АНО «Институт логики, когнитологии и развития личности»
ALT Linux

Пятая конференция разработчиков свободных программ на Протве

Обнинск, 21–23 июля 2008 года

Тезисы докладов

Москва,
Институт Логики,
2008

В книге собраны тезисы докладов, одобренных Программным комитетом Пятой конференции разработчиков свободных программ. Круг рассматриваемых тем весьма широк: от новейших системных и прикладных разработок до правовых проблем, вопросов организации работы в проектах и аналитики.

Программа конференции

21 июля

10.00–12.00: Регистрация

12.00–12.30: Кофе

Доклады спонсоров
12.30–14.00

14.00–14.45: Обед

Дневное заседание
14.45–16.45

16.45–17.15: Кофе

Вечернее заседание
17.15–19.15

22 июля

Утреннее заседание
09.30–11.30

11.30–12.00: Кофе

Дневное заседание
12.00–14.00

14.00–14.45: Обед

Дневное заседание
14.45–16.45

16.45–17.15: Кофе

Вечернее заседание
17.15–19.15

23 июля

Утреннее заседание
09.30–11.30

11.30–12.00: Кофе

Дневное заседание
12.00–14.00

Вне программы

Дмитрий Ткачёв
Москва, Т-Платформы

Программа exCellenT-Platforms — разработка программного обеспечения для архитектуры процессора Cell

Компания «Т-Платформы» — ведущий российский разработчик и производитель комплексных решений для высокопроизводительных вычислений. «Т-Платформы» — единственная российская компания, пять собственных кластерных решения которой вошли в рейтинг самых мощных суперкомпьютеров мира Top500. С декабря 2007 года компания «Т-Платформы» начала реализацию программы «exCellenT-Platforms», основная цель которой — разработка и внедрение комплексных решений на базе инновационных процессоров Cell, а также активное объединение сообществ разработчиков, что позволит значительно ускорить реализацию внутренних возможностей Cell в отечественных программных разработках.

Основная цель проекта exCellenT-Platforms — разработка и внедрение линейки комплексных решений на базе инновационных восьмиядерных процессоров CELL, а также активное объединение и развитие сообществ разработчиков, что позволит значительно ускорить реализацию внутренних возможностей Cell в отечественных программных разработках. Наличие столь мощных вычислительных решений параллельной обработки данных позволит российской науке сделать большой шаг навстречу новым открытиям, а также внесет значительный вклад в развитие экономики страны.

Процессор CELL был разработан альянсом Sony Group, Toshiba и IBM в центре STI Cell Design Center в Остине, штат Техас. Его архитектура предусматривает использование управляющего процессора на базе архитектуры Power, совместно с которым работают несколько высокопроизводительных процессорных элементов Synergistic Processor Element (SPE) с архитектурой SIMD, и широкого набора команд DMA для эффективного обмена между процессорными элементами.

Компания «Т-Платформы» и Фонд содействия малых форм предприятий в научно-технической сфере объявляют конкурс среди молодых программистов на лучшие проекты по разработке и адаптации

программного обеспечения для решений на базе восьмиядерных процессоров Cell.Конкурс с призовым фондом в1 000 000 рублей проходит в рамках программы «exCellenT-Platforms» и программы «Участник молодежного научно-инновационного конкурса» (УМНИК) Фонда содействия малых форм предприятий в научно-технической сфере. В рамках программы УМНИК Фонд содействия поддерживает научно-техническую инновационную деятельность молодежи и выделяет гранты на реализацию проектов НИОКР молодыми специалистами.

Павел Мельников
Москва, ETegro Technologies

Практическая демонстрация системы тестирования Inquisitor на серверном оборудовании ETegro Technologies

О компании ETegro Technologies

ETegro Technologies является ведущим разработчиком высокопроизводительных серверов, систем хранения данных, графических станций и индустриальных компьютеров для любых применений, от web-сервера начального уровня до центров обработки данных крупных компаний.

Инновационные продукты ETegro Technologies созданы для работы в современной ИТ-инфраструктуре, при этом предоставляя запас мощности и наращиваемости для задач завтрашнего дня. Сбалансированная архитектура, высокоскоростные объединительные среды, гибко конфигурируемые опции адаптеров ввода-вывода, подсистем хранения, высокая производительность, высочайшая надёжность — всё это делает продукты ETegro отличным выбором. Постоянные инвестиции в исследование новых технологий позволяют ETegro

Technologies быть реальным лидером ИТ-рынка. Среди наших клиентов — такие общепризнанные лидеры своих отраслей, как АК «Алроса», ОАО НК «Роснефть», ОАО РЖД, АКБ Региональный Кредит, медиа-холдинг РБК, ОАО «Вебальта», Rambler, Masterhost и многие другие.

Неизменно высокое качество продукции и профессионализм команды ETegro позволили компании за три года своего существования добиться значительных успехов и занять общее пятое место на серверном рынке России и стран СНГ с совокупной долей около 7 % (по данным IT Research, 4 кв. 2007 г.).

Вартан Хачатуров
Санкт-Петербург, ALT Linux

Проект: Sisyphus
`http://sisyphus.ru`

Опыт портирования репозитария Sisyphus на архитектуру Power™: проблемы и решения

Доклад посвящён описанию методики портирования репозитория Sisyphus на новую архитектуру на примере платформы Power™.

Основные части доклада следующие:

Bootstrap Этот раздел посвящён описанию самой нетривиальной части процедуры переноса: созданию первой сборочной среды. Здесь будут обсуждаться имеющиеся, с точки зрения автора, проблемы репозитария и пути их решения.

Инфраструктура В этом разделе будут обсуждаться вопросы создания сборочной инфраструктуры, регулярной пересборки Sisyphus, а также связанные с этим проблемы.

Демонстрация Демонстрация работы дистрибутива, основанного на Sisyphus, на процессоре Cell Broadband Engine.

Михаил Гусаров, Евгений Хоружий
Новосибирск, Минск

Проект: OpenInkpot
http://openinkpot.org/

OpenInkpot: освобождая электронные книги

OpenInkpot — проект разработки свободного дистрибутива для устройств для чтения книг на базе электронных чернил. В докладе рассказывается о предпосылках к созданию проекта, о достигнутых результатах и об устройстве дистрибутива.

В 2007 году на массовом рынке появились специализированные устройства для чтения с экранами на основе электронной бумаги (e-ink).

Электронная бумага — тип экрана, принципиально отличающийся от LCD/OLED-экранов, массово применяющихся в небольших устройствах. Отличия состоят в следующем:

- Небольшое количество оттенков серого цвета (конфигурация, давно не интересная разработчикам графических библиотек);
- Очень большое время обновления экрана (порядка 0,5 секунды);
- Мультистабильность экрана (возможность удержания изображения в выключенном состоянии).

Особенности e-ink-экранов приводят к весьма специфичным требованиям к управлению питанием. В совокупности с небольшим количеством принципиально различных устройств на рынке, а также неудовлетворительным качеством прошивок, поставляемых разработчиками устройств, это привело к созданию отдельного проекта, фокусирующегося на разработке дистрибутива Linux для использования в e-ink устройствах для чтения книг.

Оборудование и ядро Linux

Основная разработка ведётся для HanLin/lBook eReader V3. Устройство имеет микропроцессор S3C2410 (SoC, система-на-кристалле) на ядре ARM. К нему подключаются контроллер экрана, память, слот SD/MMC, USB-разъём, контроллер батареи, MP3-декодер, светодиод и кнопки.

Такой же процессор используется в проекте OpenMoko, поэтому часть драйверов (в частности, SD/MMC) были взяты из него. Написаны драйвера клавиатуры, контроллера дисплея, контроллера зарядки батареи.

Особенностью контроллера экрана является специфический параллельный интерфейс с достаточно низким быстродействием.

В драйвере контроллера дисплея используется механизм отложенного ввода-вывода (deffered IO), что позволяет объединять несколько последовательных изменений изображения за короткий промежуток времени в одну транзакцию. Данный механизм появился в ядре Linux 2.6.22.

Ядро поддерживает корректное засыпание-просыпание процессора, однако это занимает некоторое время и пока не удалось избавиться от проблем с работой драйвера e-ink контроллера после просыпания.

Дистрибутив

Разработка основана на embedded-дистрибутиве SLIND[1]. Изначально работа велась на базе OpenEmbedded[2], но неудовлетворённость системой сборки — долгие циклы полной пересборки и плохой контроль за сборочной средой — заставили взять другой, хоть и менее известный дистрибутив.

Технологическая база полностью унаследована от SLIND: используется Debian Etch в качестве базовой сборочной среды и кросс-компиляция с применением некоторых трюков для выполнения bootstrap на хост-системе. Исходный код пакетов хранится в git, а собранные пакеты — в APT-репозиториях.

Целевая система представляет собой deb-based дистрибутив, усечённый для embedded-целей, но сохраняющий высокую степень совместимости с Debian по формату source- и binary-пакетов и по инструментарию разработки.

На текущий момент работает kdrive (X server для встраиваемых устройств), «читалка» книг FBReader. Идёт работа по портированию библиотек EFL на libxcb.

В рамках Google Summer of Code идёт работа над портированием OpenInkpot на устройства Sony PRS-505 и Bookeen Cybook Gen3.

[1] `http://slind.org/`
[2] `http://openembedded.org/`

Денис Овсиенко
Москва, независимый разработчик

Проект: RackTables
http://racktables.org/

RackTables — свободное решение для управления датацентром

RackTables — свободное ПО, являющееся средством координации деятельности в центрах обработки данных (датацентрах) малого и среднего размера. Проект представляет собой типичный образец современного web-приложения и состоит из PHP-интерфейса к базе данных MySQL. Доклад описывает как принципы решения, так и вопросы, чаще всего возникающие в результате его внедрения.

Предыстория и статус

Этот продукт первоначально был создан в некоторой компании для решения её внутренних потребностей по управлению датацентром и оказался удачным решением. В феврале 2007 года в интересах дальнейшего развития он на условиях анонимности и независимости был опубликован под свободной лицензией. В таком качестве он развивается до сих пор, выпуская в среднем около одного релиза раз в полтора месяца и сформировав пользовательскую базу по всему миру. Каждый месяц несколько сот пользователей скачивают исходный код с сайта проекта, некоторые из них вносят свой вклад в его дальнейшее развитие.

Общий подход

Фокус возможностей RackTables выполнен на автоматизацию деятельности так называемых application service providers и (в меньшей степени) colocation providers. Оба вида деятельности предполагают концентрацию технических мощностей в специально сконструированных центрах обработки данных. В практике функционирования датацентров возможно выделить устоявшиеся операции, выполняемые относительно независимо друг от друга:

- постановка на инвентарный учёт;
- управление адресной ёмкостью (3-й уровень модели OSI);
- управление портовой ёмкостью;
- генерация серверов по шаблонам;
- выделение монтажного места, монтаж и перемещение оборудования;
- установка и изменение кроссировок (1-й уровень модели OSI).

В зависимости от организации работ эти операции могут выполняться как одним человеком, так и независимыми группами людей. Эти действия полностью покрыты набором базовых функций RackTables, для чего в рабочем пространстве представлены: монтажная база в виде стоек, собственно оборудование, ёмкостной ресурс в виде IP-сетей и среда передачи данных в виде данных о кроссировках. Каждое действие выполняется относительно некоторого «объекта», являющегося единицей учёта. Все объекты типизированы и, кроме минимального набора основных атрибутов, обладают зависимым от типа переменным набором дополнительных.

Типичными структурами, построенными на основе этих элементов, являются сети из серверов и маршрутизаторов, объединённые с помощью коммутаторов. В накопленной базе данных работает поиск разнородной информации по произвольному фрагменту текста.

Кроме базовых функций, RackTables предлагает специально разработанную тэговую классификацию, которая основывается на формируемом пользователем дереве тэгов. Это дерево является единым для всех фигурирующих в RackTables сущностей: пользователей, серверных стоек, объектов, IP-сетей и других элементов. Эффективность классификации, в первую очередь, определяется структурой дерева, причём отдельно взятая сущность может иметь назначенным любое количество тэгов, входящих в произвольное число иерархий. Например, функциональная роль сервера и его принадлежность могут быть никак не связаны между собой. С помощью тэгов можно быстро составить представление о какой-либо незнакомой сущности и отыскать другие такие же.

Опыт внедрения

При использовании RackTables в рабочей практике начинают действовать различные факторы.

1. Дополнительные обязанности. Для эффективной работы все внесённые в реальную картину мира изменения должны фиксироваться в документальной её копии. Облегчает задачу то, что даже сложные процессы состоят из последовательности элементарных действий, каждое из которых может быть легко сохранено с помощью подручной web-консоли. То, с какой задержкой это может происходить, является предметом соглашения сторон и характеризует степень общей достоверности базы данных.

2. Дружественное инженеру окружение. Чтение подготовленной структурированной информации всегда быстрее, чем проведение экспресс-расследования, особенно при отсутствии необходимого (физического или удалённого) доступа. Это позволяет новичкам быстрее осваивать уже имеющиеся конструкции, а опытным сотрудникам — недавно появившиеся.

3. Общий знаменатель. После некоторого периода привыкания оказывается, что возник новый способ ссылаться на предметы рабочих процессов. Для того чтобы в тексте однозначно указать на какую-либо сущность или целую их группу, достаточно привести URL, по которому они отображаются. Это оказывается удобным, особенно когда одна из сторон имеет доступ только к документации. С другой стороны, необходимо понимать, что информация должна быть достаточно полной, чтобы все стороны могли её интерпретировать однозначно. Это опять-таки скорее предмет соглашения, чем ограничение системы.

4. Выявление бизнес-логики. При построении и, как правило, дальнейших коррекциях структуры дерева тэгов становится ясно, какими категориями вообще принято оперировать в организации и насколько большими «популяциями» они представлены. Это помогает формализовать имеющиеся неформальные процессы (или скорректировать формальные).

В целом можно сделать заключение, что, опираясь на заранее оговорённые процедуры и сферы ответственности, с помощью RackTables можно снизить затраты времени на управление датацентром в несколько раз.

Михаил Пожидаев
Томск, Томский государственный университет

Возможные варианты правил конфигурирования речевого вывода в среде ALT Linux

В докладе рассматриваются вопросы централизованного хранения конфигурационной информации речевого вывода в среде дистрибутивов ALT Linux. Большое внимание уделяется выработке гибких правил организации единых механизмов обработки речи и реализации речевого интерфейса программы установки операционной системы.

Последнее время в процессе разработки речевых систем наблюдается рост активности. Появилось движение в сторону консолидации механизмов передачи речевой информации. Текущее направление разработки речевых систем позволяет говорить, что в GNU/Linux в будущем будет возможна работа без зрительного контроля в графической оконной среде, в то время как раньше речь шла только о консольных приложениях и о emacspeak. Такие крупные проекты, как OpenOffice.org, FireFox, Java Swing, GTK+, QT, заявили о реализации механизмов передачи оповещений об изменении в пользовательском интерфейсе для вывода их в речевом виде. Ведётся также разработка AT-SPI — центрального компонента для сбора такой информации, являющейся частью проекта Gnome.

Возникает серьёзный вопрос о реализации в дистрибутивах ALT Linux инструментария для централизованного конфигурирования речевых систем. Сложность в том, что важно сохранить работу инструментов, используемых в настоящий момент. К примеру, emacspeak остаётся самой развитой средой и имеет множество достоинств. Наиболее вероятным развитием событий будет существование отдельных механизмов обработки речевой информации для KDE, для Gnome и для консольных приложений. Трудно сказать, удастся ли создать компонент для «Альтератора», в котором у пользователя будет возможность менять настройки одновременно для всех систем. Вероятно, что это может оказаться неудобным, и тогда лучше предоставить настройку вывода в KDE и Gnome средствам, поставляемым вместе с этими системами, а через «Альтератор» выполнять настройку только консольных приложений.

Главный параметр, который пользователь должен уметь свободно менять, — речевой синтезатор, используемый для обработки англий-

ской и русской речи. Желательно выработать соглашения, которые позволяли бы качественно поддерживать список установленных синтезаторов. Это дало бы возможность пользователю видеть каждый новый синтезатор в списке поддерживаемых сразу после его установки. В качестве такого соглашения предлагается ввести специальную директорию, например `/etc/tts.d`, для добавления файлов, поставляемых в пакете синтезатора и содержащих необходимую информацию для конфигурирования. Содержимое такой директории возможно обрабатывать специальным модулем «Альтератора». Формат подобных файлов и множество содержащейся в них информации ещё предстоит выработать.

В настоящий момент, благодаря наличию в QT механизма генерации оповещений об изменении состояния пользовательского интерфейса, появляется интересная возможность. Разработчики QT предусмотрели наличие уровня абстракции, необходимого для перенаправления подобных оповещений не в AT" SPI, а в любой другой компонент, в том числе и созданный сторонним разработчиком. Как известно, на QT реализован оконный интерфейс платформы «Альтератор». Можно попытаться реализовать специальный мост, пересылающий оповещения в некоторый лёгкий обработчик, связанный с речевыми синтезаторами и способный работать в среде программы установки. В случае успеха возможно получить операционную систему с полностью озвученной процедурой установки, что является очень важным жизненным требованием.

Михаил Якшин
Москва, ALT Linux Team

Проект: Inquisitor
`http://www.inquisitor.ru`

Inquisitor — свободная платформа тестирования и мониторинга аппаратного обеспечения

В 2005 году, на Второй конференции разработчиков свободных программ на Протве, была впервые представлена система Inquisitor[1] в своей первой инкарнации — специализированной системы для авто-

матизированного тестирования аппаратного обеспечения компьютеров.

Разработка системы была начата в 2004 году ALT Linux по заказу компании MaxSelect. Первоначально ставилась достаточно скромная задача: нужно было просто подобрать набор тестов для нагрузочного тестирования ноутбуков и сделать некий механизм их автоматического запуска.

Время шло, проект развивался, и в августе 2007 года были заявлены цели, которые должны были быть достигнуты разработкой новой, третьей версии Inquisitor:

- Inquisitor — это *платформа* для реализации систем тестирования, сертификации и мониторинга компьютерного оборудования. Ключевое слово «платформа» в данном случае означает чёткое разделение свободного *ядра системы*, которое содержит в себе основной функционал, и *конфигурации системы*, настраиваемой под конкретное применение.

- Платформа должна распространяться *свободно на условиях GPL*.

- Система строится по *модульному принципу*. Можно легко исключать или добавлять новые модули тестирования, анализа, диагностики и т. д. Точно так же модуляризируются системные компоненты: можно использовать разные способы загрузки, разные ядра, разные репозитории исходных пакетов при сборке и т. п. Модулям предоставляется удобный API, что способствует лёгкому расширению системы.

- *Масштабируемость* — платформа Inquisitor может работать с любыми типами компьютеров и способна протестировать как один локальный компьютер, так и многие тысячи компьютеров на производстве. Для этого существуют несколько вариантов сборки:

  Standalone — вариант, устанавливаемый из пакета в уже инсталлированную ОС Linux, можно использовать для демонстрации, бенчмаркинга и знакомства с устройством платформы.

  Live — вариант, распространяемый на загрузочном Live CD или DVD, для «домашнего» использования — удобен для того, чтобы загрузиться и протестировать один компьютер.

Серверный — наиболее сложный вариант, когда есть отдельный сервер, контролирующий тестирование, с которого загружаются по сети (PXE) тестируемые машины-клиенты.

- *Гибкость*: система позволяет изменять большинство параметров работы модулей без изменения непосредственно исходных кодов. Это облегчает изучение системы, использование и обслуживание при объёмных внедрениях.
- Серверная версия ведёт *базу данных*, которая хранит все сведения обо всех компьютерах, когда-либо проходивших через Inquisitor. Если компьютер уже тестировался и теперь тестируется ещё раз после замены части оборудования, «умный» планировщик не будет выполнять все тесты заново, а сделает только нужные для проверки работоспособности вновь установленного оборудования.

Существует несколько основных классов модулей:

1. Модули анализа оборудования («detect») — предоставляют информацию о конфигурации компьютера: производители, модели, серийные номера обнаруженного оборудования.
2. Модули тестирования оборудования («test») — представляют собой этапы тестирования аппаратных компонентов. Тесты могут выдавать достаточно разнообразный поток информации: от ответов вида «прошёл — не прошёл» до развёрнутой диагностики с количественными характеристиками.
3. Модули мониторинга («monitor») — периодически записывают определённые показатели для дальнейшего построения графиков и анализа. Результаты мониторинга могут служить поводом для принятия решения о непрохождении теста.
4. Модули самоидентификации («self-id») — позволяют привязаться к какому-то уникальному идентификатору компьютера для сохранения данных о тестировании между перезагрузками.
5. Модули загрузки («boot») — позволяют автоматически загружать компьютер с различных носителей (PXE, ISO CD/DVD, HDD и т. п.).
6. Модули сборки («build») — позволяют собирать Inquisitor с использованием разных репозиториев (rpm/dpkg).

