Первая
международная конференция
разработчиков свободных программ
на Протве

Тезисы докладов

Обнинск
29–30 июля 2004

В сборнике представлены тезисы докладов, одобренных Программным комитетом Первой международной конференции разработчиков свободных программ на Протве. Другие материалы конференции можно найти на `http://conf.altlinux.ru` ISBN 5-85593-144-7

© Коллектив авторов, 2004
© ALT Linux, 2004

Программа конференции

28 июля

19:00–20:00: Заезд участников, регистрация

29 июля

10:00–12:00: Регистрация участников, кофе

Утреннее заседание
12:00–14:30

(Председатель — Алексей Новодворский)

12:00–12:15: Открытие конференции

14:30–15:30: Обеденный перерыв

Дневное заседание
15:30–17:00

(Председатель — Александр Боковой)

17:00–17:15: Кофе

Вечернее заседание
17:15–19:00

(Председатель — Алексей Смирнов)

30 июля

Утреннее заседание
10:00–11:45

Секция 1
(Председатель — Александр Боковой)

Секция 2
(Председатель — Анатолий Якушин)

11:45–12:00: Кофе

Дневное заседание
12:00–14:15

Секция 1
(Председатель — Станислав Иевлев)

Секция 2
(Председатель — Александр Боковой)

Вечернее заседание
15:00–17:00

Секция 1
(Председатель — Алексей Новодворский)

Секция 2
(Председатель — Алексей Смирнов)

УЧАСТНИКАМ ПЕРВОЙ МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ РАЗРАБОТЧИКОВ СВОБОДНЫХ ПРОГРАММ

Уважаемые Дамы и Господа, Коллеги!

Я рад приветствовать участников и организаторов первой международной конференции свободных программ!

В настоящее время, доступ к информации и знаниям определяет характер участия в жизни общества и открывает новые возможности для развития каждого. Области образования, науки и культуры в современном мире немыслимы без информатизации.

ЮНЕСКО подчёркивает возрастающую важность свободного программного обеспечения в информационном развитии общества и полагает, что полноценное развитие информационного общества во всех регионах мира невозможно без опоры на свободное программное обеспечение, более того — без опоры на сообщество разработчиков и пользователей свободных программ во всем мире.

У свободных программ большие перспективы, но реализация всего потенциала возможна только в том случае, если будут существовать активные проекты, поддерживаемые талантливыми и квалифицированными разработчиками. В этом свете мне бы хотелось ещё раз подчеркнуть важность Конференции для ЮНЕСКО как сторонника свободных программ.

Я хотел бы пожелать присутствующим плодотворной работы и всестороннего развития своих начинаний.

Директор бюро ЮНЕСКО
в г. Москва
Филипп Кео

28 июля

19:00–20:00: Заезд участников, регистрация

29 июля

10:00–12:00: Регистрация участников, кофе

Утреннее заседание

12:00–14:30

(Председатель — Алексей Новодворский)

12:00–12:15: Открытие конференции

12:15–13:00

Дмитрий Тараканов

Москва, Intel

Инструменты Intel для разработки программного обеспечения

Аннотация:
Доклад посвящён основным возможностям Intel® VTune™ Performance Analyzer, Intel C++ Compiler, Intel Threading Tools, Intel Performance Primitives, Intel Math Kernel Library. Основной уклон сделан на применение этих инструментов под ОС Linux.

Производительность — главное назначение всех инструментов от Intel. Инструменты Intel совместимы с ведущими IDE. Доступны в Windows и Linux, на всех Интеловских платформах

Продукты для разработки программного обеспечения:

- Intel® Compilers — Лучший способ добиться высокой производительности приложений на архитектурах Intel.
- Intel® VTune™ Performance Analyzer — Быстрый анализ узких мест производительности и способы их разрешения.
- Intel® Performance Libraries — Высокооптимизированные библиотеки функций широкого спектра применения.
- Intel® Threading Tools — Ускоряют и облегчают разработку многопоточных приложений

Компиляторы Intel® C, C++, FORTRAN

- Имеются для Windows и Linux -Имеются для 32 и 64-битных платформ;
- Использование последних достижений в области создания платформ и процессоров ;
- Оптимизация под архитектуру NetBurst™ (Pentium™ 4 и Xeon™);
- Оптимизация под архитектуру Itanium™ и Itanium™ 2;
- Поддержка Hyper-threading™ технологии и стандарта OpenMP;
- Интеграция в среды Windows (IDE) и Linux.

Встроенные средства SIMD-расширений:

- встроенные средства SIMD-расширений работают с упакованными данными до 128 бит в длину, что обеспечивает возможность параллельной обработки элементов данных;
- позволяют использовать Си функции вместо кодирования на языке ассемблера;
- обеспечивают доступ к основным возможностям, не реализуемым с применением обычных методик кодирования.

Автовекторизация — автоматически применяет SIMD команды в наборах команд SSE2, SSE и MMX™.

Программная конвейеризация предназначена для перекрытия итераций циклов Использует мощную поддержку программной конвейеризации, обеспечиваемую архитектурой Itanium™:

- циклический сдвиг регистров;
- специальные команды ветвления для циклов;
- большой массив регистров.

Компилятор работает автоматически без необходимости указания каких-либо ключей в командной строке. Межпроцедурная оптимизация — распространяет оптимизацию на все файлы Оптимизация по профилированию — оптимальна для кода с часто выполняемыми ветвлениями, которые трудно предсказать во время компиляции. Диспетчеризация ЦП — выбирает соответствующий код в период выполнения в зависимости от фактического типа процессора.

Автопараллелизатор компилятора Intel® Qparallel обнаруживает циклы, которые могут безопасно выполняться в параллели, и автоматически генерирует многопотоковый код для подобных циклов.

VTune™ Performance Analyzer — Быстрый анализ узких мест производительности и способы их разрешения. Помогает определить и локализовать проблемы производительности ПО посредством:

- Сбора широкого спектра показателей производительности с ОС на которой выполняется Ваше приложение;
- Обработки и отображения данных в различных видах, начиная с system-wide и заканчивая исходным кодом и процессорными инструкциями;
- Идентификации потенциальных проблем производительности и предложения вариантов их разрешения.

VTune™ Performance Analyzer Поддерживает:

- Платформы:
  - Семейство Intel® IA-32;
  - Семейство Itanium™;
  - XScale™.
- Операционные системы:
  - Microsoft Windows;
  - Linux (Red Hat, SuSe и др.);

- MRTE:
  - Java (BEA, IBM, Microsoft, Sun);
  - .NET;
  - Поддерживает локальный и удалённый сбор данных.
- Агент и коллекторы данных на удалённой машине
  - Microsoft Windows;
  - Linux;
  - MTRE.
- Управление процессом, анализ и отображение результатов на host-машине
  - Поддерживает интерфейс командной строки;
  - Интегрируется в Visual Studio 7.

Функциональность

- Sampling(time-based, event-based)
- Overtime view
- Call graph
- Граф вызовов с подробной информацией о временных затратах
- Критический путь исполнения
- Counter monitor
- Системные счётчики производительности
- Статический анализ исполняемых модулей
- Intel® Tuning Assistant
- Комментарии по проблемам, подсказки по модификации кода
- Обработка данных
- Мастера для конфигурирования коллекторов
- Сравнение и слияние результатов

- Упаковка и перенос проектов на другую машину
- Getting Started tutorial

VTune™ для Linux: две возможности

- Локальный и удалённый сбор данных.
  - VTune™ CLI2.0 устанавливается локально на Linux машине;
  - Интерфейс командной строки;
  - Коллектора на Linux машине для удалённого сбора данных;
  - Просмотр данных на Windows host-машине GUI;
- VTune™ Performance Analyzer 2.0 for Linux
  - Поддерживает
  - Платформы:
  - Семейство Intel® IA-32
  - Семейство Itanium™
  - Операционные системы: Linux* (Red Hat, SuSe и др.)
  - MRTE
  - Java (BEA, IBM, Sun) на IA-32
  - Интерфейс командной строки
  - Результаты совместимы с Windows-версией.
  - Могут быть упакованы и просмотрены с помощью Windows-версии анализатора
  - Поддерживает до 64 процессоров
  - Обеспечивается высоким уровнем клиентского сервиса
- Обзор функциональности
  - Sampling
  - Call graph
  - Source view
  - Обработка данных
  - Упаковка и перенос проектов на другую машину

  - Man-страницы и HTML-руководство

Intel® Performance Libraries — высокооптимизированные библиотеки функций широкого спектра применения.

- MKL Intel® Math Kernel Library
  - Linear Algebra: LAPACK plus BLAS (Levels 1, 2, 3)
  - Discrete Fourier Transforms (DFT)
  - Vector Statistical Library functions (VSL)
  - Vector transcendental math functions (VML)
- IPP Intel® Integrated Performance Primitives signal, image, graphic, multimedia and numeric processing functions
  - Доступны для Linux* and Windows*
  - Индивидуально оптимизированы под Pentium™ III, Pentium™ 4 Itanium™
  - IPP также под StrongARM*, Xscale™
  - GPP — Intel® Graphics Performance Primitives
  - Доступны для PDA/XScale™ с PPC 2002
  - Поддержка других ОС в процессе разработки

13:00–13:45

Александр Боковой Москва, IBM Восточная Европа/Азия, Центр компетенции Linux

Active Directory: долгий путь в Unix?

Аннотация:
При построении сетевой инфраструктуры предприятия важным критерием является возможность централизованного управления ресурсами. В то время как для конфигураций, построенных на основе одной

операционной системы, такие решения существуют, в случае использования гетерогенных сред на основе Microsoft Windows и систем с открытым исходным кодом интеграция по-прежнему достигается с трудом. В докладе рассматривается попытка построения свободной альтернативы служб Active Directory в рамках эксперимента, выполненного Центром технологий Linux (IBM) совместно с разработчиками ряда проектов со свободным программным кодом.

С выходом Windows 2000 компания Microsoft представила своё решение для централизованного управления ресурсами масштаба предприятия — службы Active Directory (ADS). За прошедшие четыре года с момента её появления, поддержка Active Directory была добавлена практически во все продукты Microsoft и, что важнее, в средства разработки приложений для сторонних производителей. В результате сложилась ситуация, когда новое программное обеспечение фактически требует развёртывания служб Active Directory.

В этой связи становится важным обеспечение приемлемого уровня интеграции свободных решений со службами Active Directory в условиях современного предприятия. Как показывает практика, существующего уровня интеграции недостаточно — использование ADS на платформе Windows давно переросло типовые инфраструктурные решения (печать и доступ к файловым ресурсам). В то же время, использование закрытых расширений стандартизированных протоколов в реализации ADS усложняет реализацию свободной альтернативы.

Для оценки объёма работы, необходимой для воплощения такой альтернативы в реальность, Центром технологий Linux компании IBM совместно с рядом открытых проектов (Samba, Heimdal, OpenLDAP) было проведено исследование, практической целью которого было построение минимального демонстрируемого решения на базе платформы GNU/Linux, позволяющего подключать существующие системы на базе Microsoft Windows XP/2003 в домен свободной реализации ADS.

Службы Active Directory основаны на использовании целого ряда открытых стандартов (DNS, DHCP, Kerberos V, LDAPv3, DCE RPC) с рядом недокументированных расширений. Реализация свободной альтернативы ADS в рамках одного свободного проекта невозможна: требуется серьёзное взаимодействие целого ряда команд разработчиков: Samba Team, MIT Kerberos V или Heimdal, OpenLDAP, Bind, DHCP и некоторых других. При этом, вносимые изменения в некоторых случаях не соответствуют стандартам и RFC, которые те или иные программные средства реализуют, что существенно осложняет их интеграцию в основной код.

В то же время, ряд существующих архитектурных решений в некоторых из перечисленных выше проектов не позволяет с необходимой гибкостью реализовать нужные расширения. Таким образом, свободная реализация ADS сталкивается не только с проблемой расшифровки и реализации нестандартных дополнений, применённых компанией Microsoft, но и с собственными внутрипроектными ограничениями некоторых критически важных компонент свободного программного обеспечения.

Тем не менее, в рамках исследования были определены необходимые условия для реализации свободной альтернативы ADS, которые обеспечили бы прозрачную интеграцию решений на основе GNU/Linux и продуктов компании Microsoft и других производителей несвободного ПО на платформе Windows. Более того, некоторые из этих условий постепенно реализуются в рамках свободного программного обеспечения, и примером тому является проект Samba 4, инфраструктура которого играет важную роль в объединении всех остальных программных решений в единое целое в рамках ADS.

13:45–14:30

Анатолий Якушин Москва, Госпиталь ветеранов войн 3

Проект: OpenOffice.ru

Состояние и перспективы разработки свободных офисных приложений

Аннотация:

Доклад посвящен текущему состоянию и перспективам разработки офисных приложений на базе свободных прикладных платформ. Рассматриваются вопросы использования популярных прикладных платформ (OpenOffice.org, Mozilla и т. д.) в корпоративной среде, средства разработки, модели реализации.

Проблемы создания комплексных информационных систем в последние десятилетия выросли в отдельную ветвь науки об управлении. Однако бурный рост этого научного направления неразрывно связан с бизнесом крупных фирм, продвигающих на рынок программные системы

и комплексы, реализующие передовые новации. Это неизбежно приводит к терминологической путанице, что неизбежно на раннем этапе развития любого научного направления, в отсутствии устоявшихся школ и авторитетов. Поэтому вначале необходимо остановиться на базовых терминах и понятиях. В данном докладе офисные приложения рассматриваются в классическом определении как совокупность аппаратных и программных средств для создания учрежденческой инфраструктуры.

В этом контексте к типично офисным задачам обычно относят:

- документооборот, осуществляющий обработку, прохождение, хранение и поиск документов;
- коммуникативные задачи по обмену информацией между сотрудниками и структурными подразделениями;
- календарные функции и планирование ресурсов.

Над этими базисными функциями в наиболее сложных системах функционируют модули поддержки принятия решений и управления. Как правило, системам документооборота придают специализированные функции — бухгалтерия, кадры, склад и т. п.

При рассмотрении офисных приложений необходимо дать краткую характеристику проприетарных систем, доминирующих на рынке в настоящее время. Здесь можно выделить набор решений от корпорации Microsoft, продукты фирмы IBM (Lotus Notes), Borland и ряд других. Анализируя подобные продукты, можно выделить основные части офисного приложения с точки зрения разработчика прикладного программного обеспечения. К ним относятся: хранилище данных, прикладная платформа и дополнительное программное обеспечение.

При рассмотрении понятия *прикладная платформа* можно заметить, что сегодня не существует единого взгляда на трактовку данного термина. Несмотря на некоторую спорность, следующее определение: прикладная офисная платформа это совокупность интегрированных переносимых средств разработки приложения, включающих в себя максимальное количество законченных полнофункциональных компонентов. Говоря о доступных свободных прикладных платформах можно выделить и кратко охарактеризовать следующие:

- ставшая уже классической триада PHP, Apache, MySql и проекты на ее основе;
- системы управления контентом Zope и Midgard;

- проект Clip;
- Mozilla;
- OpenOffice.org.

Однако наличие прикладной платформы не является гарантией создания законченного офисного приложения, каждое приложение требует высокопродуктивного масштабируемого и реплицируемого хранилища данных. В этом направлении у движения OpenSource есть значительные успехи. В качестве возможных систем хранения данных можно кратко рассмотреть:

- MySQL;
- PostgreSQL;
- SapDB и MaxDB;
- ZoDB.

Рассмотрение вопроса о хранилище данных неразрывно связано с вопросом формата хранимого документа. Сегодня бесспорным лидером здесь является формат XML, однако хранилище должно кроме собственно СУБД иметь полный набор средств для обработки этого весьма непростого в реализации формата. Для более подробного рассмотрения возможностей современных свободных прикладных платформ можно остановится на проекте OpenOffice.org.

Традиционное позиционирование OpenOffice.org, только как универсального офисного пакета является в корне неверным. Данный программный продукт может вполне успешно выступать в роли прикладной офисной платформы и наделен для этого всеми необходимыми качествами. Он содержит законченные полнофункциональные компоненты для большинства видов офисной деятельности. К ним относятся текстовый процессор, электронные таблицы, система презентаций и графический редактор. Однако кроме этого OpenOffice.org содержит развитые средства создания форм, систему доступа к хранилищам данных и встроенные языки программирования. Нативным форматом документов является XML.

Возможности OpenOffice.org можно показать на целом ряде примеров. Однако для создания полноценного офисного приложения этих возможностей все-таки недостаточно. Современные приложения уровня

предприятия требуют развитых коммуникативных возможностей и удаленного доступа к данным. Поэтому весьма серьезным является вопрос интеграции нескольких платформ на базе единого хранилища данных. Здесь весьма интересным является интеграция OpenOffice.org и Mozilla для решения вопросов обмена информацией между пользователями и реализации календарных функций. В настоящее время данные задачи реализуются в рамках проекта Glow.

В заключение следует сказать, что в настоящее время существует достаточно полный набор средств для создания офисных приложений на базе свободных программных продуктов, однако их популяризация в среде разработчиков требует от сообщества определенных усилий.

14:30–15:30: Обеденный перерыв

Дневное заседание

15:30–17:00

(Председатель — Александр Боковой)

15:30–16:15

Олег Паращенко — Санкт-Петербург, Санкт-Петербургский Государственный Университет

Проект: Protva XQuery

Подход к переносимой реализации языка XQuery

Аннотация:

XQuery, язык запросов XML, можно рассматривать как обобщённый интерфейс к древовидным данным. В работе обосновывается идея создания переносимой системы XQuery, которая может обрабатывать любые деревья, и предлагается реализация с помощью виртуальной машины.

Зачем обычным приложениям XQuery

Один из подходов к обработке XML[2] — это представление его в виде дерева, для навигации по которому используется XPath[4]. На основе XPath созданы языки для преобразования XML: XSLT[5] и XQuery[5]. XSLT является языком шаблонов, а XQuery можно представить как значительно расширенный XPath.

XML можно рассматривать как внешнее текстовое представление древовидных структур, а связанные с XML стандарты — как способы обработки таких данных. Такой подход может оказаться полезным для некоторых видов программ, например:

- текстовых процессоров;
- компиляторов и интерпретаторов.

В подобных приложениях важную роль играют сложные запросы к деревьям, например:

- найти все заголовки, не помеченные как удалённые, и создать из них оглавление;
- найти присваивания вместо сравнений в блоках условий команд if, for, while и им подобных.

Прямолинейная реализация таких приложений зачастую сложна, а код оказывается более низкого уровня, чем решаемая задача. В качестве альтернативы предлагается использовать встроенный XQuery, что должно упростить написание программ.

Подход к реализации

Основная проблема при разработке системы XQuery — это большой объём работ, поэтому не стоит писать XQuery каждый раз заново. Лучше создать переносимую реализацию и портировать её в приложения с минимальными затратами.

Для этого предлагается разработать виртуальную машину для обработки деревьев и систему XQuery на её основе. Если какому-либо приложению потребуется поддержка XQuery, то достаточно будет реализовать эту машину.

Чтобы упростить разработку, код пишется и отлаживается на языке Common Lisp, а затем транслируется в ассемблер виртуальной машины.

За основу взята система CL-XML[6]. James Anderson, её автор, проделал большую работу, так что CL-XML уже обрабатывает все примеры из спецификации XQuery.

CL-XML переводит XQuery во внутреннее промежуточное представление в виде абстрактного синтаксического дерева (AST). Этот формат можно использовать для высокоуровневой оптимизации запросов.

Большая часть языковых конструкций Common Lisp реализована с помощью макросов. Если взять CL-XML и раскрыть все макросы, то получится программа, состоящая из примитивов. Набор полученных конструкций является черновой версией виртуальной машины.

Реализация диалекта минимального Lisp (или Scheme) не составляет сложности, так как является хорошо изученной задачей [7].

Отдельный интерес представляет трансляция нашей виртуальной машины на LLVM[8] — промежуточное представление и набор инструментов для оптимизации программ. Благодаря LLVM и высокоуровневой оптимизации, код вычисления XQuery может быть эффективнее вручную написанного аналога.

