Визитка

ДЕНИС СИЛАКОВ, к. ф.-м. н., старший архитектор ЗАО «РОСА», занимается автоматизацией разработки ОС «РОСА»

Контроль качества дистрибутива Linux

В прошлом номере мы разобрались с тем, как устроена сборка дистрибутива Linux [1]. В заключительной части статьи увидим, как обеспечивается его «правильная» работа

Типичный дистрибутив Linux содержит тысячи программных компонентов, насчитывающих в сумме сотни миллионов строк кода на различных языках. В 2008 году эксперты Linux Foundation оценили стоимость разработки дистрибутива Fedora (девятой версии) в 60 тысяч человеко-лет (http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html), или в 10 млрд долларов (по расценкам того периода).

В то же время над поддержкой большинства дистрибутивов трудится не так уж и много людей – например, штатная команда РОСЫ насчитывает несколько десятков человек. А многие системы и вовсе поддерживаются силами всего нескольких энтузиастов.

Но по приведенным выше числам понятно, что даже несколько сотен разработчиков, мэйнтейнеров и тестировщиков не в состоянии полностью контролировать весь объем исходного кода дистрибутива и своевременно исправлять обнаруживаемые проблемы, не говоря уже о том, чтобы развивать все входящие в систему продукты.

Тем не менее большинство вариаций Linux является достаточно стабильными и надежными системами с регулярно обновляемым ПО, пригодными не только для домашнего применения, но и для использования в областях, очень требовательных к качеству и надежности. Расскажу как обеспечивается высокое качество системы при небольшом количестве людей, работающих над ней.

Использование наработок сообщества

Одним из ключевых факторов, позволяющим обойтись малым количеством разработчиков при создании ОС на основе Linux, является возможность переиспользовать результат работы большого сообщества, сформировавшегося вокруг этой системы. Ведь значительная часть компонентов практически любого дистрибутива Linux создается сторонними разработчиками, не имеющими к дистрибутиву никакого отношения. Эти же разработчики проводят тестирование своих программ, поэтому в дистрибутивы обычно отправляется не сырой код, а готовый продукт, прошедший определенные проверки. Конечно, многие открытые приложения разрабатываются небольшими командами, не располагающими ресурсами для серьезного тестирования. Но многие крупные приложения (такие, как Mozilla Firefox или LibreOffice) развиваются некоммерческими организациями, имеющими собственные команды тестировщиков.

Кроме того, большинство компонентов не уникальны для конкретного дистрибутива и используются во множестве других систем. И даже если непосредственные разработчики какого-то компонента пропустили в нем какие-то ошибки, то они могут быть выявлены разработчиками и тестировщиками одного из этих «других» дистрибутивов. Поскольку процессы создания дистрибутивов являются открытыми, то о выявленных ошибках сразу же становится известно всем заинтересованным лицам.

Более того, «позаимствовать» из другой системы можно не только информацию об ошибке, но и исправление для нее. Нередко ошибка обнаруживается в одном дистрибутиве, а исправления предлагаются разработчиками другого. В этом заключается одна из сильных сторон открытого процесса разработки, позволяющего свободно обмениваться знаниями и кодом. Однако при разработке дистрибутива, нацеленного на при-менение на предприятиях и сопровождаемого платной технической поддержкой, нельзя рассчитывать исключительно на помощь «со стороны». Ведь всегда есть вероятность столкнуться с проблемами, специфичными для конкретной системы. И чем меньше сходства у дистрибутива с другими представителями семейства Linux, тем эта вероятность больше. Поэтому при подготовке очередной версии дистрибутива, равно как и при внесении изменений в уже выпущенный релиз, необходимо проводить его тщательное тестирование. Каким же образом происходит тестирование дистрибутива его непосредственными разработчиками, и каковы основные сценарии работы при обнаружении ошибок? Рассмотрим эти вопросы на примере РОСЫ, однако большая часть сказанного в равной степени справедлива для всех дистрибутивов, развиваемых относительно крупными командами.

Автоматизация тестирования

Помимо использования наработок других команд, существенную помощь в экономии человеческих ресурсов дает

автоматизация процессов тестирования. Часто применяемой практикой в современном мире ИТ является «непрерывная интеграция» (Continuous Integration, CI), подразумевающая проведение регулярных сборок проекта и его тестирование в автоматическом режиме.

