы не ясна и выяснение надлежащего правообладателя само по себе требует значительных затрат. В некоторых случаях какая-либо компания или сайт получает разрешение от производителя на распространение такой программы. Чаще всего Abandonware распространяется бесплатно (как freeware), иногда – платно (как shareware).

1.5

Adware (англ. Ad, Advertisement – реклама и Software – программное обеспечение) – программное обеспечение, содержащее рекламу.

Вид программного обеспечения, при использовании которого пользователю принудительно показывается реклама.

Также adware называют вредоносное ПО, показывающее рекламу (чаще всего – в интернет-браузере).

1.6

Свободное программное обеспечение (**Free Software**) – программное обеспечение, в отношении которого пользователь обладает «четырьмя свободами»: запускать, изучать, распространять и улучшать программу.

По нынешнему законодательству большинства стран, программный продукт и его исходный код по умолчанию охраняется авторским правом, которое даёт автору (или другому правообладателю, в частности организации-нанимателю – для служебных произведений или наследникам – для умерших авторов) полную власть над распространением и изменением программы, даже в случае, когда исходный код общедоступен для обозрения.

Чтобы программное обеспечение стало «свободным», его правообладатели должны дать пользователю все четыре вышеперечисленные свободы действий. Это достигается выпуском исходного кода программного обеспечения под одной из особого рода лицензий, называемых свободными лицензиями. При этом автор программы сохраняет свои авторские права.

Свободное ПО может одновременно быть и коммерческим – существует множество бизнес-моделей, где не надо платить за каждую копию ПО. Например, платная сервисная поддержка, или коммерческая лицензия для использования свободного кода в собственническом ПО.

Подавляющее большинство открытых программ является одновременно свободными и наоборот, поскольку определения открытого и свободного программного обеспечения близки.

Разработка ПО как научное исследование

Особенность программного обеспечения состоит в том, что оно производится в одной форме – в виде исходного текста, а распространяется и используется в другой – в виде двоичной программы, машинных кодов, по которым невозможно однозначно восстановить исходный текст. Чтобы эффективно изменять программу, исправлять ошибки или даже просто точно установить, что и как делает программа, необходимо располагать её исходным текстом, поскольку при компиляции в машинный код программа утрачивает читабельность.

Первоначально создание программного обеспечения для компьютеров было в первую очередь академическим занятием. Для специалистов в области компьютерной науки каждая программа представляла собой результат научного исследования, в некотором смысле аналогичный публикации статьи. Это означает, что исходный текст программы был обязательно доступен всему научному сообществу, поскольку

любой научный результат должен быть верифицируем, то есть подтверждаться другими исследователями и быть открытым для критики. Таким образом, процесс разработки программного обеспечения более принципиально схож с научным процессом: учёный брал существующие программы, исправлял их в соответствии со своими идеями и публиковал исправленные программы – новый результат.

Однако технология производства компьютеров развивалась не менее активно, чем программное обеспечение для них. В 1970-е годы существовало огромное разнообразие различных архитектур вычислительных машин, различавшихся также и производительностью, и ценой. Естественно, для каждой архитектуры приходилось разрабатывать отдельный набор программного обеспечения. С середины 1970-х в большинстве американских университетов для академических разработок использовались компьютеры архитектуры PDP-10, что позволило сотрудникам разных университетов использовать разработки друг друга на своих машинах. Сотрудники лаборатории искусственного интеллекта Массачусетского технологического института (MIT) в конце 1970-х разработали для PDP-10 собственную операционную систему ITS (Incompatible Timesharing System, несовместимая система с разделением времени) и очень большой набор программ для неё. Исходные тексты написанных в MIT программ были общедоступны, сотрудники других университетов пользовались их исходными текстами и присылали им исправления, всё программное обеспечение в этих лабораториях было полностью академическим.

ПО как «патентованный» продукт

В условиях огромного многообразия архитектур компьютеров программное обеспечение составляло неотъемлемую часть самой машины, причём далеко не самую дорогостоящую часть. Производители компьютеров поставляли их вместе с основным программным обеспечением – по крайней мере, с операционной системой. Производство компьютеров было наукоёмким, но в основе своей коммерческим предприятием.

В ситуации, когда программное обеспечение является объектом продажи наравне с предметами обихода, на него автоматически распространяются уже не только законы научной разработки, но и свойства материальных предметов, которыми можно торговать, обмениваться, право владения и пользования которыми стоит охранять законодательно. Так программное обеспечение попало в разряд интеллектуальной собственности: то есть исходный текст программы стал рассматриваться как произведение, объект применения авторского права. Чтобы защитить свои интересы, производители компьютеров и программного обеспечения используют лицензии – вид договора между обладателем авторских прав и пользователем (покупателем) программного обеспечения. Подобные договоры заключались и с университетами: например, университету передавались исходные тексты программ и право их изменять, но запрещалось распространять их за пределами университета. Подобные ограничения означали, что тексты соответствующих программ не могли открыто обсуждаться в сообществе, то есть не существовали для научной разработки. Были у компьютеров и программного обеспечения покупатели и вне академической среды – например, банки. Таким пользователям не столь важно получить исходные тексты программ, они заинтересованы в программном обеспечении как в законченном продукте и готовы платить деньги за надёжные и удобные программы.