Система Inquisitor существует и развивается уже достаточно продолжительное время. Начатая как сравнительно небольшая система тестирования оборудования, Inquisitor перерос в систему контроля качества, а теперь, благодаря активной поддержке разработки компанией ETegro Technologies, — уже в полноценную платформу для разработки всевозможных решений на его базе.

Недавно состоялся выпуск финальной версии платформы Inquisitor 3.0, что даёт возможность open source community подключаться к развитию проекта «в ширину» путём добавления дополнительных тестов, детектов, мониторингов и других модулей. Далее ветка 3.x будет развиваться как стабильная, с добавлением возможностей, не требующих архитектурных изменений.

Литература

[1] *Якшин М.* Система автоматизированного тестирования и контроля качества оборудования «Inquisitor» // Вторая международная конференции разработчиков свободных программ на Протве. — М.: 2005.

Александр Боковой
Москва, Samba Team

Проект: Samba
`http://www.samba.org`, `http://ctdb.samba.org`,
`http://www.openchange.org`

Новая Samba

С 1992 года, когда Andrew Tridgell написал свою реализацию протокола SMB для Unix-подобных систем, чтобы получить доступ из клиента PathWorks в MS DOS к данным на сервере Sun, прошло уже более 15 лет. В 1996 году Microsoft потребовалась «прорывная технология», чтобы стать Internet-компанией, и такой технологией стал стек протоколов SMB, переименованный в CIFS, Common Internet File System. Расширенный и дополненный, десять лет спустя CIFS уже неотделим в сознании потребителей от Microsoft. Проект Samba традиционно рассматривался как попытка догнать Microsoft,

но неожиданно в 2007 на неприметных устройствах хранения фотографий, музыки и видео их разработчики из азиатских стран пишут «совместимо с протоколом Samba», а не CIFS. Тихая революция?

2007 год вообще был богат на события. Команда разработчиков проекта Samba попадает на первые страницы главных деловых газет, таких как Financial Times, за победу над Microsoft в рамках антимонопольного процесса в Европейском Союзе. Затем Microsoft публикует более 40,000 страниц документации под лицензией, дающей возможность использовать её при разработке Samba, а также впервые с 1998 года принимает участие в конференциях, посвящённых разработке Samba. В 2008 году эта документация становится доступной всем желающим. Из самой закрытой компании ИТ-мира Microsoft становится самой открытой?

Но это только вершина айсберга. В 2007 году проект выпускает версию Samba, рассчитанную на высокопроизводительные системы хранения. Кластерная Samba в составе Scale-out File Services компании IBM позволяет горизонтальное масштабирование систем хранения с доступом на основе CIFS и демонстрирует производительность минимум в четыре раза быстрее любых коммерческих аналогов, включая Microsoft, фактического «законодателя мод» в области CIFS.

Небольшая группа внутри Samba Team за пять лет сформировала основу для замены Microsoft Active Directory. Samba 4.0, ещё не выпущенная и вряд ли готовая увидеть свет до конца 2008 года, тем не менее уже стала площадкой для атаки на другие бастионы «технологий Microsoft» — Microsoft Exchange. В проекте OpenChange уже реализована практически вся функциональность клиентской части Microsoft Exchange, что позволяет, например, GNOME Evolution работать в сетях Exchange наравне с клиентами от Microsoft, а MocaBox наступать на позиции Microsoft Unified Communications.

В то же время пользователи заблудились в трёх соснах. Samba 3.0, 3.2 и 4.0 развиваются одновременно и с разным функционалом, выглядят вкусными «стогами сена», дразнят и манят уже не первый год. Однако до сих пор проект не говорит пользователям, для чего планируется каждая версия и каков режим её выпуска. В докладе автор попытается объяснить позиционирование, новую модель разработки и выпуска разных версий Samba, их новинки и возможности. Также отдельное внимание будет уделено взаимодействию с разработчиками и компаниями, включая создателей Windows Vista и протокола SMB2.

Вячеслав Пупышев
Ижевск, Удмуртский государственный университет

Использование FreePascal при обучении школьников основам программирования

Рассмотрим ситуацию, когда методика обучения школьников основам программирования разработана, и осталось только решить, в какой среде программирования это делать. Для этого нужно определиться с основными критериями. Обычно важнейшими критериями являются следующие:

- Возможность следовать методике без «забегания вперёд». Например, в изучая язык C, приходится многие моменты рассказывать по принципу «пока пишите так, потом расскажу, почему».
- Бесплатность системы программирования. Коммерческие системы слишком дороги, и необходимость их покупки может оттолкнуть потенциального ученика.
- Простота установки. Желательно иметь систему программирования всем ученикам дома. Опыта установки систем чаще всего нет.
- Возможность полной русификации. Без возможности русифицировать диагностику ошибок учить основам программирования очень сложно, ведь приходится всем ученикам переводить, что за ошибка у них «вылезла». При обучении целого класса это очень отвлекает.
- Возможность получать работающую программу вне системы программирования. Получение исполняемой программы очень хорошо мотивирует учеников.
- Возможность написания в этой же системе не только учебных, но сложных, возможно даже коммерческих, продуктов. После изучения основ программирования возможны разные пути. Кто-то остановится на том, что разовьёт своё алгоритмическое мышление и программировать больше не будет. А кому-то это дело придётся по душе, и появится большое желание научиться программировать по-настоящему. Будет очень хорошо, если эта же система позволит писать и серьёзные программы.

Разработанная мною методика развития алгоритмического мышления рассчитана на школьников старше четвёртого класса. Курс по данной методике изучается один год. Основной частью курса является активное изучение языка программирования Pascal и написание работающих программ на нём. Тут и возникли все описанные выше критерии. Оказалось, что при должной настройке можно добиться от FreePascal удовлетворения всех этих критериев. Здесь словом FreePacal буду обозначать и компилятор и интегрированную среду разработки. Если просмотреть критерии, то получится следующее:

- Возможность следовать методике без «забегания вперёд». Это позволяет сам язык Pascal.
- Бесплатность системы программирования. FreePascal поэтому и «Free».
- Простота установки. После настройки всю систему FreePascal можно просто скопировать на другой компьютер. Если скопировать в такой же каталог, то работать будет сразу.
- Возможность полной русификации. Русифицировать диагностику ошибок можно, указав в настройках соответствующий файл. А русификация системы помощи требует дополнительных ухищрений, но возможна.
- Возможность получать работающую программу вне системы программирования. Тут проблемы не было никогда.
- Возможность написания в этой же системе не только учебных, но сложных, возможно даже коммерческих, продуктов. FreePascal — мощная система, поддерживающая даже диалекты Delphi.

Конечно есть и претензии к FreePascal. Отладчик сыроват, система помощи уступает даже TurboPascal, диагностика ошибок тоже уступает TurboPascal. Трудности в русификацией под Unix.

Lazarus

Более современно выглядит интегрированная оболочка Lazarus. Исследование, подобное оболочке FreePascal, есть и для неё. Важным преимуществом Lazarus является современный оконный интерфейс. Поскольку дети уже привыкли к внешнему виду, продиктованному операционной системой Windows.

Алексей Дьяченко, Евгений Цыганцов, Виктор Мяэотс
Москва, ГОУ Центр Образования «Технологии обучения»

Проект: Moodle
`http://www.moodle.org`

Среда дистанционного обучения Moodle

Moodle является одной из самых популярных сред дистанционного обучения в мире. Количество зарегистрированных инсталляций приближается к 50 тысячам. Система используется в десятках тысяч учебных заведений в 199 странах мира и переведена на 75 языков. Moodle давно и успешно используется в России и странах СНГ как в высшем, так и в среднем образовании, а так же в качестве корпоративной системы повышения квалификации.

Среда дистанционного обучения Moodle — это один из флагманов в данной отрасли, разрабатываемый по принципам Open Source под лицензией GNU GPL. Moodle реализует богатый функционал, сравнимый с ведущими коммерческими системами, а в чём-то даже их опережающий. Это возможно благодаря широкому сообществу пользователей и разработчиков со всего мира, поддерживающему данный продукт. Многие разработчики Moodle сами являются преподавателями или преподавали в прошлом, благодаря чему данный продукт отличают удобство и простота использования, сочетающиеся с широким спектром поддерживаемых методических приёмов и форм учебного взаимодействия.

Moodle начал своё развитие в качестве Open Source проекта в ноябре 2001 года, а первый релиз версии 1.0 вышел 20 августа 2002 года. Инициатором и ведущим разработчиком проекта является Martin Dougiamas из Австралии. Практически с самого начала Moodle вызвал широкий интерес среди учебных заведений во всем мире, в том числе и в России, где он с 2003 года используется в проекте Департамента Образования города Москвы по дистанционному обучению детей-инвалидов (i-Школа).

К настоящему моменту Moodle насчитывает почти 50 тысяч зарегистрированных инсталляций по всему миру, в которых создано больше 2 миллионов курсов и обучается больше 20 миллионов пользователей. Интерфейс Moodle локализован на 75 языков, а сообщество

пользователей, зарегистрировавшихся на сайте проекта, превышает 400 тысяч пользователей из 193 стран мира. Moodle применяется как частными репетиторами, так и огромными университетами, обучающими по 200 тысяч студентов. В России зарегистрировано почти 300 инсталляций Moodle.

Помимо базового функционала, уже ставшего стандартным для любой среды дистанционного обучения и включающего возможность использования в курсе учебных материалов как в веб-форматах, так и в виде произвольных файлов, форумов, чатов, внутренней почты, автоматических тестов и пакетов SCORM, Moodle поддерживает множество собственных форм взаимодействия, выводящих дистанционный учебный процесс совершенно на другой уровень эффективности. В их числе возможность создавать произвольные, в том числе и нечисловые, шкалы оценивания, выставления оценок преподавателями и сокурсниками за активность в форуме, сбора и рецензирования работ в текстовом формате, в виде произвольных файлов и в виде произвольной активности, не отражающейся в системе, глоссариев, баз данных с настраиваемым форматом, wiki, тестов Hot Potatoes (расширенный формат тестов, например, в виде кроссвордов), опросов, учебных карточек, предъявляемых последовательно, вразнобой или в зависимости от корректности решения теста в предыдущей карточке (модуль «лекция») и семинаров (ученикам предлагается сначала сдать работы, а потом разобрать работы своих коллег по заданным критериям). Вся активность пользователей в системе фиксируется и отображается в виде индивидуальных отчётов о деятельности (портфолио), в которых на одной странице можно видеть все выполненные задания, сданные работы и полученные рецензии и оценки по каждому курсу.

Благодаря этим возможностям, Moodle можно использовать как для стандартного дистанционного обучения, так и для поддержки очного обучения или проведения тестирования.

Весь отображаемый пользователю текст в Moodle может быть обработан с помощью указанного набора фильтров. В базовой версии поставляется широкий набор фильтров, включая автосвязывание с записями в глоссарии или базе данных, замену ссылок на мультимедийные файлы проигрыванием их во встроенном плеере, графическое отображение формул в алгебраическом или TeX форматах, многоязыковую поддержку внутри одного текста, а также актуальный на некоторых сайтах фильтр ненормативной лексики.

Управление доступом в Moodle, начиная с версии 1.7, осуществляется на основе настраиваемой системы ролей и полномочий, которые можно назначать и переопределять в различных контекстах: всей системы, категории курсов, одного курса, элемента курса.

Большое внимание уделено интеграции Moodle в информационную инфраструктуру учебного заведения, для чего предусмотрено множество вариантов аутентификации пользователей и подписки на курсы на основе информации из внешних источников.

Интерфейс Moodle может быть настроен путём выбора подходящей темы оформления и настройки состава блоков, отображаемых в правой и левой колонках. Шаблон интерфейса и набор блоков в каждом курсе могут быть индивидуальными.

Гибкостью и широтой функционала Moodle обязана своей модульной архитектуре, на которую указывает её название. Помимо модулей локализации и шаблонов оформления, Moodle поддерживает модули элементов курсов, типов ресурсов, видов заданий в тестах, формата экспорта и импорта тестов, типов полей в модуле «база данных», форматов заданий, блоков, фильтров, различных видов отчётов, форматов курсов, аутентификации пользователей и подписки на курсы. Помимо модулей, входящих в базовую поставку, существует широкий набор дополнительных модулей, которые можно найти на сайте проекта. Благодаря поддержке всех этих модулей, функционал Moodle можно наращивать и изменять без исправления кода базовой системы, которое могло бы привести к проблемам с переходом на следующую версию. Следуя этому принципу, можно легко избежать ситуации, когда переход на следующую версию невозможен из-за огромного количества модификаций кода, а поддержка собственной ветки требует всё большего количества ресурсов.

Удобен Moodle и в администрировании. Сейчас уже многие веб-приложения на php поддерживают автоматическое обновление базы данных при смене версии, в Moodle же возможность автоматически обновиться с любой из предыдущих версий на любую последующую существовала с самого начала, когда этой возможности не было практически нигде. Почти всеми настройками можно управлять через панель администрирования Moodle, и только в самых экзотических случаях требуется ручная правка конфигурационного файла.

Немаловажную роль в популярности Moodle играет дружественное и открытое для новых пользователей сообщество. Базовым сайтом проекта является сайт `http://moodle.org`, построенный на осно-

ве Moodle. Помимо стандартных для сайтов Open Source проектов разделов, на сайте существуют разделы сообществ разработчиков и пользователей (в виде курсов Moodle). Главным из них является англоязычный раздел «Using Moodle». Для неанглоязычных пользователей заведены разделы в категории «Community Discussion», в которой есть раздел и для русскоязычных пользователей «Russian Moodle». Всё основное общение разработчиков, пользователей и локализаторов происходит в этих сообществах, что является более дружественным по отношению к новым пользователям способом общения, чем традиционные списки рассылки. Сторонние разработчики модулей, шаблонов оформления и учебных ресурсов могут опубликовать их в соответствующих разделах основного сайта. Для оповещения разработчиков об ошибках, запроса новых возможностей или отправки патчей традиционно используется Tracker; в настоящее время это JURA. Вся документация, дополняющая встроенную справку, разрабатывается и хранится в wiki на базе движка MediaWiki, поэтому участвовать в составлении и переводе документации могут все желающие.

Координацию проекта осуществляет австралийская компания Moodle PTY Ltd., основателем которой является лидер проекта Martin Dougiamas. В дополнение к поддержке сообщества, коммерческую поддержку Moodle осуществляют многочисленные партнёры Moodle во всем мире, работу которых контролирует Moodle PTY Ltd. Благодаря своему статусу, партнёры получают право приоритетного рассмотрения своих сообщений об ошибках в Moodle и возможность непосредственного консультирования с ведущими разработчикам ядра Moodle, в обмен выплачивая Moodle PTY Ltd. отчисления.

Работа над каждой локализацией выделена в самостоятельный проект, некоторые из участников которого имеют доступ к файлам локализации в репозитории исходного кода. После загрузки изменений локализации в репозиторий в течении суток они становятся доступны для автоматического обновления через панель администратора в каждой инсталляции Moodle. Остальные участники могут присылать свои дополнения к локализации, и после проверки его соответствия принятой терминологии они также включаются в перевод. Официальный список русскоязычной терминологии Moodle опубликован в разделе «Russian Moodle», любые изменения и дополнения к нему принимаются голосованием участников сообщества. Такая схема позволяет сохранить целостность перевода и избежать расщепления, чреватого дублированием работы.

В дополнение к русскоязычному разделу на основном сайте проекта, существует также сообщество русскоязычных преподавателей, использующих Moodle, на сайте `http://www.infoco.ru`, посвящённое, в основном, методическим и организационным вопросам дистанционного образования. Ежегодно, в конце марта, в городе Железноводске (Минеральные Воды) проводится конференция русскоязычных пользователей Moodle в рамках международной конференции Информационные Технологии в Науке и Образовании. На конференцию собираются представители ВУЗов и других образовательных учреждений.

Таким образом, Moodle можно назвать одной из самых популярных и динамично развивающихся сред дистанционного обучения как среди коммерческих, так и некоммерческих продуктов. Moodle поддерживает весь традиционный функционал сред дистанционного обучения, множество дополнительных функций и может быть гибко надстроен и адаптирован к нуждам конкретного образовательного учреждения. А благодаря дружественному сообществу и широкому кругу партнёров, пользователи Moodle могут получить любую необходимую поддержку.

Евгений Цыганцов, Алексей Дьяченко, Виктор Мяэотс
Москва, ГОУ Центр образования «Технологии обучения»

Проект: «Электронный деканат»
`http://sourceforge.net/projects/freedeansoffice/`

Модуль «Электронный деканат» для СДО Moodle

Многие отечественные ВУЗы при внедрении информационных технологий в управление учебным процессом сталкиваются с отсутствием подходящего открытого ПО, а также высокой стоимостью имеющихся на рынке решений автоматизации для ВУЗов Первоочередной задачей Open Source проекта «Электронный деканат» является адаптация СДО Moodle к особенностям организации учебного процесса в отечественных учебных заведениях, а в перспективе — разработка гибкой системы автоматизации бизнес-процессов в ВУЗах. Система разрабатывается как модуль СДО Moodle и сама имеет развитую модульную архитектуру, позволяющую адаптировать её под потребности каждой организации без модификации кода базовой системы.

Выбирая средства для реализации дистанционного обучения, многие учебные заведения обращают свой взгляд на СДО Moodle. И это не случайно. Moodle очень удобна для решения этой задачи. Среди её достоинств кроссплатформенность, русифицированный дружественный интерфейс, обширная справочная система, широкий набор методов подачи материала. Одним из основных достоинств является универсальность с точки зрения организации учебного процесса — СДО Moodle реализует среду обучения, в которой студенты могут взаимодействовать с учебными материалами, с преподавателями и друг с другом. Она стала основой популярности Moodle, позволяя применять эту систему для организации самых разных видов обучения в организациях разных типов.

Однако в Moodle нет групп, как их понимают в отечественных учебных заведениях, учебного плана, расписания, ведомостей и других неотъемлемых атрибутов реального учебного процесса практически любого нашего образовательного заведения.

Поэтому организации, начинающие внедрение Moodle, сталкиваются с проблемой организации учебного процесса, обеспечения отчётности, а также контроля за учебным процессом.

Таким образом, существует насущная потребность в адаптации СДО Moodle к традициям отечественной системы образования. Данную задачу призвана решить система «Электронный деканат» для СДО Moodle (далее ЭД), разрабатываемая сообществом российских программистов как открытый проект под лицензией GNU GPL[1].

Исходной версией для разработки послужил программный продукт, разработанный для внутренних нужд и реализующий следующие возможности:

1. Организовать учебный процесс по учебным периодам (семестрам и т. п.). Администратор создаёт учебные периоды, указывает даты их начала и окончания.

2. Организовать учебный процесс для групп студентов. Реализован механизм создания учебных групп, в которые зачисляются студенты. Предусмотрена возможность создания и регистрации групп студентов путём загрузки из текстового файла.

[1]Подробную информацию можно найти по адресам в интернете `http://sourceforge.net/projects/freedeansoffice/` и `http://www.infoco.ru/course/view.php?id=19`.

3. Создавать учебный план для групп и студентов на учебный период. Из текстового файла загружаются списки дисциплин, изучаемых группами за один учебный период. Автоматически происходит зачисление студентов групп на соответствующие учебные курсы Moodle.
4. Создавать и редактировать расписание для групп и отдельных учеников. Контролировать проведение занятий преподавателями в режиме реального времени.
5. Автоматизировать процесс регистрации студентов в Moodle, зачисления студентов в группы, подписку на учебные курсы.
6. Просматривать все оценки ученика или группы по всем предметам.
7. Просматривать все итоговые оценки ученика или группы по всем предметам за учебный период.
8. Просматривать ведомость группы.
9. Распечатывать или сохранять в excel различные ведомости.

Однако при переходе к разработке в режиме открытого проекта выявилась проблема монолитности существующей версии, сильно затрудняющая работу распределённой команды разработчиков. Поэтому было принято решение вначале реализовать новое ядро системы, а затем перенести на неё разработанный функционал уже в виде модулей.

Архитектурно ЭД для Moodle — это модуль типа блок. Он сам также имеет модульную структуру, поддерживая различные типы плагинов, в которые вынесена вся бизнес-логика. Стандартизация плагинов позволяет легко дополнять ЭД новыми функциями, облегчая совместимость нового плагина с ЭД и другими плагинами. А также использовать уже написанные плагины для реализации новых функций, а не писать всё заново. Таким образом, в зависимости от набора установленных плагинов, ЭД может быть приспособлен для использования в самых разных организациях.