Дистрибутив Linux – отличный пример проекта, в котором использование CI более чем актуально. Ведь в разрабатываемой версии дистрибутива ежедневно могут обновляться десятки и даже сотни компонентов, так что вопрос регулярного тестирования дистрибутива в его текущем состоянии стоит остро. Поэтому при разработке основных вариаций Linux используется непрерывная интеграция, в той или иной степени.

Во второй части статьи про РОСУ я уже рассказывал о средствах контроля качества репозиториев. Подобные инструменты используются во многих дистрибутивах, однако они сводятся к статическому анализу пакетов и позволяют выявлять достаточно узкий класс проблем. Более актуальной является проверка функционирования компонентов дистрибутива при реальной работе. Для этого в РОСЕ при разработке новой версии дистрибутива производится регулярная сборка установочных образов системы (файлов в формате ISO, которые можно записывать на диск или флеш-носитель или просто подключать к виртуальной машине). Каждый образ отправляется на тестирование, в рамках которого осуществляются следующие проверки (в полностью автоматическом режиме, без участия человека):

- запускается ли ОС с образа в Live-режиме, не требующем установки на жесткий диск;
- можно ли установить ОС с образа (при этом запускается установщик и имитируется нажатие клавиш, позволяющее установить систему с параметрами по умолчанию);
- корректно ли подключаются репозитории после установки системы и можно ли произвести обновление пакетов до самых свежих версий из репозиториев;
- наконец, внутри установленной системы запускаются автоматические тесты для различных компонентов ОС.

Все действия производятся в специально создаваемой для каждого тестирования чистой виртуальной машине. В качестве средства виртуализации используется Qemu в связке с KVM.

Автоматические тесты запускаются с помощью autotest (http://autotest.github.com) – открытой системы автоматического тестирования, изначально созданной для ядра Linux, но сейчас применяемой и в других проектах. Что касается конкретных тестов, то в РОСЕ применяются свободно распространяемые тестовые наборы, такие как dbench (стресс-тест для компонентов ядра, отвечающих за работу файловых систем), xfstest (тесты для файловых систем), тесты Linux Standard Base (прежде всего нацеленные на проверку API и ABI библиотек) и другие. В будущем планируется добавить тесты не только для системных компонентов, но и для приложений с графическим интерфейсом (в первую очередь уникальных программ собственного производства).

Результаты тестирования сборок РОСЫ общедоступны, и интересующиеся могут посмотреть на них на странице http://fba.rosalinux.ru/autotest. Здесь представлены отчеты всех автоматических тестов, запущенных с помощью autotest, а также пошаговые снимки экрана в процессе установки системы и ее запуска в Live-режиме.

Тестирование выпущенных и разрабатываемых версий

Полномасштабное тестирование всей системы производится только для версий, находящихся в разработке. Целенаправленного тестирования уже выпущенных версий не производится, за исключением подготовки отдельных образов (сервис-паков), включающих все обновления, накопившиеся с момента релиза. Основным источником сообщений об ошибках в уже выпущенных версиях являются пользователи, обращающиеся в HelpDesk или Bugzilla.

При подготовке же очередного релиза, помимо автоматических тестов, перед выпуском каждой промежуточной версии производятся полномасштабные проверки всех компонентов системы. По результатам этих проверок принимается решение о том, удовлетворяет ли промежуточная версия соответствующим критериям качества и можно ли ее выпускать.

В РОСЕ тестированием системы занимается специально выделенная команда QA, о которой я уже упоминал в первой части статьи. При тестировании члены QA руководствуются планами тестирования, включающими:

- проверку работы установщика системы (правильно ли определяется оборудование, корректно ли система делит машину с уже установленными ОС и т.д.);
- проверку возможности обновления старой версии системы до новой (если она предусмотрена);
- проверку корректности загрузки установленной системы (здесь важно проверить не просто факт успешности загрузки, но и посмотреть, не возникают ли при этом сообщения о потенциальных проблемах, и нет ли компонентов, существенно замедляющих загрузку);
- проверку работы устройств, подключенных к машине;
- тестирование приложений, входящих в состав системы по умолчанию, в соответствии с типичными сценариями их использования.