Статус

Проект находится в начальной стадии. Пока только готов план работ, проведены первые эксперименты и подобраны инструменты для реализации. Промежуточные отчёты будут выкладываться на странице проекта Protva XQuery[1], код будет распространяться по лицензии LGPL.

На конференции будут представлены черновая версия виртуальной машины и первые результаты.

Первой вехой проекта будет добавление поддержки XQuery в XSLT-процессор xsltproc[9].

Список литературы

[1] Проект Protva XQuery `http://xmlhack.ru/protva/`

[2] Extensible Markup Language (XML) `http://www.w3.org/XML/`

[3] XML Path Language (XPath) `http://www.w3.org/TR/xpath`

[4] The Extensible Stylesheet Language Family (XSL) `http://www.w3.org/Style/XSL/`

[5] XML Query (XQuery) http://www.w3.org/XML/Query

[6] CL-XML: Common Lisp support for XML http://cl-xml.org/

[7] Scheme Compiler Technology/Implementation Techniques and Optimization http://library.readscheme.org/page8.html

[8] The LLVM Compiler Infrastructure Project. http://llvm.cs.uiuc.edu/

[9] The XSLT C library for Gnome http://xmlsoft.org/XSLT/

16:15–17:00

Андрей Орлов — Москва, Инфо Индастриез Групп

Павел Узорин — Москва, Интеко АГ

Александр Разоренов — Москва, Инфо Индастриез Групп

Проект: Neural.RU

Информационно-справочная система rPAS

Аннотация:
rPAS — информационно-справочная система, ориентированная на создание интеллектуального хранилища данных — средства, позволяющего минимизировать затраты на поиск и подбор данных, нужных для работы. rPAS использует различные алгоритмы самообучения для выработки стратегии размещения информации, адаптируя её к изменяющимся потребностям пользователей.

Цель создания информационно-справочных систем — обеспечение работы с большими массивами документов с приемлемой (оптимальной) скоростью поиска необходимой информации.

Интеллектуальное хранилище, помимо традиционного поиска, обеспечивает автоматическое размещение документа в рубрикаторе и его перемещение в процессе хранения между рубриками, отражая изменяющиеся потребности пользователей системы, а также связывает его с другими документами. Кроме того, хранилище проводит анализ своего

содержимого, позволяя помочь в контроле актуальности информации, и проводя прогнозирование характера информации, которая может быть востребована в дальнейшем (т. н. «упреждающее индексирование»).

Жизненный цикл любого документа начинается с размещения в информационно-справочной системе. При размещении документа проводится анализ его проводится анализ его содержимого, в результате которого выделяются отличительные признаки и составляется векторное описание (например, признак—вес признака), позволяющее определить смысловое сходство документов в терминах расстояний между векторами. Множество документов может быть разбито на группы, соответствующие группам векторов, расположенных вблизи друг друга. Это позволяет составить и поддерживать автоматический рубрикатор документов.

После передачи документа на хранение, начинается следующий этап жизненного цикла — работа с документом, включающая в себя запрос документа из хранилища пользователями, размещение ссылок на него в индивидуальном рубрикаторе и связывание с другими документами. Это обычная деятельность пользователей любой информационно-справочной системы, которую можно охарактеризовать как упорядочение данных для оптимизации обслуживания потребности в них.

Анализ запросов документов из рубрикатора позволяет определить факт сходства между некоторыми рубриками и документами на основании предположения о сходстве документов, используемых совместно (запрошенных одними и теми же пользователями, связанными между собой и т. п.). На основе этих данных корректируются правила составления векторных описаний и словари признаков, что приводит к постепенной адаптации структуры рубрикатора к некоторым усреднённым потребностям пользователей.

Система производит постоянный мониторинг активности пользователей, что позволяет составить и поддерживать актуальной модель интересов пользователей системы. В соответствии с этой моделью, можно предсказать потребности в размещённом документе и связать документ так же, как это сделал бы пользователь системы. Для дополнительной настройки такого самостоятельного поведения системы возможно указать необходимость выполнения определённых действий (например пересылку документа) в ответ на такие события как изменение размещения документа или его связывание.

В системе могут быть инициированы различные аналитические процедуры, позволяющие на основе составляемых в процессе разбора документов словарей и рубрикатора выявлять неполноту данных в храни-

лище и прогнозировать возможность возникновения потребности в информации определённого рода в ближайшем будущем. Результаты могут доводиться до сведения заинтересованных пользователей или использоваться самой системой для получения дополнительной информации из внешних источников.

Для создания такой информационно-справочной системы потребовалась разработка специального объектно-ориентированного сервера приложений rPAS. rPAS имеет клиент-северную архитектуру, в которой сервер обеспечивает хранение и обработку документов, а клиентские приложения предоставляют интерфейс операторам или служат коннекторами к другими внешним источникам или потребителям данных.

Документы хранятся в виде объектов, каждый из которых может предоставлять один или более интерфейсов. Интерфейс является унифицированным, независимым от типа способом работы с объектом, известном клиентским приложениям. Это позволяет исключить перепрограммирование клиентских приложений до тех пор, пока для работы с объектами новых типов достаточно уже существующих интерфейсов, поэтому в rPAS возможно создание клиентских приложений с достаточно сложным интерфейсом без особых опасений о совместимости с будущими версиями объектной модели. Клиент-серверное взаимодействие может осуществляться посредством различных протоколов, основным из которых является семейство протоколов CORBA.

В настоящее время закончена разработка первой версии rPAS, включающей в себя сервер, простую объектную модель, ориентированную на хранение и редактирование документов, браузер хранимых объектов, коннектор к почтовой службе и некоторым другим источникам данных. Независимо от целей его создания, в текущем состоянии rPAS может применяться как простая, объектно-ориентированная клиент-серверная среда.

Это позволило начать работы по реализации алгоритмов, обеспечивающих использование rPAS в качестве информационно-справочной системы. Работы находятся в различной степени завершённости и направлены на решение простой тестовой задачи: создание настраиваемого классификатора входного потока документов, полученных, например, просмотром новостных лент или электронной почты. Хотя работы над основными алгоритмами находятся в начальной стадии, существующий каркас и их упрощённые версии позволили провести тестовую эксплуатацию rPAS для разбора входного потока почты и новостей, что показало принципиальную правильность выбора архитектуры.

17:00–17:15: Кофе

Вечернее заседание

17:15–19:00

(Председатель — Алексей Смирнов)

17:15–18:00

Алексей Воинов, Георгий Курячий Москва, ALT Linux

Модульный подход к управлению ОС: проект Alterator

Аннотация:

Цель проекта Alterator — создание полностью открытого инструмента, который позволит перевести на пользовательский уровень процедуру настройки и сопровождения типовых решений на основе UNIX-подобной ОС и её служб. Задача проекта — упростить процедуру разработки таких решений на всех стадиях, от формирования пользовательской модели и создания интерфейса до задания команд, непосредственно управляющих системой.

Печальная реальность сегодняшнего времени — т. н. «неподготовленный администратор», человек, обязанности которого приближаются к обязанностям «настоящего» системного администратора (высококвалифицированного пользователя), а знания зачастую не превосходят знаний «оператора ЭВМ» (как включать, как делать такие-то стандартные действия, куда звонить, если не работает).

Если система предназначена для решения комплекса *типовых* задач (например, WWW-сервер + почта + межсетевой экран), под каждую из них находится готовое решение, с большей или меньшей степенью удовлетворяющее потребителя. Поэтому нет необходимости в том, чтобы администратор имел квалификацию разработчика.

Как правило, обслуживание таких систем — также комплекс *типовых* действий оператора, не требующий, за редким исключением, квалификации администратора. В этом случае удобна «шаблонная» организация интерфейса (например, в виде кнопок и меню), ориентированная

на выбор *готового* решения из множества. «Конструктивная» же организация интерфейса (например, изменение файла настроек), напротив, неудобна, так как ориентирована на *самостоятельную* постановку и решение задачи.

Обычные методы организации «шаблонного» интерфейса: Webmin, LinuxConf, DrakX и т. п. созданы, по всей видимости, по единой схеме *замены* действий администратора их наглядным представлением. Следствием такой схемы обычно является жёсткая привязка интерфейса к реальным элементам управления. Каждое дополнение такого продукта функциональностями требует повторения процесса разработки и интеграции их в готовую программу. Всё это приводит к очень негибкой, целиком зависящей от коллектива разработчиков структуре.

Цель проекта Alterator — создание полностью открытого инструмента, который позволит перевести на пользовательский уровень процедуру настройки и сопровождения типовых решений на основе UNIX-подобной ОС и её служб. Задача проекта — упростить процедуру разработки таких решений на всех стадиях, от формирования пользовательской модели и создания интерфейса до задания команд, непосредственно управляющих системой.

Для этого в проекте предусмотрено разделение потока управления на уровни: горизонтальные (интерфейс—модель—реализация) и вертикальные (соответственно задачам). Каждый элемент такой сетки — отдельная программа, получающая данные со стандартного ввода и/или в командной строке.

Горизонтальное деление предполагает, что модели среды и языки представления данных и заданий на всех уровнях различны и соответствуют терминологии, используемой, соответственно, дизайнером, оператором и системным программистом. Верхний уровень (модель представления, язык LOO) заведует тем, как объекты и действия над ними отображаются в конкретном варианте GUI. Центральный уровень (пользовательская модель, язык WOO) соответствует задаче в терминах «неподготовленного пользователя». Нижний уровень (модель реализации, язык HOO) реализует конкретные действия над системой и её объектами, которые надо произвести для решения задачи.

Вертикальное деление предполагает, что любая WOO-команда может обрабатываться сразу несколькими обработчиками-трансляторами WOO в HOO (commander), каждый из которых использует только необходимую часть передаваемого WOO-объекта. Для этого они регистрируют типы обрабатываемых WOO-объектов и команд в интегрирующем модуле (chooser).

Неизбежные при этом конфликты в результирующем потоке HOO-команд разрешаются при помощи упорядочивания по зависимостям формальной проверки на непротиворечивость. Стоит отдавать себе отчёт, что обратное преобразование (низкоуровневого HOO в высокоуровневый WOO) далеко не всегда возможно, однако каждый commander может сделать некоторые предположения на основании результата работы *всей* очереди HOO-команд и знания породившей её WOO-команды.

Для унификации интерфейса HOO-команд используется AdmFS, виртуальная файловая система, верхняя половина которой — стандартное файловое API, а нижняя — вызов соответствующей программы-hook, которая и производит специфические для конкретной операционной системы действия.

Таким образом, компоненты к Alterator можно разрабатывать на *любом* языке программирования. Для этого достаточно соблюдать *текстовые* протоколы управления WOO и HOO, использовать интерфейс командной строки, файловое API.

Добавление очередной функциональности (новый commander) в Alterator не требует изучения внутренности других commander'ов, достаточно соблюдения протоколов WOO и HOO. Добавления дополнительных системных (hook) и интерфейсных (look) модулей может вообще не потребоваться, и в любом случае их можно разрабатывать независимо друг от друга.

Наконец, разделение по уровням позволяет адаптировать Alterator к конкретной версии операционной системы, изменяя один только уровень hook (под AdmFS), а к конкретному внешнему виду — только уровень look (над chooser).

Необходимые для Alterator механизмы были частично реализованы в пилотном проекте «ИВК Кольчуга».

13:00–13:45

Дмитрий Левин Москва, ALT Linux

Проект: ALT Linux Team

hasher: технология безопасной сборки пакетов

Аннотация:

Рассматривается задача безопасной воспроизводимой сборки пакетов в большом репозитории на примере Sisyphus, изучаются требования, накладываемые этой задачей на архитектуру системы сборки, рассматривается архитектура hasher'а и существенные моменты её реализации. Приводятся примеры производных решений на базе hasher'а.

Традиционная схема сборки дистрибутивов

Давным-давно, когда дистрибутивы операционных систем помещались на один компакт-диск вместе с исходным кодом, а создавали их узкие группы специалистов, никакой сборочной технологии по существу не было, а отдельные элементы дистрибутива собирались прямо в хост-системе, полученной из полностью установленного дистрибутива.

Со временем инструментальные дистрибутивы выросли в размере, увеличилось число принимающих участие в разработке дистрибутивов, и в результате сборка дистрибутива в хост-системе стала небезопасной, ненадёжной и неудобной.

Требования к сборочной технологии

Технология сборки элементов дистрибутива (пакетов) должна:

- не снижать уровень безопасности хост-системы;
- обеспечивать собственную безопасность от атак со стороны пакетов;
- обеспечивать безопасность сборки пакетов от атак со стороны других пакетов;
- гарантировать надёжность (воспроизводимость) результатов сборки;
- обеспечивать приемлемый уровень производительности.

Архитектура hasher'а

В основе архитектуры hasher'а лежит трёхпользовательская модель: вызывающий непривилегированный пользователь (*C*) и два непривилегированных вспомогательных псевдопользователя; первый (*R*) играет роль root в порождаемой сборочной среде, второй (*U*) — обычного пользователя, собирающего программы.

Переключение между вызывающим и вспомогательными пользователями осуществляется с помощью специальной привилегированной программы (вызываемой посредством sudo), написанной с применением параноидальных мер защиты от непривилегированных пользователей. Кроме того, с помощью этой программы удаляются процессы, запущенные вспомогательными псевдопользователями и не завершившиеся в срок, а также создаются устройства. Наконец, эта программа предоставляет возможность контролировать ресурсы, выделяемые процессам непривилегированных пользователей, для защиты от DOS-атак.

Путь пакета через сборочную систему в общих чертах выглядит следующим образом:

1. Пользователь *C* порождает среду (aptbox) для работы с apt.

2. Полностью удаляется сборочная среда, возможно оставшаяся от предыдущей сборки. Удаление происходит последовательно в chroot пользователем *U*, в chroot пользователем *R* и, наконец, пользователем *C*.

3. Пользователь *C* создаёт каркас новой сборочной среды, состоящий из вспомогательных каталогов и вспомогательных статически слинкованных программ (ash, find и cpio). С помощью вспомогательной привилегированной программы создаётся фиксированный набор устройств, достаточный для нормального функционирования сборочной среды и при этом не несущий угрозы host-системе.

4. Порождается базовая установочная среда, представляющая собой набор средств, необходимых для штатной установки пакетов в эту среду. Пользователь *C* с помощью apt определяет набор пакетов, необходимых для порождения базовой установочной среды. Пользователь *R* с помощью вспомогательных статически слинкованных программ распаковывает эти пакеты.

5. Порождается базовая сборочная среда, представляющая собой набор средств, необходимых для сборки любого пакета.

Пользователь *C* с помощью apt определяет набор пакетов, пользователь *R* устанавливает их.

6. Порождается сборочная среда для данного пакета. Пользователь *U* извлекает сборочные зависимости пакета, пользователь *C* с помощью apt определяет набор пакетов для установки, и пользователь *R* устанавливает их.

7. Пользователь *U* осуществляет сборку пакета.

Такая схема призвана исключить атаки вида $U \rightarrow R$, $U \rightarrow C$, $R \rightarrow C$, а также все виды атак на root.

Для повышения производительности, особенно важной при сборке большого числа пакетов, применяется кэширование базовой сборочной среды.

С помощью средств apt реализована возможность использования собранных ранее пакетов для сборки последующих пакетов.

beehive: распределённая сборка с помощью hasher'a.

Благодаря свойству воспроизводимости результатов сборки hasher можно использовать для параллельной сборки большого числа пакетов на нескольких серверах. Таким образом удаётся достичь разумного времени сборки при средних вычислительных ресурсах. Открывается возможность организовать регулярное тестирование на пересобираемость большого репозитория пакетов, что и было сделано на примере Sisyphus с помощью средства параллельной сборки beehive.

30 июля

Утреннее заседание

10:00–11:45

Секция 1

(Председатель — Александр Боковой)

10:00–10:25

Дмитрий Тараканов Москва, Intel

Разработка эффективных приложений для процессоров семейства Intel® XScale™

Аннотация:

Доклад посвящён основным характеристикам, возможностям и преимуществам процессоров семейства XScale™. Будут приведены данные тестирования производительности процессоров семейства XScale™ и других процессоров.

Рынок для процессоров Intel® XScale™. Процессоры семейства PXA2xx для карманных устройств:

- RISK архитектура;
- Высокая производительность — частоты до 600Mhz;
- Низкое потребление;
- Интерфейс с видео/фото камерами ;
- Полная линейка процессоров для решений разного уровня;

- Объединение коммуникационной и вычислительной функциональности;
- Портирование приложений традиционных для десктопа на карманные устройства.

Архитектура ARM*:

- 16 32-битных регистра общего назначения
- 4Гб адресного пространства
- Байтная адресация
- 8/16/32 битные инструкции чтения записи
- Требуется выравнивание данных
- Упрощённый набор команд
- 32-битные опкоды, выравненные на 32 бита
- Вся арифметика 32-битная целая
- Нет FPU, нет деления, нет call/return, нет стека
- Дополнительные команды доступны через сопроцессоры MMU/MPU, cache control, FP, DSP etc.

Сравнение XScale™ vs. P4;
Среда разработки: Windows* 2000/XP
Целевые процессоры: Intel® PXA25x/26x
Следующие поколения процессоров Intel® XScale™
Целевые ОС:
Windows* CE 3.0/.NET
Symbian* OS
Palm* OS
Библиотеки поддержки: Intel® IPP
Поддержка приложений сторонних разработчиков: ARM ADS
Microsoft* Windows* CE .NET Platform Builder / eMbedded* Visual C++*
GNU
Почему рекомендуется использовать компилятор Intel®? Разработан производителем процессора:

- Полная поддержка микроархитектуры Intel® XScale™
- Получение оптимизированного кода
- Первоочередная поддержка новых особенностей аппаратуры
- Совместим с компилятором Microsoft* C++ compiler
- Приложения Intel® дополняют и развивают приложения Microsoft для MS Platform Builder (разработка систем) для MS eMbedded Visual C++ (разработка приложений)
- Intel® C++ compiler может быть легко выключен/включён.

Совместим с компилятором Microsoft По синтаксису опций, например:
`{1|2|3}` — установка уровня оптимизации;
`/Zi` — создание отладочной информации;
По синтаксису прагм, например:

```
#pragma optimize( "[optlist]", {on | off} )
```

Компилятор Intel® запускает компилятор Microsoft для генерации файлов PCH для просмотра исходных текстов (source browser). Возможность смешивания объектных модулей Intel и Microsoft*; Объектный формат совместим с Microsoft*; Компоновка прикладных объектов с библиотеками MS*;

- Поддержка режима Thumb компании ARM* (16-бит)
- Включение в проекте ARM: опция `/Q_cgopt=thumb`
- Полностью Thumb проект: создаётся обычным путём в eMbedded Visual C++
- Нет необходимости прикомпоновывать специальную библиотеку
- Доступен для Microsoft Windows CE .NET
- Предоставляется дополнительная оптимизированная библиотека run-time
- Быстрая эмуляция операций FP, целочисленное деление
- Оптимизированные версии strcmp, strcpy
- Замещает стандартные библиотеки MS*

Top 5 полезных советов:

1. Исправьте ошибки перед тем как начинать оптимизацию.
2. Используйте компилятор Intel!
   - Лучшие run-time библиотеки.
   - векторизация и оптимизация.
   - Помогите компилятору!
3. Используйте VTune для анализа производительности и поиска узких мест!
4. Оптимизируйте использование кэша!
   - Кэш-промах стоит 150 циклов.
   - Используйте preload() (works on ARM, too).
   - Advanced: XScale™ имеет дополнительный mini cache и позволяет замораживать данные в кэше.
5. Ручная оптимизация критических мест.
   - Используйте DSP-расширения, iMPT, WMMX.
   - Оптимизируйте ветвления.
   - Оптимизируйте циклы.
6. Используйте оптимизированные IPP/GPP библиотеки.

