АНО «Институт логики, когнитологии и развития личности»
ALT Linux
НОУ «ИПС-Университет г. Переславля им. А. К. Айламазяна»
Институт Программных Систем РАН

Девятая конференция «Свободное программное обеспечение в высшей школе»

Переславль, 25–26 января 2014 года

Тезисы докладов

Москва,
Альт Линукс,
2014

Девятая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переславль, 25–26 января 2014 года. М.: Альт Линукс, 2014. — 144 с. : ил.

В книге собраны тезисы докладов, одобренных Программным комитетом девятой конференции «Свободное программное обеспечение в высшей школе».

ISBN 978-5-905167-15-7

Программа конференции

25 января

11.00-12.00 Регистрация участников и заселение

Дневное заседание
12.00–15.00

12.00 А. Е. Новодворский. Открытие. Информация оргкомитета

12.10–12.40 С. М. Абрамов, член кор. РАН.

12.40–13.10 Н. Н. Непейвода, проф.

13.10–13.40 Е. А. Роганов

13.40–15.00 Перерыв на обед

Вечернее заседание
15.00–19.30

15.00–15.30 А. С. Черепанов

15.30–16.00 А. Г. Боковой

16.00–16.30 Д. А. Костюк

26 января

Утреннее заседание
9.30–13.30

Дневное заседание
14.50–18.30

Вне программы

Николай Непейвода
Переславль-Залесский, ИПС РАН

О необходимых знаниях и умениях для программистов суперкомпьютеров

Суперкомпьютеры высшего уровня в последние годы представляют собой быстро изменяющиеся сложные структуры, часто построенные на базе разнородных блоков. Это связано с тем, что они подошли к трём барьерам: скорость света, выделение тепла (предел Ландауэра) и предел Чейтина (предел сложности конструкций, которые может понять человек). Казалось бы, последний предел самый невинный. Но, как отмечают многие, именно он сейчас стал определяющим в очень многих случаях. Появился термин «агрессивное невежество» программистов, которое приводит к тому, что программа оказывается полностью неадекватна системе на которой она исполняется.

Параллельные алгоритмы с самого начала строятся по другим законам, чем последовательные. Для их создания нужно логическое мышление, а логика как наука практически выброшена из учебных программ в России. «Математическая логика», под которой понимают классическую и в которой полностью опущен как неформализуемый раздел, посвящённый взаимосвязям формального и содержательного, лишь ещё более отупляет. Философская логика не лучше. А давать науку в комплексе не хватает ни часов, ни сил, ни методических разработок.

Логически созданная программа с самого начала максимально совместна (незаслуженно забытый термин из Алгола-68), что означает возможность легко уложить на любую подходящую для неё архитектуру. Это даёт отнюдь не оптимальное для конкретной аппаратуры решение, но на порядок лучше того, которое получается «распараллеливанием» изначально последовательной программы. Далее уже нужно одновременно и в комплексе видеть как алгоритм, так и ресурсные требования, что требует более высокого уровня знаний и умений.

Разные архитектуры суперкомпьютеров требуют для получения решения выше среднего различных неклассических логик. Это дополнительная объективная трудность: человек, который, преодолев

недостатки образования, вошёл в одну из архитектур (например, в матрично-конвейерную), теряется, когда надо программировать, скажем, на ассоциативном процессоре обработки данных. Здесь нужен опыт преобразования математических структур (алгебраические концепции). Ситуация ещё обостряется в связи с перспективой перехода к алгебраическим процессорам. Поэтому современный программист суперкомпьютеров, не знающий алгебры, может освоить максимум одну структуру и перестроить мозги под неё. При переходе на другую его ум полностью сломается. А в бурно развивающейся отрасли такое недопустимо.