В отношении последнего пункта необходимо отметить, что в отличие от некоторых коммерческих ОС дистрибутивы Linux по умолчанию поставляются с богатым набором прикладного ПО – офисным пакетом, графическими редакторами, мультимедиа-проигрывателями и прочими программами, которые понадобятся большинству пользователей. Тестировщики должны убедиться, что основные прикладные программы можно по крайней мере запустить и начать использовать.

При обнаружении ошибок в баг-трекере РОСЫ заводятся соответствующие сообщения и производится анализ критичности (об этом процессе будет рассказано ниже). По результатам проверок лидер QA составляет сводный отчет и передает его разработчикам и техническому комитету вместе с заключением команды QA о возможности выпуска дистрибутива. На основе этого отчета ТК принимает решение о выпуске – как правило, это решение следует вердикту QA, но иногда ТК может решить, что наличие определенных ошибок не является препятствием к выпуску.

Важной составляющей тестирования предварительных версий ОС является привлечение добровольцев из числа пользователей дистрибутива. Поскольку процесс разработки открыт, то никаких затруднений с этим не возникает – любой желающий может скачать предварительную версию системы и попробовать ее в действии. Привлечение большого числа тестировщиков-добровольцев позволяет решить следующие проблемы:

- увеличить широту охвата проверяемого оборудования; одной из часто встречающихся проблем в Linux является некорректная работа тех или иных специфических устройств, а проверить все возможные сочетания оборудования силами штатных тестировщиков не представляется возможным;
- провести тщательное тестирование тех или иных приложений, ведь у каждого человека есть свой набор часто используемых программ, в которых он заметит даже незначительные ошибки.

Ошибки, найденные добровольными участниками тестирования, также заносятся в баг-трекер РОСЫ и учитываются при составлении отчета QA.

Таким образом, официальным местом отслеживания известных проблем и статуса работ по их исправлению является система учета ошибок (баг-трекер), в качестве которой в РОСЕ (и многих других открытых проектах) используется Bugzilla. Как отмечалось в первой части статьи, имеется дополнительная система для взаимодействия с пользователями – HelpDesk, однако разработчики не принимают участия в ее работе. Если какое-то обращение в HelpDesk признается ошибкой в дистрибутиве, требующей внимания разработчиков, то представители QA заводят соответствующее сообщение в Bugzilla.

На основе анализа Bugzilla можно сделать выводы о том, насколько активно развивается ОС, какие проблемы являются наиболее актуальными, и получить другую статистику, позволяющую судить о «здоровье» дистрибутива.

Для того чтобы проводить такой анализ или просто эффективно использовать систему учета ошибок, необходимо понимать, как устроен процесс обработки каждого сообщения о проблеме и что означают различные атрибуты, которые выставляются для него в Bugzilla или других системах.

Первичный анализ сообщения об ошибке

Как правило, количество сообщений об ошибках достаточно велико, они поступают постоянно, исправлять их моментально не представляется возможным. Ошибки бывают разные – некоторые сводятся к косметическим недочетам и опечаткам в тексте сообщений, другие являются критическими и мешают нормальному функционированию системы, третьи являются серьезными, но проявляются только в очень специфических условиях и так далее.

Кроме того, вокруг некоторых сообщений разворачиваются бурные дискуссии: ошибка ли это или особенность ОС, обусловленная ее архитектурой? Такие «ошибки» не приводят к краху приложений или проблемам с безопасностью, как правило, они сводятся к тому, что система работает не так, как ожидает пользователь. Однако пользователи бывают разные, и ожидания у них – тоже. И разработчикам необходимо найти решение, которое устроит всех пользователей либо сведет количество недовольных к минимуму.

Поэтому при поступлении сообщения об ошибке проводится проверка, действительно ли обнаружена ошибка в дистрибутиве или некорректное поведение является следствием некорректных действий пользователя. Сообщения, признанные ошибками, переводятся в статус CONFIRMED. Такой статус могут выставлять только определенные люди – основные разработчики, мэйнтейнеры и представители команд безопасности и QA.