Поддерживаются следующие типы плагинов:

1. Плагин интерфейса — обеспечивает взаимодействие с пользователем системы. Реализует интерфейс пользователя.
2. Плагин «справочник». Обеспечивает работу с базой данных. Упрощает использование стандартных операций и служит слоем для инкапсуляции SQL-запросов.

3. Плагин синхронизации — обеспечивает двунаправленную синхронизацию данных ЭД и внешних систем. В том числе, через плагины данного типа происходит обмен данными с Moodle. Это позволяет снизить зависимость остального кода от API внешних систем и структур данных во внешних базах данных.
4. Плагин бизнес-процессов. Позволяет задать для каждого типа объектов возможные состояния, переходы между ними и сопутствующие переходам действия. Например, перевод студента из состояния «обучается» в состояние «в академическом отпуске», а далее в «отчислен» или вновь «обучается» и т. п.
5. Плагин библиотеки. Плагин вспомогательных функций и классов, используемых плагинами, которые названы выше. Например, плагин навигации содержит функции, реализующие иерархическую панель перемещения по ЭД, и используется в плагинах интерфейса.

Интерфейс всех плагинов поддерживает автоматические установку и удаление, выполняемые из панели администрирования ЭД.

Перспективы развития:

1. Перенос на новую архитектуру ЭД возможностей существующей версии преобразованием её кода в плагины ЭД.
2. Дополнение ЭД плагинами для создания структуры, полностью соответствующей сложившейся в учебных заведениях.
3. Реализация всех действий участников и организаторов учебного процесса через ЭД.
4. Автоматизация организации и управления учебным процессом.
5. Приведение используемой документации в соответствие с принятыми стандартами делопроизводства в образовательных учреждениях.
6. Автоматизация документооборота.

Решить эти задачи невозможно без тесного и неформального диалога между программистами и представителями ВУЗов — педагогами, администрацией, бухгалтерией, представителями учебной части и др. Большое внимание уделяется мнению будущих пользователей о том, что и как они хотят делать в ЭД, их пожеланиям к интерфейсу ЭД. Для этого на сайте `http://infoco.ru` открыт соответствующий раздел, в котором идёт обсуждение структуры и возможностей

ЭД, модели бизнес-процессов, протекающих в учебных заведениях, и её реализация в ЭД. Там же открыто несколько wiki, в которых редактируются форматы типовых бланков отчётной документации для средней и высшей школы.

Исходный код ЭД доступен в разделе проекта на сайте `http://sourceforge.net`. Там же находятся формы для оповещения разработчиков об ошибках, отправки пожеланий или фрагментов кода.

Таким образом, модуль «Электронный деканат» является продуктом не только с открытым исходным кодом, но и разрабатываемым в виде открытого проекта. Связка СДО Moodle + «Электронный деканат» полезна организациям, которые внедрили или только собираются внедрять дистанционное обучение, прежде всего ВУЗам но не только им: модульная архитектура и открытость исходных кодов даёт возможность адаптации под нужды любых организаций. ЭД даёт возможность автоматизации управления учебным процессом и переноса привычной среды очного обучения на дистанционные курсы. Кроме того, ЭД разрабатывается российским сообществом программистов. Это даёт лёгкую и быструю обратную связь и возможность принять участие в разработке нужных вам возможностей. Тем самым сэкономив время, силы и деньги.

Владимир Рубанов, Константин Власов, Андрей Смачев
Москва, ИСП РАН

Проект: LSB Infrastructure
`http://ispras.linuxfoundation.org/`

Анализ совместимости Linux-приложений с различными дистрибутивами

В докладе представлены средства для анализа приложений на совместимость с различными дистрибутивами Linux на основе базы знаний и инфраструктуры, созданных в Linux Foundation. Анализ основывается на сопоставлении сведений о наличии различных версий библиотек и их функций в различных дистрибутивах с требуемыми приложением внешними библиотеками и функциями.

Linux как платформа для приложений

В настоящее время в мире насчитывается более 500 публичных дистрибутивов Linux[1]. Каждый из таких дистрибутивов представляет собой уникальную комбинацию определённых версий/модификаций *базовых компонентов*, таких как ядро, библиотеки, системные утилиты, приложения и т. д. В данном докладе дистрибутивы Linux рассматриваются с точки зрения платформы для обеспечения функционирования сторонних приложений (то есть не включённых в комплект поставки самим производителем дистрибутива). С этой точки зрения главными компонентами дистрибутива являются собственно ядро операционной системы и набор разделяемых библиотек, которые предоставляют приложениям прикладные бинарные интерфейсы (ABI) в виде функций или глобальных данных.

Проблема заключается в том, что в разных дистрибутивах могут поставляться разные версии библиотек, отличающихся как версиями базового кода от разработчиков самих библиотек, так и изменениями, внесёнными разработчиками дистрибутива. В итоге, это может означать разные интерфейсы для приложений, как по составу, так и по поведению. Именно поэтому написать стороннее приложение, которое будет работать на всех дистрибутивах, да ещё и без перекомпиляции (что важно для многих производителей ПО), может оказаться не таким простым делом. Производители дистрибутивов стараются смягчить эту проблему, поставляя несколько версий одной и той же библиотеки в составе дистрибутива, чтобы разные приложения могли использовать необходимые им версии. Однако разработчикам приложений достаточно сложно найти централизованные сведения о составе различных дистрибутивов для анализа переносимости своих приложений. Именно поэтому актуальной задачей становится сбор такой информации и разработка средств автоматизированного анализа переносимости приложений между дистрибутивами, сведения о которых имеются в такой базе данных.

Linux Foundation Database

Для стандартизации, защиты и продвижения Linux крупнейшими ИТ-компаниями, среди которых стоит упомянуть IBM, Intel,

[1] См. например, `http://lwn.net/Distributions/`.

HP, Novell, Oracle, был создан международный консорциум **Linux Foundation**, который в настоящее время представляет собой основную в мире площадку, объединяющую усилия и экспертизу различных организаций и лиц, заинтересованных в успешном развитии Linux.

Для централизованного анализа экосистемы Linux в рамках деятельности Linux Foundation была создана база данных, содержащая информацию о составе различных дистрибутивов (наличие определённых библиотек и интерфейсов), о внешних зависимостях реальных приложений и о стандартизованных в рамках Linux Standard Base (LSB) элементах.

В настоящее время эта база данных включает более 80 таблиц с суммарно 25 миллионами записей. Основную часть данных занимают сведения о дистрибутивах и приложениях. По состоянию на июнь 2008 года база содержит информацию о 41 дистрибутиве и о 1067 приложениях. Эти сведения используются в том числе для поддержки принятия решений по развитию стандарта LSB.

Содержимое базы данных доступно для удобного просмотра с поиском и кросс-навигацией с помощью специализированной информационной системы **LSB Database Navigator**[2].

Linux Application Checker

Для проведения собственно анализа конкретных приложений на совместимость с различными дистрибутивами предназначен инструмент Linux Application Checker, созданный в рамках проекта LSB Infrastructure[3] в российском Центре верификации ОС Linux[4].

Linux Application Checker предоставляет пользователю веб-интерфейс на основе собственного встроенного простого веб-сервера и опирается на локальную копию специальным образом подготовленных данных из главной базы данных Linux Foundation. Таким образом, инструмент полностью независим и может использоваться на изолированных машинах.

В процессе работы пользователю предлагается задать целевое приложение в виде набора исполняемых файлов и поставляемых с приложением библиотек. Такие файлы можно задавать по отдельности или

[2] `http://linuxfoundation.org/navigator/`
[3] `http://ispras.linuxfoundation.org/`
[4] `http://linuxtesting.org`

указывать целые пакеты rpm, deb, tar.gz, из которых все найденные в ELF формате файлы будут автоматически трактоваться как относящиеся к приложению. Кроме того, в качестве целевого приложения можно указывать просто одно из установленных в системе — в таком случае все компоненты приложения будут автоматически получены из менеджера пакетов.

В процессе анализа заданного таким образом приложения на основе ELF считывается информация о зависимостях каждого компонента (список библиотек берется из DT_NEEDED записей секции .dynamic, а список интерфейсов формируется на основе фильтрации списка символов из .dynsym и .symtab секций). Затем исключаются внутренние зависимости между компонентами приложения и формируется объединение внешних библиотек (идентифицируемых по soname) и конкретных интерфейсов, необходимых приложению от дистрибутива.

Полученный список сопоставляется с предоставляемыми различными дистрибутивами версиями библиотек и интерфейсов, и формируется отчёт о совместимости заданного приложения с конкретными дистрибутивами. Отчёт создаётся в виде набора связанных HTML-файлов от общей сводки до конкретных деталей. Где возможно, выводятся сведения и рекомендации из базы знаний Linux Foundation, например рекомендации, чем заменить устаревшие (deprecated) интерфейсы, советы включить определённую библиотеку в состав поставки приложения или советы исключить неиспользуемые библиотеки (присутствующие в DT_NEEDED зависимостях, но без реально используемых интерфейсов).

В случае если заданное приложение совместимо со стандартом LSB, пользователю будет предложено сертифицировать его путём достаточно простой процедуры регистрации в онлайн сертификационной системе LSB.

Денис Силаков, Владимир Рубанов
Москва, ИСП РАН

Проект: LSB Navigator
http://ispras.linux-foundation.org/index.php/LSB_DB_Navigator

LSB Navigator — онлайн-справочник для разработчиков Linux-приложений

В докладе представлена веб-система LSB Navigator, являющаяся составной частью новой версии Linux Development Network (LDN). Инициатива LDN позиционируется Linux Foundation как аналог MSDN для Linux. LSB Navigator позволяет разработчикам быстро находить информацию о конкретных интерфейсах программирования и библиотеках Linux, их статусе и ссылки на документацию.

Linux Development Network

Linux предоставляет разработчикам приложений богатый выбор инструментария и библиотек для реализации своих идей. Более того, для воплощения одной и той же функциональности зачастую существует несколько альтернатив. В результате, программисты оказываются перед выбором — какие именно библиотеки использовать? Какие функции из этих библиотек стоит использовать, а какие нет? Поиск ответа на такие вопросы осложняется тем, что каждая библиотека имеет свою собственную документацию; при этом качество описания различных библиотек сильно разнится, и для использования многих из них полезными оказываются дополнительные источники информации. В результате, при создании сложного программного продукта разработчикам приходится иметь дело со множеством разрозненных спецификаций, статей, и т. п.

В ОС Windows разработчикам предоставляется Microsoft Development Network (MSDN) — целая электронная библиотека, объединяющая различные источники документации по программированию в Windows. С целью создания аналога MSDN для Linux консорциум Linux Foundation выступил с инициативой Linux Development Network (LDN), призванной собрать в одном месте спецификации, статьи, ссылки на документацию и другую информацию, которая может быть полезна при разработке приложений под Linux.

LSB Navigator

Одной из составных частей LDN является веб-система LSB Navigator, предоставляющая пользователям доступ к информации, хранящейся в базе данных стандарта Linux Standard Base (LSB). Наиболее подробными являются данные об элементах (библиотеках, функциях, командах), включённых в LSB, однако пользователь может получить сведения и по большому количеству объектов, не входящих в стандарт. Например, число бинарных интерфейсов (функций и глобальных переменных) в разрабатываемой версии LSB приближается к 40 тыс., в то время как всего в базе данных содержится информация о почти трёх миллионах интерфейсов.

В соответствии с характером предоставляемых пользователю данных, LSB Navigator разделён на три основных секции — *LSB Elements*, *Distributions and Applications* и *Workgroup Services*.

LSB Elements

Данная секция позволяет получить информацию об элементах стандарта LSB, к числу которых относятся:

- библиотеки;
- классы;
- бинарные интерфейсы;
- команды;
- модули интерпретируемых языков.

Каждый из этих элементов имеет т. н. «домашнюю страницу», на которой представлена вся информация о данном элементе, присутствующая в базе данных LSB. Одним из важнейших для разработчиков аспектов можно назвать наличие ссылки на документацию для интерфейсов, команд и модулей интерпретируемых языков. По возможности, предоставляется URL непосредственно на описание элемента. В случае, когда предоставить такую ссылку невозможно — например, некоторые функции описаны в спецификациях, доступных только в виде PostScript- либо PDF-документов, — предоставляется ссылка, по которой можно скачать спецификацию.

Ссылки на документацию предоставляются только для элементов, включённых в последнюю версию LSB. Для функций, рассматривавшихся в качестве кандидатов на включение в LSB, но отклонённых

рабочей группой, а также для интерфейсов, объявленных устаревшими, приводятся причины, по которым функция не входит в стандарт, и указывается, что можно использовать в качестве альтернативы.

Также для каждого элемента LSB на его домашней странице доступны данные о статусе элемента в различных версиях стандарта и его присутствии в различных дистрибутивах Linux. Для всех сущностей, кроме команд, доступна статистика по использованию в приложениях. Для каждого интерфейса, входящего в LSB, можно получить информацию о сертификационных тестах, нацеленных на проверку его реализации в дистрибутивах.

Помимо элементов стандарта, секция LSB Elements предоставляет информацию о тесно связанных с ними сущностях. К таковым относятся:

- заголовочные файлы, которые необходимо подключать для использования тех или иных функций;
- типы, необходимые для использования функций;
- константы и макроопределения, которые также могут быть полезны при разработке приложений.

Distributions and Applications

Данная секция содержит сведения о библиотеках, классах, командах и бинарных интерфейсах, предоставляемых дистрибутивами и используемых приложениями. Для каждого такого элемента существует «домашняя страница», аналогичная домашним страницам элементов LSB; однако информации на страницах элементов, которые никогда не входили в LSB, существенно меньше — можно получить список дистрибутивов, предоставляющих данный элемент, и список приложений, его использующих. Для функций, объявленных устаревшими разработчиками библиотек, в которые они входят, на домашней странице приводится соответствующий комментарий с предложением альтернативной замены.

Для дистрибутивов и приложений также существуют «домашние страницы», на которых приводится информация обо всех версиях дистрибутива либо приложения, данные о которых содержатся в базе LSB, а также ссылка на страницу производителя в Интернете. Для приложений приводятся отдельные списки используемых библиотек,

бинарных интерфейсов и модулей интерпретируемых языков, которые не входят в LSB.

Workgroup Services

Как следует из названия, эта секция предназначена, в основном, для членов рабочей группы LSB, однако информация, содержащаяся здесь, может быть интересна любому человеку, интересующемуся LSB. Здесь можно получить статистические данные о развитии стандарта (сколько элементов каждого вида появилось/было исключено в каждой версии), список спецификаций, на которые ссылается LSB, данные о покрытии интерфейсов сертификационными тестами.

Подраздел *Applications Statistics* содержит статистику использования в приложениях функций и библиотек, как входящих, так и не входящих в LSB. Страница *LSB Rating of Applications* этого подраздела отображает степень совместимости приложений, информация о которых загружена в базу данных LSB, со стандартом.

Заключение

LSB Navigator — лишь часть Linux Development Network. Объём предоставляемой LSB Navigator информации достаточно велик, однако это, в основном, ссылки на внешние источники документации и данные, собираемые автоматически с помощью соответствующих инструментов. В рамках превращения LDN в действительно полезную для разработчиков систему эта информация будет дополняться различными статьями и документацией компаний-партнёров, написанными специальными техническими писателями и просто активными членами сообщества.

Михаил Шигорин
Киев, ООО «Медиа Мэджик»

Проект: ALTSP
http://www.freesource.info/wiki/Dokumentacija/LTSP5
http://www.magic.kiev.ua/ru/solutions/servers/altsp5/

Linux Terminal Server Project

Преимущества использования терминальных технологий для целей миграции на свободное ПО и повторного использования морально устаревшего «железа». Участие в проектах ALT Linux и LTSP: порознь и вместе. ALTSP как уникальный сплав «лучшего из двух миров» — LTSP4/5.

Доистория

В 1999 году уже существовало как движение свободного ПО, так и намерение разработчиков ядра Linux вплотную заняться настольными системами. Этим воспользовались в одной детройтской фирме, занимавшейся обслуживанием госпиталей; так появилась первая версия LTSP.

События развивались довольно быстро, и в 2004 был выпущен LTSP4 с очень неплохими эксплуатационными характеристиками: работа на 12M RAM, загрузка в полминуты, поддержка балансировки нагрузки и локально запускаемых на клиенте приложений (например, для интернет-телефонии с доступом к микрофону).

LTSP5

В 2005 году начались работы над фреймворком по составлению терминальных решений из существующих дистрибутивов, а не над специализированным мини-дистрибутивом. Эта работа выполнялась в тесном сотрудничестве с проектом Ubuntu.

При том, что технологически переработка представляла собой прорыв, на практике многие ценные свойства были утеряны или разменяны на другие. Например, существенно возросли аппаратные требования к терминалам, сильно возросло время их загрузки, исчезла

поддержка LocalApps и кластеризации. В отличие от того, что задумывалось, ситуация с безопасностью также ухудшилась.

Всё это было оправдано происшедшим заложением фундамента для дальнейшей метадистрибутивной разработки: теперь LTSP предоставляет методологию по поддержке загрузки и обеспечения работы клиентов и компоненты, готовые к использованию для интеграции такого режима в существующие дистрибутивы.

В 2007 году к разработке LTSP5 присоединились участники проектов Gentoo, openSUSE, Fedora, Slackware, принося с собой кусочки специфического для их дистрибутивов кода, исправляя и улучшая общий.

ALTSP5

В том же 2007 году был доведён до бета-состояния и использован в деле форк LTSP5 на базе ALT Linux, сочетавший в себе как наработки по использованию дистрибутива из пятой ветки, так и зрелые компоненты из четвёртой; наш офис[1] успешно переехал на технологию ALTSP5 в конце весны.

К концу года был наработан дистрибутив, который обеспечил на стенде загрузку и нормальную работу Pentium 166/32M как одного из тонких клиентов, подключённых к более мощной машине — даже при использовании KDE, OpenOffice.org и Eclipse.

Почему форк?

Рассматривались разные варианты. В итоге остановились на LTSP5 в качестве основы, но были «выкинуты» те части LTSP 5.0, которые были сочтены незрелыми, и втянуты элементы LTSP 4.2, которые доказали свою лучшую пригодность (NFS root и использование XDMCP).

Upstream merge

Результат вскоре был работоспособен, но сильно отличался от основного направления разработки LTSP5. После ряда обсуждений было решено всё-таки попробовать объединить ветки весной 2008, что позволило сократить объём различий примерно вчетверо.

Текущее состояние

ALTSP5 остался таким же «гибридным» вариантом: используется специальная сборка ядра с патчами для надёжного свопа по сети; потребность в памяти терминала получилось свести к сопоставимому с требуемым для LTSP4.2 объёму — 16M.

Доступны снапшоты терминального решения на основе ALT Linux 4.0 Desktop и школьного дистрибутива Линукс Терминал, которые позволяют воспользоваться терминальным сервером непосредственно после установки (при возможности загрузиться по PXE).

Написана документация[2] и система управления клиентами — обе скорее в минимальном объёме: документации чем меньше *приходится* читать — тем лучше, а вот улучшение модуля Alterator — один из главных участков дальнейшей работы.

Планы

Существует намерение разрешить возникший «конфликт поколений» LTSP (в первую очередь по части требований к ресурсам) за счёт реализации поддержки множественных транспортов в стиле LTSP4/5 и расширяя их — для:

- загрузки терминалов (NFS/NBD);
- регистрации в системе (XDMCP/LDM/RDP/...);
- работы с приложениями (X11/X11+ssh/RDP/...);
- работы с локальными устройствами (ltspfs/RDP/...).

Дистрибутивами по факту вовсю пользуются и в виде снапшотов, причём они весьма близки к состоянию релиза; работа в направлении выпусков также ведётся.

Контакты

Загрузить бета-версии (и будущие релизы) можно с серверов `ftp.linux.kiev.ua` и `beta.altlinux.org`; подписаться на рассылку, где идёт обсуждение разработки и применения — по адресу `https://lists.altlinux.org/mailman/listinfo/ltsp-server` .

Докладчика можно найти как `mike@altlinux.org` (email/jabber).

Литература

[1] Терминал-сервер. — 2007.
http://www.magic.kiev.ua/ru/solutions/servers/altsp5/.

[2] *Chumachenko, A.* Ltsp5 в altlinux. — 2007.
http://freesource.info/wiki/Dokumentacija/LTSP5.