Но здесь ещё один важный аспект. Если теория и практика резко различаются по уровню, то начинаются нежелательные эффекты. Скажем, математический идиотизм, когда математика развита до третьего уровня (преобразования структур функций над объектами), а практика на нулевом. Но ещё хуже случай, когда практика на первом. Тогда математик самоуверенно выдаёт практические рекомендации отрицательной ценности. А вот если практика у него на втором (хотя бы освоение на дилетантском уровне всей системы стилей программирования), рекомендации становятся адекватными, он может посоветовать программисту, как перестроить структуры под новую архитектуру компьютеров. Аналогично, если теория на втором, а программирование на третьем, то программист сможет внятно поставить вопрос аналитику, подобрать адекватный инструмент и увидеть взаимосвязь программных структур и ресурсов в данной конкретной безумной архитектуре сверхпродвинутого суперкомпьютера.

Поэтому я полностью согласен с предложением С. М. Абрамова: суперкомпьютерная отрасль требует обучения студентов с самого начала по совершенно другим программам, чем обычные информатики. Мышление на уровне конкретных моделей неизбежно приводит к агрессивному невежеству в данной области. Нужно мышление по крайней мере на уровень выше (метамодели). На самом деле адекватно области мышление на уровне преобразований метамоделей. И очень важно здесь сбалансировать теоретические и практическое вопросы (в теории овладеть преобразованиями структур, а не фиксированной структурой чисел и их конкретных преобразований; в практике — концепциями и преобразованиями из одной формы в другую, а не отдельным «автоматом Калашникова»). В теории — логика и математика на базе алгебры, на практике — СПО.

Евгений Роганов
Москва, МГИУ

Проект «Портал поддержки образовательного процесса»

Аннотация

Образовательные порталы сейчас широко используются при любой форме обучения, однако качество размещаемых на них учебных материалов часто не соответствует современным требованиям, так как создающие эти материалы преподаватели не обладают достаточными знаниями в области современных информационных технологий. Решить эту проблему помогает создание портала технической поддержки образовательного процесса.

Актуальность задачи

Веб-технологии сейчас становятся неотъемлемой частью инструментария, используемого при любой форме обучения. Без образовательного веб-портала представить себе современный университет просто невозможно. Вне зависимости от того, как организован этот портал, на нём содержатся учебные материалы различных форматов, создаваемые преподавателями.

Как, где и с использованием каких технологий преподаватели создают эти материалы? Часто это происходит на домашнем компьютере и сводится к подготовке документа в одном из стандартных для эпохи офис-технологий форматов (например, с использованием Microsoft Office или Libre Office). Более подготовленные преподаватели, знакомые с системой компьютерной вёрстки TEX, способны создавать pdf-документы. Некоторые умеют использовать новейшие разработки в области веб-технологий и конструировать красивые и удобные в использовании гипертекстовые документы и презентации.

В результате учебные материалы, размещаемые на образовательном портале университета, оказываются очень разнородными, не стандартизованными и в значительной мере не удовлетворяющими современным требованиям. Преподавателям, которые пытаются применять для создания материалов новейшие технологии, приходится преодолевать значительные технические трудности, устанавливая на

свой компьютер необходимое программное обеспечение и постоянно обновляя его.

Наличие портала технической поддержки образовательного процесса, на котором преподаватели университета могут подготавливать стандартизованные учебные материалы с использованием новейших технологий, существенно повышает качество учебного процесса. Это помогает основной массе преподавателей преодолеть желание «оставаться на месте», используя лишь давно знакомые им методы подготовки документов, и значительно облегчает жизнь тем из преподавателей, которые сами стремятся применять современные технологии.

Инструментарий

К настоящему моменту времени среди свободных программных продуктов и технологий имеются все необходимые компоненты для создания веб-приложения, которое может выполнять функции портала поддержки образовательного процесса. Среди них (список не полный): Linux[1] — базовая ОС, Ruby[2] — язык для скриптов пред- и постобработки документов, Ruby on Rails[3] — среда разработки веб-приложений, Haskell[4] — функциональный язык, незаменимый при решении ряда задач, Pandoc[5] — конвертер документов, Markdown[6] — простой и удобный язык разметки, LanguageTool[7] — программа проверки грамматики и стиля, TeX Live[8] — современный дистрибутив LaTeX, HTML5[9] — новый стандарт языка HTML, всё лучше поддерживаемый браузерами, MathJax[10] — библиотека для визуализации математических формул в браузерах, jQuery[11] — библиотека, обеспечивающая взаимодействие JavaScript и HTML, Twitter Bootstrap[12] — библиотека HTML и CSS шаблонов, Reveal.js[13] — одно из средств создания HTML-презентаций.