Если сообщение об ошибке заводится человеком, имеющим полномочия выставлять статус CONFIRMED, то этот статус выставляется сразу. В частности, так помечаются все отчеты, поступающие от HelpDesk, ведь для них уже проведен предварительный анализ и подтвержден факт наличия ошибки. Остальные сообщения по умолчанию имеют статус UNCONFIRMED и должны быть проанализированы членами команды Bug Triage («сортировщиками» ошибок). Более точно, в обязанности команды входит:

- анализ сообщений в статусе UNCONFIRMED и либо их перевод в CONFIRMED (если проблему удается воспроизвести), либо отклонение (перевод статуса в INVALID);
- уточнение, какие из поддерживаемых версий дистрибутива подвержены ошибке;
- определение пакета, к которому относится ошибка, и назначение ответственного за ее исправление (как правило, им является мэйнтейнер соответствующего пакета).

Отклонение сообщения происходит, если в ходе обсуждения с пользователем выясняется, что ошибка вызвана некорректными настройками системы либо действиями, не предусмотренными штатным режимом работы приложения. Пользователю при этом даются рекомендации, каким образом надо настроить систему, чтобы приложение работало правильно. Также статус INVALID выставляется, если воспроизвести ошибку не удается, а пользователь в течение долгого времени (в РОСЕ принят срок в три месяца) не отвечает на уточняющие вопросы. Если же наличие проблемы признается, то производится первичный анализ ошибки с участием человека (или команды), назначенного ответственным за ее исправление. В рамках этого анализа определяется первоисточник проблемы. Если ошибка не является уникальной для конкретного дистрибутива, а изначально присутствует в коде, взятом от сторонних разработчиков, то для такого отчета в Bugzilla выставляется атрибут Upstream, который может принимать следующие значения:

unknown – разработчикам исходного компонента еще неизвестно об ошибке;

known – ошибка известна и планируется ее исправление;

wontfix – разработчики не считают такое поведение ошибкой.

Если ошибка уже исправлена в новых версиях ПО, то атрибут Upstream не выставляется, а сразу производится обновление соответствующего пакета в дистрибутиве либо перенос исправлений в ту версию, которая используется в ОС.

Выставление атрибута Upstream отнюдь не значит, что разработчики дистрибутива отказываются исправлять проблему и перекладывают ответственность на создателей оригинального продукта. Этот флаг – всего лишь помощь мэйнтейнерам, напоминающая, что решение данной проблемы будет актуально и для других систем и его можно искать у других дистрибутивов либо вырабатывать совместно.

В некоторых ситуациях разработчики дистрибутива могу согласиться с наличием проблемы, но отказаться от ее исправления, например, если проблемный компонент признается устаревшим и в скором будущем планируется заменить его на более совершенный аналог. Тогда сообщение об ошибке отклоняется со статусом WONTFIX. При первичном анализе происходит оценка критичности ошибки, от которой зависит приоритет ее исправления. Для критичности ошибки предусмотрены следующие значения:

blocker – ошибка нарушает работоспособность всей системы или большой группы пакетов, препятствуя загрузке системы, установке обновлений или приводя к потере данных пользователя; как правило, такие ошибки возникают только на ранних этапах подготовки очередной версии дистрибутива, и их исправление является необходимым условием выпуска версии;
critical – ошибка приводит к неработоспособности одного из ключевых компонентов системы у большинства пользователей или приводит к потере данных при использовании этого компонента; также в РОСЕ критической ошибкой считается невозможность пересобрать пакет из исходного кода в текущем окружении;
major – пакет, в котором обнаружена ошибка, непригоден для использования;
normal – «обычная» ошибка, не попадающая в другие категории;
minor – небольшая ошибка, в целом не влияющая на функциональность системы либо имеющая простые способы обхода; в эту же категорию попадают ошибки, являющиеся серьезными с точки зрения нарушения функциональности системы, но проявляющиеся только в специфических условиях у небольшого количества пользователей;
trivial – тривиальные ошибки наподобие опечаток в документации;
enhancement – ошибки как таковой нет, но приложение делает не то, что ожидает от него пользователь; такие отчеты – не сообщение об ошибке, а запрос на добавление функциональности в существующий компонент.

После такого анализа начинается непосредственная работа над предоставлением исправлений. Отмечу, что, несмотря на длинное описание, процесс первичного анализа многих ошибок занимает несколько минут, так что не следует воспринимать его как излишнюю бюрократию, тормозящую процесс разработки.