Однако компьютеры развивались очень быстро, и бывшие вполне современными в 1970-е PDP-10 к началу 1980-х уже устарели, и значительно отставали по производительности от более современных машин. Однако ни для одной из новых архитектур уже не было операционной системы и прочего программного обеспечения, разработанного исключительно в академической среде и по её правилам. Теперь университеты должны были покупать новые компьютеры с новым программным обеспечением и выполнять условия лицензии, ограничивающей их права на разработку и распространение ПО – иначе говоря, ограничивающей возможность научной модели разработки программного обеспечения.

В это время в лаборатории искусственного интеллекта MIT разрабатывались так называемые LISP-машины, умевшие на аппаратном уровне интерпретировать язык программирования, похожий на LISP – развитый и перспективный язык программирования. На LISP же была написана операционная система для таких машин и всё программное обеспечение для них. В начале 1980-х некоторые сотрудники лаборатории искусственного интеллекта выкупили у MIT права на LISP-машины и математическую систему Macsyma и основали собственные коммерческие компании для дальнейшей разработки в этой области. Очень многие сотрудники лаборатории перешли работать в эти компании, после чего все их дальнейшие разработки уже становились закрытыми для научного сообщества. Новые LISP-машины распространялись с лицензиями, запрещающими пользователям модифицировать и распространять исходные тексты

программ. Программы, которые раньше для сотрудников MIT были аналогом научных публикаций, стали принадлежащим кому-то патентованным продуктом.

Одному из сотрудников, оставшемуся в лаборатории искусственного интеллекта MIT, Ричарду Столлману, такое положение дел казалось недопустимым нарушением открытого научного процесса разработки программного обеспечения. Он в одиночку пытался в рамках прежней академической модели развивать LISP-машины и открыто реализовывать изменения, аналогичные сделанным в рамках закрытой коммерческой разработки, чтобы LISP-машины MIT могли конкурировать с патентованными аналогами. Конечно, эта попытка угнаться за активной разработкой целой компании была обречена на неудачу.

Тогда в поисках единомышленников Ричард Столлман создаёт некоммерческую организацию «Фонд свободного программного обеспечения». Своей основной целью Фонд ставит сохранение программного обеспечения, процесс разработки которого всегда будет гарантированно открытым, а исходные тексты всегда доступны. Более масштабная цель Фонда – разработка операционной системы, целиком состоящей из открыто разрабатываемого программного обеспечения. Декларируя такую цель, Столлман, фактически, хотел вернуть представлявшееся ему идеальным состояние, когда в MIT работали в собственной операционной системе для PDP-10.

Операционная система, разрабатываемая в рамках Фонда, должна была стать совместимой с операционной системой UNIX. К началу 1980-х UNIX очень широко использовался, в том числе и в академической среде. Для этой операционной системы существовало много программ, свободно распространявшихся в научном сообществе, поэтому хотелось, чтобы эти программы работали и в новой – свободной – операционной системе. Эта будущая операционная система получила название GNU.

Определение свободного ПО

Для того, чтобы сохранить модель научного сотрудничества между разработчиками, необходимо было обеспечить, чтобы исходные тексты программ, написанных разработчиками, оставались доступными для чтения и критики всему научному сообществу. Для этого Ричард Столлман сформулировал понятие **свободное программное обеспечение**, в котором отразились принципы открытой разработки программ в научном сообществе, сложившемся в американских университетах в 1970-е годы. Столлман явно сформулировал эти принципы, они же – **критерии свободного программного обеспечения**. Эти критерии оговаривают те права, которые автор свободной программы передаёт любому пользователю.

- Программу можно использовать с любой целью («**нулевая свобода**»)
- Можно изучать, как программа работает и адаптировать её для своих целей («**первая свобода**»). Условием этого является доступность исходного текста программы.
- Можно распространять копии программы – в помощь товарищу («**вторая свобода**»).
- Программу можно улучшать и публиковать свою улучшенную версию – с тем, чтобы принести пользу всему сообществу («**третья свобода**»). Условием этого является доступность исходного текста программы.