Андрей Михеев, Алексей Русаков
Москва, Консалтинговая группа Руна

Проект: RunaWFE
http://sourceforge.net/projects/runawfe

Workflow Server и Workflow Desktop — совместные дистрибутивы Альт Линукс и Консалтинговой группы Руна на базе дистрибутивов ALT Linux 4.0 Server и ALT Linux 4.0 Personal Desktop

Программный продукт решает задачу автоматизации процессов административного управления. Дистрибутивы содержат клиентские и серверные компоненты workflow-системы. Основная задача системы — раздавать задания исполнителям и контролировать их выполнение. Система является открытой, распространяется под лицензией LGPL. Достоинствами продукта являются простота установки и эксплуатации, наличие подробной документации.

Введение

Процессный подход к организации административного управления получает всё большее распространение. В соответствии с этим подходом деятельность по управлению представляется в виде множества процессов — наборов заданий, выполняемых как людьми, так и информационными системами.

Последовательность заданий определяется графом процесса, который менеджер или бизнес-аналитик может быстро изменять при помощи редактора бизнес-процессов.

Процессный подход реализуют компьютерные системы, называемые workflow-системами. Эти системы позволяют быстро разрабатывать и изменять процессы, не меняя кода системы.

Передача информации между исполнителями заданий происходит при помощи переменных процесса. В случае если в переменных хранить документы, workflow-систему можно использовать для автоматизации документооборота.

Краткое описание системы

Система состоит из:

- Собственно workflow-системы;
- Графического редактора процессов;
- Клиента-оповещателя о поступивших заданиях.

Основные функции

Собственно workflow система:

- Работа с определениями и экземплярами процессов;
- Работа со списками заданий;
- Визуализация форм, соответствующих заданиям;
- Работа с системой через web-браузер;
- Предоставление возможности работы с системой приложениям специального вида (ботам[1]);
- Авторизация и аутентификация пользователей.

Графический редактор процессов:

- Редактирование графа процесса;
- Создание и редактирование графических форм заданий;
- Создание и назначение ролей ;
- Создание переменных.

Клиент-оповещатель о поступивших заданиях:

- Оповещение о поступивших заданиях;

[1] В частности, боты могут моделировать работу сотрудника предприятия.

- Работа с системой через специальное приложение-клиент.

Система является как бы конвейером, перенесённым с производства в офис, позволяет работнику выполнять поступившие задачи, не отвлекаясь на:

- Получение необходимой для выполнения задания информации;
- Передачу результатов своего труда другим работникам;
- Изучение должностных инструкций.

Всё необходимое возникает на экране пользователя при «клике» на задании (в частности, на экране может быть написана инструкция — как надо выполнять это задание).

Исполнителями могут быть как люди, так и специальные компьютерные приложения — боты.

Используя ботов, можно при помощи системы решить задачу интеграции разнородных приложений предприятия в единую систему (КИС).

Состав дистрибутивов

Дистрибутивы представляют собой два диска:

- Установочный диск для клиентской части (Клиентский дистрибутив);
- Установочный диск для сервреной части (Серверный дистрибутив).

При помощи клиентского дистрибутива можно установить следующие компоненты:

- Клиент — web-ссылка на клиентский web-интерфейс;
- Клиент-оповещатель о поступивших заданиях;
- Графический редактор бизнес-процессов;
- Документация;
- Симулятор бизнес-процессов (устанавливаемый на клиенте компонент, при помощи которого можно тестировать разработанные бизнес-процессы).

После того, как компоненты установлены, с ними можно работать через меню системы Меню/Офис/Runa WFE и значки на рабочем

столе (при установке RunaWFE в уже существующую ОС значки на рабочий стол не устанавливаются).

При помощи серверного дистрибутива можно установить следующие компоненты:

- Runa WFE сервер;
- Бот-станция.

При помощи дистрибутивов можно доустановить RunaWFE на уже установленную систему ALT Linux 4.0 Personal Desktop или ALT Linux 4.0 Server.

Литература и ссылки

1. OnLine demo системы доступно по адресу: `http://wf.runa.ru/Online_Demo`
2. Ссылка на сайт проекта: `http://wf.runa.ru`
3. Что такое системы управления бизнес-процессами: А. Михеев, М. Орлов. Цикл статей: «Перспективы workflow систем». PCWEEK

- `http://www.pcweek.ru/themes/detail.php?ID=67765`
- `http://www.pcweek.ru/themes/detail.php?ID=68049`
- `http://www.pcweek.ru/themes/detail.php?ID=69002`
- `http://www.pcweek.ru/themes/detail.php?ID=71354`
- `http://wf.runa.ru/images/9/93/Article04_examples.pdf`

Алексей Турбин
Рязань, ALT Linux Team

Проект: Sisyphus
http://sisyphus.ru

Сборочная система git.alt

Новая система сборки rpm-пакетов поддерживает целостность репозитория за счёт транзакционных переходов, при которых полностью вычисляются характеристики репозитория. По умолчанию разрешены только переходы, которые не ухудшают характеристик. Система ориентирована на асинхронное продвижение транзакций, а окончательная сериализация и учёт взаимного влияния между транзакциями осуществляется при помощи *слияний* (merge). Для этого система поддерживает внутреннюю структуру данных — интроспективный *метарепозиторий*.

Во введении дан обзор средств совместной разработки ALT Linux Team, созданных на основе распределённой системы контроля версий GIT.

Введение

Сборку rpm-пакетов можно рассматривать как процесс, который реализует функцию B(S,C)->P, где S — src.rpm-пакет с исходным кодом, C (chroot) — сборочная среда, P — собранные rpm-пакеты. Надёжная сборка rpm-пакетов, т. е. воспроизводимость процесса сборки B (build), осуществляется с помощью программы hasher [Левин 2004].

Позже была разработана программа gear [Левин 2007], которая позволяет хранить исходный код src.rpm-пакетов в системе контроля версий GIT [Торвальдс 2005]. *Gear-репозиторием* G мы называем git-репозиторий с исходным кодом, из которого можно получить src.rpm-пакет для сборки: G->S.

Несмотря на некоторые преимущества src.rpm-пакетов (такие, как сохранение полностью оригинального тарболла, а также сама возможность распространения исходного кода в виде, готовом для сборки), использование gear-репозиториев значительно упрощает *совместную разработку* пакетов. Вообще, использование системы контроля версий стимулирует переход от пассивной *поддержки* пакетов к более

интенсивной работе с исходным кодом, даёт больше возможностей для *разработки*. Это преимущество можно считать решающим, поэтому было предложено использовать gear-репозитории как основной формат хранения исходного кода пакетов в ALT Linux Team. (При этом src.rpm-пакеты продолжают существовать лишь как промежуточный «полуфабрикат» для сборки; в дальнейшем возможен полный отказ от хранения src.rpm-пакетов.)

Сервер совместной разработки gear-репозиториев ALT Linux Team мы кратко обозначаем как `git.alt`. Одной из подсистем `git.alt` является `girar` (архив gear-репозиториев); `girar` осуществляет доступ разработчиков к gear-репозиториям. Каждый разработчик имеет собственные копии gear-репозиториев, в которые он может вносить, вообще говоря, достаточно произвольные изменения (предлагая, таким образом, включить эти изменения в очередную версию пакета). Реализована система почтовых уведомлений, которая упрощает обмен изменениями. При этом фактический обмен и аккумуляция изменений осуществляется средствами распределённой системы контроля версий `GIT`.

Несмотря на то, что любой разработчик может внести изменения в пакет, окончательную версию пакета могут подготовить только один или несколько разработчиков, за которыми закреплён этот пакет (maintainers). Контроль осуществляется подсистемой `girar ACL`, и «неавторизированные» версии пакетов по умолчанию отвергаются (вместе с тем, сохранена возможность для т. н. NMU, non-maintainer upload).

Когда новая версия пакета окончательно подготовлена, разработчик делает в gear-репозитории *метку* (tag) с криптографической подписью и инициирует запрос на сборку пакета. Запрос обрабатывается сборочной системой `girar-builder`, которая является предметом настоящего доклада. Сборочная система взаимодействует с *кеширующим сервером* gear-репозиториев. Если запрос на сборку был обработан успешно, то собранные rpm-пакеты помещаются в репозиторий rpm-пакетов, а gear-репозиторий с новой меткой публикуется на кеширующем сервере. Названия gear-репозиториев на кеширующем сервере совпадают с именами src.rpm-пакетов (это требование не является обязательным для gear-репозиториев разработчиков). Таким образом, кеширующий сервер предоставляет доступ к «официальным» gear-репозиториям, содержимое которых соответствует фактическим сборкам пакетов.

Задания и транзакции

Задание состоит из одного или нескольких gear-репозиториев (и их меток), отправленных на сборку. Сборочная система сначала осуществляет *первичную сборку* пакетов. Если при первичной сборке пакетов возникли грубые ошибки, то задание автоматически отменяется. В противном случае, если все пакеты в задании успешно собрались, то *открывается транзакция*: генерируется временный репозиторий, в котором выполняется ряд *проверок* (вычисляются *характеристики* нового репозитория). По умолчанию переход в новое состояние разрешён, только если характеристики репозитория не ухудшились. Если же собранные пакеты ухудшают характеристики репозитория, то составляется *список нарушений*, и задание переводится в *отложенный режим*.

Выделим следующие проверки, выполняемые сборочной системой:

- **Неудовлетворённые зависимости** (unmet dependencies). Если при помещении пакетов в репозиторий возникают новые неудовлетворенные зависимости, то по умолчанию такой переход не может быть разрешён.
- **Тестовая пересборка пакетов** в наибольшей степени обнаруживает взаимное влияние между пакетами. Эта проверка может быть довольно ресурсоемкой: должны быть пересобраны все пакеты, у которых в сборочной среде `C` оказывается хотя бы один новый пакет из транзакции. Если же хотя бы один новый пакет их транзакции входит в базовую сборочную среду, то потребуется полная тестовая пересборка всего репозитория rpm-пакетов. Сборочная система фиксирует не только саму возможность пересборки, но и изменение зависимостей у пересобранных пакетов.
- **Идентичность noarch пакетов** при сборке на всех архитектурах.
- **Нарушение ACL** у какого-либо пакета в задании не относится, строго говоря, к характеристикам репозитория; однако включение его в общий список нарушений транзакции упрощает NMU.

Воздействовать на задания в отложенном режиме можно, в основном, двумя способами:

- **Добавление новых пакетов**, которые исправляют нарушения. Например, если у библиотеки изменилось имя (soname), то

в репозитории могут образоваться неудовлетворённые зависимости на старое имя библиотеки. Тогда в задание можно добавить все пакеты, которые зависят от старой библиотеки, чтобы пересобрать их с новой библиотекой.

- **Разрешение нарушений.** Например, администратор `git.alt` может разрешить нарушение ACL, если «неавторизированная» версия пакета (NMU) содержит исправления критических ошибок.

Таким образом, задания могут переводиться в отложенный режим и позднее возобновляться. Если все нарушения были сняты, то выполняется транзакционный переход репозитория в новое состояние.

Метарепозиторий

Необходимость учёта взаимных влияний между транзакциями приводит к идее *метарепозитория* — структуры данных, используемой сборочной системой, которая также представляет самостоятельный интерес для разработчиков. Метарепозиторий организован как обычный git-репозиторий, который содержит каталоги по имени src.rpm-пакетов `S`. В этих каталогах хранится вся существенная информация о пакетах, в частности:

- **Сборочные зависимости** (BuildRequires) src.rpm-пакетов. Хранение сборочных зависимостей упрощает запуск тестовой пересборки пакетов.
- **Зависимости** собранных пакетов. Изменение зависимостей при тестовой пересборке позволяет изучать взаимное влияние между пакетами.
- **Логи сборки** пакетов. Изменения в логах сборки пакетов позволяет анализировать свойства пакетов, которые трудно учесть формально (например, новые предупреждения при компиляции).

История изменений в метарепозитории соответствует истории продвижения транзакций. В каждый момент времени основное состояние (master) метарепозитория соответствует текущему (стабильному) состоянию репозитория rpm-пакетов. Когда открывается новая транзакция, в метарепозитории создается новый *бранч* (в терминах `GIT`), имя которого соответствует целочисленному идентификатору зада-

ния; дальнейшие изменения в бранче метарепозитория соответствуют этапам продвижения транзакции.

Когда отложенное задание вновь активизируется, может потребоваться *слияние* (merge) master в бранч, которое учитывает, в частности, возможные изменения в сборочной среде `C`. Наконец, при транзакционном переходе осуществляется другой тип слияния: окончательное слияние бранча в master (в простейшем случае оно может быть реализовано при помощи т. н. fast forward). Оба типа слияний являются не чисто «текстовыми» (как в системе контроля версий), а «логическими», т. е. результат слияния определяется дополнительной логикой работы сборочной системы.

Литература

[Левин 2004] *Дмитрий Левин*, Hasher: технология безопасной сборки пакетов // Первая международная конференция разработчиков свободных программ на Протве. Тезисы докладов. М., 2004. С. 28–30.

[Левин 2007] *Дмитрий Левин*, От SRPMS к GEAR // Четвёртая международная конференция разработчиков свободных программ на Протве. Тезисы докладов. М., 2007. С. 14–18.

[Торвальдс 2005] *Линус Торвальдс*, GIT — the information manager from hell.
`http://git.or.cz` и др.

Ринат Биков
Ижевск, УдГУ

Проект: SEvents
`http://sourceforge.net/projects/sevents`

Система событийного программирования SEvents

Введение

Событийное программирование — стиль программирования, при котором вычисления активизируются в результате некоторого собы-

тия в системе и являются реакцией на происшедшее событие, изменяя локальные характеристики системы[1].

Событийное программирование можно разделить на программирование от событий и от приоритетов. *SEvents* — система событийного программирования с фактором случайности, которая в некоторой степени реализует стиль программирования от приоритетов. Однако она имеет некоторые особенности.

Описание

SEvents обладает следующими чертами:

- Сценарий событий описывается на специальном языке скриптов;
- Переменные в сценарии могут быть трёх типов: *int*, *boolean*, *double*;
- Все переменные глобальные;
- Описания всех событий и глобальных переменных располагаются в одном файле;
- Условия выполнения событий произвольны;
- Нет циклов, условных операторов, массивов и других структур;
- Объекты сценария реализуются в виде динамических библиотек;
- Предоставляются минимальные возможности визуализации объектов;
- Обработчики неотделимы от событий;
- Обработчики активных событий выполняются псевдопараллельно, согласно приоритетам[1];
- Обработчик события состоит из последовательности операторов, имеющих приоритет события;
- Соблюдение приоритетов операторов системой является необязательным;
- Производится полное журналирование изменений системы.

Данная система событийного программирования в первую очередь предназначена для обучения студентов, поэтому главной задачей при

[1] С поправкой на особенность, о которой будет рассказано ниже.

её разработке было полное журналирование. Также не последняя роль отводилась простоте системы. Для усложнения задачи синхронизации введена вероятность *«выскакивания»* оператора за пределы своего приоритета, в чём и заключается необязательность соблюдения приоритетов.

Выводы

SEvents — система событийного программирования с открытым кодом, позволяющая уйти от ограничений, налагаемых прикладными реализациями событийного программирования и существующими ЯП, в которых невозможно объявить событием ситуацию, когда значение одной переменной больше другой. Снятие данного ограничения значительно расширяет возможности при написании программ событийного стиля.

Литература

[1] *Н.Н.Непейвода.* Стили и методы программирования, Москва: Интернет-Университет Информационных Технологий, 2005.

Станислав Иевлев
Москва, ALT Linux

Проект: alterator
`http://wiki.sisyphus.ru/Alterator`

ALT Linux Installer

После нескольких лет разработки в ALT Linux была создана новая современная программа установки операционной системы. Она обладает высокой функциональностью (широкий выбор источников установки, возможность автоустановки), простым интерфейсом, модульностью и широкими возможностями по адаптации к конкретному продукту.

Инсталлятор — неотъемлемая часть любого дистрибутива. Основная задача инсталлятора — корректное развёртывание системы на оборудовании пользователя. От него же зависит и общее впечатление от системы в целом.

Инсталлятор ALT Linux есть результат объединения следующих продуктов: общая инфраструктура (installer), штатный конфигуратор системы (alterator) и профиль установки (installer-desktop, installer-server и т. д.).

Alterator — модульная система с поддержкой трёх различных интерфейсов взаимодействия с пользователем: командная строка, традиционный GUI и http-интерфейс через web-браузер. В инсталляторе и центре управления системой используются одни и те же модули. Благодаря этому достаточно легко отлаживать отдельные компоненты инсталлятора.

Installer состоит из трёх стадий. Первая, загрузчик, поддерживает массу способов загрузки основной части инсталлятора: диск, nfs-сервер, cdrom, ftp-сервер (анонимный и с паролем), http-сервер. В первых двух случаях возможна работа как с развёрнутым дистрибутивом, так и с iso-образом непосредственно. Вторая стадия инсталлятора выполняет основную часть работы — разбивку диска и установку базовой системы. Третья стадия работает уже внутри установленной системы. Благодаря такому разбиению на каждом этапе имеет место минимальный расход оперативной памяти, что позволяет устанавливаться на достаточно слабые машины.

Инсталлятор работает в стиле пошагового выполнения модулей конфигуратора. Шаги могут быть интерактивные и неинтерактивные. Последние оформлены в виде скриптов и разделены на следующие группы: initinstall.d — инициализация инсталлятора, preinstall.d — перенос настроек из среды инсталлятора в установленную систему (между второй и третьей стадиями инсталлятора), postinstall.d — изменения в установленной системе по окончанию работы инсталлятора.

Каждый модуль конфигуратора, работающий в инсталляторе, может иметь свой набор preinstall.d-скриптов. Таким образом, когда составитель дистрибутива заменяет модуль на эквивалентный, прозрачным образом происходит замена и необходимых для его работы скриптов.

Профиль инсталлятора — содержит описание шагов, описание порядка исполнения шагов, а также набор дополнительных скриптов. Для наиболее распространённых случаев можно воспользоваться готовыми коллекциями скриптов из пакетов серии installer-feature. Например, installer-feature-pxeboot делает все необходимые настройки

в системе, чтобы сразу после установки она могла функционировать как tftp-сервер.

Таким образом, инсталлятор может быть очень точно подстроен под требования конкретного выпускаемого продукта, а наличие большого количества готовых компонент на все случаи жизни позволяет свести к минимуму возможные ошибки при составлении профиля.

К сказанному осталось только добавить, что инсталлятор имеет отладочный режим (что позволяет запустить его в случае сбоя автоопределения оборудования) и режим неинтерактивной автоустановки по заранее составленному сценарию.

Владислав Завьялов
Москва, ALT Linux

Проект: Alterator
http://wiki.sisyphus.ru/Alterator

Разработка модулей alterator

Платформа alterator является основой для систем инсталляции и конфигурации в дистрибутивах ALT Linux. Для конфигурации каждого компонента операционной системы создаётся специальный модуль, который предоставляет пользователю графический интерфейс для изменения тех или иных параметров системы. Alterator обеспечивает удобную инфраструктуру для создания подобных модулей. В докладе рассказывается про современное устройство модулей alterator и особенностях их создания.

Архитектура alterator и его модулей определяется основной решаемой задачей — созданием интерфейса к настройкам операционной системы.

Каждый модуль состоит из описания пользовательского интерфейса и бакенда — части, предназначенной для собственно изменений настроек системы. Бакенды могут создаваться на произвольных языках программирования. Существует инфраструктура, упрощающая создание бакендов на языках perl и shell. Разрабатывается подобная инфраструктура для создания бакендов на ruby.

Сейчас существует два вида интерфейсов, которые описываются отдельно. Web-интерфейс описывается html-шаблоном специального вида: предусмотрено заполнение формы информацией из бакен-

да и отправка этой информации обратно в бакенд. Кроме того, предусмотрена javascript-библиотека для создания различных динамических эффектов: переключение активности и видимости элементов html в зависимости от действий пользователя, специальные элементы, представляющие изображения часов и календаря...

Также существует возможность создания интерфейса для специального графического браузера на базе QT. Описание интерфейса выполняется на языке scheme, для чего создана библиотека, работающая с простым набором графических элементов, которая позволяет обмениваться информацией с бакендом и т. п.

Alterator предоставляет среду для выполнения бакендов и связь их с соответствующими интерфейсами. Устройство альтератора позволяет использовать модуль в различных режимах: к бакенду можно обращаться из командной строки, можно показывать его интерфейс в отдельном окне или обёрнутым в меню центра управления системой, или в составе пошагового инсталлятора.

В alterator также существуют единые системы сборки и локализации модулей, общая для html- и qt-интерфейсов система справки.