10:25–10:50

Никита Винокуров Москва, IBM Восточная Европа/Азия, Центр компетенции Linux

Linux on POWER

Аннотация:
Цели, ближайшие перспективы и проблемы развития ПО с открытым исходным кодом и ОС Линукс в частности на платформе PowerPC. Взгляды компании IBM на процесс портирования и использования ОС Линукс на PowerPC. Особенности архитектуры PowerPC.

Взгляд IBM на развитие Linux on Power

Технология POWER присутствует на рынке уже более 13 лет и используется в промышленной эксплуатации среди 64-битных серверов с 1994 года. Linux — это зрелая, защищённая и проверенная операционная система. Фактически — это самая быстрорастущая операционная система, поддерживаемая сегодня на серверах.

IBM продолжает фокусировать свои действия на помощи клиентам в переходе на открытые ИТ-системы, помогая им снизить стоимость их владения. Использование POWER-архитектуры простирается от игровых приставок до суперкомпьютеров. IBM активно участвует в усилении Linux-on-Power как надёжной, открытой и мощной вычислительной платформы.

Поддержка разработчиков

Компания IBM, продвигая Power как более быструю и надёжную платформу для запуска Linux-приложений нежели Intel, заинтересована в том, чтобы разработчики портировали свои приложения с Intel-Linux и Windows архитектур на PowerPC+Linux и всячески поддерживает этот процесс. Например, организован форум разработчиков, осуществляющих миграцию, который также поддерживается разработчиками из Linux Technology Center компании IBM `http://linuxppc.org`. В регионах работают центры инноваций, включённых в программу тестирования Linux-приложений на технике IBM `http://www.developer.ibm.com/spc/`, помогая разработчикам ПО тестировать свои приложения без покупки техники.

Линейка продуктов

Вычислительные системы, основанные на архитектуре PowerPC, представлены на рынке в виде двух линеек серверов eServer: iSeries и pSeries. Эти две серии машин используют, фактически, один тип процессора — Power4. Различаются они только лишь позиционированием на рынке и способностью организовывать логические разделы (LPAR). Сейчас появились новые версии машин iSeries, работающих на Power5.

Дистрибутивы

В данный момент, основными дистрибьюторами ОС Linux, которые заявили о поддержке платформы PowerPC являются SuSE, Red-

Hat, TurboLinux, YellowDogLinux и Gentoo. Причём, компания IBM, в свою очередь, объявила поддерживаемыми только три из них: SuSE, RedHat и TurboLinx. Во все эти дистрибутивы входит набор для кросс-компиляции приложений в 64-битный код (кроме Gentoo, где 64-х битная платформа является "родной"), поскольку основная масса приложений, кроме ядра и нескольких системных утилит работает в 32-битном режиме.

Приложения

Все приложения, входящие в названные выше дистрибутивы портированы на платформу PowerPC. Практически все они работают в 32-битном режиме. Портированы также ключевые продукты компании IBM: DB2, Java, VisualAge, WebSphere, Lotus Domino. Существуют в портах такие приложения как Mozilla и OpenOffice.org. Однако, с портированием приложений с использованием 64-х бит, существуют проблемы, которые открытому сообществу разработчиков предстоит решить `http://linuxppc64.org,http://dev.gentoo.org/\~{}tgall`

Архитектура PowerPC

Технологии

Два ядра на одном чипе. Общий кэш L3. Медные соединения. Кремний на изоляторе. Distributed Switch. Позволяют добиться более высокой надёжности и производительности за счёт снижения тепловыделения и увеличения интеграции (170млн транзисторов). Конструкция Distributed Switch и организация SMP обеспечивают практически линейный рост производительности с увеличением числа процессоров (до 32-х). В Power5 объявлена поддержка MultiThreading — одновременного выполнения двух потоков инструкций одним ядром.

LPAR

Поддержка логических разделов. Одновременно, на каждой машине может быть запущено несколько операционных систем в так называемых, логических разделах. Ресурсы машины, такие как процессор, память и устройства ввода-вывода разделяются между операционными системами. Например, на линейке серверов pSeries, минимальное количество процессоров и карт ввода-вывода на один LPAR может быть

выделено 1 и 2 соответственно. В iSeries, процессор можно делить на большее количество LPAR (до 32-х).

Пока Linux не умеет использовать динамическое перераспределение процессоров, памяти и устройств ввода-вывода в LPAR, хотя такая возможность существует, но работа в этом направлении ведётся (ветка 2.6.x ядра). Использование LPAR позволяет снизить удельную стоимость аппаратного обеспечения, приходящегося на операционную среду.

Архитектура процессора

Процессор PowerPC — это RISC (Reduced Instruction Set Computing) процессор. Все вычислительные операции производятся в регистрах. А работа с памятью ограничивается только загрузкой/выгрузкой регистров. Команды процессора PowerPC разделяются на 3 уровня спецификации: Уровень приложения (наиболее переносимый), Виртуальный уровень (управление кэшированием, таймером, специфический для разных реализаций) и Уровень операционной среды (управление памятью, исключительными ситуациями). Уменьшенное ядро, таким образом позволяет использовать интеграционные мощности для оптимизации работы процессора, например, организации конвейера.

Платформа PowerPC, по мнению IBM, является наиболее перспективной с технологической точки зрения и должна занять доминирующие позиции как на серверном рынке, так и на рынке персональных вычислений в ближайшем будущем.

10:50–11:15

Антон Качалов Москва, ALT Linux

Opie — свободная графическая среда для мобильных устройств

Аннотация:
Доклад посвящён проекту разработки свободной графической среды Opie, разрабатываемой для КПК и других устройств, работающих под управлением ОС Linux. Также в докладе освещаются вопросы состояния проекта и перспектив его разработки.

Opie родился в начале 2002 года как ветвь Qtopia, разработанного Trolltech. На сегодняшний день команда Opie насчитывает более полусотни разработчиков. Opie переведён на десяток языков, включая русский. Благодаря усилиям разработчиков и пожеланиям пользователей, в интерфейс постоянно вносятся улучшения.

Opie построен на библиотеке QT Embedded — реализация QT для встраиваемых устройств. Opie рассчитан на работу под управлением ОС Linux на следующих устройствах: Sharp Zaurus, HP iPAQ, Casio Cassiopeia. Взаимодействие с приложениями Opie обеспечивается посредством Touch pad (эмуляция мышки), а также с помощью клавиатуры. В Opie есть средства для настройки и работы с сетью. Многие приложения имеют интеграцию, например, для работы с IrDa. Opie может работать со следующими классами устройств: IrDa, Wlan, Ethernel, BlueTooth. Кроме того, данные можно синхронизировать с такими приложениями как KDE PIM, MS Outlook и Qtopia Desktop. Многие программы, скомпилированные для Qtopia, будут работать в Opie. В ближайших планах стоит полный перевод приложений на модульный интерфейс, позволяющий ускорить запуск программ. При разработке Opie за базовый язык программирования был взят C++.

В рамках проекта Opie автор работает над приложением, облегчающим манипуляцию пакетами. Решение основано на использовании apt как клиент-серверного приложения: хранение хэшей и все вычисления возлагаются на хост-систему, а на клиентской стороне хранится список пакетов и зависимостей для миграции с одной хост-системы на другую и установку пакетов в обход использования apt.

Так же планируется расширение возможностей hasher для нативной сборки пакетов на системах с другой архитектурой и со слабой дисковой подсистемой. В этом решении в роли диска, на котором будет происходить сборка, выступает nfs-root (или afs-root). Схема работы проста: hasher работает на хост-системе, а rpmbuild выполняется на сборочной системе. Дисковые операции (такие как распаковка/архивирование файлов, копирование/перемещение) будут выполняться на хост-системе.

11:15–11:40

Надежда Плотникова Нижний Новгород, Intel

Разработка многопоточных приложений для архитектур ia32/ia64 под Linux с использованием Intel® Threading Tools

Аннотация:
В докладе будут рассмотрены реализации потоков в POSIX стандарте и библиотека OpenMP; особое внимание будет уделено проблемам, с которыми сталкивается разработчик многопоточных приложений. Будет показано, как с помощью Intel® Threading Tools находятся и устраняются эти проблемы (включая потенциальные) и как анализируется производительность приложений.

Многопоточное программирование и отладка программ под Linux с помощью Intel® Threading Tools. Существует большое количество способов распараллеливания как на уровне архитектуры, так и программно. Параллелизм в модели общей памяти является наиболее доступным и привлекательным для конечного пользователя, т. к. позволяет использовать имеющиеся ресурсы (2-4 CPU SMP машины, такие как, например, ZeonTM, ItaniumTM Platform Family, Pentium® 4 with HT и т. п.) максимально эффективно. Это достигается путём использования потоков.

Существуют несколько вариантов реализации многопоточности, делящиеся на 2 подкласса: явные и неявные потоки. В первом случае это потоки в стандарте POSIX, использующие Linux native API и требующие явного использования синхронизационных примитивов, а во втором случае это библиотека OpenMP и возможность автоматического распараллеливания циклов с помощью компилятора.

Однако применение всех типов потоков — задача непростая и порождает целый класс специфических ошибок, таких как data races, dead locks и т. п., на поиск и устранение которых может потребоваться много времени. Intel представляет набор инструментов для отладки и анализа производительности многопоточных приложений: Intel® Thread Checker и Thread Profiler. Использование Intel® Thread Checker позволяет найти практически все известные, а также потенциально возможные ошибки распараллеливания. Thread Profiler показывает картину исполнения потоков, все объекты синхронизации и прочую информацию о потоках.

Секция 2

(Председатель — Анатолий Якушин)

10:00–10:25

Роман Савоченко Украина:Днепродзержинск, НИП DIYA

Проект: OpenSCADA

Проект «OpenSCADA»

Аннотация:
OpenSCADA представляет собой открытую SCADA(Supervisory Control And Data Acquisition) систему, построенную по принципам модульности, кроссплатформенности и масштабируемости. Программа OpenSCADA предназначена для сбора, архивирования, визуализации информации, выдачи управляющих воздействий, а также других родственных операций характерных для полнофункциональной SCADA системы.

OpenSCADA представляет собой открытую SCADA систему, построенную по принципам модульности, многоплатформенности и масштабируемости. SCADA (Supervisory Control And Data Acquisition) с английского означает: супервизорные системы контроля и сбора данных.

В качестве политики разработки данной системы выбраны open source принципы. Выбор данной политики определяется необходимостью создания открытой, надёжной и общедоступной SCADA системы. Система OpenSCADA предназначена для выполнения как обычных функций SCADA систем, так и для использования в смежных областях информационных технологий.

Система OpenSCADA может использоваться:

- на промышленных объектах в качестве полнофункциональной SCADA системы;
- внутри PLC (программируемый логический контроллер) в качестве среды исполнения;
- встраиваться в различное оборудование;

- для построения различных моделей, симуляторов (технологических, химических, физических, электрических процессов);
- на персональных компьютерах, серверах и кластерах. Для сбора, обработки, представления и архивации информации о системе и её окружения.

В качестве базовой (хостовой) операционной системы (ОС) для разработки и использования выбрана ОС Linux, которая является стандартной POSIX совместимой ОС. Кроме того ОС Linux является оптимальным компромиссом в вопросах:

- надёжности;
- гибкости/масштабируемости;
- доступности;
- популярности и распространённости.

Поскольку система OpenSCADA разрабатывается на стандартной POSIX ОС, по принципам кроссплатформенности, то её адаптация на остальные ОС не составит проблемы.

Сердцем системы является модульное ядро. В зависимости от того, какие модули подключены, система может выполнять как функции различных серверов так и функции клиентов клиент-серверной архитектуры. Собственно, данная схема позволяет реализовывать распределённые клиент-серверные системы любой сложности.

Также, для достижения высокого быстродействия, модульный принцип, позволяет объединять различные функции в одной программе. Архитектурно, система OpenSCADA состоит из подсистем:

- безопасность;
- базы данных;
- транспорты;
- протоколы обмена;
- контроллеры сбора данных;
- параметры;

- архивы;
- пользовательские интерфейсы;
- управление модулями;
- специальные функции;

Именно модульность подсистем определяет модульность системы в целом. Модульными являются подсистемы:

- базы данных;
- транспорты;
- протоколы обмена;
- контроллеры сбора данных;
- архивы;
- пользовательские интерфейсы;
- специальные функции;

Исходя из принципа модульности, указанные выше подсистемы могут свободно изменять свою функциональность путём подключения модулей соответствующего типа. Модульное ядро системы OpenSCADA, выполняется в виде статической и динамической библиотек. Это позволяет встраивать функциональность системы в существующие программы, а также создавать новые программы на основе ядра OpenSCADA. Кроме того, модульное ядро является самодостаточным и может использоваться самостоятельно посредством небольшой инициализирующей программы.

Модули хранятся в совместно используемых библиотеках. Причём одна библиотека может содержать несколько модулей. Данный метод хранения является основным, поскольку поддерживается практически всеми современными ОС. Хотя это не исключает возможности разработки других методов хранения модулей.

Проектирование системы OpenSCADA, начато в сентябре 2002г на основе опыта работы в сфере автоматизации технологических процессов. Реализация системы OpenSCADA, начата в феврале 2003г. На 27.07.2004 проект находится в стадии разработки и имеет версию 0.2.5

Разработчики: Савоченко Роман Алексеевич (rom_as@fromru.com)
Сторонние эксперты: Бортник Тимофей Владимирович (timbortnik@hotbox.ru)
Спонсоры проекта: Научно Инновационное Предприятие «DIYA» (diyaon@alice.dp.ua)

10:25–10:50

Андрей Паскаль Йошкар-Ола,
Валерий Гражданкин Оренбург,

Проект: Ананас http://ananas.lrn.ru

Открытая платформа для систем бухгалтерского и оперативного учёта

Аннотация:
Доклад кратко освещает цели, задачи и историю проекта по созданию открытого программного обеспечения автоматизации учетной деятельности предприятий, знакомит с состоянием разработок, осуществляемых в настоящее время в рамках проекта. Доклад содержит элементы анализа рынка свободного программного обеспечения учётной деятельности как в России (СНГ) так и за рубежом. Доклад будет интересен всем, кто интересуется направлениями развития и перспективами проектов по разработке свободного программного обеспечения. Особый интерес доклад представляет для представителей компаний, планирующих активное использование свободного программного обеспечения в своей информационной инфраструктуре и для автоматизации своих бизнес-процессов.

Краткая история проекта

Программа «Товарная наценка» на Borland Delphi. Первая реализация для Gnu/Linux. Tcl/Tk и xbase — платформа xbtk. Лицензирование и переход на QT/MySQL. Новый этап — полнофункциональная, настраиваемая, многоплатформенная среда. История сотрудничества вольных программистов.

Свободные проекты учётных систем

Зарубежные проекты: SQL Ledger, GnuCash, GnuE, OpenMFG.

SQL Ledger — наиболее интересный с точки зрения реального бизнес-применения проект на сегодняшней день. Программное обеспечение распространяется на условия GNU GPL лицензии. Система имеет богатый набор реализованных функций и успешно применяется на практике многими пользователями. Проект имеет высокие темпы развития уже на протяжении нескольких лет. К специфике можно отнести изначальную ориентацию системы на бухгалтеров, что выражается в сквозной увязке всех операций с бухгалтерским планом счетов. То есть, прежде всего, это система именно бухгалтерского учёта, хотя на сегодняшний момент система имеет гораздо больше возможностей, чем ведение проводок по счетам. В том числе поддерживается управление запасами, ведение расчётов с контрагентами, выписка счетов, оформление заказов, расчёт налогов, производственный модуль (сборка). Система функционирует только в режиме web-приложения, может быть установлена как на GNU/Linux так и на MS Windows платформе, имеет ограниченные возможности по настройке, локализованный на многие языки интерфейс пользователя. Как отмечают участники проекта, в системе пока нет модулей начисления зарплаты и CRM, но их появления можно ожидать в скором времени. Автор и создатель программы работает над проектом полный рабочий день, чем и объясняется динамика развития проекта в настоящее время.

GnuCash — известная по многим дистрибутивам Gnu/Linux система персонального учёта финансов. Успешно развивается, имеет богатый набор реализованных функций.

GnuE — один из наиболее амбициозных проектов, не имеющий в настоящий момент реальных внедрений и не готовый к реальной эксплуатации. Проект проходит стадию создания инструментария — настраиваемой прикладной платформы, которая должна обеспечить базу для разработки прикладных модулей.

OpenMFG — использует смешанную модель лицензирования и хотя имеет определённые успехи, не может рассматриваться как успешный OpenSource проект в области автоматизации.

Российский рынок — уроки Нау

На российском рынке все ещё не появилась свободная система, реальное использование которой представляло бы интерес для широкого

бизнес-сообщества. Тем не менее, «Гонка на черепахах» продолжается, и периодически появляются новости о создании новых проектов или о прогрессе в уже существующих. Из наиболее интересных событий, произошедших в недавнем времени на российском рынке автоматизации, затрагивающих тему свободных систем, можно назвать попытку компании НауМен оседлать нарастающую волну популярности открытых систем применительно к зарабатыванию денег на рынке автоматизации бизнес-процессов предприятий.

В момент начала рекламной кампании НауМен у одного из авторов было опасение, что реклама свободного распространяемого учётного ПО может превратиться в антирекламу, как только заказчики получат негативный опыт работы с новоиспечённой компанией. К большой радости свободного сообщества, этого не произошло благодаря тому, что, столкнувшись с трудностями окупаемости своей рекламной кампании и продвижения своего бренда, НауМен просто-напросто отказался от свободного распространения того, что называлось продуктами, и вернулся к традиционной бизнес-модели на основе проприетарного ПО.

Можно сделать вывод, что попытка НауМен получить в короткий срок значительную прибыль от продажи свободных продуктов для автоматизации и их поддержки не увенчалась успехом. Думаю, объяснением тому могут быть несколько причин. Но прежде чем их назвать, хочу сказать, что возникшую ситуацию я бы не стал рассматривать как неудачу организации бизнеса на основе открытых продуктов для автоматизации. Дело в том, что отсутствовало ключевое звено такого бизнеса — открытый продукт, хорошо известный, популярный и свободно доступный. Так что главной причиной неудач НауМен на рынке свободного ПО, на мой взгляд, явилось непонимание руководством компании его специфики. Для того, чтобы бизнес на открытом ПО мог случиться, одна существенная предпосылка — наличие широко известного, популярного продукта — является действительно необходимой.

Теоретические вопросы разработки систем учёта. Технические особенности и основные возможности прикладной платформы «Ананас»

Любая система автоматизации учёта, по сути, предназначена для регистрации, хранения и анализа событий, фактов и информации об объектах окружающего мира. И описать их можно по-разному.

В связи с этим при разработке системы автоматизации учёта возникают вопросы формализации информации о событиях и объектах окру-

жающего мира. Одним из способов описания является создание специальных структур хранения информации о структуре объектов, называемой «Метаданные».

Объект «Метаданные» описывает и хранит сведения о структуре регистрируемой в системе учёта информации, не зависящие от реализации хранилища. Минимальный набор метаобъектов, необходимый для реализации учёта может быть сведён к двум объектам: регистрации факта и хранения информации об элементах окружающего мира.

Базовые объекты системы учёта реализуются в виде объекта «Документ» — для регистрации фактов, событий и связанной с ними информации, и объекта «Каталог» (или справочник) — для хранения информации об элементах окружающего мира.

К вспомогательным объектам системы учёта можно отнести «Регистр сведений», «Регистр накопления», «Журнал документов», формы ввода/вывода данных, запроса и анализа информации, зарегистрированной в системе.

Вариантов практической реализации систем учёта множество. В части научного подхода к организации хранения данных мы использовали работы д. т. н. О. О. Варламова. В вопросах, относящихся к организации представления и хранения данных объектов в таблицах сервера реляционной БД (статьи, диссертация «Системный анализ и синтез моделей данных и методы обработки информации в самоорганизующихся комплексах оперативной диагностики», Москва, 2003), модели Чена («сущность—связь», ER-модель). При выборе подхода к созданию системы учёта мы сделали ставку на расширяемость и многоплатформенность. Следствием этого стал выбор инструментария разработки: QT, MySQL и Postgresql, GNU/Linux.