Обработка ошибок и внедрение исправлений

Ошибки со статусом CONFIRMED подлежат исправлению разработчиками и мэйнтейнерами дистрибутива. Когда разработчик берется за исправление той или иной ошибки, он переводит его статус в IN_PROGRESS. По возможности определяется срок, к которому будет готово исправление, в терминах версий дистрибутива (как правило, такой подход используется для ошибок в разрабатываемой ветке системы – для них указывается, к какой промежуточной версии планируется предоставить исправления).

После внесения необходимых изменений в пакеты ошибка переводится в статус RESOLVED FIXED. Если ПО, в котором обнаружена ошибка, входит в состав официально поддерживаемых пакетов уже выпущенной версии дистрибутива (либо промежуточной версии на той стадии, когда запрещено внесение изменений без согласования с QA), то разработчик запрашивает проверку факта исправления со стороны команды QA. Запрос делается формально выставлением флага «Request for update».

После этого к работе приступает команда QA, проверяющая два аспекта: действительно ли проблема исправлена и не привело ли исправление к появлению других проблем.

При внесении серьезных изменений в ключевые компоненты ОС QA запрашивает анализ изменений со стороны Security team – нет ли у этих новых версий известных проблем с безопасностью.

Нередко в ходе поверки между тестировщиками и мэйнтейнерами идут активные дискуссии. Последние могут уточнять, какие изменения работы системы после обновления являются ожидаемыми и на какие аспекты следует обратить особое внимание. QA, в свою очередь, может спрашивать мэйнтейнеров, действительно ли то или иное изменение поведения является корректным. Все такие обсуждения рекомендуется вести непосредственно в баг-трекере, избегая личной переписки и других приватных способов обмена сообщениями. Это позволяет хранить историю обсуждений в общедоступном месте, что полезно при анализе причин тех или иных изменений разработчиками, не принимавшими непосредственного участия в их обсуждении и реализации. Кроме того, системы учета ошибок многих дистрибутивов (в том числе и РОСЫ) индексируются поисковыми машинами, так что разработчики и пользователи других систем также могут применять накапливающуюся в них базу знаний.

По результатам проверки QA выставляют переключатель «qa_verified», а команду безопасности «secteam_verified» – в значение «+» (одобрено) или «-» (отклонено). Если хотя бы одна команда пришла к выводу о недопустимости предложенного исправления, то проблема отправляется на доработку, а сообщение в Bugzilla возвращается в статус CONFIRMED.

Если же проблем не выявлено, то выставляется запрос на публикацию обновленного пакета установкой переключателя «published» в значение «?». Отчеты с таким статусом просматриваются менеджером репозитория, который дополнительно анализирует, не нарушит ли обновление целостность репозитория и не приведет ли к конфликтам

Рисунок 1. Типичный вид обработанного сообщения об ошибке

с другими пакетами. Если и здесь проблем нет, то обновленный пакет помещается во внутренний репозиторий РОСЫ, расположенный в сборочной среде ABF.

Каждое обновление официально поддерживаемого пакета сопровождается бюллетенем, в котором вкратце описано, какую проблему это обновление исправляет. Бюллетень формируется разработчиком, дорабатывается сотрудником QA и им же привязывается к конкретной сборке пакета в ABF. В ABF можно просмотреть список всех бюллетеней к той или иной версии системы и получить представление о том, насколько активно поддерживается дистрибутив и какого рода проблемы в нем исправляются.

Отмечу, что ABF предоставляет очень удобные возможности по подготовке пакетов с исправлениями. В частности, в подсистеме сборки в отдельный шаг выделена публикация собранного пакета в репозиторий. Вместо публикации в целевой репозиторий пакет после сборки может быть помещен в так называемый контейнер – небольшой репозиторий, содержащий только результаты сборки конкретного проекта. Контейнер можно подключить к системе наравне с другими репозиториями. При подготовке исправлений разработчики сначала собирают обновленные пакеты в соответствующие контейнеры. После подключения такого контейнера в системе должно появиться уведомление о доступном обновлении (отсутствие уведомления означает, что мэйнтейнер при пересборке забыл увеличить версию пакета и ОС не отличает его от уже установленного, в таком случае пакет необходимо отправить на доработку). Члену команды QA, желающему проверить исправление, остается только установить обновление штатными средствами системы.