Евгений Синельников, Дмитрий Масленников
Саратов, ООО «Этерсофт»

Проблемы построения интегрированной сетевой инфраструктуры на основе GNU/Linux

Доклад посвящён проблематике администрирования корпоративных IT-систем. В докладе рассмотрены возможные пути решения проблем администратора с точки зрения разработчика. Приведён обзор технологий, используемых при построении таких решений, как Microsoft Active Directory, Novell eDirectory, FreeIPA.

При разворачивании и предварительной настройке корпоративных IT-систем, перед администратором ставится огромное количество различных задач. Для решения этих задач существуют специализированные программные продукты. Но, к сожалению, практически все такие продукты являются коммерческими и имеют высокую стоимость.

Существует мнение, что большинство популярных дистрибутивов GNU/Linux уже содержат всё необходимое для построения таких решений. Так что администратор может обойтись только ими. Это мнение основано на том, что в современные дистрибутивы действительно входит ряд ключевых технологических решений, позволяющих обеспечить специфические потребности большинства даже самых искушённых пользователей.

Тем не менее объёмы работ, необходимые для разворачивания и администрирования даже не самых сложных вариантов настройки, ставят большие проблемы перед достаточно опытными администраторами. Второй стороной этой проблемы является то, что получающиеся в результате решения являются уникальными, недокументированными и, как следствие, порождают огромные проблемы при смене персонала.

Единый сетевой вход в систему

Самая первая и самая проработанная проблема, с которой сталкиваются администраторы при разворачивании сетевой инфраструктуры предприятия, — это организация единого сетевого входа в систему (сетевого «логина»). Для этого необходимо организовать единую базу данных пользователей, выбрать механизм аутентификации и правильно настроить все клиентские компьютеры предприятия.

Для доступа к данным в большинстве современных реализаций централизованных хранилищ сетевых объектов принято использовать протокол LDAP[1], позволяющий представлять данные в виде дерева каталогов объектов. Этот протокол используется в коммерческих продуктах Microsoft Active Directory[2] и Novel eDirectory[3].

Администратор, желающий построить систему на бесплатных компонентах, может выбрать Fedora DirectoryServer[4], OpenLDAP[5], ApacheDS[6] и другие реализации. К сожалению, свободные реализации обычно сильно проигрывают в возможностях средств администрирования и разворачивания. Кроме того, большинство руководств по настройке LDAP-серверов ориентированы на самые простые решения и не раскрывают преимущества иерархической организации данных. Как следствие, на практике это приводит к довольно упрощённым вариантам использования этих мощных решений.

В ряде случаев для этого используются реляционные базы данных. Выбор хранилища при этом определяет модули, необходимые

для настройки клиентов. В GNU/Linux это PAM[7] и NSS-модули[8] для аутентификации и локальной авторизации соответственно.

В итоге можно заключить, что под Linux существуют продукты, позволяющие решить данную задачу. Но есть и проблемы: от администратора потребуются существенные знания по этому вопросу. Администратор Active Directory вообще может не знать о том, что в нём используются LDAP и Kerberos. Закрытые системы идут по пути разворачивания готового интегрированного решения, тогда как при использовании открытых продуктов потребуется настройка отдельных компонент. При этом настройка их становится значительно сложнее, так как приходится разбираться в деталях их взаимодействия, чтобы все они заработали совместно.

Безопасная аутентификация

Безопасная система аутентификации, используемая в большинстве современных корпоративных IT-систем, основана на протоколе Kerberos[9]. Популярность этого протокола связана с тем, что это один из немногих протоколов, не требующих передачи сертификатов во время инициализации сессии, защищённый при этом от атаки «подслушивание» (то есть не передающий по сети пароли) и «человек посередине», а также имеющий отлаженную свободную реализацию и позволяющий осуществлять прозрачный доступ к сетевым ресурсам (без дополнительного ввода пароля) на основе выдаваемых на время сессии билетов. Зависимостями на библиотеку libkrb5 пронизаны многие пакеты в различных дистрибутивах (от SSH до KDE).

При этом наличие клиентской поддержки не отменяет ряда проблем, связанных с особенностями использования и настройки сервера KDC (Kerberos Distribution Center). Эти проблемы, прежде всего, связаны с неприспособленностью протокола для применения вне специально сформированной среды, представляющей собой совокупность дополнительных сервисов. Причина необходимости дополнительных сервисов — отсутствие в протоколе собственных средств авторизации и зависимость от сервера доменных имён (DNS). То есть всё, что может этот протокол, — это подтвердить, является ли пользователь или служба тем, за кого себя выдаёт.

Несмотря на все преимущества, проблемы, связанные с особенностями использования Kerberos, имеют довольно неприятные последствия — многим администраторам просто неизвестно о возмож-

ностях этого протокола, поскольку типовые решения, построенные на нём, мало распространены. Кроме того, при использовании LDAP-серверов появляется проблема совместного администрирования. Одним из интересных решений, в этом плане, является инициатива компании Red Hat с проектом FreeIPA[10].

Централизованное управление

В плане удалённого управления GNU/Linux-решениями существует три основных подхода: управление через Web (облегчённый вариант, требующий только наличия броузера); прямой доступ к удалённому терминалу (обычно для этого используется SSH); специализированные средства управления через удалённый вызов процедур (RPC).

Основными проблемами удалённого управления через Web (HTTP или HTTPS) и SSH является то, что для администрирования разных сервисов, находящихся на разных серверах, необходимо будет явно подключиться к каждому из них. Например, чтобы завести пользователя в среде с OpenLDAP и Kerberos, потребуется в каждом из них проделать свою часть работы: в LDAP нужно создать соответствующий объект, а в Kerberos — соответствующий «принципал» (единица аутентификации в Kerberos).

При этом использование RPC и «толстых» клиентов позволяет администратору создать пользователя и задать для него пароль, предоставив клиенту самому подключиться ко всем необходимым серверам и выполнить все необходимые действия. Это снижает вероятность ошибки и уменьшает время, затрачиваемое на администрирование системы. Кроме того, так проще предоставить администратору возможность воспользоваться всеми компонентами системы из скриптов для автоматизации своих задач. Но реализация RPC и «толстых» клиентов намного сложнее, чем Web или SSH. Видимо, последний фактор и повлиял на то, что в свободных проектах используется в основном Web. Коммерческие продукты используют в основном механизмы RPC.

Локация сервисов

Проблема поиска сервисов в сети является одной из первых проблем, с которой сталкивается администратор, желая дать пользователю возможность удобно находить и обращаться к сетевым ресур-

сам. Для решения этих проблем используются различные средства. Основными из них являются DNS (в частности, SRV и TXT-записи) и DHCP. В GNU/Linux-решениях для сетей без выделенного сервера также может помочь сервис Avahi[11].

К сожалению, некоторая трудоёмкость получения параметров, предоставляемых этими службами в прикладных программах, а также необходимость специальной, не менее простой настройки всех этих сервисов, приводит к тому, что этими средствами в открытых системах зачастую не пользуются. Закрытые решения берут большую часть работы на себя (например, любой компьютер в Active Directory самостоятельно заносит свой адрес в DNS-сервер, если имеет на это право), и не требуют кропотливой настройки от администратора.

Отдельным образом решается задача просмотра сетевых каталогов, разделяемых по протоколу CIFS/SMB, что связано прежде всего с огромной популярностью данного сервиса и отсутствием общесистемного решения. Для этого существует огромное число различных утилит (smbfs, konqueror, nautilus, smb4k и т. д.)

Создание новых сервисов и интегрирование существующих

Проблемы добавления новых сервисов и их интеграции в существующую инфраструктуру актуальны до сих пор. Ни одно, даже самое обширное и разноплановое решение, не может покрыть все запросы всех предприятий и организаций. Поэтому возникает задача добавления новых сервисов. Чтобы вписаться в существующую инфраструктуру, сервис должен использовать единую базу пользователей, общий механизм аутентификации и пр. Но практика показывает, что большая часть решений реализует всё самостоятельно. Причиной этого является сложность и запутанность API для интеграции в систему. А это, в свою очередь, приводит к тому, что разобраться в устройстве системы, не являясь её разработчиком, практически невозможно. Это касается, в основном, закрытых решений, так что у открытых есть шанс исправить данное положение.

Те, кто сомневается с привёденным утверждением, могут попробовать написать аналог mpd[12], который будет использовать Active Directory[2] для аутентификации и авторизации пользователей, пытающихся управлять им.

С другой стороны, mpd уже реализован, но к интеграции в централизованно управляемую инфраструктуру не приспособлен, что поро-

ждает сложности в его администрировании. Интеграция же подобных решений позволит существенно сократить время, затрачиваемое на их администрирование.

Администрирование рабочих станций

У администратора на предприятии, в котором существует большое количество клиентских станций, неизбежно появляется проблема автоматизации их настройки и перенастройки, обслуживания, установки и обновления ПО и пр. При этом необходимо изменять различные настройки рабочих станций в зависимости от их использования. Свободные средства для автоматизации этих действий авторам неизвестны.

Другая проблема появляется при развёртывании интегрированных систем. Эта проблема состоит в настройке аутентификационных и авторизационных клиентских модулей. Суть проблемы с настройкой клиентских модулей заключается в существовании множества подходов к настройке модульной системы PAM[7] и NSS[8].

Следствием этой проблемы является наличие в различных дистрибутивах собственных средств настройки аутентификации и авторизации, не приспособленных к расширению. В целом, это приводит к тому, что даже в рамках дистрибутивов не существует готовых средств для организации процесса развёртывания специализированных решений.

Масштабируемость

Проблемы масштабируемости интегрированных инфраструктур прежде всего связаны с вопросами репликации (механизм прозрачной синхронизации), которые в случае мультимастерной репликации (механизм взаимной синхронизации на запись) одновременно решают ещё и вопросы резервного копирования. Выбор того или иного метода репликации определяется особенностями использования интегрированной среды. В случае одномастерной репликации проблема сводится к переводу различных сервисов к сохранению данных в едином хранилище, в случае мультимастерной вопрос осложняется проблемами разрешения неизбежных коллизий.

В заключение хотелось бы отметить, что рассмотренный ряд проблем не является полным. Тем не менее поставленные проблемы,

по мнению авторов, проявляются при администрировании корпоративных IT-систем чаще всего.

Литература

[1] Lightweight directory access protocol.
http://ru.wikipedia.org/wiki/LDAP.

[2] Microsoft active directory.
http://ru.wikipedia.org/wiki/Active_Directory.

[3] Novell edirectory.
http://ru.wikipedia.org/wiki/Novell_eDirectory.

[4] Fedora directory server.
http://ru.wikipedia.org/wiki/Fedora_Directory_Server.

[5] Openldap.
http://ru.wikipedia.org/wiki/OpenLDAP.

[6] The apache directory project.
http://directory.apache.org/.

[7] Pluggable authentication modules.
http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules.

[8] Name service switch.
http://en.wikipedia.org/wiki/Name_Service_Switch.

[9] Kerberos.
http://en.wikipedia.org/wiki/Kerberos_(protocol).

[10] Freeipa.
http://www.freeipa.org/.

[11] Avahi.
http://en.wikipedia.org/wiki/Avahi_(software).

[12] Music player daemon.
http://en.wikipedia.org/wiki/MPD_(Music_Player).

Дмитрий Масленников, Евгений Синельников
Саратов, ООО «Этерсофт»

Проект: Tartarus
http://www.tartarus.ru

Tartarus. Интегрированная среда для построения сетевых сервисов

В докладе рассказывается о создании открытой среды для разработки сетевых сервисов с едиными безопасными механизмами взаимодействия, аутентификации и авторизации. Данная среда используется для построения комплексного решения по администрированию парка компьютеров типового предприятия. Проект позиционируется как альтернатива известным средствам Microsoft Active Directory и Novell eDirectory под GNU/Linux.

Приступая к созданию интегрированного решения для администрирования сети и парка компьютеров предприятия, в первую очередь мы ставили перед собой следующие задачи:

- Расширяемость — новые компоненты должны легко добавляться к системе.
- Безопасность — безопасная аутентификация и шифрование всех важных данных, передаваемых по сети.
- Простота разворачивания и администрирования — стандартный вариант системы должен легко разворачиваться, не требуя администратора с высокой квалификацией. Дальнейшее обслуживание также должно быть максимально простым и удобным.

Мы старались не ограничивать круг решаемых задач. В первую очередь мы решаем задачи организации единой базы данных всех пользователей предприятия, автоматической настройки рабочих станций, предоставления доступа к интернет с рабочих станций и учёта его использования, администрирования почтового сервиса. Это базовый набор, который мы собираемся предоставить, но его можно неограниченно расширять, добавляя произвольные компоненты.

В качестве протокола для безопасной аутентификации мы выбрали протокол Kerberos, как и практически все аналогичные проекты. Для взаимодействия компонентов системы мы выбрали ICE (The Internet Communications Engine)[1]. Для того чтобы использовать

[1] http://www.zeroc.com/ice.html

Kerberos совместно с ICE, нам пришлось внести небольшие изменения в его компонент для работы с SSL.

В целом, вся система состоит из сервисов. Сервисом мы называем произвольное сетевое приложение, конфигурацию которого можно менять программно по сети, используя протокол ICE, если пользователь, который пытается это сделать, зарегистрирован в системе, имеет соответствующие права и аутентифицировался через Kerberos.

Мы не ставим перед собой задачи написать все необходимые нам сервисы. Чаще всего мы просто выбираем то или иное приложение, уже реализующее необходимый нам функционал (BIND, PostgreSQL, MIT-KDC и пр.), и добавляем к нему вспомогательное приложение-конфигуратор, задачей которого является предоставление ICE-RPC API для конфигурирования сервиса. Конфигуратор ставиться на тот же физический или виртуальный сервер, что и само основное приложение. Так мы решаем задачу удалённого управления нашими сервисами.

Основой системы является базовый набор сервисов, без которых система функционировать не может. Этими компонентами являются:

- Сервер KDC — обязательный компонент Kerberos. Мы используем реализацию MIT Kerberos[2].
- Сервер DNS, который практически необходим для Kerberos. Сейчас мы используем PowerDNS[3].
- Системная база данных (SysDB) — основная база данных Tartarus, которая хранит информацию обо всех пользователях, группах безопасности, компьютерах и сервисах системы. Она должна быть расширяемой для новых типов объектов и иерархической, с контейнерами, которые могут содержать другие контейнеры или дочерние лист-объекты. Часто в качестве такой базы данных используют какой-либо из имеющихся LDAP-серверов. После продолжительного тестирования и рассмотрения данного вопроса мы тем не менее решили написать свой сервис SysDB со своим протоколом, основанным на ICE.

В качестве опциональных сервисов мы планируем предложить:

- Сервис DHCP. Позволяет задавать некоторые настройки клиентов (в основном, настройки сети) автоматически по сети в момент загрузки.

[2] `http://web.mit.edu/Kerberos/`
[3] `http://www.powerdns.com`

- Сервис предоставления доступа к файлам и директориям по сети (с помощью NFSv4).
- Почтовый сервис. Позволяет создавать пользователям почтовые ящики, задавать квоты, создавать рассылки, управлять единой адресной книгой и т. п.
- Сервис административных сценариев. Этот сервис был придуман нами как решение проблемы автоматической настройки и перенастройки рабочих станций по информации с сервера. Административные сценарии выполняются на клиенте и могут производить произвольные действия. Поведение каждого сценария может быть изменено его параметрами. Набор сценариев и их параметров мы называем политикой. Администратор может создать необходимое число политик и применить их к отдельным пользователям, компьютерам, их группам или сразу к веткам в иерархии SysDB. Это позволит гибко настраивать рабочие станции, как сами по себе, так и в зависимости от пользователя, вошедшего в систему. Сервис будет поставляться с большим количеством готовых сценариев, которые будут способны настраивать пользовательский интерфейс, устанавливать дополнительное программное обеспечение, монтировать домашний каталог пользователя или работать с перемещаемым профилем и многое другое. Мы планируем использовать этот механизм также в качестве замены политик безопасности, используемых в Microsoft Active Directory. Если же администратор все-таки не нашёл сценария, который выполняет требующиеся ему действия, он может написать новый сценарий сам, используя для этого произвольный язык программирования.
- Сервисы для управления доступом к сети Интернет и учёта его использования. Это набор сервисов, которые будут управлять работой межсетевого экрана, proxy-сервера, программного обеспечения для учёта трафика таким образом, чтобы максимально упростить администратору решение всех задач данной тематики.

Первоначальная установка производится мастером, который, задав несколько простых вопросов (доменное имя, логин и пароль первого администратора, некоторые детали настройки сети), настраивает и запускает все базовые компоненты на текущем компьютере. В дальнейшем они могут быть перенесены на различные компьютеры для

балансировки нагрузки. Дополнительные сервисы просто устанавливаются и запускаются на произвольном компьютере, зарегистрированном в SysDB, то есть настроенном на работу в системе или на самом сервере. При этом автоматически происходит регистрация новых сервисов в SysDB.

Настройка клиентских компьютеров производится мастером, которому указывается доменное имя системы, к которой мы подключаемся, а так же логин и пароль администратора, который имеет права на все необходимые действия.

Компьютер, подключённый к системе, настраивается таким образом, чтобы пользователи, зарегистрированные в SysDB, могли работать на нём. Он также получает возможность самостоятельно аутентифицироваться и использовать сервисы Tartarus, в частности, для автоматического создания соответствующей ему записи на сервере DNS, получения информации о назначенных политиках и пр. Всё это выполняется различными клиентскими компонентами Tartatus, которые должны быть установлены при регистрации.

Администрирование системы может совешаться с любого компьютера. Для этого мы разработали единый графический интерфейс администрирования, который расширяется модулями, написанными на языке Python. Некоторые модули осуществляют управление сервисом какого-либо одного типа, например, модуль по управлению DNS. Большинство же воздействуют сразу на несколько. Так, создавая пользователя, мы делаем запись в SysDB и, одновременно задавая ему пароль, создаём соответствующую запись в KDC. Всё это происходит прозрачно для администратора. Интерфейс написан с использованием популярной библиотеки Qt4. Внешний вид с подгруженными модулями по управлению базой системных объектов и сервером DNS показан на рисунке 1.

Кроме того, всеми функциями системы можно воспользоваться из языка программирования C++, Python, Ruby и PHP, что позволяет администратору автоматизировать процесс управления системой. Для администраторов, которые привыкли использовать bash в своей работе, мы предоставляем консольные утилиты администрирования.

Тем, кому не достаточно наших сервисов, могут разработать собственные. Мы старались максимально упростить написание расширений для нашей системы. Для примера, ниже приведена реализация минимального сервиса Hello. Сервис предоставляет функцию

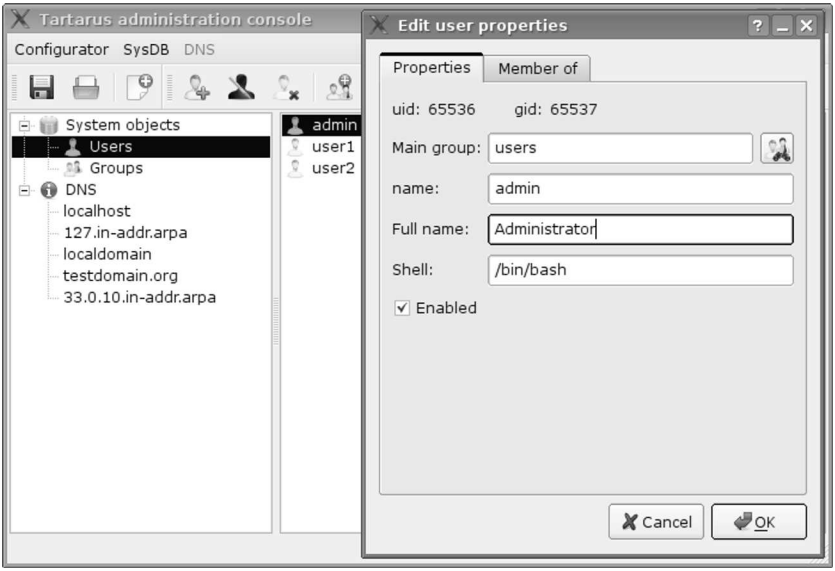

Рис. 1: Внешний вид Tartarus

getGreeting, которая получает имя пользователя, проверяет, совпадает ли оно с тем, что говорит Kerberos, и приветствует обратившегося.