Только удовлетворяющая всем принципам программа может считаться свободной, то есть гарантированно открытой и доступной для научного сообщества. Нужно подчеркнуть, что эти принципы оговаривают только *доступность* программ для всеобщего использования, критики и улучшения, но никак не оговаривают связанные с распространением программ денежные отношения, в том числе *не предполагают и бесплатности*. В англоязычных текстах здесь часто возникает путаница, поскольку слово «free» по-английски означает не только «свободное», но и «бесплатное» и нередко употребляется по отношению к бесплатному программному обеспечению, которое распространяется без взимания платы за использование, но которое недоступно для изменения сообществом, потому что его исходные тексты не опубликованы. Такое бесплатное ПО вовсе не является свободным. Наоборот, свободное ПО вполне можно распространять (и распространяют), взимая при этом плату, однако, соблюдая при этом критерии свободы: каждому пользователю предоставляется право получить исходные тексты программ, изменять их и распространять далее. Всякое программное обеспечение, пользователям которого не предоставляется такого права, является несвободным – независимо от любых других условий.

Открытый доступ к исходным текстам программ является *ключевым* признаком свободного ПО, поэтому предложенный несколько позднее Эриком Реймондом термин «*open source software*» (ПО с открытым исходным текстом) некоторым представляется даже более удачным для обозначения феномена свободного программного обеспечения, чем изначально предложенный Столлманом «free software». Хотя стоит отметить, что Столлман настаивает на различии этих двух понятий, так как open source обладает только одной, а не всеми четырьмя свободами, присущими Свободному ПО.

Общественная лицензия GNU (GNU's Not UNIX – «GNU – не Unix»)

Декларировав критерии свободного ПО, члены Фонда свободного ПО стали распространять свои программы в соответствии с этими принципами, никак не оформляя это документально: иначе говоря, первоначально свободные программы распространялись вообще без лицензии. Однако произошедший с самим Ричардом Столлманом прецедент (см. ниже) убедил его в том, что документальное оформление необходимо для свободного ПО.

Ричард Столлман занимался разработкой текстового редактора Emacs на основе исходных текстов Джеймса Гослинга. Тогда Гослинг свободно раздавал свои исходные тексты всем заинтересованным. Однако в какой-то момент Гослинг продал права на распространение Emacs компании UniPress, и компания попросила Столлмана прекратить распространение его версии Emacs, так как права принадлежат им. Этот инцидент заставил Столлмана переписать заново те части исходного текста Emacs, которые теперь принадлежали UniPress, после чего он разработал собственную лицензию на программное обеспечение.

Лицензия, сформулированная Столлманом, должна была работать так же, как и лицензии на патентованное программное обеспечение: это типовой договор автора программы (обладателя авторских прав) с пользователем, в котором автор, среди прочего, оговаривает права пользователя по отношению к программе. В отличие от типовой коммерческой лицензии, лицензия Столлмана предоставляет пользователю права, являющиеся критериями свободной программы: получать исходные тексты программ, изменять их, распространять изменённые и неизменённые версии. Кроме того, в этой лицензии оговаривается принципиальное для Столлмана условие распространения свободного ПО: ни один пользователь, сделавший модифицированную версию свободной программы, не имеет права распространять её, не соблюдая всех принципов свободного ПО. Иначе говоря, модификацию свободной программы нельзя сделать несвободной. По этой причине лицензию GNU прозвали «вирусной лицензией»: она как бы «заражает» программу, становясь её неотъемлемой частью.

Лицензия, содержащая такие условия, получила название «copyleft». По-английски Право копирования, одно из авторских прав, называется «copyright», поэтому «copyleft» можно условно перевести как «лево копирования» или «авторское лево». Здесь игра слов: right – омоним, обозначающий как Право в юридическом смысле, так и имеющий значение «правый» (справа), указывающее относительное пространственное расположение сторон; в свою очередь английское left – левая сторона. Одновременно в этой игре слов слышаться коннотации с традиционными социально−политическими понятиями «правые» и «левые». Иногда говорят, что условие «copyleft» прямо противоположно по смыслу авторскому праву: если авторское право призвано ограничить пользователя в копировании и распространении копий продукта, то «авторское лево», наоборот, строго запрещает его ограничивать. Впоследствии лицензия Столлмана получила название GNU General Public License («Общественная лицензия GNU»), сокращённо GNU GPL или просто GPL.

В настоящее время помимо GPL известны и другие лицензии, под которыми может распространяться свободное ПО. Самая распространённая из таких лицензий – Лицензия BSD (Berkeley Software Distribution license – Программная лицензия университета Беркли). Лицензия BSD отличается от GNU GPL главным образом тем, что в ней отсутствует условие «copyleft», то есть на основании свободного ПО, распространяемого под этой лицензией, можно производить несвободные модификации. Однако пока лицензия BSD и другие лицензии соответствуют принципам свободного ПО, объявленным Фондом, они остаются лицензиями на свободное программное обеспечение.