При исправлении проблем в официально неподдерживаемых пакетах QA и команда безопасности не привлекаются. Обновленный пакет предлагается к публикации сразу после внесения исправлений. Менеджер репозитория проверяет, что пакет действительно не подлежит проверке QA, что обновление не нарушает целостности репозитория и не оказывает влияния на официально поддерживаемые программы, и при отсутствии проблем помещает пакет во внутренний репозиторий ABF.

Однако и на этом процесс еще не заканчивается – между помещением пакета во внутренний репозиторий на ABF и его появлением в официальных репозиториях, доступных пользователям, введена искусственная задержка в несколько часов. При этом системы всех разработчиков РОСЫ настроены на внутренние репозитории ABF (полагаю, достаточно очевидно, что разработчики РОСЫ работают в системе собственного производства). Системы же конечных пользователей настроены на использование официальных репозиториев или их зеркал. Поэтому все разработчики получают обновления немного раньше, чем они становятся доступны пользователям. И в случае каких-либо проблем, у разработчиков есть время, чтобы откатить изменения. Все разработчики дистрибутива являются тестовым полигоном, на котором апробируются все обновления перед тем, как отправиться к пользователям.

Исправления проблем с безопасностью

Описанные выше процессы применяются для обработки практически всех сообщений об ошибках, за исключением обновлений, исправляющих проблемы с безопасностью системы, ведь такие обновления необходимо доставить пользователям как можно быстрее. За обнаружением уязвимостей в ПО дистрибутива следит Команда безопасности (Security team).

При обнаружении проблем с безопасностью эта команда заводит соответствующие сообщения об ошибках, доступные только разработчикам дистрибутива, но она обладает полномочиями самостоятельно публиковать исправленные пакеты в репозиторий, в обход команды QA и даже менеджера репозитория. Впрочем, модификации критически важных системных пакетов (ядра, glibc, systemd и тому подобного, где даже тривиальное изменение может нарушить функциональность всей системы) подвергаются обязательному анализу всеми командами. Однако такие случаи возникают достаточно редко и обрабатываются оперативно.

***

Итак, существенную роль в обеспечении качества дистрибутива Linux играют его пользователи, а также все мировое сообщество свободного ПО. Однако если система претендует на промышленную применимость, то наличие такой взаимопомощи не избавляет от необходимости иметь собственные эффективные процессы обнаружения, отслеживания и исправления ошибок. Во многих дистрибутивах эти процессы построены схожим образом, и существенную роль в них играет автоматизация. Одним из ключевых инфраструктурных компонентов, поддерживающих обеспечение качества ОС (как и многих других продуктов), является система учета ошибок, в которой происходит обсуждение обнаруживаемых проблем и способов их решения.

Рассмотренные в статье процедуры работы с сообщениями об ошибках могут показаться сложными и требующими существенных усилий для выполнения всех формальностей. Однако многие задачи выполняются автоматически соответствующими инструментами в инфраструктуре разработки. Например, извещения об изменениях статусов рассылаются всем заинтересованным лицам по e-mail (в частности, QA, Security team и менеджер репозитория получают письма о выставлении переключателей «qa_verified», «secteam_verified» и «published» соответственно). Кроме того, Bugzilla предоставляет мощные возможности по построению аналитических отчетов и поиску ошибок по определенным критериям, что позволяет эффективно работать при наличии сотен тысяч, а то и миллионов сообщений.

Тестирование системы – это задача, где пользователи могут оказать заметную помощь, не обладая глубокими техническими знаниями. Не все любят формализм, однако для эффективной разработки большой и сложной системы, каковой является дистрибутив Linux, наличие определенных формальных процессов неизбежно, иначе возникает хаос. Одна из задач разработчиков дистрибутива – сделать формальные процессы прозрачными, не раздражающими и не отнимающими много времени. Тогда и пользователи будут довольны, внося посильный вклад в развитие системы, и разработчики получат пользу, а не будут разгребать сотни бесполезных сообщений об ошибках. EOF

1. Силаков Д. Что такое дистрибутив Linux? Разработка дистрибутива Linux на примере РОСЫ. Часть 2. //«Системный администратор», №3, 2013 г. – С.83-87.