Для этого сначала описываем наш интерфейс:

```
module Tartarus { module iface {
module Hello {
interface Greet
{
 /**
 * Returns personal greeting for person named /name/
 **/
 string getGreeting(string name);
};
};
};};
```

А затем реализуем его на Python (поддерживается реализация на C++, но она чуть более многословна):

```
# Hello servant implementation
import Ice, IceSSL, Tartarus
import Tartarus.iface.Hello as I
class GreetI(I.Greet):
    def getGreeting(self,name,ctx):
        princ = IceSSL.getConnectionInfo(ctx.con).krb5Princ
        # princ has format name@REALM
        n = princ[:princ.rfind('@')]
        if name == n
            return "Hello, %s!" % name
        else:
            return "You are %s, not %s!" % (n, name)
```

Это всё, что необходимо для нового сервиса. Использовать полученный сервис столь же просто.

На сегодняшний день нами уже реализованы все базовые сервисы, а также некоторые компоненты поддержки системы со стороны клиента. Уже сейчас доступны такие возможности, как вход сетевого пользователя в систему и автоматическая регистрация компьютера в DNS. Реализованы мастера для разворачивания базовой системы и подключения клиентских компьютеров к ней. Для всех завершённых сервисов имеются полнофункциональные графические инструменты администрирования.

Реализация дополнительных сервисов планируется к концу этого года.

Егор Гребнев
Москва, ALT Linux

Государственные НИОКР в области ПО с открытым кодом в Евросоюзе

Если в США участие государственного сектора в разработке свободного ПО сводится преимущественно к университетским разработкам и единичным государственным заказам, то в Европе ПО с открытым кодом с начала 2000-х гг. является важным направлением государственных исследовательских программ, и на НИОКР в этой области выделяются миллионы евро. Результативность государственных вложений в развитие ПО с открытым кодом в Европе не бесспорна. Несмотря на то, что некоторые проекты сумели получить довольно большой общественный резонанс (прежде всего, проекты из серии FLOSS* под руководством нидерландского исследователя Ришаба Гоша), большая их часть, по-видимому, не представляет большого интереса ни для пользователей свободных программ, ни для самих разработчиков. С прекращением финансирования сайты перестают развиваться, а репозитории исходных текстов перестают обновляться.

Разработка свободного ПО как элемент государственной политики

Государственные проекты по разработке свободного ПО — не новое явление, но нынешняя политика стран Европы и Латинской Америки по государственной поддержке разработки свободных программ обладает важной особенностью: государство стремится поддерживать свободное ПО как *направление*, в отрыве от конкретных проектов, считая это направление перспективным для национальной экономики и политических интересов государства.

В Европе государственные НИОКР в области свободного ПО имеют наиболее длительную историю, бюджетные траты наиболее высоки, а количество проектов представляет наиболее полный материал для обобщений. В масштабах Евросоюза разработка свободных программ была впервые акцентирована в рамках Пятой (1998–2002), Ше-

стой (2002–2006) и Седьмой (2007–2013) рамочных программах научно-исследовательских проектов[1].

Успех и неуспех государственных НИОКР

В Пятой рамочной программе было 29 проектов, связанных со свободным ПО. Из них только два проекта в долгосрочной перспективе оказались жизнеспособными: OPENECG (бюджет 458 тыс. евро), посвящённый созданию открытых стандартов и обеспечению межсистемной совместимости в области ПО по созданию электрокардиограмм, и OROCOS (бюджет 215 тыс. евро), посвящённый созданию инструментария для приложений по управлению робототехникой. Сайты этих проектов продолжают обновляться и развиваться, в то время как остальные либо были «заморожены», либо закрылись. Это не значит, что проекты не принесли результатов: например, проект ACEOS (бюджет 240 тыс. евро) по портированию Linux на архитектуру TriCore позволил разработчикам в дальнейшем коммерциализовать выполненную работу. Однако этот проект стоит несколько особняком, поскольку его задача, будучи раз выполненной, не требовала существенных затрат в дальнейшем. В большинстве случаев, заимствовав внешние атрибуты свободных проектов (наличие списка рассылки, раздел загрузки с текущими релизами ПО), проекты Еврокомиссии, связанные с выпуском свободного ПО, не сумели создать жизнеспособных сообществ, без которых эти атрибуты во многом лишены смысла.

Не все проекты в рамках Рамочных программ направлены на создание конечного ПО. Исследовательские проекты FLOSS (бюджет 371 тыс. евро), FLOSSPOLS (бюджет 450 тыс. евро), выполненные под руководством нидерландского исследователя Ришаба Гоша, сыграли большую роль в восполнении информационного пробела по вопросам использования свободного ПО в государственных учреждениях. Косвенным подтверждением значимости этих проектов является то, что предложенная в них аббревиатура FLOSS (Free/Libre and Open Source Software) получила широкое распространение как один из терминов для совокупного обозначения свободного ПО и ПО с открытым кодом.

[1] Список проектов по 5-й и 6-й программам: `http://ec.europa.eu/information_society/activities/opensource/european_activities/index_en.htm`

В настоящее время в рамках Шестой программы ведутся целых три проекта, посвящённых качеству свободного ПО: SQO-OSS, QUALOSS и QUALIPSO с очень внушительными бюджетами: 1,64 млн. евро, 2,05 млн. евро и 10,42 млн. евро. Тем не менее, несмотря на величину затрат, публичные веб-сайты этих проектов вызывают, скорее, скромные ожидания в отношении их результатов.

Итоги и перспективы

Ошибки организации НИОКР Еврокомиссии в области свободного ПО можно усматривать в нескольких факторах:

1. НИОКР Евросоюза направлены на поддержку новых, ещё не доказавших своей коммерческой перспективности проектов — и большая часть этих проектов после получения финансирования в полном масштабе остаётся по-прежнему бесперспективной;
2. правила организации исследовательских программ Еврокомиссии не способствуют ротации исполнителей: поскольку одни и те же лица могут одновременно являться участниками и экспертами, то эти участники имеют возможность содействовать друг другу в получении всё новых проектов, представляющих всё меньшую ценность за пределами сформировавшегося «сообщества».

Хорошо известный пример более прозрачной системы финансирования проектов по разработке свободного ПО, нацеленной на предоставление помощи проектам, уже доказавшим свою перспективность, — это Google Summer of Code. Возможно, в будущем преимущества этой более открытой модели будут заимствованы и государствами.

Однако пока что о выработке эффективной модели государственного финансирования НИОКР в области свободного ПО говорить рано — как на общеевропейском уровне, так и в отдельных государствах. И если рабочая группа по свободному ПО в рамках парижского полюса компетенции System@tic[2], благодаря механизму тщательного отбора перспективных проектов и большому числу компетентных разработчиков свободного ПО в парижском регионе внушает надежды, то итальянская инициатива по финансированию разработки свободных программ в размере 10 млн. евро (детали которой так и не были

[2] `http://www.systematic-paris-region.org`

в полной мере раскрыты)[3] скорее представляет пример нерациональной траты бюджетных средств.

Вартан Хачатуров
Санкт-Петербург, СПбГУ, Экономический факультет

Свободное и открытое ПО: экономический взгляд на явление

Доклад посвящён обзору работ в области экономической теории, содержащих попытки экономического анализа FOSS как явления общественной жизни, а также его влияния на различные характеристики экономики, такие как экономический рост и темпы технологического прогресса.

Будут кратко рассмотрены работы в области микроэкономики, посвящённые анализу рынка программного обеспечения, монопольной власти на нём и влияния открытого ПО, последние работы в области макроэкономики — о влиянии FOSS на разделение труда и, через отдачу труда, на экономический рост.

Также мы рассмотрим некоторые из взглядов экономистов-представителей неоинституционализма на правовые и экономические аспекты свободных лицензий.

Литература

[1] *Schmalensee, R.* Antitrust issues in schumpeterian industries / R. Schmalensee // *The American Economic Review.* — 2000. — Vol. 90, no. 2.

[2] *Gilbert, R. J.* An economist's guide to u.s. v.ricmic microsoft / R. J. Gilbert, M. L. Katz // *The Journal of Economic Perspectives.* — 2001. — Vol. 15, no. 2.

[3] *Romer, P. M.* Endogenuos technological change / P. M. Romer // *The Journal Of Political Economy.* — 1990. — Vol. 98, no. 2.

[4] *Lerner, J.* Some simple economics of opensource / J. Lerner, J. Tirole // *The Journal of Industrial Economics.* — 2002. — Vol. 50, no. 2.

[3] `http://ec.europa.eu/idabc/en/document/7391/502`

[5] *Lerner, J.* The economics of technology sharing: Opensource and beyond / J. Lerner, J. Tirole // *The Journal of Economic Perspectives.* — 2005. — Vol. 19, no. 2.

[6] *Varian, H. R.* Copying and copyright / H. R. Varian // *The Journal of Economic Perspectives.* — 2005. — Vol. 19, no. 2.

[7] *Stewart, D.* Social status in an opensource community / D. Stewart // *American Sociological Review.* — 2005. — Vol. 70, no. 5.

[8] *Benkler, Y.* Coase's penguin or linux and the nature of firm / Y. Benkler // *The Yale Law Journal.* — 2002. — Vol. 112.

Анатолий Якушин
Москва, ALT Linux

Кого накормит волшебный котёл? Экономические аспекты свободного программного обеспечения

В докладе делается попытка определить место свободного программного обеспечения в современных экономических воззрениях на производство и реализацию программных продуктов, как по результатам анализа данных в существующей литературе, так и путём практического исследования существующих программных комплексов; а также исследуются современные методики количественной и качественной оценки эффективности информационных систем и свободное ПО для их исчисления.

Актуальность проблемы

Интерес к свободному программному обеспечению породил в последнее время значительное количество публикаций, связанных с вопросами построения бизнеса на основе СПО. Данные публикации отличаются весьма широким спектром мнений, от отказа свободному программному обеспечению в праве на участие в современном рынке производства программных продуктов и противопоставления его «коммерческому ПО», до приписывания СПО магических свойств «волшебного котла», создающего прибыль уже одними свойствами свободы[1]. Отсутствие единого взгляда на проблему заставило ав-

тора произвести попытку анализа существующих стратегий и тактик производства и распространения программного обеспечения.

Материалы и методы

В ходе исследования были рассмотрены 23 крупные информационные системы, созданные на основе как свободных, так и проприетарных продуктов. Ко всем информационным системам применялись комплексные методы анализа, включающие в себя моделирование архитектуры ИС, исследование жизненного цикла программного продукта, исчисление совокупной стоимости владения ИС. Для исчисления TCO использовалась свободная программа tcotool [2]. Кроме этого, анализировались доступные из открытых источников бизнес-стратегии ведущих производителей программного обеспечения.

Результаты

После проведённого анализа было выявлено несколько закономерностей:

- С экономической точки зрения не отмечено никаких принципиальных различий в архитектуре и особенностях жизненного цикла программных продуктов в зависимости от характера лицензионного договора.
- Совокупная стоимость владения информационными системами, созданными на основе продуктов с проприетарным и свободными лицензионными договорами, при прочих равных условиях различается только на стоимость, затраченную на приобретение лицензии.
- Совокупная стоимость владения информационной системой, основанной на программных продуктах со свободными лицензиями, снижается при её тиражировании; тогда как ССВ проприетарной информационной системы при тиражировании возрастает.
- В случае государственного заказа информационной системы, рыночная модель её финансирования в большинстве случаев оказывается недостаточной. Для создания устойчивой государственной информационной системы необходимо дополнительное финансирование по принципу покупки «общественного блага».

В целом, можно согласиться с положением о свободном и несвободном коммерческом программном обеспечении, приведённым в [3]. Однако в процессе анализа бизнес-стратегий различных производителей ПО было отмечено следующее правило. Практически все успешные стратегии свободного ПО основывались исключительно на продаже услуг по адаптации, установке и поддержке, а собственно программный продукт разрабатывался за рамками коммерческого контекста; тогда как для проприетарного ПО более характерна компромиссная модель, основанная как на продаже лицензий, так и на оказании услуг.

Литература

1. Eric Steven Raymond «The Magic Cauldron» `http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/`
2. TCOTOOL `http://www.tcotool.org/`
3. М. Отставнов «Перспективы свободного программного обеспечения в сфере государственного управления и бюджетном секторе экономики».

Евгений Чичкарёв
г. Мариуполь, Приазовский государственный технический университет

Моделирование физических полей: особенности разработки и использования OpenSource-приложений

Обзор программного обеспечения для решения уравнений в частных производных — расчёт теплового поля, электромагнитного поля, поля напряжений, поля скоростей и т. п. Обсуждаются возможности пакетов типа FreeFEM, а также разработка соответствующих приложений в среде Octave, Scilab, SciPy; возможности символьного решения в среде Maxima. Представлены примеры решения научных, технических и учебных задач, краткий анализ необходимых алгоритмов.

ОС Linux — удобная платформа для численных расчётов, которые необходимы во многих инженерных и научных областях (в гидродинамике, теории упругости, теплопроводности, электромагнетизме и других). С математической стороны расчёт сводится к построению того

или иного поля — поля концентраций, скоростей, температур и др. — и решению дифференциального уравнения или системы в частных производных. В большинстве случаев применяются стандартные методы — конечных элементов (наиболее часто), конечных объёмов, конечных разностей.

Для разработки пакетов используются почти исключительно объектно-ориентированные языки: C++, Java, Python. Все пакеты базируются на ряде алгоритмов и библиотек, а именно:

- базовые процедуры работы с векторами и матрицами (BLAS);
- базовые алгоритмы линейной алгебры (LAPACK);
- оптимизированная по скорости выполнения численных расчётов библиотека ATLAS.

Для специфичных задач решения систем линейных уравнений, формирующихся при решении уравнений с частными производными, используют более специализированные библиотеки: UMFPACK; SuperLU; TAUCS (библиотека включает прямые и итерационные алгоритмы); PETSc (возможность распараллеливания по MPI, связь с python); Aztec (библиотека для параллельного итерационного решения линейных систем, эффективная, с доступным исходным кодом, но несвободной лицензией); TNT (библиотека для программ на C++).

Все программы для работы с уравнениями в частных производных используют и разреженные матрицы, т. к. обязательно возникает необходимость решать системы линейных уравнений с очень большим числом переменных, но при этом соответствующие матрицы имеют большое количество нулевых элементов, т. е. являются разреженными. Использование специальных библиотек позволяет хранить в памяти только ненулевые элементы и значительно ускорить процесс решения. В частности, UMFPACK, SuperLU, PETSc включают компоненты для работы с разреженными матрицами. Для работы на Python разработан пакет Python-sparse (есть в Debian, Ubuntu). Разработано и довольно большое количество библиотек и компонентов для матричных расчётов на Java (Jama и т. п.).

Для решения задач, связанных с решением систем уравнений в частных производных, известные пакеты включают библиотеку компонентов, набор скриптов для сборки и макроязык, позволяющий описать решаемую задачу (перечень известных пакетов с краткой характеристикой приведён в докладе). Для решения широкого круга задач, связанных именно с решением уравнений в частных производ-

ных, предназначены достаточно гибкие, документированные и простые в использовании FreeFEM++/FreeFEM3D, GetDP, Gerris. При работе с python/scipy удобным выбором является FiPy (библиотека для Python, отличается удобной записью задачи, приближенная к математической записи в сильной форме; получается простой и краткий код, решает задачи типа реакция—диффузия—конвекция в 1–2–3D), либо GetFEM++ (имеется интерфейс для python или MatLab).

Все пакеты универсального назначения для численных расчётов — Octave, SciLab, SciPy — включают функции для операций с разреженными матрицами и решения соответствующих систем линейных уравнений. Для MatLab, SciLab разработан пакет расширений OpenFem, предназначенный для решения разнообразных задач методом конечных элементов. Как показало тестирование методов решения ряда задач тепломассообмена (т. е. параболических), для областей 1D и 2D с не слишком сложной геометрией в среде Scilab, Octave, SciPy целесообразно использование метода линий. Для эллиптических задач эффективно использование разреженных матриц с соответствующими солверами (например, для SciLab — методы сопряжённых градиентов, минимальных невязок).

Для решения гидродинамических задач (уравнений типа Навье—Стокса и т. п.) обычно используется специализированные пакеты (gerris, Dolfin/FEniCS, OpenFOAM, FeatFlow и др.). Однако для достаточно простых задач (двумерный несжимаемый поток) возможно и решение в рамках SciLab и Octave. В докладе представлены примеры расчётов, проанализированы возможности визуализации результатов более сложных расчётов.

Дмитрий Сподарец
Одесса, Одесский национальный университет им. И. И. Мечникова, [Root@UA]Media

Проект: Аппаратно-программный комплекс TERM
`http://www.term.rootuamedia.org.ua`

Разработка аппаратно-программного комплекса TERM-1 по измерению температур быстропротекающих процессов на основе открытого программного обеспечения

Проведение экспериментальных исследований сопровождается измерениями физических величин или их аналогов в виде электрических сигналов. Обработка полученных данных — процесс громоздкий. Например, при использовании термопары регистрируется термоэдс для получения конечных результатов. В таком случае необходимо проводить градуировку термопары и затем обрабатывать нужные нам данные. Именно поэтому представляет собой интерес автоматизация процесса обработки.

В теплофизических процессах одним из основных параметров исследования является температура. Поэтому была поставлена задача: создать быстродействующее устройство, измеряющее температуру в реальном времени и подключаемое к компьютеру, а также создание программного обеспечения, отображающего цифровые значения температуры и график зависимости температуры от времени.

Существует большое количество аналогичных приборов, но все они или недостаточно функциональны, или попросту дорогие.

Поэтому, основываясь на ОС Linux и языке C++ с применением библиотек QT4 и MathGL, была предложена разработка аппаратно-программного комплекса по измерению температур. Свободное программное обеспечение было выбрано по причине повышенной надёжности, мобильности, открытости кода и возможности дальнейшего развития путём сотрудничества с сообществом.

Универсальность комплекса обеспечивается путём расширения модульной структуры.

На первом этапе был разработан аппаратный модуль TERM-1 с программным комплексом, в основу которого заложено ядро, взаимодействующее с тремя программными модулями. Первый служит

для получения данных со звуковой карты, второй — с изображения, сделанного при помощи цифрового фотоаппарата (для определения яркостной температуры тела), третий — для проведения градуировки и отладки всего комплекса.

TERM-1 представляет собой преобразователь «напряжение — частота», на входе которого стоит операционный усилитель.

Основной принцип работы комплекса следующий. При нагреве термопары генерируется напряжение и подаётся на TERM-1, где первоначально усиливается, а затем преобразуется в импульсный сигнал, частота которого линейно зависит от входящего напряжения. Далее импульсы подаются на звуковую карту. На последнем этапе модуль программы считывает со звуковой карты напряжение и подсчитывает количество пришедших импульсов за определённый промежуток времени. Далее ядро преобразует полученное количество импульсов в численное значение температуры, строит график временного изменения температуры.

Комплекс удачно прошёл испытания при измерении температур горения различных газов.

Планируется работа по усовершенствованию метода измерения яркостной температуры тела.

Михаил Быков
Москва

Проект: Warehouse
`http://demo.store.rinet.net`
`git://store.rinet.net/~michael/repo/warehouse.gitmaster`

Приложение «Склад интернет-провайдера» на Ruby-on-Rails

ООО «Кроникс плюс» (торговая марка «Ринет») существует на интернет-рынке с доисторических времён. Это осколок Большого Взрыва, породившего Релком, Демос и бесчисленное множество продолжателей интернет-провайдинга на всей территории бывшего СССР. Неудивительно, что в нём существует очень благожелательная атмосфера по отношению к решениям на основе программного обеспечения со свободными лицензиями — просто потому, что к этому

привыкло ядро компании. В последние годы с серверов СПО уверенно распространяется на рабочие станции. Это происходит само собой, незаметно и ежедневно. В «курилках» обсуждают, как поставить Ubuntu на eeePC или как запустить игрушку под wine. В бухгалтерии — точнее, в управлении сетью компании, в её финансовом планировании решения на основе СПО очень быстро заменяют коммерческие решения. Это происходит постепенно и естественно.

Когда нужды бизнеса потребовали внедрения CRM, руководство компании изначально заняло позицию, что не будет внедрять проприетарные решения. Изначально примитивные CRM задачи решались с помощью самописных текстовых скриптов + EXCEL.

Сначала вариант программы «Склад» был написан, как у всех, путём модификации стандартного компонента. Он работал успешно. Склад был выбран как первый элемент финансовой системы, причём предполагалось, что он будет написан достаточно быстро, так как задача, во-первых, простая, а во вторых, есть работающий прототип, соответственно ТЗ имеет конечную размерность.

Так появилась идея постепенно, шаг за шагом, замещать блоки учёта и анализа с решений на основе 1С на решения на основе веб-технологий. Эта идея в Ринете и воплощается, и приложение, рассматриваемое в настоящем докладе, является частью этого «заговора против мирового зла».