Основной проблемой, с которой мы столкнулись, стала реализация хранилища информации объектов системы учёта с использованием свободно распространяемых серверов баз данных, а также вопросы разграничения функций между сервером БД и прикладной программой.

В результате начала вырисовываться структура платформы «Ананас». В настоящее время платформа состоит из следующих приложений: Designer (инструмент разработки прикладной системы учёта) , GUI-engine («исполнитель» созданных в Designer приложений), WEB-engine (исполнитель web-приложений).

Состояние разработки платформы «Ананас» можно описать двумя словами: работа кипит. В настоящее время практически проработано большинство вопросов по концепции, набору и функциям основных объектов платформы, способов описания и хранения структуры метаданных

и экземпляров объектов системы учёта, создан дизайнер и исполнитель приложений. Активно разрабатываются базовые визуальные и невизуальные объекты платформы и инструментарий по их использованию и настройке. В процессе работы над платформой «Ананас» возникали и возникают самые разные вопросы и проблемы, в частности, слабое знание инструментария разработки новыми членами команды, нехватка опыта и теоретических знаний в отдельных областях, касающихся проекта.

10:50–11:15

Юрий Хныкин — Ижевск, ITK CLIP team

Андрей Черепанов — Москва, EAS team

Проект: CLIP, R2D2, EAS

Разработка систем автоматизации бизнеса

Аннотация:

1. CLIP — средство разработки
2. r2d2 — веб-платформа автоматизации бизнеса
3. E/AS — клиент серверная платформа
4. Перспективы развития систем автоматизации бизнеса.

CLIP — средство разработки

CLIP — это не только компилятор популярного языка Clipper на платформах Unix, это полноценная среда для разработки прикладных программ, включающая компилятор, препроцессор, библиотеки, почти полностью соответствующие проверенному годами Clipper, а также около 2000 функций из популярных библиотек и виджетсетов. Кроме того CLIP обеспечивает доступ к большинству распространённых SQL-серверов, поддерживает объектный стиль программирования, позволяет использовать динамические библиотеки и модули на байт-коде.

Основные проблемы при переносе унаследованных приложений с Clipper'а на CLIP связаны с различиями и особенностями операционных систем.

Дальнейшее развитие CLIP происходит в сторону использования современных технологий таких как:

- объектная база данных (CODB), позволяющая хранить объекты любого класса с любым составом атрибутов и размеров.
- объектный сервер приложений COBRA (CLIP Object Broker and Application Server), позволяющий исполнять бизнес-логику на серверной стороне, динамически подключать новые методы классов и бизнес-процессов без остановки работы сервера.
- пользовательский интерфейс, одинаково удобный для разработчиков, инженеров бизнес-процессов и, конечно, пользователей.

На основе этих возможностей CLIP строятся два проекта, имеющие в основе разные пути реализации пользовательского интерфейса и бизнес-процедур: r2d2 и E/AS.

r2d2 — web-ориентированная платформа для создания корпоративных информационных систем

Использует Mozilla как основу пользовательского интерфейса. Проект ориентирован на решение задач различного рода учёта, планирования и анализа: бухгалтерского, управленческого, финансового и т. д.. В качестве объектов учёта и анализа могут выступать любые объекты, имеющие структурированный набор атрибутов, а также различного рода процессы, протекающие с участием объектов учёта. Основное отличие r2d2 от аналогичных учётных программ в том, что всё это реализовано на ОБД и практически не имеет никаких ограничений по количеству классов данных или по количеству атрибутов в объектах.

Вся многомерная/многоуровневая аналитическая информация находится в специализированном OLAP, ориентированном на объекты и двойную запись. Вся аналитическая информация доступна из OLAP, что обеспечивает высокую скорость работы с учётной информацией и гибкость построения сложных запросов. Запрос клиента может быть сформирован и отражён в ответе с помощью обычных HTML-форм из любого браузера.

Для организации полноценного пользовательского интерфейса r2d2 разработан специальный модуль к Mozilla на базе форматов XUL, CSS, JS с набором библиотек для построения пользовательских или создания собственных расширений.

3. E/AS — клиент-серверная платформа

Ещё одним примером развивающегося проекта с открытым исходным кодом является проект E/AS (Enterprise automation system — система автоматизации предприятия). Платформой для реализации этого проекта был выбран CLIP. В рамках проекта реализованы следующие решения:

- разработана платформонезависимая библиотека интерфейса, поддерживающая различные виджетсеты через механизм драйверов;
- создан язык описания интерфейсов и печатных форм на базе XML и UIML;
- предусмотрена возможность автогенерации форм на основании метаданных базы;
- осуществлена возможность подключения модулей на байт-коде, загружаемых с сервера и подключение модулей к серверу базы данных, в том числе и конфигуратор автоматизируемых бизнес-процессов.

Указанные технологии обеспечивают гибкость платформы, крайне необходимую для проектирования таких задач бизнеса как задачи бухгалтерского учёта, задачи финансового учёта, документооборота, корпоративного и оперативного управления.

Перспективы развития систем автоматизации бизнеса

Сегодня сложилась реальная ситуация подрыва доминирования монополистов — производителей программного обеспечения, связанная с распространением ОС Linux на рынке настольных компьютеров. Эта же тенденция может затронуть и производителей систем автоматизации бизнеса, ориентированных только на Windows и не предпринимающих усилий по созданию кросс-платформенных приложений.

В этих условиях весьма вероятно появление новой мощной силы. Мы считаем, что такой силой может послужить сообщество прикладных программистов программ с открытым исходным кодом.

11:15–11:40
Михаил Шигорин Киев,

TYPO3: консистентная модульная система управления веб-контентом

Аннотация:

- Краткий обзор основных концепций и возможностей TYPO3 с точки зрения веб-разработчика;
- разделение обязанностей: логика, контент, дизайн;
- разделение привилегий: пользователи «оболочки» и «изнанки» сайта;
- наращивание возможностей: модули на основе общей БД;
- в реальности: открытый код, решения «под ключ»;
- сравнение с более поздними технологиями разделения контента и представления (например, XML/XSLT); возможности интеграции.

11:45–12:00: Кофе

Дневное заседание
12:00–14:15

Секция 1
(Председатель — Станислав Иевлев)

12:00–12:25

Алексей Воинов — Москва, ALT Linux

Проект: WindowMaker

Состояние разработки проекта WindowMaker

Аннотация:
В последнее время у многих могло сложиться впечатление о том, что проект WindowMaker заброшен и больше не развивается. На самом деле это не так. В докладе описывается текущее состояние проекта, а также планы обновлённой команды разработчиков.

WindowMaker — популярный менеджер окон для X11. Проект был начат Альфредо Койма в 1997 году сразу после выхода AfterStep 1.0, и предлагался в качестве одного из путей развития AfterStep. Однако, остальные разработчики не поддержали Альфредо.

К 2001 году вокруг проекта сложилась любопытная ситуация. В проекте выделилось «узкое» место — Дэн Паску. Из-за того, что он не доверял ничьему коду, он переписывал все присылаемые ему патчи, используя оригинал только как идею. Со временем он просто перестал успевать обрабатывать весь присылаемый ему материал. Если учесть возросшую занятость в «реальной жизни» у обоих программистов проекта, становится понятно, что развитие проекта начало замедлятся уже давно. Стороннему наблюдателю это стало заметно значительно позже. В конце 2002 года была выпущена последняя версия WindowMaker. К середине 2003 изменения в cvs практически прекратились.

На самом деле, основная работа окончательно переместилась туда, где ей всегда уделялось больше времени: к создателям патчей. Как это ни удивительно, несмотря на почти гарантированное непопадание

исправления в основную ветку проекта, люди продолжали их делать. Если собрать все патчи, включая те, которые были созданы уже после прекращения изменений в проекте, то получится вполне современный менеджер окон. В настоящее время в пакете WindowMaker, который попадает в Sisyphus используется 36 патчей от разных разработчиков.

В декабре 2003 года я рассказывал о патчах для WindowMaker и об их создателях на виртуальной конференции Umeet. Я говорил, что наличие патчей — это очень хорошо, поскольку майнтейнер проекта может вовсе не желать поддерживать чей-то чужой код. Однако «простые» пользователи могут при желании получить нужную им функциональность, общаясь только с создателем патча. При этом все остаются довольны: майнтейнер проекта доволен, потому что ему не добавляется лишней работы, автор патча доволен, потому что его работа заметна и не тонет в тысячах строк основного проекта. В этом смысле, проект WindowMaker, пожалуй, уникален. Я не встречал больше нигде патчей, не вошедших в основную ветку, которые бы распространялись в официальном тарболе.

Наличие большого количества патчей от разных разработчиков усложняет работу тем, кто хочет применить их все. Обновление патчей — серьёзная задача, требующая много времени.

В начале 2004 года на irc-канале #windowmaker выделилась группа разработчиков, желающих нормального развития проекта. Был создан репозиторий, в котором началась работа по созданию нового оконного менеджера на базе существующего кода WindowMaker. Приблизительно через неделю после начала работы старая команда разработчиков WindowMaker была поставлена в известность о новом проекте. В результате длинного разговора было решено новый проект не создавать, а продолжить разработку WindowMaker силами обновлённой команды.

Разработка патчей, на самом деле, накладывает значительные ограничения на тех, кто их пишет. Мало кто будет озабочен прикладыванием к своей версии программы патча, который не добавляет какого-то заметного изменения. Это обстоятельство не способствует улучшению качества кода основной версии, если за этим не следит майнтейнер. Именно поэтому первоочередной задачей новой команды стала чистка кода.

Другие (сформулированные) задачи включают:

- адаптацию и интеграцию уже разработанных патчей в основную ветвь проекта;

- поддержку динамической загрузки модулей, добавляющих новые возможности;
- улучшение интерфейса пользователя, включая все диалоговые окна и программу конфигурирования.

Адаптация патчей необходима, потому что многие из них разработаны так, чтобы вносить минимум исправлений в существующий исходный код, а это далеко не всегда приводит к лучшим решениям. Разделение на модули позволит сделать WindowMaker намного более гибким и позволит изменять набор возможностей во время работы без перекомпиляции. Сейчас, например, есть вариант компиляции lite, который отключает довольно много кода, но это невозможно изменить во время работы. У существующего интерфейса есть масса недостатков. Например, при запуске WindowMaker с переводом некоторые переведённые надписи не умещаются в отведённых им позициях.

Важно, чтобы отношения новой команды с пользователями складывались не хуже, чем у старой команды. Пока это удаётся, но прошло ещё слишком мало времени, чтобы говорить наверняка. Нам ещё не присылали новых патчей, мы ещё не выпустили ни одной новой версии. Версии WindowMaker из новой ветви начнут выходить только после того, как будет выпущена версия 1.0.

12:25–12:50

Денис Слюсарев

Москва, ainmarh lab

Проект: QPLATFORM

Основы QPLATFORM

Аннотация:
QPLATFORM — это комплекс программных средств для разработки CGI(fastcgi)-приложений, рассчитанных на высокую нагрузку. Главными достоинствами проекта являются: язык разработки — C, встраиваемые в бинарный код программы html-шаблоны, динамическое кэширование контента (с вероятностью попадания в кэш около 90% при типичных задачах) и система управления пользовательскими сессиями.

В комплекс входят:

- FastCGI система организации работы CGI-приложений, как отдельных программ (демонов/сервисов), которые после запуска постоянно находятся в памяти и обрабатывают приходящие к ним от web-сервера запросы.
- QTP (или Q-Lang) препроцессор шаблонов, рассчитанных на высокую нагрузку и оптимизированных для встраивания непосредственно в код программы.
- CGIX — модифицированная версия CGIC, оптимизированная для работы с FastCGI и QTP (или Q-Lang).
- QCM — система динамического кэширования.
- QSM — система контроля пользовательских сессий.
- QDBC — система, унифицирующая работу с базами данных (в данный момент находится в стадии разработки).

Преимущества использования QPLATFORM:

- Высокая скорость работы приложений.
- Наличие необходимого инструментария для быстрого построения cgi-приложений.
- Возможность post-кэширования обработанных запросов.
- Возможность автоматического динамического кэширования (libqcm).
- Возможность получения «Content-Length» для результата работы, а, следовательно, возможность использования http-акселераторов на полную мощность.
- Система QSM (session manager) позволяет с помощью сессий решать проблемы безопасности.

Основной причиной появления проекта QPLATFORM является отсутствие на данный момент инструментария для создания в короткие сроки быстрых cgi-приложений на языке C. Существует достаточно большое количество шаблонных библиотек, XML/XSL-процессоров, которые являются очень громоздкими и непроизводительными. Так, при создании даже небольших проектов программисты сталкиваются со

сложной дилеммой: если выбрать интерпретируемый язык как платформу для создания web-приложений, то при возрастании уровня посещаемости появятся проблемы с производительностью; при выборе же компилируемого языка разработка займёт очень много времени, которое, возможно, будет потрачено даром, если проект не будет иметь успеха. Компромисса не существует, поэтому приходится жертвовать либо одним, либо другим. А QPLATFORM можно рассматривать как золотую середину: длительность разработки не больше, чем при разработке на PHP, однако скорость выше, чем при создании приложения с использованием большинства существующих библиотек для C.

Существует несколько причин низкой производительности современных web-приложений. Самое главное — это база данных (будь то MySQL, PostgreSQL, или даже Oracle), которая занимает большую часть ресурсов сервера. Вторая причина — использование интерпретируемого языка (PHP, Perl, Python и др.), И третья — низкая скорость работы самой системы, например, запуск приложения при каждом запросе (если мы избрали путь CGI).

Обойти все эти проблемы или свести их долю к минимуму можно следующим образом.

1. Чтобы избежать чрезмерного использования базы данных, нужно кэшировать контент, который сохраняет своё наполнение между изменениями, то есть построить правила, определяющие принадлежность, наличие и устаревание кэша для конкретной страницы. QCM (QPLATFORM Cache Manager) позволяет делать это посредством определения префикса и имён GET/POST полей, по значениям которых будет определяться местоположение данной страницы в иерархии кэша; а также с помощью определения имени тестера (timestamp), позволяющего разбить кэш на группы, чтобы при внесении изменений в какую-то конкретную группу можно было объявить кэш этой группы устаревшим.

   Единственное ограничение, с которым сталкивается разработчик приложения QPLATFORM, — это невозможность кэшировать страницы, показываемые залогиненным пользователям (т. к. кэширование производится для всей страницы целиком); конечно, если не используются html-frame'ы или не задействован механизм обновления данных через JavaScript, то есть чтобы страница целиком оставалась одна и та же. Но это ограничение мы планируем в ближайших версиях обойти.

2. Использование языка C позволяет достичь максимальной скорости обработки данных.

3. В случае с QTP приложение запускается только в самом начале, а не для каждого запроса заново, и обрабатываются запросы в бесконечном цикле. Чтобы не тратить время на парсинг шаблонов, превратим, используя QTP (QPLATFORM Template Preprocessor), .html-шаблон в обычную программу на C и, согласно правилам QPLATFORM, опишем call-back процедуры для заполнения шаблона данными. При правильном построении QTP шаблонов можно добиться полной независимости друг от друга дизайна и cgi-приложения. QSM (QPLATFORM Session Manager) позволяет хранить сессии пользователей в Shared Memory, что также ускоряет работу web-приложения.

И, наконец, рассмотрим то, что волнует менеджеров, которые явно не хотят долгое время платить программистам и не видеть результатов, — скорость разработки. Стандартное web-приложение должно:

1. Получить данные из базы.

2. Произвести некие преобразования данных.

3. Вывести данные на страницу.

4. Иметь возможность взаимодействовать с системой.

Для этого как раз и создавалась система QPLATFORM. Она позволяет:

1. Получить данные из базы через QDBC (QPLATFORM DataBase Connector), воспользовавшись удобным унифицированным интерфейсом.

2. Произвести над данными преобразования посредством языка C. Описать правила кэширования контента.

3. Воспользоваться QTP для создания гибких и удобных в использовании шаблонов.

4. Получить возможность взаимодействия с системой по умолчанию.

12:50–13:15

Артём Кастелин, Александр Ковтушенко Москва,
МГТУ им. Н.Э. Баумана

Инструмент для визуализации трассы выполнения параллельной программы — TV 1.0

Аннотация:
Предлагаем инструмент визуализации истории выполнения трассы параллельной программы, выполняющейся в среде MPI. Инструмент представляется полезным для поиска возможностей оптимизации параллельной программы.

Параллельное программирование все ещё остаётся экзотической областью, редко используемой в прикладном программировании. Возможности для организации параллельных вычислений существенно выросли — прежде всего в виде «железа». Реально доступны (в том числе и не слишком дороги) решения в виде «кластеров», т. е. вычислительных систем, построенных на серийных процессорах, и специально для этого применения производимых коммутаторов. Имеется целый ряд сред программирования для подобных систем, причём необходимый набор инструментов доступен и в виде открытого программного обеспечения.

Эти средства остаются на наш взгляд невостребованными по двум взаимообусловленным причинам: во-первых, разработка практичного прикладного параллельного алгоритма требует специальных навыков, вероятно — специальной подготовки. Во-вторых, слаб или почти отсутствует спрос на вычислительно ёмкие прикладные решения, т. е. нет постановки задачи, требующей параллельного счета. Таких постановки могут быть связаны с самыми разными проблемами промышленности или задачами управления: замена натурных испытаний вычислительным экспериментом, оптимизация управляющего воздействия и т. д. Недостаток специалистов, способных формулировать такие постановки задачи и решать их, является существенным препятствием.

Используемый нами набор средств для создания параллельного программного обеспечения включает в себя: средство для создания параллельных программ на основе стандарта MPI (в основе — библиотека реализующая программный интерфейс MPI), системные средства выполнения полученной параллельной программы на кластере. Стандарт MPI предлагает широкий выбор средств коммутации и синхронизации.

Потребность в параллельной программе появляется тогда, когда вычислительная ёмкость задачи высока, поэтому эффективность параллельного алгоритма, эффективность использования аппаратуры вычислителя — важнейшая цель. Её очень проблематично достичь «сходу», не варьируя параллельный алгоритм. Отладка параллельной программы (по большей части) — это последовательность попыток приблизиться к приемлемой скорости счета. Одна из основных проблем — процесс выполнения программы имеет существенно больше (по сравнению с «обычной» последовательной программой) факторов, влияющих на динамику вычислений.

Одним из ценных средств экономии сил в этой борьбе является предлагаемая пара инструментов — библиотека для автоматической (точнее — «почти автоматической») фиксации событий времени выполнения параллельной программы + диалоговая среда для просмотра/анализа накопленных данных.

Имеется достаточно длинный список подобных инструментов. Сошлёмся на две из них. Во-первых, это профилировщик+визуализатор Vampir немецкой компании Palas — продукт наиболее распространённый, по отзывам, по документации — имеющий много достоинств, но коммерческий. Во-вторых, это визуализатор ParaGraph (восходит к 1990 г.), в настоящее время в качестве трассировщика использует библиотеку MPICL (разрабатываемая Patrick H. Worley с 1997 г в Oak Ridge National Laboratory) — оба продукта являются открытыми. Однако использование ParaGraph не слишком комфортно. Программа реализована на базе непосредственно X Window, на наш взгляд ей недостаёт средств управления.

Предлагаемый нами визуализатор TV 1.0 также как ParaGraph использует библиотеку MPICL, имеет при этом следующие отличия:

- Визуализатор разработан на основе среды Qt, что позволило сделать его более удобным в работе (при этом система Vampir рассмотрена в качестве аналога).
- Имеется в том числе и сборка визуализатора для выполнения на ОС MS Windows (трассы формируются в виде текстовых файлов, т. е. переносимы), что позволяет использовать её «любому пользователю».

Надеемся, что TV 1.0 окажется полезным дополнением в ящике инструментов разработчика параллельных прикладных программ на основе MPI.