Сообщество разработчиков и пользователей

Главное условие существования свободного ПО – все-таки не лицензия, а люди, которые готовы бесплатно делиться текстами своих программ и совершенствовать тексты чужих. Свободное ПО унаследовало модель открытой научной разработки, а вместе с ней – и академическую модель взаимодействия между учёными, вылившуюся в специфическую организацию сообщества разработчиков и пользователей.

Взаимопомощь

У любого пользователя программного обеспечения непременно возникают вопросы, когда он пытается применить его для решения своих задач. Пользователь несвободной (патентованной) программы платит за неё производителю, который взамен предоставляет ему некоторые гарантии, одна из которых – отвечать на вопросы о работе программы. Специально для этого производитель организует *службу*

поддержки, которая по телефону, электронной почте и другим средствам связи отвечает на вопросы пользователей.

Пользователь свободно распространяемой программы не получает вместе с ней никаких гарантий: автор сделал её исходный текст открытым для общества, но при этом не взял на себя обязательств объяснять всем, как работает программа. Хотя, справедливости ради стоит заметить, что любая несвободная программа в 99 % случаях тоже поставляется «как есть» и без гарантий. Поскольку сообщество пользователей большинства программ распределено по всему миру, для организации взаимодействия в нём наиболее активные пользователи (а зачастую и сами авторы) организуют (реже – используют существующие) списки рассылки, форумы и другие средства общения в Интернете. Для накопления и рубрикации информации по программе (в частности, списков часто задаваемых вопросов (ЧаВо; англ. FAQ – frequently asked questions), а также организации более сложных форм взаимодействия (совместной разработки, баг-трекинга) создаются web-сайты, посвящённые программе.

Исправление ошибок

В любой программе непременно имеются ошибки. Производитель патентованной программы создаёт и оплачивает работу отдела контроля качества (QA – Quality assurance), который занимается поиском ошибок. Тем не менее, некоторые ошибки этот отдел пропускает, и они достигают пользователя. Пользователь несвободной программы, столкнувшись с ошибкой, не может выявить её причину (поскольку ему недоступны ни исходные тексты программы, ни даже отладочная информация), но, скорее всего, способен описать ошибку и условия, в которых она происходит. Он может сообщить об ошибке производителю программы (обычно посредством обращения всё в ту же службу поддержки), и если там решат, что ошибка действительно в программе, а не в работе пользователя, о ней будет сообщено разработчикам. В итоге пользователь может ожидать, что в следующей версии программы ошибка будет исправлена. Нередко обновление несвободной программы приравнивается производителем к приобретению новой копии, что влечет за собой соответствующие издержки.

Диагностика ошибки, произошедшей на компьютере пользователя – задача не из лёгких, поскольку у сотрудников службы поддержки (и тем более программистов фирмы) нет к нему доступа. Поэтому отделами поддержки широко практикуются программы, выдающие разнообразную информацию о компьютере пользователя, а в сложных случаях и пресловутая отладочная информация (сотрудник просит пользователя прогнать программу в «диагностическом режиме» (как правило, при помощи недокументированной настройки, либо пользователю присылается отладочная версия нужного модуля) и отправить ему полученный файл отчёта).

У свободно распространяемой программы обычно нет оплачиваемого отдела контроля качества. Значит, пользователь может столкнуться с ещё большим количеством ошибок, чем в патентованной программе. Тем актуальнее для него возможность сообщить об ошибке разработчикам программы. Раньше в сопровождающей программу документации было принято указывать электронный адрес, по которому разработчики принимали сообщения об ошибках (bug report). Некоторые вводили стереотипную форму для таких сообщений, чтобы облегчить и автоматизировать их обработку. Уже это требует существенно более высокой связности сообщества во всём мире, существенно большей, чем достаточно для закрытой разработки.

Разработчики и контролёры-испытатели патентованного продукта могут ходить на службу в один и тот же офис и там обмениваться информацией или тратить определённую долю рабочего времени на составление и анализ строгих отчётностей, содержащих сообщения об ошибках и рапорты об устранении неисправностей. Такая организация труда эффективна, если круг разработчиков невелик, а ввести общую дисциплину относительно легко. Для открытого же проекта круг и взаимное расположение потенциальных разработчиков не ограничены ничем, поэтому эффективность разработки в гораздо большей степени зависит от того, насколько просто всем членам сообщества договариваться между собой, а также от «сознательности» пользователей.