В качестве платформы для разработки приложения «Склад интернет-провайдера» была выбрана Ruby-on-Rails. Это платформа, которая позволяет даже весьма среднему программисту быстро писать читаемый, поддерживаемый и масштабируемый код и его без напряжения развивать и надстраивать. Хотя теоретическая задача «Склад», вообще говоря, предельна проста, при использовании PHP через год работы код, вероятнее всего, стал бы совершенно нечитаемым. И разработка бы остановилась или существенно затруднилась. Приложение на Ruby всё время просит упрощения, совершенствования кода. Вдобавок, приложение написано в REST-стиле, что позволяет легко связывать его с другими приложениями, разбивать на части и как угодно масштабировать.

Что такое Склад, говоря вообще? Это Учёт и Контроль. Есть список материалов, когда-либо поступивших на склад. Это модель материалов Asset (Модель—Контроллер—представление). Модель Материалы-Asset — это, по сути, просто список названий материалов. Каждому названию соответствует множество приходов — модель Income,

и расходов, или выдач — Yield. Есть модель Объектов сети — по историческим причинам она называется LoginRoot, и модель персонала — Person. Материал asset (здесь с маленькой буквы, потому что здесь это не класс Asset, а его объект, то есть конкретный материал) поступает на склад. Создаётся приход income, в котором хранятся дата, поставщик, цена, количество, идентификатор кладовщика и пр. Количество материала на складе увеличивается на величину этого прихода. Затем монтажник (основная Роль в Персонале) создаёт заказ-требование выдать ему этот материал. Материал ему выдаётся. При этом количество материала на складе уменьшается на соответствующую величину. А в списке материалов монтажника появляется то же количество этого материала. На следующем шаге монтажник списывает с себя некоторое количество материала на Объект Сети. В результате, мы может посчитать, сколько стоит данный объект сети в каждый момент времени и вся сеть вообще. Вот как бы каркас всей архитектуры приложения. Теперь несколько уточнений, способных сильно запутать эту очень простую картинку.

Складов может быть несколько, и материалы можно передавать со склада на склад. В частности, есть склад Ремонт. Монтажники могут не только списывать материалы на объекты сети, но и передавать их друг другу. Материалы могут исчезать и появляться ниоткуда — это необъяснимое изменение количества в записях базы данных носит учёное название «инвентаризация». Инвентаризации могут подвергаться как материалы на складе, так и материалы у монтажника. Сложность здесь в том, что, в результате. количество затраченных средств и количество средств, учтённых в разного рода расходах, должны в точности и до копейки друг другу равняться.

Во-вторых, как оприходованные материальные ценности соответствуют выданным? Вводится понятие текущей партии материала. Это первый (самый ранний по времени из ещё не выданных) приход — income. По мере того, как материал выдаётся со склада, текущее количество в текущей партии уменьшается. Текущей ценой является цена единицы материала этой партии. Как только текущее количество становится равным нулю, текущей партией становится следующая партия со своей, как правило большей, ценой.

В-третьих, некоторые материалы являются «более материальными», чем другие. Они имеют инвентарные номера. Только они и списываются на объекты сети. Это могут быть инструменты или ценное оборудование. Менее ценные материалы пренебрежительно назы-

ваются в силу своей малой ценности «малоценкой» и списываются не на объекты сети, а сразу в момент их выдачи монтажнику. Инвентарные номера в момент выдачи можно вводить как вручную, так и с помощью лазерного считывателя. А в момент списания на объект сети монтажник не может воспользоваться лазерным считывателем, даже если бы он был у каждого монтажника. Материал находится в этот момент уже на объекте сети, а монтажник — в офисе компании. Эта трудность будет решена только в будущем, но уже относительно недалёком — при большем распространении и удешевлении мобильных устройств, когда монтажник сможет заполнять отчёт о выполнении работы непосредственно в момент подключения оборудования. К сожалению, между инвентарным номером, например, роутера и его мак-адресом в настоящее время в нашем приложении однозначного соответствия нет, так что автоматизировать процесс списания оборудования на объект сети нельзя.

Здесь перечислены только самые необходимые для понимания работы приложения элементы его архитектуры. Суть дела видится не в фиксации приходов и расходов склада, это тривиальная задача. Суть дела — в возможности пересчёта всего дерева записей при внесении исправлений задним числом, при сохранении целостности данных и корректности их для целей бухгалтерского учёта.

К недостаткам приложения следует отнести (помимо весьма среднего пока что качества кода) использование для отрисовки бесконечного количества разнообразных табличек плагина activescaffold. Это слишком мощный и сложный для данной цели плагин. Здесь было бы хорошо использовать нечто гораздо более простое. Для отрисовки каждой таблички activescaffold требует создания своего контроллера. Большое количество вспомогательных контроллеров очень загрязняет код. Этот плагин будет заменён на более подходящий в будущих версиях приложения. Не реализовано также пока (за ненадобностью) архивирование устаревших данных.

Пара слов о связи с остальными частями инфраструктуры компании. Приложение считывает «снаружи» структуру подразделения компании, список её сотрудников и список объектов сети. Это реализовано пока простым считыванием от соседей их данных в формате XML. Соседние части инфраструктуры не могут работать в REST-стиле, поэтому считывание данных впоследствии можно будет реализовать, используя протокол SOAP. Но поскольку речь идёт именно о периодическом считывании, а не об активном обмене, то сейчас

SOAP представляется излишним. Простое считывание данных в формате XML прозрачней, а в будущем, может быть, удастся использовать REST.

В системе реализована центральная аутентификация, т. е. single-sine-on: пользователь регистрируется на привычном ему сервере, который прописывает ему специального вида cookie с полем domain равным имени нашего общего домена, .rinet.net. А авторизация реализована так — пользователям в момент считывания списка добавляются уже в самом приложении определённые роли, а именно «кладовщик», «администратор», «монтажник». Пользователь получает доступ к тем возможностям, которые диктуются его ролью. Это неуклюже, но поскольку сервер, хранящий данные о персонале, не знает о роли пользователя на складе, то пока иного выхода нет. В будущем роли будут также распределяться унифицировано.

На нашем опыте отчётливо видно, что написание отдельных приложений, выполняющих свои задачи, в наше время не проблема. Проблема в архитектуре системы в целом. Отдельные отлаженные компоненты и удобные плагины, существующие в настоящее время, не рассчитаны на работу в системе приложений. У нас нет опыта построения распределённых систем, нет опыта разработки структуры данных всего предприятия в целом. Как только требуется сопряжение компонентов системы, код становится неуклюжим, написанным по конкретному случаю, а работа системы ненадёжной. Требуется, во-первых, целый слой готовых компонентов для построения распределённых систем, а во-вторых, открытая культура такой разработки — образцы, учебники, success-story и так далее. Понятно, почему это так — один работает со своей задачей на PHP, другой — со своей на Ruby, третий — со своей на Java. Вдобавок, нужно заранее увидеть всю структуру данных всего предприятия и внедрить её жёсткой рукой. А это сложно. Если же эта задача будет выполнена, то во всём остальном задачу создания систем на основе современных веб-технологий, полностью замещающих 1С во всей её функциональности, можно оценить как трудоёмкую и дающую массу простора для всякого творчества, но в принципе тривиальную и не имеющую никаких подводных камней. Она будет реализована во многих вариантах в ближайшее же время. И именно так, как это происходит у нас — разнородные веб-приложения, «заточенные» под выполнение узко определённых задач и обменивающиеся данными по протоколам SOAP или/и REST или

аналогам. Немедленно вслед за этим появятся и массовые решения, готовые «из коробки».

Демонстрационную версию программы можно посмотреть по адресу `demo.store.rinet.net`. Код доступен по адресу `git clone: git://store.rinet.net/~michael/repo/warehouse.gitmaster`. Процесс установки и запуска описан в файле `Install`.

Андрей Черепанов
Москва, E/AS Foundation

Проблемы разработки свободного учётного программного обеспечения

Идея создания свободной платформы для учётных задач стоит буквально с самого начала внедрения Linux и свободного программного обеспечения в России. Под учётными задачами будем рассматривать автоматизацию бизнес-процессов, как правило, сводящуюся к автоматизации бухгалтерского, налогового, финансового и оперативного учёта.

Для фирм и организаций уже есть вполне удовлетворительные по качеству и совместимости офисные пакеты, средства коммуникаций и совместной работы. Но при всей наработанной массе свободного программного обеспечения именно отсутствие развитой свободной платформы учётных задач не позволяет обеспечить широкое развёртывание Linux в коммерческих компаниях.

Пока ещё наше государство не готово перейти на принципы упрощённой отчётности для налоговых и статистических органов, принятых в Европе и США. Именно это привело к излишнему усложнению программ автоматизации, которые рассчитаны на нашу сложную систему предоставления отчётности. Исходя из этого есть ряд следствий, касающихся учётного программного обеспечения.

Во-первых, иностранные программы (SAP ERP, Axapta, Compierre и пр.) концептуально не покрывают в своей базовой поставке требования к учётным задачам в России и СНГ. Несмотря на наличие локализации, методика ведения учёта в них сильно отличается от практики осуществления российского учёта и сложившихся привычек бухгалтеров.

Во-вторых, отечественные учётные программы не только успешно конкурируют с мировыми брендами учётного ПО, но и фактически обеспечили их монопольное положение на отечественном рынке.

Исходя их этих факторов, можно предположить, что на рынке учётного ПО (по крайней мере, в СНГ) практически можно осуществить создание продукта, который бы нашёл свою нишу и, возможно, конкурировал бы с аналогами. Это следует, помимо национальной специфики, из природы учётных задач, в которых будет востребованы решения, не опирающиеся на фиксированную логику, а обеспечивающие гибкость доводки и автоматизации. Ведь бизнесы различных фирм сильно отличаются друг от друга и побеждают решения, на которых можно сравнительно быстро и гибко автоматизировать учёт специфических бизнес-процессов.

Ярким примером, подтверждающим этот тезис, является конкурентная борьба между отечественными продуктами 1С:Предприятие и Парус. Несмотря на то, что последний гораздо проще для ведения учёта, но не обладает широкими возможностями по настройке и изменению логики, в отличие от 1С:Предприятия, в котором была сделана ставка не на конечный функционал программы, а на платформу разработки различных решений. Это обеспечило коммерческий успех 1С:Предприятия и фактически монопольное положение его на российском рынке.

Точно такие же сравнения можно было сделать и применительно к кустарному учётному ПО, созданным в 90-х годах прошлого века. На сегодня его распространение практически сошло на нет, оно осталось в узких нишах, для которых адаптация готовых решений на платформе 1С было экономически нецелесообразно (например, страховые компании).

Тем не менее, у 1С:Предприятия (да и у других аналогичных коммерческих продуктов) есть ряд узких моментов, которые были предназначены для формирования широкой сети высокооплачиваемых внедренцев.

- Проприетарное программное обеспечение. Позволяет производителю платформы получать доход за счёт продажи коробок, а не конечных внедрений.

- Монолитная конфигурация. Весьма удобно для продажи коробочных решений и высоких цен на услуги внедренцев, которые фактически переписывают конфигурацию заново.

- Файл-серверная архитектура. Вызвано наработанной массой кода.
- Ориентация на одну платформу. Разработка только под Windows привела к ситуации, когда у проприетарного разработчика нет ресурсов для портирования приложений на другие платформы.

Сегодня существует ряд проектов свободных учётных платформ, доказавших свою жизнеспособность. Прежде всего, это отечественная разработка Ананас, а также успешные западные проекты GNUe и Compiere. Однако и они не получили широкого распространения. Попробуем определить их недостатки:

- Повторение проприетарных аналогов. Считается, что слепое копирование проприетарных аналогов гораздо проще создания своей уникальной разработки. Хотя сходство внешнего вида и функциональности может привлечь пользователей проприетарных аналогов, разработчики становятся заложниками функциональных недостатков и ограничений оригинала, они вынуждены постоянно следовать за коммерческим разработчиком и, как следствие, вечно доделывать свой продукт. Едва ли не каждый год появляется очередной проект, который нацелен на копирование 1С:Предприятия под свободной лицензией.
- Перфекционизм. Часто разработчики свободного ПО стремятся сделать идеальный законченный продукт, что приводит к затягиванию стадии проектирования и подбора инструментария, а также неоправданной задержке выхода готового продукта.
- Амбиции руководства. Является следствием предыдущего пункта. В проекте свободного ПО руководитель является истиной в последней инстанции, на первых порах контролируя все аспекты администрирования проекта и определения его развития. Практически всегда нежелание менять поставленные цели приводит к стагнации развития проекта. Второй причиной амбиций является неверный подбор архитектуры программного обеспечения и нежелание перерабатывать существующую базу кода, которая не позволяет оперативно подключаться к проекту сторонним разработчикам.
- Недостаток свободного времени. Это основной бич проектов свободного программного обеспечения. Самым эффективным ре-

шением этой проблемы является распределение работ на максимально большое количество людей с уменьшением доли в коде каждого из них.

Итак, если найти пути преодоления недостатков, найденных у конкурентов, и использовать сильные стороны их продуктов, то можно создать жизнеспособный проект учётного программного обеспечения.

Попробуем найти пути обхода озвученных выше недостатков проприетарных и свободных проектов учётных платформ.

1. Разрабатываемое программное обеспечение должно создаваться и распространяться под свободной лицензией. Однако критически важно вовлечь в процесс развития платформы разных, в том числе и коммерческих, разработчиков, поэтому необходимо обеспечить достаточную независимость платформы от прикладных решений, с одной стороны, и друг от друга, с другой стороны. В этом случае возможно создание гибридных свободных и проприетарных компонентов, обеспечивающих решение разнообразных задач.
2. Монолитная конфигурация должна быть разделена на любое количество компонентов. Это позволит повысить мобильность решений на рынке при условии даже малого числа заинтересованных лиц.
3. Файл-серверная архитектура должна быть заменена на современное трёхзвенное клиент-серверное решение, состоящее из клиента, сервера приложений и сервера баз данных.
4. Программное обеспечение платформы и прикладных решений должно создаваться под основные платформы: Unix, Windows и Mac OS X. Для этого подобранный инструментарий должен обеспечивать переносимость между указанными платформами.
5. Платформа не должна слепо копировать аналоги, но может и должна использовать эффективные решения, реализованные в продуктах конкурентов.
6. Прикладные решения должны быть отделены от собственно платформы, обеспечивающей их функционирование. Это позволит быстро наращивать функциональность всей системы в целом и зафиксировать API. Кроме того, это позволит использовать лицензии для прикладных решений, отличные от свободной лицензии платформы и базового инструментария.

7. Создание не только платформы, но и инфраструктуры для обмена решениями позволит привлечь больше людей и распределить разработку.

8. Сама платформа должна обеспечивать замену любого функционального блока, чтобы не замыкаться на одной технологии, а использовать навыки программирования на разных языках разных разработчиков. При этом важно обеспечить достаточность и неизменность API для совместимости блоков друг с другом;

9. Наиболее оптимальным выходом из сегодняшней патовой ситуации со свободным учётным программным обеспечением является создание независимого от разных проектов хранилища описания прикладных конфигураций. Это позволит разным проектам получить актуальную и полную базу прикладных решений, которую они могут адаптировать для своих продуктов. Естественно, должны быть определены форматы передачи подобных данных. В идеальном случае любая существующая или новая платформа может получать из этого хранилища любую прикладную конфигурацию. Также возможно создание веб-ориентированных решений по учётным задачам.

Таким образом, при соблюдении последнего условия мы можем рассчитывать на увеличение специализации, при которой разработчики свободных учётных платформ концентрируются на создании качественного продукта, поддерживающего актуальную и широкую базу прикладных решений.

Максим Тюрин, Александр Фейлик
Ровно, Украина, ООО «Кратос»

eКлючи от системы или использование смарт-карт в GNU/Linux

В современном мире существует проблема, доставляющая множество неудобств. Это проблема хранения и запоминания паролей для авторизации.

Одно из решений безопасной авторизации без запоминания множества паролей — использование смарт-карт для авторизации.

В операционной системе GNU/Linux смарт-карты могут использовать:

– клиент OpenSSH для авторизации по ключу;
– PAM для авторизации в системе;
– GnuPG для хранения секретной части ключа;
– продукты Mozilla для работы с сертификатами.

Также можно использовать смарт-карты для работы с шифрованными файловыми системами.

Введение

Жизнь современного человека немыслима без множества электронных устройств, для работы с которыми необходимо помнить пароли и пин-коды. Часто для человека составляет проблему запомнить даже один пароль, удовлетворяющий политике секретности (8–12 знаков, включающих буквы, цифры и спецсимволы). И мало что так выводит человека из равновесия, как забытый важный пароль. Системному администратору в работе необходимо оперировать множеством паролей (часто больше 50 различных паролей) и запомнить такое их количество попросту невозможно.

Что чаще всего используют для решения этой проблемы:

- один пароль на всё;
- записывание паролей в блокнот;
- использование программы менеджера паролей на компьютере или КПК.

Но все эти решения или катастрофически снижают безопасность, или очень неудобны в использовании. Кроме того, нельзя не вспомнить о проблеме сохранности пароля в тайне. Очень часто пользователи сообщают друг другу свои пароли.

Для решения подобных проблем можно использовать несколько решений. Например:

1. Использование биометрических данных, в частности сканера отпечатков пальцев;
2. Использование OTP (One Time Password);
3. Использование смарт-карт.

Каждое из этих решений имеет свои преимущества и недостатки. Решающими факторами для нас стали повышенная безопасность,

возможность хранения различных ключей для нескольких задач и относительная дешевизна решения. А также — это должно было быть доступным решением, представленным на украинском рынке. Поэтому мы выбрали использование Смарт-карт, а именно Aladdin eToken Pro.

Aladdin eToken Pro — это смарт карта в формате USB-брелока. Сделана на чипе Infineon и работает под управлением Siemens CardOS M4. Устройство имеет аппаратную поддержку алгоритмов RSA, DES, TripleDES, SHA-1, MAC. Также есть возможность аппаратной генерации ключей RSA.

Поддержка программным обеспечением

В OS GNU/Linux есть два варианта работы с Aladdin eToken PRO. Первый — использование совместимого с интерфейсом Windows(R) SCard PC/SC Lite. Второй — использование реализации стандарта PKCS#15 OpenSC.

Для использовании в GNU/Linux мы выбрали второй вариант из-за лучшей поддержки программным обеспечением, но пришлось решить проблему совместимости с Windows(R) — так как используются различные протоколы. Проблема решилась достаточно просто — дублированием сертификатов. При необходимости сертификат заливался в eToken в двух экземплярах: в GNU/Linux — с помощью OpenSC, и в Microsofs Windows — с помощью ПО от производителя.

OpenSSH

В OpenSSH существует два вида авторизации:

- по паролю;
- по ключу.

У нас используется авторизация по ключу как более безопасная. Но сразу возникает проблема с хранением секретной части ключа. Если хранить на жёстком диске — будешь привязан к одному рабочему месту, если хранить на «флешке» — ключ будет очень слабо защищён. Если в своей машине ещё можно быть уверенным, то на чужой машине нельзя.

eToken для хранения ключа ssh подошёл очень хорошо. Из eToken нельзя «вытянуть» закрытую часть ключа без сложного и дорогосто-

ящего взлома самого устройства; да и потеря устройства будет не так опасна для безопасности, как потеря «флешки» с ключом ssh.

PAM

Существует два модуля PAM для работы со смарт-картами.

Pam P11 — простой модуль, не имеющий конфигурационного файла. Для авторизации с помощью этого модуля необходимо просто поместить открытую часть ключа или сертификата, хранящегося в смарт-карте, в домашний каталог пользователя.

Pam PKCS#11 — полнофункциональный модуль с возможностью верификации данных из смарт-карты с помощью цепочки сертификатов, списка отзыва сертификатов (CRL), LDAP, Active Directory, Kerberos; также включает менеджер событий, в котором можно указать действия, выполняемые при извлечении или вставке смарт-карты.

GnuPG

GnuPG умеет работать только с одним типом смарт-карт — OpenPGP card `http://g10code.com/p-card.html`, но существует проект gnupg-pkcs11-scd, предоставляющий из себя GnuPG-совместимый демон с поддержкой стандарта PKCS#11. С помощью этого демона GnuPG может работать с другими смарт-картами.

Mozilla

Продукты от mozilla.org и производные от них умеют работать со смарт-картами, поддерживающими стандарт PKCS#11. Например: сертификаты, хранящиеся в смарт-карте, могут использоваться для авторизации на WEB или почтовых серверах.