13:15–13:40

Виктор Капустин, Анна Корсун Санкт-Петербург,
Санкт-Петербургский Государственный университет

Проект: Web-based IDEF Functional Modeler
http://sourceforge.net/projects/web-based-idef/

Инструмент для функционального моделирования

Аннотация:
Функциональное моделирование (IDEF0) использует несложный графический язык. Программные средства поддержки этого языка, однако, достаточно дороги (в основном, за счет избыточной функциональности) и работают, в основном, на платформе MS Windows. Мы используем браузер для обеспечения минимально необходимых для создания IDEF0 диаграмм возможностей, создавая кросс- платформенную среду моделирования. Программный инструментарий — HTML, JavaScript и CSS. Современные (почти-) DOM-совместимые браузеры позволили изолировать несовместимости в отдельный модуль-обертку. Мы отказались от внешне привлекательного пути кодирования HTML-элементов непосредственно в JavaScript и использования свойства innerHTML. Вместо этого мы строго разделяем код и разметку, выделяя графические элементы в отдельные шаблоны, каждый из которых хранится в отдельном HTML-документе, подгружаемом в свой (обычно невидимый) iframe. Это заметно усложняет програмирование, но, мы надеемся, позволит привлечь независимых дизайнеров для качественной прорисовки элементов интерфейса. Поскольку компоненты разрабатываемого ПО можно использовать в составе других программных средств, распространение его планируется осуществлять под GLPL.

Функциональное моделирование (IDEF0) использует несложный графический язык. Программные средства поддержки этого языка, однако, достаточно дороги (в основном, за счёт избыточной функциональности) и работают, в основном, на платформе MS Windows. Мы используем браузер для обеспечения минимально необходимых для создания IDEF0 диаграмм возможностей, создавая кросс-платформенную среду

[1]Работа частично поддержана Российским фондом фундаментальных исследований, грант 03-07-90299.

моделирования. Программный инструментарий — HTML, JavaScript и CSS. Современные (почти-) DOM-совместимые браузеры позволили изолировать несовместимости в отдельный модуль-обёртку.

Мы отказались от внешне привлекательного пути кодирования HTML-элементов непосредственно в JavaScript и использования свойства innerHTML. Вместо этого мы строго разделяем код и разметку, выделяя графические элементы в отдельные шаблоны, каждый из которых хранится в отдельном HTML-документе, подгружаемом в свой (обычно невидимый) iframe. Это заметно усложняет программирование, но, мы надеемся, позволит привлечь независимых дизайнеров для качественной прорисовки элементов интерфейса.

Поскольку компоненты разрабатываемого ПО можно использовать в составе других программных средств, распространение его планируется осуществлять под GLPL. Обсуждается возможная смена базовой технологии проекта (XML, CSS3).

Функциональное моделирование (IDEF0) находит своё применение в отраслях, связанных с созданием сложных организационных и информационных систем, поддержкой жизненного цикла сложных систем и изделий. IDEF0 был предложен более 30 лет назад в авиакосмической промышленности США, и стандартизован в США[1] и в России[2]. В то время как эти стандарты доступны свободно, программные средства поддержки IDEF0 выпускаются рядом известных фирм. Эти программные средства имеют тяжеловесные графические интерфейсы, перегружены различными факультативными функциями, дороги, что является препятствием для их массового применения.

Графический язык IDEF0, в основном, ограничен горизонтальными и вертикальными прямыми линиями и фигурами, образуемыми такими линиями. Такая графика легко реализуется в HTML, что привело к идее разработки инструмента поддержки создания IDEF0-диаграмм в виде веб-приложения.

Аналитик, создающий функциональную модель, должен иметь возможность манипуляции графическими объектами модели (прямоугольниками, линиями) и текстовой информацией, связанной с этими объектами. Объем этой информации и данных о положении объектов может быть значительным, и реализация поддержки каждого действия аналитика в виде серверных программ (скриптов) сделает работу аналитика

[1]Integration Definition for Function Modeling (IDEF0)/FIPS Publication 183. — 1993

[2]Рекомендации по стандартизации Р50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции: Методология функционального моделирования. — Госстандарт, 2001

крайне некомфортной из-за полной перерисовки окна браузера. Наш подход к созданию инструмента для функционального моделирования заключается в реализации поддержки манипуляции объектами модели непосредственно в браузере (HTML-документе) с помощью скриптового языка ECMAscript (JavaScript 1.5). Использование именно этого языка связано с практически повсеместной его поддержкой браузерами всех основных производителей (Mozilla/Netscape, Microsoft Internet Explorer — MSIE, Opera) на многих платформах (Windows, Linux/Unix, Mac).

Браузеры обеспечивают три различных уровня программных интерфейсов (API), позволяющих изменять отображаемый документ: (1) непосредственный вывод HTML-текста в документ (document.write()), (2) изменение содержимого выбранного элемента документа (innerHTML) и (3) стандартизованные не только для браузеров API объектной модели документа (методы DOM). Первоначально одним из авторов был создан ранний прототип инструмента для функционального моделирования с использованием второго из перечисленных подходов. Компоненты графического интерфейса пользователя (ГИП) задавались в виде программных констант. Этот подход, эффективно работающий для несложных интерфейсов в MSIE, оказался по нескольким причинам совершенно неработоспособным в Mozilla. Во-первых, изменение строковых значений атрибутов элемента документа путём изменения свойства innerHTML в Mozilla (в отличие от MSIE) не отражаются в дереве DOM документа (за исключением ссылок на обработчики событий, где все обстоит ровно наоборот: изменение названия функции — обработчика события в Mozilla приводит к изменению обработчика, а в MSIE — нет). Во-вторых, Mozilla вообще более строго следует стандарту DOM, в соответствии с которым любое изменение свойства innerHTML любого элемента документа порождает формально новый документ.

В результате прототип инструмента для функционального моделирования был переработан с использованием исключительно DOM API. Для каждого графического компонента проекта (блока IDEF0, панелей свойств блока/объектного потока и пр.) используется отдельный HTML-документ. Такой документ загружается в невидимый потоковый кадр (iframe) и используется как шаблон для порождения видимых компонентов ГИП. Это позволяет полностью отделить дизайн элементов ГИП от кода. Оказалось, что стандарт DOM 2 не полностью поддерживается даже последними (IE 6.1, Mozilla 1.7) версиями браузеров (в частности, в части передачи информации о событиях их обработчикам, копирования поддеревьев между кадрами и др.), что пока задержива-

ет завершение отладки части ГИП, связанной с перемещением блоков и изменением их размеров. Для сохранения результатов формирования функциональной модели написан простой серверный скрипт на языке Perl.

Для превращения разработанного прототипа в реальный инструмент моделирования, обладающий минимальной необходимой функциональностью, предстоит решить ещё несколько задач. Можно продолжать реализацию ГИП инструмента для функционального моделирования в HTML/XHTML, однако мы планируем исследовать возможность отказаться от HTML и перейти к использованию XML/XSLT/CSS, что позволит упростить разработку конвертеров в промышленные коммуникативные стандарты, используемые для обмена данными о IDEF0 функциональных моделях: IDL и XML, а также стандартизовать те элементы ГИП, которые используются для задания параметров моделей.

Текущее состояние кода размещено в CVS проекта Web-based IDEF Functional Modeler (`http://sourceforge.net/projects/web-based-idef/`). Используемая Open Source лицензия — GLPL, поскольку компоненты разрабатываемого ПО можно использовать в составе других программных средств.

13:40–14:05

Александр Сенько Москва, БЕН РАН / МГТУ им. Н.Э. Баумана

Михаил Якшин Москва, ALT Linux / МГТУ им. Н.Э. Баумана

Открытая система виртуальной интеграции гетерогенных баз данных на основе технологии XMPP/Jabber

Аннотация:

Настоящая система представляет собой систему виртуальной интеграции гетерогенных баз данных и предоставляет пользователям — унифицированный интерфейс доступа к подключенным к системе базам данных, а поставщикам данных — способ интеграции их информационных ресурсов в систему. В качестве транспортного протокола используется XMPP/Jabber.

В настоящее время людям, обращающимся к сети Интернет при поиске информации, приходится работать со множеством разнообразных баз данных, сильно различающихся между собой по структуре, тематической направленности, полноте представляемых данных. Многие организации, сталкивающиеся с необходимостью предоставления доступа к своим информационным ресурсам, разрабатывают специальные системы для работы с конкретными информационными массивами.

Пользователям в таком случае предоставляется специализированный интерфейс доступа к информации, содержащейся в поддерживаемой базе данных, следовательно, при работе с такими системами приходится сталкиваться с различными подходами к поиску и навигации. Зачастую большая часть времени тратится на изучение интерфейса, предоставляемого системой, и его возможностей, а не на работу с данными. В случае необходимости поиска по нескольким массивам данных пользователю приходится несколько раз составлять различные варианты одного и того же поискового запроса к каждой из используемых систем, а потом вручную объединять полученные данные.

Эти причины обуславливают целесообразность разработки программно-организационного комплекса, предоставляющего универсальный интерфейс доступа к базам данных информационных ресурсов, различным по структуре и содержанию.

Одним из наиболее интересных подходов к интеграции разнородных БД в настоящее время считается виртуальная интеграция данных. Суть ее состоит в следующем: для каждой БД создаётся программа-посредник, преобразующая запросы из принятого в системе единого (глобального) формата в запросы, специфичные для данной базы, и обратно — результат обработки запроса преобразуется в тот же единый формат. Таким образом, пользователь реально работает с одним форматом данных при обращении к различным базам.

Сейчас на смену бинарным форматам приходят более гибкие, самым перспективным из которых является XML; появляются и новые подходы к организации транспортного уровня для систем подобного класса. В качестве одного из таких решений выступает протокол XMPP/Jabber — не так давно разработанный на базе системы мгновенного обмена сообщениями (сходной с популярной ICQ), а сейчас эволюционировавший в полноценный транспортный протокол XML-роутинга.

На данный момент создан действующий минимальный прототип системы, предоставляющий универсальный интерфейс доступа к гетерогенным библиографическим базам данных. На этом этапе не рассмат-

ривались проблемы объединения результатов распределённого поиска, полученных от нескольких баз данных, их ранжирования и т. п.

Основные задачи проекта:

- организация серверной части проекта — транспорта для Jabber-сервера и модуля-конвертера для него, обеспечивающего работу с одной базой данных;
- создание удобного интерфейса клиентской части системы для создания поисковых запросов и отображения результатов в соответствии с ГОСТ 7.1-84;
- разработка формата и протокола обмена данными между клиентом и сервером.

Серверная часть разработанной системы состоит из ядра, отвечающего за проведение политики безопасности, динамическую загрузку внешних модулей и предоставление метаинформации и модулей, содержащих конвертеры внутреннего представления данных из обслуживаемых БД в глобальный формат и обратно, а также средства работы с БД.

Для описываемой системы был разработан специализированный Jabber-клиент, предоставляющий универсальный интерфейс для составления поисковых запросов, базирующийся на логике глобального формата. Пользовательский интерфейс выглядит следующим образом: в окне строится дерево всех сущностей, имеющих соответствующие элементы XML в формате. Каждому узлу дерева пользователь может поставить в соответствие строку поиска и некоторые атрибуты, семантика которых описана в требованиях к глобальному формату.

Разработанный поисковый интерфейс довольно универсален и позволяет одинаково легко вести поиск как по хорошо структурированным данным, так и по слабоструктурированным, благодаря следующим правилам:

1. Строка поиска, заданная в узле, распространяется на все потомки этого узла. Таким образом, мы получаем два крайних варианта: задав строку в листе дерева, мы можем искать точное совпадение заданного поля, а задав строку в корне дерева, мы задаём поиск по всем полям, и множество промежуточных вариантов, задающих поиск по некоторой группе полей.
2. Для каждого поля можно задать фильтрацию по некоторым атрибутам. Это позволяет, например, найти все публикации, к

которым имеет отношение заданный человек в различных качествах, или только монографии заданного автора.

3. Каждая частная база данных может предоставлять лишь некое подмножество полей, определённых в глобальном формате, или предоставлять некоторую информацию в слабоструктурированном виде. Если база не предоставляет поле, к которому относится запрос, то поиск будет проводиться в родительском элементе (где может быть некое более общее описание).

Описанные подходы были реализованы в виде системы, обеспечивающей поиск и передачу библиографической информации. Эта система может рассматриваться в качестве прототипа для исследования возможностей создания более сложных систем с использованием вышеназванных технологий. Как показала практика, использование в качестве транспортного уровня XMPP/Jabber значительно упрощает построение подобных систем и предоставляет достаточно много дополнительных возможностей по сравнению с другими известными подходами.

Секция 2

(Председатель — Александр Боковой)

12:00–12:25

Александр Котельников Санкт-Петербург, ALT Linux

XML как универсальное хранилище данных

Аннотация:
Доклад о проведённой разработке формата государственной документации, основанного на XML, и его реализации на базе OpenOffice.org.

В рамках НИОКР, проводимых ALT Linux по госконтракту «Разработка рекомендаций и предложений по использованию стандартов, необходимых для распространения ИКТ. Разработка предложений по развитию в Российской Федерации рынка программного обеспечения со свободной лицензией. Разработка методических рекомендаций по использованию программного обеспечения со свободной лицензией в деятельности органов государственной власти Российской Федерации», был

создан основанный на XML формат и его пилотная реализация в виде XML-фильтра из и в OpenOffice.org XML File Format.

Решение базировать реализацию ГОСТовского формата на платформе OpenOffice.org было принято поскольку, в частности, это чуть ли не единственный полноценный кроссплатформенный WYSIWYG-текстовый процессор. Также положительными факторами, послужившими в пользу такого выбора явились open source лицензия OpenOffice.org и то, что его базовые форматы — в частности, формат текстовых документов — являются XML-форматами. Встроенный в OpenOffice.org XSTL-процессор позволил свести всю работу по реализации к написанию двух XSL-преобразований.

В докладе будет рассказано об основных, наиболее важных, моментах проделанной разработки, как самого формата, так и его реализации, а также о предшествовавшим разработке обсуждениям, приведшим к выбору XML в качестве основы.

Предполагается рассказать в сравнительной форме о форматах OpenOffice.org XML File Format и HTML + CSS2.

Ни одна технология, какой бы замечательной она ни была, не может не иметь минусов и недостатков, ограничивающих её применение. XML не является исключением. Будет рассказано и о некоторых неудачных свойствах этого рода, присущих XML.

Планируется проиллюстрировать некоторые технические подробности простыми примерами.

12:25–12:50

Валентина Ванеева

Томск, ALT Linux

Проект: Mozilla.ru

Разработка XPCOM-компонентов Mozilla для работы с базой данных SQLite 3

Аннотация:

В работе рассказывается о проекте по расширению существующего модуля sql для платформы Mozilla. Автором была реализована поддержка локальной базы данных SQLite, а также были сделаны тестовые сборки Mozilla, включающие в себя эту базу данных. Также

рассматриваются возможные применения связки Mozilla и SQLite и sql-модуля в целом.

Почти всем известно, что Mozilla (мы рассматриваем Mozilla Application Suite) — пакет приложений для работы в Интернете. Но, кроме этого, Mozilla — ещё и многофункциональная платформа для разработки приложений, и важнейшие её технологии — XPCOM, XUL, JavaScript, RDF.

XPCOM (кроссплатформенное общее управление объектами) — механизм для управления компонентами Mozilla (какими именно, можно посмотреть в подкаталоге components/ каталога установки Mozilla). XPCOM-объекты — это самый нижний уровень Mozilla. Если быть короче, XPCOM немного напоминает CORBA и очень похож на COM. XUL (язык описания интерфейсов на основе XML) — язык разметки, предназначенный для быстрого и удобного создания графического интерфейса пользователя. Наконец, RDF (схема описания ресурсов; тоже подмножество XML) используется для хранения различной служебной информации. С помощью RDF реализуются, например, шаблоны XUL.

Именно эти технологии делают Mozilla платформой для эффективной разработки и нужны для описываемого проекта. В то же время этой платформе ощутимо не хватает средств для работы с системами управления реляционными базами данных (СУБД). На данный момент между Mozilla и СУБД почти всегда есть какой-то промежуточный уровень, обычно это веб-сервер, общающийся с Mozilla и СУБД посредством сценариев. Кроме того, есть проекты вроде MozPHP (http://mozphp.mozdev.org/), позволяющие использовать PHP локально в том числе и для работы с базами данных. Более «правильный» способ — добавление средств работы с базами данных сразу в Mozilla в виде XPCOM-компонентов. Наличие поддержки SQL в Mozilla значительно расширило бы возможности и пользователей, и разработчиков.

Веб-приложения обычно разрабатываются таким образом, что состоят из трёх частей: клиента, сервера базы данных и между ними — веб-сервера. Проблема этой связки в том, что такие функции, обычно выполняемые веб-сервером, как аутентификация/авторизация, также дублируются и сервером базы данных. Поэтому связка из двух частей: клиент и сервер базы данных — представляется более эффективной.

Добавить такую поддержку попытался Ян Варга, автор относительно недавно появившегося дереве CVS Mozilla модуля SQL (http://mozilla.org/projects/sql/). Этот модуль реализует единый интер-

фейс для доступа к различным базам данных, но пока присутствует поддержка только PostgreSQL. В число функций этого модуля входят:

- подключение к базе;
- формирование и отправка SQL-запроса;
- получение результатов и их представление в формате RDF;
- возможность вызова перечисленных выше функций через JavaScript.

Этого достаточно для быстрой разработки приложений, использующих СУБД.

Существуют и другие проекты, реализующие работу с СУБД: MySQLClient (`http://mysqlclient.mozdev.org/`) и MozSQLXPCOM (`http://mozysqlxpcom.mozdev.org/`), но они предназначены для работы только с MySQL.

Помимо работы с удалёнными базами данных представляется также интересно и работа с локальными базами, не требующими подключения к сети. Яркий пример такой СУБД — SQLite (`http://www.sqlite.org/`). Среди её преимуществ:

- она имеет небольшой размер и может входить в установочный комплект Mozilla;
- она кроссплатформенная;
- поддерживает большое подмножество SQL, а это обеспечивает переносимость на клиент-серверные реляционные БД;
- локальную базу данных можно копировать и перемещать с машины на машину как обычный файл;
- она популярна, входит в комплект поставки PHP;
- она не требует настройки и специальных усилий по администрированию;
- авторы отказались от копирайта.

Область применения локальных баз — работа со словарями, справочными системами, системами сбора статистики, работа в offline, распространение данных и т. д.

Именно поэтому в рамках данного проекта в модуле SQL автором была реализована поддержка SQLite. То есть был разработан XPCOM-компонент для SQLite, работающий с локальной базой SQLite, который был включён в сборку Mozilla для трёх платформ: Windows, Linux, Mac OS X.

Проделанная работа может оказаться полезной, например, проекту EVM (`http://evm.mozdev.org/`), где для хранения почты используется база данных, причём работа с ней осуществляется через модуль sql. Существует также патч (`http://bugzilla.mozilla.org/show_bug.cgi?id=245745`) для хранения журнала посещений в локальной базе данных (SQLite), его также можно модифицировать, чтобы он использовал модуль sql.

Далее можно добавлять в sql поддержку других СУБД, драйверов ODBC/JDBC. Можно также разрабатывать приложения для работы с SQLite и обычными, удалёнными базами данных. Можно разработать пользовательский интерфейс для работы с базами данных по примеру OpenOffice.org.

Ссылки на использованный материал:

- *N. McFarlane* Rapid Application Development With Mozilla;
- материалы W3C по стандарту RDF, `http://www.w3c.org/RDF/`;
- материалы по XUL, `http://www.mozilla.org/projects/xul/`;
- материалы по XPCOM, `http://www.mozilla.org/projects/xpcom/`.

12:50–13:15

Олег Паращенко

Санкт-Петербург, xmlhack.ru

Проект: TeXML

TEXML: XML-нотация для TEX

Аннотация:

В работе описывается один из способов преобразования XML-документов в формат LATEX. Для этого предлагается использовать TEX ML, промежуточное представление нотации TEX в виде XML.