Простому и упорядоченному приёму и перенаправлению сообщений об ошибках служат системы отслеживания ошибок (Bug Tracking System), самые известные из которых разработаны участниками больших проектов для себя, а благодаря свободным лицензиям используются повсеместно. Таковы GNUTS (разработанная в GNU), Bugzilla (Mozilla Foundation), JitterBug (проект Samba) или Debian BTS. Более ранние версии ориентируются на электронную почту, более поздние включают в себя Web-интерфейс. Например, при помощи Bugzilla организуется сайт в Internet, на котором пользователь может заполнить форму сообщения об ошибке. Каждое сообщение имеет свой номер, по которому можно попасть на

«персональную» страницу данной ошибки, где отражаются все происходящие по её поводу события, от первоначального сообщения (открытия) до исправления (закрытия). При каждом изменении в состоянии ошибки Bugzilla рассылает всем заинтересованным лицам (включая, естественно, сообщившего об ошибке и занимающихся данной программой разработчиков) письма по электронной почте. Поскольку Bugzilla позволяет оставлять комментарии и прикладывать файлы, она является полноценным средством для общения пользователя с разработчиком по поводу ошибки в программе.

Принципиальное преимущество пользователя свободной программы заключается в том, что у него, в отличие от пользователей несвободных программ, всегда есть возможность заглянуть в исходные тексты. Конечно, для многих пользователей исходные тексты не более понятны, чем двоичные исполняемые файлы. Однако при достаточном уровне познаний в программировании пользователь может сам установить причину ошибки в программе, а то и устранить её, исправив соответствующим образом исходный текст. А если пользователь заинтересован в развитии программы, то с его стороны будет разумно не только сообщить автору об ошибке, но и прислать ему свои исправления к исходному тексту программы: автору останется только применить эти исправления к тексту программы, если он найдёт их корректными и уместными. Пересылать автору исправленный текст программы целиком непрактично: он может быть очень большим (десятки тысяч строк), и автору будет нелегко разобраться, что же изменено (а вдруг изменения сделаны неграмотно?).

Чтобы облегчить и автоматизировать процесс внесения исправлений, Ларри Уолл в 1984 году разработал утилиту patch («заплатка»), которая в формализованном (но хорошо понятном человеку) виде описывает операции редактирования, которые нужно произвести, чтобы получить новую версию текста. С появлением этой утилиты пользователь, обнаруживший и исправивший ошибку в программе, мог прислать автору небольшую заплатку, по которой автор мог понять, какие изменения предлагаются, и автоматически «приложить» их к своему исходному тексту. С появлением patch гораздо больше пользователей стало включаться в разработку программ с доступным исходным текстом, немалую роль и здесь сыграла сеть Usenet (см. статью об этом Тима О’Рейли). Файлы-заплатки с исправлениями – обязательный атрибут сегодняшней разработки свободных программ.

Если пользователю программы не хватает в ней какой-то функции, то при должной квалификации он вполне может запрограммировать её сам и включить в исходный текст программы. Естественно, ему выгодно, чтобы его дополнение попало в «главный», авторский вариант программы (его называют «upstream») и появлялось во всех последующих версиях: можно точно так же оформить его в виде patch и выслать автору. Этой возможности лишён пользователь несвободной программы, даже если он достаточно квалифицирован. Единственный способ включить в программу нужную ему функцию – обратиться к производителю (если программа патентованная) с соответствующей просьбой, и надеяться, что производитель сочтёт предложенную функцию действительно необходимой.

Чем больше у свободной программы активных пользователей, готовых вносить исправления и дополнения и делиться ими, тем надёжнее работает и быстрее развивается программа. Причём такая свободная модель отслеживания и исправления ошибок для программы, у которой тысячи активных пользователей, может оказаться гораздо более эффективной, чем у любой патентованной программы: ни одна компания не может себе позволить такой огромный штат сотрудников в отделе контроля качества. Поэтому действительно популярная свободная программа может оказаться гораздо надёжнее патентованных аналогов.

Написать большую программу в одиночку довольно сложно и даже не всегда возможно, особенно если автор занимается этим в свободное от работы время. Большинство современных свободных программ пишется группой разработчиков. Даже если начинал писать программу один человек, и она оказалась интересной, к разработке могут присоединиться активные пользователи. Чтобы они могли не только вносить отдельные исправления, но и вообще всю разработку вести совместно, нужны специальные инструменты. Помимо patch, для организации совместной разработки ПО применяются системы контроля версий. Функции системы контроля версий состоят в том, чтобы организовать доступ к исходным текстам программы для нескольких разработчиков и хранить историю всех изменений в исходных текстах, позволяя объединять и отменять изменения и пр. Самая ранняя свободная система контроля версий, RCS использовалась ещё на заре свободного ПО абонентами сети Usenet, затем на смену ей пришла более развитая CVS, но сегодня и она считается во многом устаревшей, и всё чаще заменяется Subversion, Arch и другими.