Шифрованные файловые системы

Так как существует возможность расшифровывать небольшой блок данных с помощью смарт-карты с выдачей результата в stdin, то это можно использовать для работы с шифрованными файловыми системами.

Для этого берётся блок случайных данных, шифруется открытой частью ключа, хранящегося в смарт-карте, и сохраняется на диске.

После чего с помощью небольшой обвязки на shell или скриптовом языке этот блок данных используется в качестве пароля для шифрованной файловой системы.

Литература

[1] *OpenSC* Свободный проект, состоящий из нескольких библиотек и утилит, реализующих стандарты PKCS#11 и PKCS#15 для работы со смарт-картами, также включает в себя модули PAM, использующие смарт-карты. `http://www.opensc-project.org`

[2] *OpenSSH* Проект свободной реализации SSH. `http://www.openssh.org`

[3] *GnuPG* Проект свободной реализации стандарта OpenPGP. `http://gnupg.org`

[4] *gnupg-pkcs11-scd* Свободный демон для работы GnuPG со смарт-картами. `http://gnupg-pkcs11.sourceforge.net`

[5] *Александр Похабов* Железный login: ломаем зубы грубой силе. `http://www.samag.ru/cgi-bin/go.pl?q=articles;n=12.2004;a=11`

Алексей Куклин
Москва, SUDO.su

Резервное копирование в реальной жизни — анализ задачи, организационные и технические аспекты, существующие открытые решения

В докладе приводится краткий системный анализ задачи резервного копирования, освещаются основные организационные и технические аспекты в рамках существующих информационных систем. Целью доклада является информирование слушателей о существующих технических решениях, доступных при ограниченном бюджете и организационно-административных мерах, призванных повысить эффективность процесса резервного копирования и доступность резервированных материалов как для конечных пользователей, так и для системных администраторов.

Термины и определения

Резервное копирование — процесс копирования информации на резервные носители для обеспечения возможности восстановления в случае сбоя.

Пользовательские данные — информация, хранящаяся в информационной системе, представляющая ценность для пользователя.

Служебные данные — информация, хранящаяся в информационной системе, используемая или создаваемая ИС в процессе функционирования.

Первичные данные — данные, которые невозможно получить из каких-либо других доступных данных.

Вторичные данные — данные, которые можно получить из каких-либо других доступных при помощи некоторого процесса.

Резервное копирование уникальных данных — РК, при котором гарантируется сохранность первичных пользовательских данных, но не регламентировано время восстановления рабочего места/информационной системы после сбоя.

Резервное копирование рабочего места/информационной системы — РК, при котором гарантируется как сохранность первичных пользовательских данных, так и максимальный срок восстановления работоспособности РМ/ИС.

Точка отката — момент времени, после которого в случае восстановления из РК изменения в данных будут потеряны.

Задача резервного копирования

Резервное копирование преследует, как правило, одну либо некоторые из следующих целей:

- сохранность первичных данных;
- возможность восстановления функционирования рабочего места/информационной системы;
- возможность откатиться к версии данных на определённый момент времени.

Организационные моменты

Основными организационными моментами являются:

- выделение первичных данных и разделение из по классу важности;
- контроль за регулярностью резервного копирования;
- минимизация хранения ненужной информации.

Технические средства

Технические средства можно разделить на общедоступные и специализированные. К общедоступным относятся различные оптические диски (CD, DVD, в некотором будущем, возможно, blueray), flash-накопители и жёсткие диски.

К специализированным относятся ленточные накопители, ленточные библиотеки и другие, менее распространённые носители.

В целом, на сегодняшний день по различным соображениям в домашнем и SoHo секторе наиболее рациональным является использование носителей, основанных на жёстких дисках.

Программные средства

Программные средства следует рассматривать с разделением на следующие группы:

- базовые инструменты, такие, как tar/gzip/rsync/dump;
- индивидуальные средства, рассчитанные на применение на отдельном рабочем месте;
- отдельные решения, рассчитанные на использование на «отдельно стоящих» серверах;
- сетевые системы, обеспечивающие централизованное копирование.

Роман Савоченко
Днепродзержинск, Украина, ООО НИП «ДІЯ»

Проект: OpenSCADA
http://oscada.diyaorg.dp.ua

OpenSCADA. Открытое решение для построения АСУ-ТП.

Доклад посвящён проекту создания открытого решения для построения АСУ-ТП, SCADA-системе OpenSCADA. Система OpenSCADA является полностью свободной и публикуется на условиях лицензии GPL. На данный момент разработка находится на этапе концепт-релиза, активно стабилизируется и тестируется для выпуска промышленной версии в конце этого года.

Современные автоматизированные системы управления технологическими процессами (АСУ-ТП), в основном, строятся на закрытых (коммерческих) решениях мировых производителей. Не последним компонентом АСУ-ТП, а именно глазами ТП, является человеко-машинный интерфейс (HMI), который строится на программном обеспечении (ПО) класса SCADA (Supervisory Control and Data Acquisition). Однако ввиду закрытой природы как SCADA, так и, часто, программируемых логических контроллеров (PLC), а также стремления к коммерческому вытеснению конкурентов любой ценой, усложняется контроль над ними конечным пользователем и взаимодействие между различными коммерческими решениями в гетерогенном окружении. Ярким показателем этих проблем является факт наличия множества мелких коммерческих решений SCADA систем от интеграторов и фирм разработчиков АСУ-ТП.

Для решения проблем коммерческих SCADA-систем в 2002 году был основан проект открытой (свободной) SCADA-системы OpenSCADA. С 2003 года проект прошёл путь от реализации отдельных подсистем к концепт-релизу (2008 г.). Текущая версия системы OpenSCADA 0.6.1 представляет собой первый стабилизирующий релиз ветки концепт-релиза (0.6). Основными целями данного проекта являются: открытость, надёжность, масштабируемость, многоплатформенность, безопасность, финансовая доступность, предоставление удобного интерфейса пользователя. Для распространения программы выбраны условиях лицензии GPL v2.

Система OpenSCADA предназначена для сбора, архивирования, визуализации информации, выдачи управляющих воздействий, а также других родственных операций, характерных для полнофункциональной SCADA системы. Благодаря высокому уровню абстракции и модульности, система может использоваться во многих смежных областях:

- на промышленных объектах, в качестве полнофункциональной SCADA системы;
- во встраиваемых (embedded) системах, в качестве среды исполнения, в том числе внутри PLC;
- для построения различных математических моделей и имитаторов (технологических, химических, физических, электрических процессов);
- на персональных компьютерах, серверах и кластерах для сбора, обработки, представления и архивации информации о системе и её окружении.

Благодаря модульности система OpenSCADA может выступать как в роли различных серверов, так и в роли разнообразных клиентов, а также совмещать эти функции в одной программе. Это позволяет реализовывать клиент-серверную архитектуру SCADA системы на базе одних и тех же компонентов/модулей, экономя при этом машинную память, дисковое пространство, а также ценное время программистов. Фактически система OpenSCADA обладает свойством организации больше, чем клиент-серверные архитектуры, а именно возможностью построения распределённых архитектур со смешанной структурой узлов.

Ширина возможностей системы OpenSCADA определяется гибкостью (масштабируемостью) ядра системы OpenSCADA, которое обладает высокой степенью модульности. Глубина возможностей определяется наполненностью модульных подсистем модулями. Версия 0.6.1 системы OpenSCADA в общем содержит 30 модулей семи модульных подсистем:

- Подсистема «БД» содержит 4 модуля поддержки БД: DBF, SQLite, MySQL и FireBird.
- Подсистема «Сбор данных» содержит 9 модулей поддержки следующих источников данных: платы сбора данных от Diamond

Systems; данные ОС; вычислитель на формальном языке блочных схем; вычислитель на Java-подобном языке высокого уровня; чистая реализация механизма логического уровня источника данных; сетевые источники данных по протоколу SNMP; PLC фирмы Siemens серии S7; различные PLC посредством семейства протоколов ModBus; источники удалённых OpenSCADA-станций.

- Подсистема «Транспорты» содержит один модуль транспортов на основе сокетов.
- Подсистема «Транспортные протоколы» содержит два модуля протоколов: HTTP и собственного протокола OpenSCADA.
- Подсистема «Специальные» представлена тремя модулями библиотек функций окружения пользовательского программирования и библиотекой тестирования компонентов системы OpenSCADA.
- Подсистема «Пользовательские интерфейсы» представлена шестью модулями поддержки пользовательских интерфейсов как конфигурации OpenSCADA, так и построения пользовательских интерфейсов контроля и управления на основе Web технологий и технологии библиотеки QT.

OpenSCADA как представитель класса свободного ПО уже сейчас позволяет решать многие задачи предметной области. Кроме того, разработка системы OpenSCADA находится на переломном этапе, когда совместными усилиями можно опробовать и адаптировать систему к собственным прикладным задачам, получив в результате всесторонне развитую открытую SCADA-систему промышленного уровня к версии 0.7.0.

Вадим Лебедев
Москва, МГУПИ

Проект: OpenSCADA
http://oscada.diyaorg.dp.ua/

OpenSCADA. Поддержка OPC

Реализация клиент-серверного взаимодействия по протоколу OPC XML DA (Линукс), аккумулирование данных в СУБД MySQL для решения задач промышленной автоматизации с помощью программного комплекса OpenSCADA.

Проблема стандартизации в сфере информационных технологий стоит давно и во многих областях весьма остро. Данная проблематика не в последнюю очередь затронула и промышленный сектор. Промышленная автоматизация имеет свою специфику: это как требования технического характера (например, наличие жёсткого реального времени и требования к оборудованию), так и срок службы системы в целом, измеряемый зачастую десятилетиями. Следствием указанной специфики является необходимость наличия устоявшихся стандартов, также актуальная для систем автоматизации.

На сегодняшний день для большинства сред промышленной автоматизации (автоматизированных систем управления технологическими процессами — АСУТП) для обмена данным используется протокол OLE for Process Control (OPC). OPC — программная технология на базе Windows-технологий (OLE, ActiveX, COM/DCOM), предоставляющая единый интерфейс для управления объектами автоматизации и технологическими процессами. Открытых альтернатив, реализующих аналогичный функционал и поддерживаемых таким количеством производителей, на сегодня, к сожалению, не существует. Первая рабочая версия стандарта была выпущена некоммерческой организацией OPC Foundation в 1996 году. До 2005 года все спецификации были доступны в Интернете для свободной загрузки. Позже политика OPC Foundation изменилась на сдерживающую распространение стандарта; спецификации и прочие материалы стали доступны только для членов фонда OPC. Стать членом фонда может любая компания, изъявившая желание и уплатившая вступительный взнос. Изначально (как, впрочем, и сейчас) стандарт декларировался

как полностью открытый, однако изменение политики фонда говорит о том, что ситуация может измениться в худшую сторону.

OPC описывает набор стандартов различного назначения. В силу архитектуры протоколов, базирующейся на COM/DCOM, в основном, требуется среда, в которой и для которой разрабатывались стандарты, — Windows. Однако данное ограничение противоречит одному из главных постулатов — возможности интеграции, кроссплатформенности. Один из стандартов OPC, OPC XML DA, базируется на открытом протоколе обмена XML-сообщениями, специфицированном W3C, SOAP[1]. Таким образом, данный стандарт платформенно независим.

В данной работе освещаются вопросы реализации клиент-серверного взаимодействия с внешними источниками данных, соответствующего спецификации OPC XML DA 1.0[2]. Взаимодействие осуществляется с использованием реализованного на базе python фреймворка PyOPC. Также приводится описание реализованной автором возможности аккумулирования данных в СУБД MySQL, предназначенных для дальнейшего взаимодействия с системой OpenSCADA. Результат работы будет распространятся под открытой лицензией GPL.

Литература

[1] Soap version 1.2 part 0: Primer (second edition). — 2003. `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`.

[2] Opc xml-da specification. — 2003.

Zbigniew Braniecki
Warsaw, Mozilla Corporation

Project: Firefox
http://www.mozilla.com

Mozilla vision of the Internet world

The Internet is one of the fastest moving elements of the modern world. The Internet is changing and growing faster than the law, economics or school system. In result it's hard to make it secure and educate people on how to use it in a safe way.

Mozilla Foundation is a worldwide organization managing the Mozilla project. It is guided by the values of transparency, openness and mutual trust which it uses to influence the Internet world.

Mozilla uses a unique model of organization — with a non-profit foundation on top, and several thousands volunteers at the very centre of the project. This hybrid balance of commercial quality and open nature brings the best of open source and free software into the professional and business environment, a guided path for others to follow.

The best known product of the Mozilla project is the Firefox web browser, which has recently been released in its third version. Firefox is in a unique position as a browser giving both — the choice and innovation to users and raising the quality and security of the Internet. Firefox is not only a product, but also a tool that used to ensure that the core values of the Internet persist such as its open nature and freedom of choice whilst rising the security and reliability of the Internet. Mozilla goal is to make the Internet a good place for business, education and people everywhere.

To ensure this, Mozilla created a powerful and open Web browser, that can bring balance and competition back to the Web, raise the importance of open standards and reiginite the development of the Web. Mozilla is also growing beyond Firefox, to reach the values that inspired Mozilla Manifesto to the areas such as communication with Mozilla Messaging and mobile with Mozilla Mobile. Mozilla is targeting the other markets with the very same values that lies behind Firefox.

Russia is a huge and rapidly growing Internet market with potential for being the biggest Internet user base in Europe. Additionally Russia presents a great interest in open source and free software technologies. It's a source of valuable contributors and local projects existing in the FLOSS ecosystem. Mozilla wants to support this effort.

Фёдор Зуев
Иркутск

Новации в российском законодательстве и свободный софт

За последние годы был принят или вступил в силу ряд законов, потенциально способных оказать существенное влияние на положение свободного софта в России. Равно как и разработчиков софта вообще.

I) С 1 января 2008 года в действие вступила IV часть Гражданского Кодекса, заменяющая все действовавшие законы об исключительных правах, в том числе закон об авторском праве, и создающая целый ряд совершенно новых юридических монополий. Закон содержит множество неопределенно-широких формулировок, которых мы не будем касаться до тех пор, пока судебной практикой им не будет дано толкование. Прокомментируем однако несколько относительно ясных моментов.

Отчуждение. Статья 1234 ГК вводит понятие «отчуждения исключительного права», понятие, которое ранее существовало в патентном праве, а теперь применяется и к копирайту. В отличии от существовавших до того лицензий, которые были отношениями обязательственными (и стало быть, существовали лишь до тех пор, пока существовал лицензиар), отчуждаемое исключительное право «переходит к приобретателю в полном объеме», отчуждение рвет все связи произведения с прежним правообладателем, все обязательства перед ним.

Существенным моментом здесь является то, что отчуждение здесь вовсе не обязательно является добровольным договором. Согласно ст 1241 исключительное право может быть отобрано «при наложении взыскания на имущество правообладателя», то есть за долги. Согласно статье 1284 такое взыскание не может быть наложено на исключительное авторское право принадлежащее непосредственно автору, но во всех других случаях — может.

Таким образом, многие организации — разработчики софта, в том числе свободного софта, ничуть не разбогатев по существу, оказываются обладателями ценных ликвидных активов. Аналогичная ситуация с недвижимостью породила полукриминальное явление, из-

вестное как «рейдерство». Навряд-ли в софтостроении события будут развиваться столь же бурно, как на рынке недвижимости, однако исключить появление софтверных рейдеров нельзя.

Возможным средством обороны от них, в особенности для организаций работающих со свободным софтом, может быть изменение трудовых договоров таким образом, чтобы исключительное авторское право оставалось у авторов, а фирма обладала бы лишь широкой лицензией на них.

Сложный объект. Статья 1240 ГК вводит новую концепцию «использования результата интеллектуальной деятельности в составе сложного объекта», в соответствии с которой «в случае, когда лицо, организовавшее создание сложного объекта, приобретает право использования результата интеллектуальной деятельности, специально созданного или создаваемого для включения в такой сложный объект, соответствующий договор считается договором об отчуждении исключительного права, если иное не предусмотрено соглашением сторон». Впрочем, даже если договором и предусмотрено иное, то «условия лицензионного договора, ограничивающие использование результата интеллектуальной деятельности в составе сложного объекта, недействительны».

Таким образом в известном вопросе о правах на программу ее майнтейнера и авторов патчей баланс в России радикально сдвигается в сторону майнтейнера. К нему переходят права на даже весьма значительный по объему и сложный код, если он крепко привязан к программе и явно не может использоваться независимо от нее (самостоятельно или в составе другой программы).

Полезно это или вредно? С одной стороны упрощается администрирование сложных свободных проектов средней сложности, открывается возможность оперативно проапгрейдить лицензию или добавить в нее необходимое исключение. Однако этим же самым ослабляется юридическая связь между разработчиками свободного софта.

Служебные произведения. Изменилось определение служебного произведения (то есть такого, исключительные права на которое принадлежат нанимателю). Теперь это не «созданные по служебному заданию» а «созданные в пределах установленных для работника трудовых обязанностей». Что, скажем осторожно, ставит под вопрос принадлежность прав (и возможности публикации под свободной лицензией) в широко распространенном случае, когда произведение (программа целиком, или патч) создается в служебных интересах и

может быть даже в служебное время — но по собственной инициативе работника. Майнтейнерам свободных программ надо внимательнее относиться к этому моменту при приеме патчей.

II) 4 мая 2008 года были приняты поправки в закон «О лицензировании отдельных видов деятельности», включающие в число видов деятельности, на которую требуется особое государственное разрешение, ни много ни мало как «деятельность по изготовлению экземпляров компьютерных программ». Правда, в весьма оригинальной формулировке: «деятельность по изготовлению экземпляров аудиовизуальных произведений, программ для электронных вычислительных машин (программ для ЭВМ), баз данных и фонограмм на любых видах носителей (за исключением случаев, если указанная деятельность самостоятельно осуществляется лицами, обладающими правами на использование указанных объектов авторских и смежных прав в силу федерального закона или договора)».

На первый взгляд выглядит как тавтология: «вы не имеете права без государственного разрешения изготавливать экземпляры программ, кроме тех случаев, когда вы имеете на это право». На самом деле закон этот направлен в первую очередь против заводов по производству компакт-дисков, действующих на основании не лицензии, а договора поручения. Но последствия из него вытекают и для остальной компьютерной отрасли. Состоят они, в общем, в том, что если раньше мы отчитывались за правомерность использования программы, за всю цепочку лицензий только перед правообладателями, то теперь такого отчета могут в любой момент потребовать чиновники лицензирующего ведомства.

Таким образом различные аспекты публичных лицензий могут в любой момент превратиться из отвлеченной теоретико-правовой премудрости в насущную бюрократическую необходимость.

Имеются основания полагать, что этот закон противоречит Бернской Конвенции, однако в существующей общеполитической ситуации вопрос этот имеет сугубо теоретическое значение.

III) Также в последние годы произошло большое количество изменений в области изобретенного в 1997 году уголовного копирайта — статьи 146 УК РФ и связанных с ней норм в УПК РФ. Юридический разбор этих изменений в данном случае мало что даст, поскольку це-

лью и результатом этих изменений было не установление определенного правового режима, а (в сочетании с другими действиями, явными и менее явными) установление фактического контроля над правоохранительными и судебными органами со стороны конкретной группы проприетарно-софтверных фирм. Сами по себе подробности этой истории непосредственно свободного софта не касаются. Но учитывая, что свободный софт для этих фирм является конкурентом в той же степени, что и пиратские издатели, неосновательны рассуждения, что ее результаты свободного софта никогда не коснутся (или даже пойдут ему на пользу). Полученная власть будет использоваться в том числе и против свободного софта.

Первый прецедент такого рода уже известен — это «дело Валентина Киселева», осужденного за написание альтернативной реализации сервера игры World of Warcraft, которую он распространял на условиях GPL. Это первый в России и даже кажется в мире случай уголовного приговора за написание свободной программы. В сообществе популярно мнения о якбы наличии неких особых позорных обстоятельств, отделяющих Киселева от фрисофтверного мейнстрима. На деле технически и юридически обстоятельства дела вполне аналогичны любой другой ситуации написания свободного аналога проприетарной программы или свободной реализации проприетарного протокола. Различия, поскольку они есть, лежат в социальной отрасли. Игровая индустрия известна своими историческими связями с организаованной преступностью (рулетка, лас-вегас, «однорукие бандиты» — помните) и как следствие, гораздо более склонны избавляться от конкуренции силовыми методами. Но если такие методы покажут свою успешность и безопасность — за ней последуют многие.

Сообществу свободного софта следовало бы задуматься над системой коллективной безопасности, коллективной защиты от враждебных акций такого рода.