Зачем нужен TEXML

TEXML[1] — это промежуточный формат для представления нотации TEX в виде XML. Он упрощает создание печатных версий документов, когда выполнены следующие условия:

- исходные документы представлены в виде XML,
- для создания печатной версии используется LATEX,
- XSLT — лучший способ преобразования XML-данных.

Известны несколько проектов, в которых XML преобразуется в LATEX с помощью XSLT. Все они сталкивались с тем, что XSLT неэффективен, если результат трансформации — не XML.

Эту проблему можно обойти, используя промежуточный этап. Вначале XSLT эффективно преобразует XML в TEX ML, а затем полученный TEX ML сериализуется в TEX.

Краткое описание TEXML

Основные конструкции языка TEXML видны на примере:

```
<TeXML>
 <cmd name="documentclass">
  <opt>a4paper</opt>
  <parm>minimal</parm>
 </cmd>
 <env name="document">
  Это &mdash; пример.
 </env>
</TeXML>
```

Результат сериализации:

```
\documentclass[a4paper]{minimal}
\begin{document}
Это~\textemdash{} пример.
\end{document}
```

Главные задачи сериализатора:

- замена специальных символов (например, «<» на «\textless»),

- обработка unicode (например, вывод «д» или «\cyrchar\cyrd{}»),
- разбиение лигатур (например, вместо «--- » — «-{}-{}-»).

Помимо этих существенных преобразований, TEX ML предоставляет также расширенные возможности.

- В язык встроена поддержка структурных конструкций LATEX: окружений, групп, команд с параметрами.
- Благодаря автоматическому форматированию создаваемого кода, итоговые документы хорошо выглядят.

Преимущества TEXML

Простой код на TEXML

```
<group><cmd name="it" gr="0"/>\пример</group>
```

соответствует такому TEX-фрагменту:

```
{\it \textbackslash{}пример}
```

Если изучить историю проектов tbook[2], xsltml[3], dblatex[4], db2latex[5] и других, создающих TEX-код с помощью XSLT, то выяснится, что во всех них были подобные проблемы:

- не всегда экранируются специальные символы,
- иногда забывается пробел после команды и получается «\it*текст*» вместо «\it *текст*»,
- может теряться открывающая или закрывающая фигурная скобка,
- из-за объективных сложностей, системы ориентированы на западноевропейскую кодировку.

Эти и другие проблемы не возникают при использовании TEXML.

Другие версии TEXML

При поиске решения для публикации XML через LATEX выяснилось, что Douglas Lovell предложил подход TEX ML ещё в 1999-м году[6], и была даже реализация на языке Java. Этот проект, TEX MLatté[7], был заброшен несколько лет назад, но от него осталась описание, которое послужило основой для новой версии TEX ML.

Существует также TEX MLapis[8], обработчик подмножества TEX ML на языке Perl, но его возможности ограничены.

Список литературы

[1] TeXML: an XML vocabulary for TeX `http://getfo.org/texml/`

[2] The tbook system for XML Authoring
`http://tbookdtd.sourceforge.net/`

[3] XSLT MathML Library `http://xsltml.sourceforge.net/`

[4] DocBook to LaTeX/ConTeXt Publishing
`http://dblatex.sourceforge.net/`

[5] DB2LaTeX XSL Stylesheets `http://db2latex.sourceforge.net/`

[6] Douglas Lovell, TeXML: Typesetting XML with TeX
`http://www.tug.org/TUG99-web/pdf/lovell.pdf`

[7] TEXMLatté `http://www.alphaworks.ibm.com/tech/texml`

[8] TEXMLapis
`http://www.bluweb.com/us/chouser/proj/texmlapis/`

13:15–13:30

Виталий Останин Санкт-Петербург, ЗАО «Взлёт»

Проект: ALT Linux Documentation Project

Средства для разработки и представления документации с использованием единого источника

Аннотация:

В докладе описывается подход к созданию документации с использованием единого источника. Описывается XML как базовый формат разметки, и DTD DocBook/XML как структура документов. Рассмотрены основные принципы модульной разработки документов, инструменты для создания и преобразования документов в другие форматы.

Семантическая разметка — первичность информации и вторичность внешнего вида

При создании документации основная задача — это запись смысла, информации о предмете документирования, и лишь потом визуальное представление этого смысла. Отделение смысла от оформления называется семантической разметкой. При таком подходе в документах размечается только информация, а внешний вид формируется с помощью дополнительных преобразований.

В проекте ALT Linux Documentation[1] (ALT Docs), посвящённом созданию документации, в качестве базового формата документации был выбран язык разметки XML[2]. Основными причинами для его выбора стали:

- общедоступная спецификация;
- простота (это текстовый формат, похожий на HTML);
- расширяемость (возможность добавлять собственные теги для разметки метаинформации);
- абстрагирование от кодировки документов (возможность создавать документы или фрагменты одного документа в произвольной кодировке);

- набор полезных спецификаций для включения документов (XInclude[3] и XPath[4]);
- независимость от платформы и инструментария.

В качестве структуры документов был выбран тип документов DTD DocBook/XML[6], разработанный для создания документации. В данной схеме семантическая разметка создаётся в DocBook/XML, а преобразование информации в различные форматы выполняется с помощью XSL-стилей[5]. Собственно, XSL-стили и определяют внешний вид информации.

Для DTD DocBook/XML существуют XSL-стили, которые позволяют получать из одного и того же исходного документа выходные форматы для печатных публикаций (PDF и PS), online-публикаций (XHTML/HTML одним файлом или разбитым на много файлов), справочных систем (man pages, Eclipse Help, Java Help, HTML Help).

Модульная разработка документов

При написании документации часто встречается ситуация, когда разные фрагменты документа пишут разные авторы. При создании документации из модульных фрагментов нужно иметь возможность объединять документы и их сопутствующие файлы (например, файлы с изображениями). При этом каждый фрагмент автор пишет как самостоятельный документ в произвольной кодировке.

Для объединения документов можно использовать спецификацию XInclude[3] (которая пока находится в стадии рекомендации, однако уже поддерживается многими программными средствами. При этом в результирующем документе нужно обеспечить уникальность идентификаторов тегов (id) и уникальность имён файлов с изображениями (images) из разных фрагментов. Такой уникальности можно достичь либо специальным преобразованием id/images, либо соблюдением общего соглашения об именовании id/images.

Ещё один момент, который нужно учитывать при модульной разработке документации — это сбор имён авторов из фрагментов для вывода в общем списке авторов, например, на титульной странице.

Для всех этих задач в ALT Linux Documentation Project разработан набор стилей, доступный по адресу `http://docs.altlinux.ru/releases/xsl/current/common/`

Средства для создания и публикации документов

Для создания документов в формате DocBook/XML можно использовать как обычный текстовый редактор, так и специализированные редакторы — emacs, vim, conglomerate[1], tkxmlive[2]. Документы можно создавать в любой кодировке (которую поддерживает XML-парсер), правильно указанной в объявлении XML.

Для обработки и преобразования документов в проекте ALT Docs используются библиотеки и утилиты libxml2/libxslt[7]. Несмотря на название — парсер и инструментарий для Gnome — это независимый от Gnome[8] кроссплатформенный набор инструментов, написанный на C, компактный, быстрый и не требующий дополнительной установки Java.

Парсер libxml2 поддерживает большинство спецификаций семейства XML, в том числе XML Inclusions 1.0[3], XPath 1.0[4], а также многие расширения EXSLT[9], что позволяет максимально использовать возможности стилей DocBook XSL.

Для выполнения всех этапов сборки можно использовать Gnu Make `http://www.gnu.org/software/make/`.

Список литературы

[1] ALT Linux Documentation Project `http://docs.altlinux.ru`

[2] Extensible Markup Language (XML) 1.0
`http://www.w3.org/TR/REC-xml/`

[3] XML Inclusions (XInclude) Version 1.0
`http://www.w3.org/TR/xinclude/`

[4] XML Path Language (XPath) `http://www.w3.org/TR/xpath`

[5] The Extensible Stylesheet Language Family (XSL)
`http://www.w3.org/Style/XSL/`

[6] DocBook DTD `http://docbook.org`

[7] The XML C parser and toolkit of Gnome `http://xmlsoft.org/`

[8] GNOME: The Free Software Desktop Project
`http://www.gnome.org/`

[1] `http://www.conglomerate.org`
[2] `http://tkxmlive.sourceforge.net`

[9] EXSLT is a community initiative to provide extensions to XSLT
http://www.exslt.org/

[10] FOP (Formatting Objects Processor)
http://xml.apache.org/fop/index.html

13:30–13:55

Алексей Крюков — Москва, МГУ им. М.В. Ломоносова, Исторический факультет

Проект: СОЛУНЬ http://www.thessalonica.org.ru

Разработка расширений для OOo Writer на примере проекта СОЛУНЬ

Аннотация:
В докладе рассматривается процесс переноса отработанных решений с платформы MS Word на OpenOffice.org, а также сравнительные достоинства используемых для этой цели средств разработки (Basic, Python, Java). Основное внимание уделяется программированию поиска/замены и ввода различных национальных символов в текстовых документах. Попутно предполагается обратить внимание на основные подводные камни, ждущие разработчика на этом пути.

Проект СОЛУНЬ (в латинском написании — Thessalonica), которому посвящено настоящее сообщение, сам по себе имеет узкоспециальное назначение, будучи предназначен для облегчения жизни филологов-классиков и историков-эллинистов. Тем не менее, представляется, что знакомство с его историей (достаточно сказать, что версия этой программы для OpenOffice.org прошла в своём развитии три этапа, будучи последовательно переписана на Basic, Python и Java) способно предохранить от большого количества синяков и шишек многих разработчиков расширений для OpenOffice.org.

СОЛУНЬ начала своё существование в виде макропакета для Microsoft Word 97 и выше, предназначенного для решения двух ключевых проблем, с которыми ежедневно сталкивается любой специалист, чья работа связана с классическим греческим языком. Во-первых, отсутствие

общепринятой 8-битной кодовой страницы, которая покрывала бы акцентированные греческие символы, породило множество используемых для этой цели несовместимых друг с другом кодировок. Соответственно, становится актуальной задача преобразования текста из этих кодировок в Юникод и обратно. Во-вторых, необходимо обеспечить возможность доступа ко всему множеству акцентированных символов с клавиатуры.

Познакомившись с OpenOffice.org, я должен был прежде всего ответить для себя на вопрос, предоставляет ли он необходимые возможности для реализации аналогичных функций. Поначалу мне казалось, что реализация конвертера во всяком случае не должна вызывать затруднений: ведь для этого требуется лишь возможность многократного поиска и замены фрагментов текста с нужным форматированием. В то же время я не видел возможности обеспечить ввод акцентированных символов внутренними средствами OpenOffice.org, так как настраиваемость клавиатуры в нем ограничена и заведомо не идёт ни в какое сравнение с уникально богатыми в этом отношении возможностями MS Word. Практическая работа над проектом показала ошибочность обоих этих предположений.

Точно так же не оправдалась надежда малой кровью перевести готовые макросы с VBA на StarBasic. Главная проблема заключалась даже не в различии объектной модели, а прежде всего в том, что разработку проекта пришлось начать с реализации ряда чисто технических функций. Например, пытаясь обеспечить сохранение настроек, я столкнулся с тем фактом, что API OpenOffice.org предоставляет лишь низкоуровневый интерфейс для доступа к реестру приложения. Соответственно, потребовалось затратить определённые усилия на создание модуля, который предоставлял бы некий набор функций, сходных с привычными по WordBasic getPrivateProfileString/setPrivateProfileString. Аналогичным образом дело обстояло с доступом к элементам диалоговых окон. Совсем уж нетривиальный код потребовался для того, чтобы создать списки доступных приложению шрифтов и языков, необходимые для формирования пользовательского диалога, в котором можно было бы выбрать нужные параметры форматирования. Надо сказать, что усилия, затраченные на этом этапе, не пропали впустую: отлаженные модули Basic впоследствии оказалось не так сложно портировать на другие языки.

Лишь завершив этот предварительный этап работы, я смог перейти к к программированию алгоритма поиска и замены символов, и тут меня постигло жестокое разочарование: оказалось, что поиск форматированного текста в OpenOffice.org работает некорректно, и полагаться

на него нельзя[1]. Наиболее естественным обходным путём был поиск фрагментов текста без указания форматирования с последующей его проверкой, но и этот вариант оказался чреват непредсказуемыми последствиями. Лишь после выхода стабильной версии OpenOffice.org 1.1 удалось найти приемлемое решение, состоявшее в отказе от множества последовательных операций поиска: теперь поиск выполняется единственный раз, с длинным регулярным выражением в качестве аргумента. Зато оказалось, что API OpenOffice.org предоставляет возможность перехватывать клавиатурные нажатия, адресованные активному окну, в результате чего удалось реализовать менеджер раскладок клавиатуры, обладающий даже бóльшими возможностями, нежели соответствующая часть макропакета для MS Word.

По мере разрастания проекта становилось всё очевиднее, что большинство его модулей выглядело бы намного изящнее, будучи реализовано в качестве объектов. Увы, такая возможность недоступна в StarBasic (в отличие от MS VBA!). Собственно, в этом заключалась главная причина, заставившая меня обратиться к Python, а затем и к Java. Моё первоначальное предпочтение Python было обусловлено следующими двумя обстоятельствами. Во-первых, Python сходен с Basic в том отношении, что оба эти языка позволяют напрямую обращаться к свойствам и методам каждого объекта UNO, в то время как в Java такой объект должен быть вначале представлен в качестве реализации того или иного интерфейса (это делается с помощью вызовов UnoRuntime.queryInterface()). Вторая причина выглядит несколько неожиданной, но, в сущности, создаёт едва ли не больше затруднений: дело в том, что Makefile'ы для представленных в OpenOffice.org SDK примерных проектов на Java отнюдь не отличаются прозрачностью и, самое главное, слишком тесно привязаны к структуре каталогов самого SDK, ввиду чего извлечь оттуда полюбившийся образец оказывается не так-то просто.

Реализация компонентов OpenOffice.org на Python, безусловно, технически проще, однако она заставляет разработчика учитывать ряд ограничений. Самое главное из них заключается в том, что на данный момент практически не существует удобного способа инсталляции модулей Python, которые не представляли бы собой в то же время реализацию компонентов UNO. Именно поэтому приходится жёстко выдерживать схему один файл—один компонент, и, соответственно, регистрировать в качестве сервисов UNO даже те объекты, которые предназначены только для использования внутри данной программы. Поскольку

[1]Пользуясь случаем, хочу лишний раз привлечь внимание аудитории к заполненному мною по этому поводу issue 10569.

же сервис UNO непременно должен быть реализацией какого-либо интерфейса, приходится дополнять пакет описаниями новых интерфейсов, что создаёт дополнительные проблемы. Наконец, сама реализация шлюза Python—UNO, несмотря на все усилия её разработчика, в настоящее время содержит немало ошибок, которые могли бы служить темой отдельного сообщения. Так, лишь в OpenOffice.org 1.1.2 была исправлена проблема с компилированными библиотеками *.pyd под win32, из-за которой, естественно, едва ли не большую часть модулей из стандартной поставки Python нельзя было использовать на этой платформе. Такого рода ошибки в конечном счёте и сделали необходимым перенос кода СОЛУНИ на Java.

Таким образом, история проекта СОЛУНЬ служит лишним подтверждением того факта, что Java по-прежнему остаётся предпочтительным средством разработки расширений для OpenOffice.org. В то же время единство объектной модели UNO позволяет осуществлять перенос уже готовых проектов на другой язык без особых потерь, делая необходимые для этого затраты усилий не столь критичными, как это могло бы быть в иной ситуации.

13:00–13:45

Пётр Савельев Санкт-Петербург, JSC Eltel / DrWeb

Практика использования формата документов OpenOffice.org для веб-публикации

Аннотация:
Практика использования формата документов OpenOffice.org для веб-публикации

Введение

Проект посвящён использованию возможностей преобразования и публикации документов OpenOffice.org. Внутреннее представление документов OpenOffice.org определяет сравнительную лёгкость написания для них XSLT-преобразований, а также использования уже имеющегося XML для организации хранения данных в СУБД. Основной

целью проекта была минимизация усилий пользователя при подготовке документа к веб-публикации. Для реализации были выбраны, помимо OpenOffice.org, Zope и PostgreSQL.

OpenOffice.org

Офисный пакет, позволяющий производить оформление документа исключительно стилями, что заметно облегчает как непосредственно работу с документом, так и его преобразование. Ещё одной ключевой особенностью стало использование WEBDAV как протокола доступа к документам в сети.

Zope

Система публикации документов. Благодаря архитектуре (написана на Python), позволяет легко создавать собственные модули, дополняющие или перегружающие уже существующие возможности.

PostgreSQL

СУРБД, одной из ключевых особенностей которой является возможность создания хранимых процедур и функций как на SQL-подобном диалекте, так и на Python, Tcl или Perl. Возможно также подключать бинарные модули (.so), написанные на C. Модуль xml (автор John Gray) был использован в качестве базы для добавления функциональности XSLT-преобразований.

Организация и развитие проекта

В качестве результата работы надо проектом видится система, позволяющая сохранять документы прямо из OpenOffice.org (через WEBDAV), производящая автоматический импорт при сохранении, и автоматический экспорт в различные форматы при запросе определённых URL.

Уровень базы данных

В PostgreSQL хранится XML, импортированный из документов OpenOffice.org. Предполагается импорт документа в виде одной XML-записи, на данный момент различные части документа (styles.xml, content.xml, meta.xml etc.) хранятся в различных таблицах.

Уровень приложений

Обработка XML с помощью XSLT производится также в PostgreSQL с помощью модуля xml, с добавленным функциями XSLT-преобразования и несколько изменённой функцией выборки по XPath. На момент написания заметки, практически вся добавленная автором функциональность в том или ином виде уже присутствовала в модуле xml2 того же John Gray.

Использование Zope в качестве сервера приложений на данный момент было отложено из-за большего, чем в PostgreSQL, количества кода для обеспечения устойчивой работы приложений. В частности, Zope не лимитирует использование приложениями как памяти, так и процессорных ресурсов.

Уровень представления

Обработку запрошенных URL производит продукт для Zope, перегружающий возможности классов ObjectManager и SimpleItem. На данный момент, один из классов продукта, DBFS, производит отображение набора хранящихся в БД документов в виде набора URL, соответствующих ID записей в БД. Однако, в работе находится версия продукта, производящего отображение из БД от одного до k произвольных записей в одном документе. Это даст возможность комбинировать хранящиеся документы и более гибко организовывать URL доступа к ним в Zope.

Заключение

На данный момент проект находится в начальной стадии развития. В результате работы должна получиться система управления документами, в качестве формата хранения использующая XML OpenOffice.org. Система должна предоставлять возможность простой публикации документов (Save as... в OpenOffice.org), возможность экспорта с помощью XSLT-преобразований (на данный момент написаны преобразования OpenOffice.org-[X]HTML), и возможность контроля версий хранящихся документов и online-поиска. Предполагается возможность экспорта, комбинирующего содержимое нескольких документов. Это также даст возможность оформления документа уже хранящимися в БД наборами стилей.

14:15–15:00: Обеденный перерыв

Вечернее заседание
15:00–17:00

Секция 1
(Председатель — Алексей Новодворский)

15:00–15:30
Николай Гарбуз, Аскар Рахимбердиев Москва, Инфра-Ресурс

Проект: OpenOffice.ru

Eclipse SDK

Аннотация:
Согласно результатам анонимного опроса посетителей сетевого проекта JUGA.RU, почётное третье место по популярности среди разработчиков в России занимает интегральная среда разработки Eclipse SDK. Однако, если внимательно рассмотреть результаты голосования, то Eclipse SDK используют только 12 разработчиков из 100. Первое и второе место по популярности, а также основную долю на рынке делят две другие замечательные среды разработки: Borland Jbuilder (28Авторы ставят перед собой задачу разобраться, почему разработчики выбирают средства, которые дороже Eclipse SDK более чем в 100 (!!!) раз. Настоящая работа посвящена исследованию выявленного парадокса через изучение истории, философии, архитектуры платформы Eclipse SDK, а также сравнительного анализа особенностей и отличий на примерах реальных проектов.