Нужно заметить, что преимущества свободной разработки для пользователя не следует преувеличивать. Не все свободные программы в равной степени доступны для изменения пользователям, и это

совершенно не связано с лицензией на их распространение. Важный фактор здесь – объём программы: если в ней десятки тысяч строк (как, например, в OpenOffice.org), то даже квалифицированному пользователю потребуется слишком много времени, чтобы разобраться, что к чему. Рассчитывать же на то, что разработчики ответят на все замечания и предложения пользователя немедленным исправлением программы тоже нельзя, поскольку они не несут перед пользователем никаких обязательств по качеству программы. В этом отношении пользователь патентованной программы может оказаться в лучшем положении.

Очень многие свойства сообщества разработчиков и пользователей свободных программ проистекают из того, что все его участники обычно занимаются этой программой из интереса или потому, что эта программа – необходимый для них инструмент (например, зарабатывания денег). Время, потраченное ими на программу, не оплачивается, поэтому нет никакой надежды, что обстоятельства не переменятся и разработка не прекратится вовсе. Нередки случаи, когда разработка программы начинается благодаря одному автору-энтузиасту, который привлекает многих к участию в разработке, а потом энтузиазм лидера гаснет, а вместе с ним затухает и разработка. К сожалению, сегодня существуют тысячи свободных программ, так никогда и не достигших версии 1.0, хотя «выгорание» лидеров и не единственная этому причина. Кроме того, программа может быть необходимой, но «неинтересной», а потому не найдётся и свободных разработчиков.

Место свободных программ на сегодняшнем рынке ПО очень значительно, и многие коммерческие и государственные предприятия используют свободное ПО прямо или опосредованно. Собственно, опосредованно все пользователи Internet задействуют, например, свободную программу BIND, предоставляющую службу DNS. Многие организации, особенно предоставляющие услуги через Internet, используют свободный web-сервер Apache, от работы которого непосредственно зависит их прибыль, не говоря уже о серверах на платформе Linux. Выгода использования свободного ПО очевидна: за него не приходится платить, а если приходится – оно стоит гораздо дешевле патентованных аналогов. Главный недостаток с точки зрения коммерческого пользователя: разработчики свободных программ не несут никаких обязательств по качеству программы, кроме моральных. Поэтому сегодня большие корпорации, например, Intel или IBM, находят необходимым поддерживать проекты по разработке свободного ПО, оплачивая сотрудников, которые работают в рамках этих проектов.

Философия

В европейской культуре долго вырабатывались правила собственности по отношению к материальным ценностям. И вполне логично, что эти правила были распространены на ценности нематериальные – в том числе и программные продукты (когда они начали представлять самостоятельную ценность). Однако, у программных продуктов есть принципиальное отличие от материальных объектов – их можно легко копировать. Создание же копии материального продукта почти равно затратам на создание оригинала.

Из-за указанного различия для ПО не действует принцип «пользоваться вещью одновременно может только один человек» (и использование ее кем-то другим автоматически наносит первому ущерб из-за неполучения блага от нее), по причине которого и существует понятие «хозяин». Поэтому попытка и тут действовать по этому принципу – закреплять право использования программы за одним каким-то человеком – интуитивно воспринимается как противоречащая природе вещей. Неудивительно, что при таком искусственном, силовом насаждении возникает множество неурядиц, каждую из которых приходится решать искусственными, а зачастую и противоестественными методами.

Например, приходится симулировать «ущерб из-за неполучения блага», который «наносится» «хозяину» программы при её безущербном копировании. Обычно это – «упущенная выгода», то есть та прибыль, которую хозяин мог бы получить, но не получил из-за того, что продукт скопировали. Приходится изобретать хитроумную аппаратуру, мешающую копированию, или причиняющую при этом ущерб. Приходится вводить в законодательство особую категорию прав – условно назовём её «патент» – ограничивающую злоупотребления – и свободу – всего человечества в пользу хозяина патента. Причём далеко не всегда хозяин патента и автор изобретения – один и тот же человек (в таких случаях противоестественность данных мер лишь усугубляется).

Несвободные программы называют «проприетарными» или «собственническими» (англ. proprietary). Иногда их неправильно называют просто «коммерческими», что неверно: получать выгоду от программы можно различными способами, и многие успешные свободные проекты это подтверждают.

1.7

Careware (от англ. care – забота и англ. software – программное обеспечение) – вид условно-бесплатного программного обеспечения shareware. Автор данного вида ПО требует, чтобы оплата за него шла на благотворительность. Синоним **charityware**.

Это нетрадиционные условия использования программ, предполагающие скорее не продажу, а обмен программного продукта на что-либо ценное для автора. Как правило, такой обмен необязателен (ваш «товар» расценивается скорее как знак внимания), и вы можете пользоваться программным обеспечением Careware так же, как Freeware.