Не умаляя достоинств лидеров рынка (Borland Jbuilder и IntelliJ IDEA) с первого взгляда бросается главное отличие между популярными средами разработки на Java. Eclipse SDK распространяется под свободной лицензией, а точнее под лицензией EPL (Eclipse Public License), что на практике означает «бери и пользуйся бесплатно», в то время как конкуренты и лидеры опроса распространяются под патентными (Proprietary) лицензиями, что на практике выглядит так:

```
IntelliJ IDEA           $499.5
Borland Jbuilder X DE   $1000.
```

Как известно, на свете ничего не бывает бесплатным, в т. ч. и Eclipse SDK совсем бесплатно получить нельзя. Даже бесплатно загружая дистрибутив Eclipse SDK из сети Интернет, все равно придётся заплатить за трафик. Зная размер дистрибутива Eclipse SDK (85MB) и стоимость трафика ($50GB), можно легко рассчитать его стоимость ().

Зная фактическую стоимость сред разработки Java-программ, их можно сравнить в универсальном денежном измерителе.

```
IntelliJ IDEA           $499.5
Borland Jbuilder X DE   $1000.
Eclipse SDK             $
```

Как видим, и в денежном измерителе продукция компании Borland Jbuilder X DE оказалась лидером. Хочется верить, что лидерство продуктов с патентными лицензиями в России не связано с нарушением авторских прав их производителей. Но с другой стороны, хочется понять: насколько и почему Eclipse SDK хуже своих ближайших конкурентов, если разработчики выбирают средства, которые дороже более чем в 100 (!!!) раз.

Настоящая работа посвящена исследованию выявленного парадокса через изучение истории, философии, архитектуры платформы Eclipse SDK, а также сравнительного анализа особенностей и отличий на примерах реальных проектов.

Тезисы:

1. Eclipse — это не IDE, а платформа разработки интегрированных приложений.

2. Работа в Eclipse не ограничивается разработкой приложений Java.

3. Открытость платформы Eclipse определяется не только лицензией, но и возможностью расширения её свойств.

4. Eclipse — не монолитное приложение, а набор расширяемых подсистем: Platform, JTD и PDE.

5. Практически любая подсистема Eclipse может быть расширена сторонними разработчиками для решения разнообразных задач.

6. SWT — средство создания графического интерфейса, объединяющее эффективность сервисов операционной системы с переносимостью Java.

7. JTD осуществляет поддержку рефакторинга, одного из главных инструментов разработки ПО, в том числе в экстремальном программировании.

15:30–15:55

Георгий Курячий Москва, ALT Linux

Организация человеко-машинного взаимодействия в ОС

Аннотация:
В докладе обсуждаются два существующих способа организации человеко-машинного взаимодействия: процедурные и проективные системы. Рассмотрены принципы построения этих систем и показаны некоторые следствия для пользователя, работающего в системе одного или другого типа, в частности, кривая научения, круг задач, которые можно решать при помощи системы, эффективность работы пользователя, отношение к работе с системой.

Человек боится того, что не в состоянии понять. Сложные машины и в особенности компьютеры вызывают у современного человека страх едва ли меньший, чем в давние времена духи и черти (см. "R.U.R."или "Терминатор 3"). Когда поведение сложной машины непредсказуемо, человек склонен приписывать ей душу, волю или, по крайней мере, интеллект — черты исключительно человеческие. В человеко-машинных *системах*, рассчитанных на совместную работу и человека и машины, этот страх чрезвычайно вреден.

Чтобы избавить пользователя от страха перед машиной, нужно сделать её понятной. К тому ведёт два пути. Первый: максимально упрощая и скрывая «ненужные» подробности устройства машины, рассказать на понятном языке, как можно *воспользоваться* машиной для решения наиболее распространённых задач. Второй: *обучить* человека устройству машины и дать ему инструмент, пользуясь которым и зная, как

«всё работает», он смог бы самостоятельно ставить и решать любые задачи, на которые ориентирована машина.

Первый способ годится для неподготовленного пользователя (однако кривая научения должна очень скоро становиться пологой). От разработчиков системы он требует составления «легенды» — упрощённой (вообще говоря — не соответствующей действительности) схемы устройства машины, пользуясь которой человек мог бы всё же принимать решения. Для работы человека в системе составляются *инструкции*, описывающие типовые ситуации и способ использования машины в них, формируя *навык* работы. Основной работой человека при этом становится *выполнение* этих инструкций (точнее, предписанных ими процедур), поэтому системы, устроенные таким способом, мы будем называть *процедурными*. Основная точка принятия решений в таких системах — *выбор* одного из предложенных вариантов использования.

Второй способ даёт *поначалу* довольно пологую кривую научения (нужно время на освоение архитектуры системы, средств управления ею, практику в постановке и решении задач с её помощью). С течением времени, однако, количество и сложность задач, решаемых с помощью системы, устроенной сообразно этому способу, начинает ощутимо расти. Такой способ требует постоянного и толкового *объяснения* процессов, происходящих в машине, без чего её освоение делается почти невозможным. Взаимодействие человека и машины в этом случае возможно только на основе *знаний* о работе системы. Основная точка принятия решений в таких системах — правильная постановка задачи (составление *проекта* будущей системы) и грамотная её реализация. Такие системы мы станем называть *проективными*.

Итак, сформулируем четыре принципа организации *процедурных систем*.

Принцип ограниченной осведомлённости

Знание многих легенд может лишь сбить с толку, а инструкции всегда есть в руководствах, знание же внутренностей системы не помогает при решении задач.

Принцип перекрытия процедур

Любая задача должна быть решена посредством выбора одной или нескольких из предлагаемых процедур — иначе пришлось бы «лезть» внутрь системы.

Принцип гарантированных навыков

Пользоваться машиной может кто угодно, ему не надо для этого ничего учить; основной упор делается на простейшие операции выбора и активации.

Принцип делегирования ответственности

Разработчик системы лучше разбирается в том, как именно решать задачу пользователя, чем сам пользователь; в отсутствие разработчика его заменяет машина.

Принципы организации *проективных* систем.

Принцип информационной открытости

Прежде чем работать, изучи предмет; всё, к чему нет документации, можно считать неработающим.

Принцип умопостижимости контекста

Машина должна быть устроена так, чтобы думающий человек (не гений) смог её понять; при решении конкретной задачи количество средств её решения до́лжно сводить к небольшому числу.

Принцип распределения усилий

Человеку свойственно ошибаться, принимать решения и мыслить творчески, поэтому работать должна машина, а человек — думать; если некоторая «тупая» работа ещё не автоматизирована, должны быть лёгкие средства это сделать.

Принцип персональной ответственности

Всё, что натворил пользователь проективной системы, есть результат именно его усилий; не стоит искать «ошибки в компиляторе».

Четыре (два и два) важных следствия этих принципов.

Работа в процедурной системе подавляет инициативу (стремление *не* следовать инструкциям) — привет «скрепке»! Процедурная система — это набор готовых решений «на все случаи жизни». Если решения всё-таки нет (трудно всё предусмотреть), проще убедить пользователя, что ему подходит другое — или научить избегать подобных случаев.

Работа в проективной системе поначалу похожа на изобретение велосипедов в тренировочных целях: некоторые простые задачи приходится — и нужно! — решать самостоятельно, несмотря на то, что решения для них есть и проверены временем. Проективная система — это набор *инструментов* постановки и решения задач, поэтому соблазн всё всегда решать самому с нуля очень велик. Однако во многих случаях всё-таки имеет смысл не изобретать велосипедов, а воспользоваться имеющимся и работающим мотороллером!

Напоследок — вопрос: какие из имеющихся ОС для современных компьютеров можно отнести к проективным системам, а какие — к процедурным?

15:55–16:10

Олег Филон Гомель, ОДО Рива

Проект: Гомелькая группа пользователей Линукс
http://gomelug.agava.ru

Этапы легализации софта

Аннотация:
Предлагается план поэтапного перехода от унаследованного проприетарного ПО к свободному для компьютерных сетей любого масштаба. Рассматриваются некоторые возникающие проблемы психологического и технического характера и способы их решения

Вопрос легальности используемого ПО для любой организации является чрезвычайно важным. Во-первых, цена этого вопроса очень высока. Как правило, стоимость установленного ПО сравнима со стоимостью всего парка компьютеров. А во-вторых, если компьютеры приобретались постепенно, многие годы, то узаконить ПО может понадобиться сразу, за короткое время. Не секрет, что вопросам лицензирования и законности программного обеспечения многие не уделяют должного внимания. В результате люди оказываются совершенно неподготовленными, когда подобные вопросы всё же возникают.

Покупка ПО у крупнейших американских монополий также не может считаться приемлемым решением проблемы. Рынок программ уже и так чрезвычайно монополизирован, любые деньги, отправленные за океан, только усугубят зависимость от проприетарного софта, лишат заработка отечественных программистов. Хуже того, многие толковые специалисты в конце концов также последуют за этими деньгами. Таким образом страдает не только экономика страны, но и её интелектуальный потенциал.

Другим решением проблемы может стать продуманная подготовка и переход на использование свободного ПО. Здесь и далее под свободным ПО имеются в виду программы, распространяемые под свободными лицензиямим . Первый этап легализации софта можно условно назвать подготовительным.

В своё время компания Citrix® разработала технологию и протокол «терминального сервера» для выполнения приложений на сервере. Клиентское ПО, существующее под многими платформами, позволяет использовать клиентский компьютер только для ввода информации

пользователем и вывода на экран. Затем эта технология была куплена Microsoft® и встроена в ОС Windows® 2000. Версия «Advanced Server» этого программного продукта имеет встроенный терминальный сервер с лицензией на 90 дней без дополнительной оплаты. На подготовительном этапе нужно добиться, чтобы пользователи научились работать со своими приложениями только на сервере.

Централизация ПО на сервере потребует более мощного сервера приложений, чем обычный файл-сервер. Как всегда, наиболее критичным ресурсом является оперативная память. Строя сервер приложений, лучше запасать по 60-100 MiB RAM в расчёте на одного активного пользователя. Другим важным моментом построения сети должно стать удаление с виндового сервера всех серверных функций, не имеющих отношения к пользовательским приложениям. Т. е. с самого начала нужно включить в сеть Unix-сервер и все сетевые сервисы устанавливать на нём. Практически всё серверное ПО существует под свободными лицензиями, так что вы можете использовать его совершенно законно. Последние версии Samba позволяют использовать эту программу в качестве контроллера домена, а сервер LDAP способен обслуживать единую базу данных пользователей как для Windows, так и для Unix.

Предложить рекомендации по переучиванию пользователей работе в терминальном режиме труднее. Важным моментом может стать утверждение правил работы в локальной сети. Например, в правилах фирмы «Рива», Гомель, есть пункт об ответственности пользователей за законность устанавливаемого ими на свои компьютеры ПО. Такой пункт создаёт прямую заинтересованость в удалении с рабочих станций всех сомнительных программ.

Только после того, как все критичные для работы вашей организации приложения перенесены на Unix-платформу либо собраны на одном или нескольких серверах приложений, можно приступать ко второму этапу. На этом этапе необходимо заменить операционную систему на рабочих местах на свободную и лицензионно чистую. Например, один из дистрибутивов Linux® или FreeBSD®. Для этих платформ существует клиент терминального сервера, программа rdesktop. В идеальном случае пользователь может не заметить, что у него другая операционная система. Также важно, что для свободных платформ существует и постоянно увеличивается набор пользовательских приложений.

Наконец, третий этап. Собственно, заменив операционную систему у клиентов и упрятав весь сомнительный софт на серверах приложений, проблема легализации софта практически решена. Вы точно знаете список используемых приложений, можете приобретать только

лицензии на необходимое количество одновременно работающих пользователей. Очень важно, что весь софт на рабочих местах свободно доступен, риск незаконной установки проприетарных программ на эти компьютеры минимален, вся сеть более защищена и контролируема.

Остаётся, однако, возможность дальнейшей и радикальной модернизации вашей сети. Обе упомянутые платформы, и Linux, и FreeBSD, позволяют сделать клиентские станции бездисковыми компьютерами. Современное железо позволяет упаковать в компактный корпус довольно мощный компьютер с малым энергопотреблением. Отсутствие дисков и вентиляторов делает его бесшумным и более надёжным. Настроив Unix-сервер для обслуживания сетевой загрузки операционной системы и раздачи файловой системы по NFS'у, вы сможете решить массу административных задач. Все Unix-приложения теперь также будут храниться централизовано, как и настройки клиентских станций. Клиентский компьютер стал типовым, легко заменяемым.

Третий этап, развёртывание сети из бездисковых станций, заслуживает внимание также как одна из перспективных бизнес-моделей для свободного ПО. Как известно, обладая огромными технологическими преимуществами, открытый софт не предлагает лёгкого способа зарабатывания денег. Необходимость публикации собственных доработок программ, распространяемых по лицензии GPL, не позволяет надолго установить монопольно высокую цену на свои программные продукты. Бизнес на открытом софте больше благоприятствует сопровождению программ и серверов, т. н. «саппорту». В случае сети из бездисковых станций и двухсерверной двухплатформенной конфигурации с отдельным сервером приложений есть возможность предложить на рынок полностью готовое сетевое решение. В этом случае предлагается не только комплексное решение всех сетевых проблем, но также обеспечивается путь миграции от унаследованных проприетарных приложений к свободно-доступным открытым, гарантируется законность, надёжность и защита всего ПО и данных.

16:10–16:25

Александр Колотов

Киров, Организация: сеть магазинов «Каравай» и «Недельный запас»

Проект: lindocs

Offline документация. Прочь от маргинальности Linux

Аннотация:

За счёт чего новички становятся специалистами? Что их разделяет? Важную роль в этом вопросе играет документация. Часто перед пользователями встаёт проблема как настроить определённую программу или как решить определённую проблему. Поиск в Интернет порой очень утомляет. А как быть в том случае, когда для того чтобы выйти в Интернет нужно настроить модем, информация по настройке которого находится в Интернет? Этот доклад затрагивает тему методов накопления и обработки документации, проблемы, возникающие при этих процессах.

«Я ставлю Linux своим друзьям! Они в восторге от него!» или «Зачем популяризовать Linux? Путь его знают только гуру!» Сообщения подобные этим часто всплывают в компьютерных форумах, где затрагивается тема Linux. У каждого из нас есть своё мнение по этому вопросу и мы всегда готовы поделиться ним с оппонентами, кто по-философски объективно, кто же хулигански-агрессивно. Не будем затрагивать эту больную тему. Постараемся рассмотреть её с другой стороны: с помощью чего новичок может стать гуру?

Безусловно, немаловажную роль здесь играет документация. Документация в формате MAN и INFO имеет несколько проблем:

Неочевидность Новичку сложно догадаться о существовании такого вида документации.

Нерегулярность Не все программные продукты документированы таким образом.

Неактуальность Информация часто оказывается устаревшей.

Сухость изложения Налицо проблема «разработчиками для разработчиков».

Локализация Отсутствие документации на родном для пользователя языке.

Данные проблемы в дальнейшем будем использовать для оценки других видов документации. Документация, сопровождающая дистрибутив операционной системы, несомненно, полезна — нет проблем с локализацией и актуальностью, но обычно можно говорить о лаконичности: затрагивается установка, настройка и использование лишь небольшого количества программ. В непериодических бумажных изданиях — книгах — отсутствуют проблемы с локализацией и лаконичностью, присутствует высокая наглядность. Но существует проблема «старения» книги — информация в ней со временем теряет актуальность. Периодические же издания (газеты и журналы) практически лишены всех недостатков.

Отдельно нужно рассматривать Интернет. Объем документации в нем даже избыточный. Здесь к услугам пользователя как домашние страницы проектов, так и неофициальные сайты поддержки, форумы, чаты, конференции, дискуссионные листы, рассылки. На первом месте стоит умение пользоваться услугами мощных поисковых систем. Для комфорта пользователей создаются всевозможные энциклопедии — накопительные и ссылочные.

Следует отметить сложность накопления данных, предоставленных во всевозможных электронных конференциях и форумах. Их модераторы или участники по возможности ведут отдельно список Ответов на Часто Задаваемые Вопросы (FAQ), но этого недостаточно — необходимо вести просто Список Ответов на Задаваемые Вопросы, то есть отслеживать и публиковать все проблемы со способами их решения, обсуждавшиеся в данном ресурсе. Причём материал не должен быть сформирован по принципу HOWTO (как сделать), где пользователь поставлен в положение обыкновенного автомата, набирающего команды методом cut-and-paste. Новичок должен понимать, что он делает. То есть в ходе описания неплохо было бы показать, что настройка не всегда может проходить гладко, что могут возникнуть некоторые проблемы, показать методы решения этих проблем.

Как минус, в Интернете, вполне логично, проявляются проблемы «всегда online» и лицензионной чистоты публикуемых материалов. Для решения проблемы «всегда online» и активной помощи пользователям предлагается организовать проект offline документации. В него включаются все online-материалы, доступные в сети и представляющие информационную ценность. Каким критериям должен отвечать подобный проект:

1. Понятный для пользователя язык. 2. Информативность. 3. Структурность построения и в идеале контекстный поиск. 4. Кроссплатформенность. Документация без проблем должна быть доступна для просмотра на любой платформе.

В качестве примера можно рассмотреть сборник offline документации lindocs[1]. Как возникла идея создания сборника? После огромного скопления различных информационных материалов, касающихся Linux и unix-подобных систем, возникла идея их каталогизировать. В качестве форматов хранения информации были выбраны HTML и plain text, а также PDF, так как просмотр информации в таком случае не представлял особого труда как в Windows, так и в Linux. Изначально, была выбрана древовидная структура каталога, которая, как в дальнейшем оказалось, не подходит для структурирования: один и тот же материал мог относиться к разным веткам древовидной структуры. Отсюда был сделан вывод, что для оглавления больше подходила бы сетевая структура. Но объем наработанной информации был к тому времени уже велик, поэтому задумка о сетевой структуре была отложена до момента увеличения количества человек, сопровождающих данный сборник.

Как происходит в текущий момент работа по накоплению в нем материалов? В Рунете существует несколько десятков сайтов, которые время от времени публикуют либо новые статьи, либо ссылки на них. С помощью wget в случае многостраничного документа, или функции «Save» браузера в случае отдельной страницы происходит скачивание материала. При сохранении браузером автоматически скачиваются каскадные таблицы стилей и невообразимое количество графической информации, которая не нужна для оффлайнового просмотра. Да и теряется текстовая информация в оформлении и изысках дизайна. Поэтому, вручную или с помощью несложных скриптов, со страниц убирается ненужное форматирование и удаляются ненужные файлы, многостраничные документы остаются нетронутыми. Затем, путём анализа заголовка или беглого просмотра содержимого, оценивается, в какой раздел сборника будет помещён материал. После многочисленных добавлений информации или коренных изменений структуры объявляется о выходе новой версии, версии нумеруются по принципу ГОД-МЕСЯЦ-ДЕНЬ. Сборник распространяется без извлечения коммерческой прибыли — заинтересованные люди либо сами приходят и переписывают документацию, либо скачивают архив сборника из Интернета. Публикация в Интернете сборника в неархивированном состоянии не производится, так как все материалы и без того доступны в сети, хотя и разбросаны.

[1] `http://www.kirov.ru/~mypost/lindocs/`

Судя по откликам в письмах и спросу на сборник, данный проект даже в таком виде нужен сообществу.

Секция 2

(Председатель — Алексей Смирнов)

15:00–15:25

Вадим Житников Москва, ООО «Компания Скид»

Проект: Maxima `http://maxima.sourceforge.net`

Свободные программы символьной математики Maxima и Axiom

Аннотация:
Программы символьной математики ориентированы на точное, аналитическое решение математических задач в максимально удобной для человека форме и находят широкое применение в научных, инженерных расчётах и в образовании. Рассматривается история, состояние, проблемы и перспективы двух свободных систем символьных вычислений Maxima (GPL) и Axiom (Modified BSD license).

Системы символьных (аналитических) вычислений (иначе ещё называемые системами компьютерной алгебры) предназначены для автоматизации самого широкого спектра математических расчётов. Причём вычисления производятся точно, аналитически — в форме, близкой к обычной математической нотации. Они прекрасно справляются с такими рутинными задачами, как раскрытие скобок, приведение подобных, дифференцирование. Более того, используя различные эвристические или специализированные алгоритмы, системы аналитических вычислений могут решать и более сложные задачи, такие как интегрирование, решение символьных и дифференциальных уравнений. Разумеется, не всякая задача имеет точное решение, и поэтому системы компьютерной алгебры могут решать задачи численно, строить графики.

Системы компьютерной алгебры имеют долгую историю, основы их развития заложены в академической среде Северной Америки. Сегодня наряду с такими известными коммерческими системами как Maple

и Mathematica, существует две свободные программы: Maxima (GPL, http://maxima.sourceforge.net/) и Axiom (Modified BSD license, http://www.nongnu.org/axiom/index.html).

В 1968 году в Массачусетском Технологическом институте (MIT) в рамках создания искусственного интеллекта началась разработка системы аналитических вычислений Macsyma. В ранний период проект, требовавший чрезвычайно больших для того времени вычислительных ресурсов, базировался на единственном компьютере марки DEC PDP-6 (позднее DEC PDP-10). С подключением этой машины к сети ARPANET возросло число пользователей и программистов Macsyma. Появилось большое число портов Macsyma на другие вычислительные машины и операционные системы. Но со временем все эти проекты, включая оригинальный в MIT, прекратили своё существование. В 1982 году права на коммерческую разработку и использование Macsyma были переданы Symbolics Inc. В то же время Министерство Энергетики США сохранило за собой права на оригинальный код Macsyma. Основываясь на нем, проф. Вильям Шелтер (William Schelter) из Техасского университета в Августине развивал свою версию программы, переименованную в Maxima. В 1998 году он получил официальное разрешение от Министерства Энергетики США выпустить Maxima под лицензией GPL 2. Тем временем в 1999 году коммерческий проект Macsyma был полностью закрыт и программа перестала быть доступной. После внезапной смерти Вильяма Шелтера в 2001 году развитие Maxima продолжается в рамках проекта базирующегося на SourceForge. Лидер проекта Джеймс Амундсон (James Amundson). В феврале 2003 г. выпущен первый официальный релиз Maxima 5.9.0, в данный момент Maxima 5.9.1 находится на стадии бета-тестирования.

История развития второй системы аналитических вычислений во многом сходна. С 1971 до начала 90х годов прошлого века система Scratchpad развивалась как научно-исследовательский проект IBM. Была проделана огромная работа, достаточно сказать, что затраты труда на разработку Scratchpad оцениваются не менее, чем 300 человеко-лет. Затем система была переименована в Axiom и передана Numerical Algorithms Group (NAG) для коммерческого использования. В 2001 году NAG сочла дальнейшее развитие Axiom нецелесообразным и выпустила систему в свободное использование. Современный лидер проекта (http://savannah.nongnu.org/projects/axiom) Тим Дэйли (Tim Daly) является разработчиком Axiom ещё со времён IBM. Уникальным свойством Axiom является строгая типизация математических выражений, что делает систему особенно полезной для развития математиче-

ских алгоритмов. Программа хорошо документирована как на уровне исходного кода (literate programming), так и для конечного пользователя (имеется книга).

Остановимся на проблемах и задачах проекта Maxima. Исходный код, доставшийся проекту в наследство от Macsyma, был в довольно разрозненном состоянии — содержал куски неполного кода и рабочие файлы. С тех пор, сначала силами Вильяма Шелтера, а потом командой разработчиков Maxima была проделана большая работа. Полностью реорганизована структура исходного кода и система построения переведена на autotools. Улучшено построение графиков, численное вычисление специальных функций и исправлено много ошибок. Усовершенствовано взаимодействие с программами оболочками. На данный момент существует несколько оболочек к Maxima — несколько режимов для GNU/Emacs и графические оболочки xmaxima, TeXmacs, symaxx/2. К сожалению, ни один из этих интерфейсов не вполне удовлетворяет современным требованиям и разработка новой полноценной кроссплатформенной графической оболочки является важнейшей задачей. Работа над исправление внутренних ошибок Maxima идёт недостаточно быстрыми темпами, что связано со следующими трудностями. Довольно высок порог вхождения для понимания внутренней работы программы. Исходный код слабо комментирован и сложен по своей структуре. Работа по усовершенствованию ядра системы требует специфической квалификации — навыков программировании на Lisp вместе с хорошей математической подготовкой.

Maxima неплохо документирована: имеется полное справочное руководство в формате texinfo, на котором основана on-line справочная система. Однако нет книги — работа начата, но пока далека от завершения. Одна из актуальных задач — перевод документации на русский язык.

Сейчас Maxima работает на 6 различных реализациях Common Lisp (GCL, Clisp, CMUCL, SBCL, OpenMCL, Allegro Common Lisp), и перенос на любую другую реализацию Common Lisp не составит труда. Система обладает высокой переносимостью и работает практически на всех современных вариантах Linux (12 процессорных архитектур Debian/GNU Linux) и UNIX, Mac OSX, Windows 9x, 2000, XP, наладонных компьютерах.

15:25–15:50

Пётр Новодворский Москва, ALT Linux

Проект: dilingvo

Интеграция свободно доступных интернет-словарей в программные продукты

Аннотация:

Доклад посвящён использованию протокола dict для интеграции общедоступных интернет-словарей при разработке программных продуктов. Рассматривается создание специального прокси http↔dict, средства нахождения нужных словарей и их пополнения.

Наряду с коммерческими программами по доступу к словарям, такими как Lingua, Multitran в Интернет существуют средства доступа к этим словарям через web. Каждая программа, обеспечивающая подобный доступ, имеет собственный интерфейс, а для того, чтобы найти подобный словарь, приходится пользоваться традиционными поисковыми системами.

Понятно, что всё это усложняет интеграцию доступа к таким словарям в операционную систему, например, такую тесную, как в случае stardict, когда высвечивается перевод к слову, которое выделил пользователь. Этим, в частности, и объясняется то, почему авторы коммерческих словарей с такой лёгкостью делают к ним бесплатный доступ через Интернет. Они руководствуются тем, что пользователи не смогут активно использовать их словари через web и им придётся покупать локальную версию их программного продукта.

Создание прокси-серверов, которые могли бы предоставлять единый интерфейс к различным web-словарям могло бы решить эту проблему.

Однако, это не решит второй проблемы, выбора нужного словаря из доступных. Эта проблема существует уже сейчас: когда пользователь выбирает в качестве основного словарного сервера dict://mova.org/, ему приходится выбирать из 62 словарей без всякой классификации. Так называемые виртуальные словари, реализованные в новой версии dictd, могут частично решить эту проблему, однако они предопределяются администратором сервера и не всегда могут удовлетворять требованиям пользователя. Кроме того, поиск сервера, содержащего нужный словарь, остаётся сложным процессом.

Специализированный DICT-сервер dilingua призван решить две описанных проблемы. Dilingua представляет шлюз пользователям протокола DICT к услугам web-словарей. Он легко настраивается под любой конкретный словарный web-интерфейс.

Фактически, он представляет собой обычную службу DICT, которая общается по протоколу, описанному в документе RFC 2229[1], однако в отличие от стандартного сервера dictd для осуществления запроса он обращается не к локальному словарю, а к словарю, находящемуся на неком web-сервере.

Для того, чтобы настроить dilingua под конкретный web интерфейс, надо указать пять параметров. Первые два параметра — это регулярные выражения, одно из которых при подаче ему на вход результирующей страницы запроса может определить, нашёлся ли перевод, а второе этот перевод из страницы выделяет. Следующие два параметра решают аналогичные задачи, но для команды match, описанной в RFC 2229. Пятый параметр задаёт классификацию словаря, о которой будет сказано позже.

Остаётся вторая проблема: как найти словарь, который удовлетворяет требованиям пользователя. Для этого был заведён DLS (сервер классификации словарей, основан на LDAP[2]), на котором сервера dict могут регистрировать имеющиеся словари и описывать их свойства. Здесь и используется пятый параметр конфигурации dilingua, содержащий информацию о классификации словарей. При старте dilingua соединяется с DLS и регистрируются на нём, учитывая классификацию. Она может включать следующие пункты:

- С какого языка переводится;
- На какой язык переводится;
- Тематика словаря;
- Тип словаря.

Таким образом, переводчик может задать интересующие его параметры, например, «Перевод с русского на английский, нефтяной глоссарий» и получить имя сервера и имя интересующего его словаря.

Появляется возможность создавать виртуальные словари, которые группируются не на основе интересов администратора какого-то dict-сервера, а следуя параметрам запроса нахождения словарей. Пользователь может поискать слово во всех словарях, которые переводят с русского на английский.

Если описанная система станет популярна, пользователь получит классифицированный доступ к огромному количеству словарей, доступных в Интернет, а разработчики смогут включить возможности этой системы в свои программы.

Список литературы

[1] *R. Faith, B. Martin*, A Dictionary Server Protocol, `http://www.ietf.org/rfc/rfc2229.txt`.

[2] *R. Weltman, M. Smith, M. Wahl*, Lightweight Directory Access Protocol (LDAP) Authorization Identity Request and Response Controls, `ftp://ftp.rfc-editor.org/in-notes/rfc3829.txt`.

15:50–16:15

Станислав Иевлев

Москва, ALT Linux

Проект: Sisyphus
`http://www.altlinux.org/index.php?module=sisyphus`
Alternatives `http://alternatives.sourceforge.net/`
Osec `http://o-security.sourceforge.net/`
Colorifer `http://colorifer.sourceforge.net/`
NDK++, NCursesXX `http://ndk-xx.sourceforge.net/`

Почему мы переписываем программы: на примере alternatives, osec, csed, ncursesxx

Аннотация:
Почему ALT Linux Team пишет свои версии известных приложений и разрабатывает новые? Чем зачастую не устраивают существующие свободные приложения? На примере нескольких приложений из Сизифа, демонстрируется как решались те или иные проблемы, какие задачи ставились и что было достигнуто. Что ещё планируется переписать и почему?

Почему при таком обилии свободных программ их приходится периодически переписывать? Характерные недостатки ПО от FSF: избыточная функциональность, совместимость с огромным количеством платформ, консервативность upstream некоторых ключевых проектов, частая несогласованность между проектами, чрезмерная оптимизация. Всё это обусловлено первоначальными целями и особенностями движения: отсутствие собственной платформы, требование максимальной распространённости ПО.

Мы же хотим от приложений: высокой масштабируемости (адекватного соответствия функциональности и имеющихся ресурсов) и максимально возможного использования готовых решений из системных библиотек, хорошей поддерживаемости кода и лёгкой адаптируемости к новым условиям.

Как выявить точки перегруженности функциональности в приложении: через конфигурационные файлы, изучение формата командной строки, анализ подлежащих протоколов. Например: конфигурационные файлы с нелинейной структурой свидетельствуют о том, что возможно тут собрана конфигурация нескольких отдельных приложений; если аргументы командной строки имеют разный контекст в разных способах вызова, значит скорее всего в этом комбайне спрятано несколько отдельных утилит, если в протоколе заложена излишняя универсальность, то приложение, реализующее этот протокол, будет скорее всего комбайном.

Как писать хорошо масштабируемые приложения? Аккуратное разделение на слои и организация конвейеров, удачный выбор структур данных, протокола, конфигурационного файла. Unix предоставляет готовые примитивы для организации конвейеров, и если приложения будут ими пользоваться, то будут содержать меньше ненужного кода. Почему-то зачастую библиотеки пытаются собрать в себе всю возможную функциональность, когда как проще разделить одну библиотеку на несколько небольших. Очень хорошо, если библиотека верхнего уровня не будет использовать недокументированные функции библиотеки нижнего уровня, ибо тогда будет легко изменять/заменять её по частям.

Идеала не достигнуть из-за изначальной сложности некоторых используемых моделей, асинхронности и параллельности процессов в системе. Несмотря на то, что хочется получить совсем простой код, этого никогда не произойдёт. Хорошее приложение всегда будет содержать массу дополнительного кода, проверяющего коды возврата из функций, защиту от race-conditions, дополнительный код для снижения соб-

ственных полномочий до уровня минимально необходимых. Приведём несколько примеров библиотек и приложений.

ncursesxx/ndk++ vs. ncurses++

curses — отличный пример неудачно спроектированной библиотеки. C++ bindings к ncurses отражают всю громоздкость и неразбериху этой библиотеки, пытаясь охватить сразу всё, что только возможно. Мы предлагаем свой вариант. Это пример правильного разделения библиотеки на слои, ncurses++ предоставляет только базовую функциональность: работа с окнами (изменение размера, перемещение, рисование) и терминалами (ввод/вывод, изменение атрибутов). ndk++ — поддержка событий и сложных виджетов. Что ещё в самом ncurses планируется переделать в том же ключе: terminfo — пример смешения вывода и получение информации из базы, ncurses — смешение ввода и вывода данных в окна.

osec vs. AIDE, tripwire, mtree

Задача проекта состояла в однократном отчёте об изменениях в файловой системе с обращением особого внимания на security-sensitive файлы. Чем не устраивают уже существующие средства контроля целостности? Перегруженной функциональностью, а также использованием собственных версий функций для обхода файловой системы и создания/ведения базы данных. Более того, все эти приложения работают с избыточными полномочиями в системе, что нежелательно с точки зрения безопасности.

Новый проект osec — пример превращения в конвейер: сборка данных, генерация отчёта, рассылка отчёта. Для работы используются готовые системные функции: fts — для быстрого и безопасного обхода файловой системы, cdb — быстрая и удобная библиотека для создания read-only баз данных. Все составные части osec устроены относительно просто.

alternatives vs. update-alternatives

Изначально в Sisyphus использовалась система альтернатив, взятая от Debian, однако в ходе эксплуатации был выявлен ряд проблем и недостатков функциональности. Новый проект alternatives призван решить эти проблемы. Это пример «комбайн вместо набора утилит».

Новые alternatives также имеют более надёжный подход к организации базы данных. Простые shell-скрипты как результат удачного формата конфигурационных файлов. Кроме того, вместо «группового» подхода к альтернативам, применяется «покомпонентный». Благодаря этому снялись ограничения на отношения master/slave и значительно упростились алгоритмы обработки.

csed vs. color-gcc

Упрощение концепции, пример когда использование стандартного конвейера наделяет приложение новой функциональностью. Если color-gcc был пригоден только для gcc, то csed может уже применяться совместно любым приложением командной строки, например, ps, tail, cat, и т. д.

16:15–16:40

Алексей Гладков Москва, ALT Linux

Новые технологии проекта Sisyphus: жизненный цикл пакета

Аннотация:
В докладе рассказывается о текущей схеме приёма пакетов, о проблемах, с которыми сталкиваются люди, следящие за приходящими пакетами. Предлагаются возможные решения этих проблем на примере новой системы проверки входящих пакетов — проекте incominger.

Проверка пакетов, поступающих в репозиторий Sisyphus, является одной из важнейших задач. С момента создания проекта Sisyphus был предпринят ряд шагов по автоматизации проверки поступающих пакетов на соответствие общим требованиям, предъявляемым к пакетам Sisyphus. В частности, была создана программа sisyphus_check. Для гарантии пересобираемости пакета была создана система hasher, позволяющая осуществить пересборку пакета в «чистом» окружении, созданном из текущего состояния Sisyphus. Следующим шагом в автоматизации прохождения пакета в репозиторий стало создание системы автоматизированной проверки входящих пакетов — проекта incominger.

Целью этой системы является выполнение стандартных действий над пакетами для осуществления входного контроля. Incominger основывается на трёх других проектах: osec, hasher и sisyphus_check.

На вход этой системы подаются пакет(или пакеты). На выходе из неё эти пакеты сортируются по двум директориям:

- директория для пакетов, проходящих в Sisyphus;
- директория для не проходящих.

В целом работу программы можно разбить на 3 стадии:

- получение пакетов;
- предварительная проверка пакетов;
- пересборка (если она нужна);
- последующая проверка;
- формирование отчёта.

Получение пакетов может производится из одного или нескольких мест. Также поддерживается несколько разных способов получения пакетов, на текущий момент поддерживаются:

- local folder — возможность забирать пакеты из локальной директории;
- package list — возможность задать файл со списком файлов;
- folder over ssh — возможность получать пакеты через ssh;
- rsync — возможность получать пакеты через rsync.

На следующей стадии производятся быстрые проверки, которые позволяют заключить, что пакет содержит грубые ошибки и дальнейшие проверки проводить не следует (например, можно даже не пробовать пересобирать пакет, если у поступившего исходного пакета отсутствует gpg подпись). Так как любой из приходящих пакетов может нести угрозу безопасности системы, то работа с ними не должна проводится в реальой (рабочей) системе. Для обеспечения безопасности проверка пакетов имеет следующий порядок:

1. Создается «путой» chroot.

2. В chroot устанавливаются пакеты необходимые для осуществления проверки на этой стадии.

3. В chroot копируется первый из очереди и проверяется. Далее копируется и проверяется следующий пакет. Этот этап повторяется для всех поступвших пакетов.

Любой не нормальный код возврата программ в chroot считается ошибкой и пакет — не прошедшим проверку. В идеале для каждого пакета должен создаваться новый chroot, но на данный момент этого не делается.

После того, как пакеты пройдут предварительные проверки, их можно попытаться пересобрать. Эта стадия не является обязательной и может быть пропущена. Необходимость этой стадии определяется в конфигурационном файле для каждого места, куда выкладываютя пакеты, предназначенные для репозитория Sisyphus.

Пересборка пакетов производится с помощью hasher. К сожалению, очень трудно выяснить, в каком порядке нужно пересобирать пакеты, потому что в исходном пакете не содержится явного списка бинарных пакетов, которые могут быть из него собраны. Поэтому все пакеты выстраиваются в единую очередь. На данный момент она формируется по дате сборки исходных пакетов. Пакеты, имена которых начинаются с префикса 'lib*' переносятся в начало очереди. Пакеты в очереди пересобираются циклически. Удачно пересобравшиеся и не пересобравшиеся пакеты удаляются из очереди. Каждый удачно собравшийся пакет помещается во временный репозиторий. Этот репозиторий используется при каждой новой попытке сборки из очереди. Если пакет не пересобрался по причине того, что для его сборки не хватает каких-либо пакетов или не хватает пакетов определённой версии, то пересборка пакета не считается неудачной и откладывается. После того как достигается конец очереди, процесс пересборки начинается сначала. Пересборка останавливается тогда, когда на очередном цикле очередь не уменьшится ни на один пакет.

После стадии пересборки в hasher становится не важно, откуда был получен пакет. И можно считать, что все пакеты, попавшие в систему incominger, правильно пересобираются.

Далее выполняется последующая проверка удачно пересобравшихся пакетов. В неё входит проверка с помощью sisyphus_check и проверка на корректную устанавливаемость. Для этого создаётся новый chroot, в нем устанавливаются все необходимое для установки проверяемого

пакета. После этого запускается программа osec, затем в chroot устанавливается проверяемый пакет и снова запускается osec. Затем пакет удаляется и опять запускается osec. Таким образом можно узнать, какие файлы были созданы и изменены после установки бинарного пакета и какие были удалены после его удаления. Если после установки пакета меняются права у файлов, ему не принадлежащих, или после удаления пакета остаются какие-нибудь принадлежащие ему файлы, то считается, что пакет содержит ошибки и он не пропускается в репозиторий Sisyphus.

На последнем этапе формируется детальный отчёт о проверке пакетов. По этому отчёту можно вынести решение, пропускать или нет пакеты в Sisyphus и если не пропускать, то по какой причине.

17:00–17:30: Кофе

17:30–18:00: Закрытие конференции

19:00–19:30: Отъезд на Linux Fest (по желанию)