Автором концепции Careware считается Поль Лютус (Paul Lutus). Сам он объясняет принцип Careware следующим образом: «Иногда деньги являются неудачным эквивалентом для некоторых сделок. Однако приобретение Арахнофилии (HTML-редактор) на условиях CareWare является сделкой… Вы можете использовать программу, при условии, что перестанете жаловаться на трудности и свою жизнь, хотя бы на время… Однако если вы не выполните этого условия, никто не постучит ночью в вашу дверь».

По назначению

2.1

Системное программное обеспечение – это комплекс программ, которые обеспечивают эффективное управление компонентами вычислительной системы, такими как процессор, оперативная память, каналы ввода-вывода, сетевое и коммуникационное оборудование и т.п.

Системное программное обеспечение реализует связь аппаратного и программного обеспечения, выступая как "межслойный интерфейс" с одной стороны которого аппаратура, а с другой приложения пользователя.

2.2

Прикладное программное обеспечение функционирует под управлением определенной операционной системы. Например, текстовый редактор Word является приложением операционной системы Windows, а текстовый редактор Edit – приложением операционной системы MS-DOS. Приложения позволяют пользователю обрабатывать текстовую, графическую, числовую, аудио- и видеоинформацию, а также работать в компьютерных сетях, не владея программированием.

Классификация **1** прикладного программного обеспечения

1. *Прикладное программное обеспечение предприятий и организаций*. Например, финансовое управление, система отношений с потребителями, сеть поставок. К этому типу относится также ведомственное ПО предприятий малого бизнеса, а также ПО отдельных подразделений внутри большого предприятия. (Примеры: Управление транспортными расходами, Служба IT поддержки)
2. *Программное обеспечение инфраструктуры предприятия*. Обеспечивает общие возможности для поддержки ПО предприятий. Это базы данных, серверы электронной почты, управление сетью и безопасностью.
3. *Программное обеспечение информационного работника*. Обслуживает потребности индивидуальных пользователей в создании и управлении информацией. Это, как правило, управление временем, ресурсами, документацией, например, текстовые редакторы, электронные таблицы, программы-клиенты для электронной почты и блогов, персональные информационные системы и медиа редакторы.
4. *Программное обеспечение для доступа к контенту*. Используется для доступа к тем или иным программам или ресурсам без их редактирования (однако может и включать функцию редактирования. Предназначено для групп или индивидуальных пользователей цифрового контента. Это, например, медиа-плееры, веб-браузеры, вспомогательные браузеры и др.
5. *Образовательное программное обеспечение* по содержанию близко к ПО для медиа и развлечений, однако в отличие от него имеет четкие требования по тестированию знаний пользователя и отслеживанию прогресса в изучении того или иного материала. Многие образовательные программы включают функции совместного пользования и многостороннего сотрудничества.

6. *Имитационное программное обеспечение*. Используется для симуляции физических или абстрактных систем в целях научных исследований, обучения или развлечения.
7. *Инструментальные программные средства в области медиа*. Обеспечивают потребности пользователей, которые производят печатные или электронные медиа ресурсы для других потребителей, на коммерческой или образовательной основе. Это программы полиграфической обработки, верстки, обработки мультимедиа, редакторы HTML, редакторы цифровой анимации, цифрового звука и т.п.
8. *Прикладные программы для проектирования и конструирования*. Используются при разработке аппаратного ("Железо") и программного обеспечения. Охватывают автоматизированный дизайн (computer aided design - CAD), автоматизированное проектирование (computer aided engineering - CAE), редактирование и компилирование языков программирования, программы интегрированной среды разработки (Integrated Development Environments), интерфейсы для прикладного программирования (Application Programmer Interfaces).

Классификация **2** прикладного программного обеспечения (школьная)

Прикладное программное обеспечение делится на:

1. приложения общего назначения,
2. приложения специального назначения,
3. обучающие программы,
4. коммуникационные программы,
5. мультимедиа-приложения,
6. компьютерные игры,
7. антивирусные программы.

Практически каждый пользователь нуждается в *приложениях общего назначения*. К ним относятся текстовые редакторы, графические редакторы, электронные таблицы, системы управления базами данных, приложения для создания мультимедиа-презентаций. Наиболее распространенные сегодня: Microsoft Office, Star Office.

Для профессионального использования в различных сферах деятельности квалифицированными пользователями компьютера используются *приложения специального назначения*. К ним относятся системы компьютерной графики, системы автоматизированного проектирования (САПР), бухгалтерские программы, компьютерные словари, системы автоматического перевода и др.

Обучающие программы используются для самообразования или в учебном процессе. Прежде всего, это программы обучения иностранным языкам, программы-репетиторы, тесты по различным предметам и др.

Большое значение приобретают *коммуникационные программы*. Это связано с развитием глобальных и локальных сетей. Сегодня разработчики операционных систем (например, Windows) включают коммуникационные программы непосредственно в состав операционной системы.

Большую пользу приносят различные *мультимедиа-приложения* (энциклопедии, справочники и так далее) на лазерных дисках, которые содержат огромный объем информации и средства быстрого ее поиска.

Компьютерные игры бывают различных типов: логические, спортивные, стратегические и так далее.

В связи с широким распространением компьютерных вирусов в отдельную группу выносят *антивирусные программы*: полифаги, ревизоры, блокировщики, иммунизаторы.

2.3

Инструментальное программное обеспечение – программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ. Обычно этот термин применяется для акцентирования отличия данного класса ПО от прикладного и системного программного обеспечения.

Виды инструментального ПО

- *Текстовые редакторы*.

- *Интегрированные среды разработки* (*IDE, Integrated development environment*). Интегрированная среда разработки – это система программных средств, используемая программистами для разработки программного обеспечения.
- *SDK (Software Development Kit)* – это набор из средств разработки, утилит и документации, который позволяет программистам создавать приложения по определённой технологии или для определённой платформы (программной или программно-аппаратной).
- *Компиляторы.* Компилятор – это транслятор, который осуществляет перевод всей исходной программы в эквивалентную ей результирующую программу на языке машинных команд (микропроцессора или виртуальной машины). Примечание: Транслятор – это программа, которая принимает на вход программу на одном языке (он в этом случае называется *исходный язык*, а программа – *исходный код*), и преобразует её в программу, написанную на другом языке (соответственно, *целевой язык* и *объектный код*).
- *Интерпретаторы*. Интерпретатор – это программа для **интерпретации**, т.е. непосредственного исполнения программ (производства вычислений, предписываемых этими программами) из исходного кода на определённом языке.
- *Линковщики (или иначе называют Компоновщики, редактор связей, англ. linker, link editor)*. Компоновщик – это программа, которая производит *компоновку* – принимает на вход один или несколько объектных модулей и собирает по ним исполняемый модуль
- *Парсеры и генераторы парсеров (см. Javacc)* – **1.** В информатике, синтаксический анализ (парсинг) – это процесс анализа входной последовательности символов с целью разбора грамматической структуры, обычно, в соответствии с заданной формальной грамматикой. *Синтаксический анализатор (парсер)* – это программа или часть программы, выполняющая синтаксический анализ. **2.** В информатике грамматический анализ (грамматический разбор, парсинг) – это процесс сопоставления линейной последовательности лексем (слов, токенов) языка с его формальной грамматикой. Результатом обычно является дерево разбора. Обычно применяется совместно с лексическим анализом, в процессе синтаксического анализа. *Грамматический анализатор (парсер)* – программа или алгоритм, осуществляющие грамматический разбор. **3.** *Parser* – это технология создания сайтов с помощью простого языка. В интернет - терминологии граббер – это скрипт, позволяющий "вытягивать" данные с других сайтов на свой. А парсер может распознавать нужную информацию в файле скачанных данных и обработать ее в соответствии с задачей. В настоящее время оба названия используются как равнозначные. **4.** *javacc* – средство создания классов на языке java для проверки и разбора структурированного текста
- *Ассемблеры*. Ассемблер – это компьютерная программа, компилятор исходного текста программы написанной на языке ассемблера, в программу на машинном коде
- *Отладчики*. Отладчик является модулем среды разработки или отдельным приложением, предназначенным для поиска ошибок в программе. Отладчик позволяет выполнять пошаговую трассировку, отслеживать значения переменных в процессе выполнения программы, устанавливать точки или условия останова и т. д.
- *Профилировщики*. Профилирование – это процедура измерения затрат времени на выполнение строк программы.
- *Генераторы документации*. Генератор документации – это программа или пакет программ, позволяющая получать документацию, предназначенную для программистов (документация на API) и/или для конечных пользователей системы, по особым образом комментированному исходному коду и, в некоторых случаях, по исполняемым модулям (полученным на выходе компилятора).
- *Средства анализа покрытия кода*. Покрытие кода – мера, используемая при тестировании программного обеспечения. Она показывает процент, насколько исходный код программы был протестирован.
- *Средства непрерывной интеграции*. Непрерывная интеграция (англ. *Continuous Integration*) – это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.
- *Средства автоматизированного тестирования*.
- *Системы управления версиями*. Система управления версиями – это программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями

позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое

- и др.

Источники:

1. http://ru.wikipedia.org
2. http://www.tspu.tula.ru/ivt/old_site/lcopy/Exponenta_RU/soft/matlab/potemkin/book/matlab/chapter4/4_3.asp.htm
3. http://www.parser.ru/
4. http://newsgrabber.net.ru/