Описание функциональности веб-портала

Портал является Rails-приложением, обеспечивающим online-преобразование документов из одного формата в другой для максимально широкого множества форматов, динамическое создание учебных материалов в гипертекстовом виде, презентаций для лекций и материалов в pdf-формате для печати из единого исходного файла в соответствии с выбранными шаблонами и набором параметров.

Входным языком является расширение Pandoc-версии языка Markdown, а для достижения требуемой функциональности используется набор Ruby-скриптов, с помощью которых выполняется предкомпиляция входного языка и, при необходимости, дополнительная посткомпиляция документов, получаемых в результате работы конвертера Pandoc.

Литература

[1] Веб-сайт ОС Linux. http://www.linux.org

[2] Ruby — лучший друг программиста. https://www.ruby-lang.org/ru

[3] Web framework. http://rubyonrails.org

[4] The Haskell Programming Language. http://www.haskell.org/haskellwiki/Haskell

[5] A universal document converter. http://johnmacfarlane.net/pandoc

[6] Язык Markdown. http://en.wikipedia.org/wiki/Markdown

[7] Proofreading software. https://www.languagetool.org

[8] A comprehensive TEX system. http://www.tug.org/texlive

[9] Язык HTML5. http://www.w3.org/TR/html5

[10] JavaScript display engine. http://www.mathjax.org

[11] JavaScript library. http://jquery.com

[12] HTML, CSS, и Javascript инструментарий. http://bootstrap-ru.com

[13] A framework for creating presentations using HTML. https://github.com/hakimel/reveal.js

Андрей Черепанов
Москва, ALT Linux

Что может предложить Альт Линукс для образования

Для образовательных учреждений различного уровня (от детских садов для школ) Альт Линукс предлагает различные решения по нескольким направлениям:

- дистрибутивы GNU/Linux как для серверов, так и для рабочих станций;
- программное обеспечение для как для обучения, так и для административной и хозяйственной деятельности;
- технологии централизованного управления компьютерами в классе и всём учреждении;
- методические материалы как в электронном виде, так и в виде книг.

Перед разработкой комплектов образовательных дистрибутивов ставятся следующие задачи:

- Нулевая стоимость пользовательских лицензий.
- Возможность централизованного управления учебным классом.
- Централизованное управление аутентификацией через сервер каталогов.
- Прозрачное использование сетевых ресурсов в режиме Single Sign-On.
- Возможность сетевой загрузки бездисковых клиентов с сохранением данных на сервере.
- Дистрибутивы должны содержать достаточное количество свободного программного обеспечения для организации учебной деятельности по образовательным стандартам.
- Дистрибутивы должны обеспечивать разнообразные и удобные каналы коммуникаций как внутри образовательного учреждения, так и с остальным миром.

- Комплект должен быть максимально локализован на русский язык.

Мы бы хотели заняться такими перспективными направлениями в области образования, которые наиболее востребованы среди преподавателей:

- Ограничения изменения параметров настройки (режим киоска, когда никакие изменения учащегося не сохраняются).
- Централизованное управление рабочими станциями со школьными дистрибутивами (управление программным обеспечением, параметрами настройки и их принудительным обновлением).
- Комбинирование функциональных комплектов программного обеспечения (например, развертывание сетевых служб на учительском компьютере).
- Терминальные классы
- Расширение документации и руководств для пользователей и администраторов.
- Сбор и включение в комплекты методических материалов в различных видах (методички, видеоуроки, презентации).

Александр Боковой, Martin Kosek
Эспоо, Финляндия, Red Hat

Проект: FreeIPA `http://freeipa.org`

Студенты и свободные проекты: опыт Red Hat Identity Management

Аннотация

Многие компании сотрудничают с университетами и привлекают студентов для получения практики. Результаты работы практикантов, как правило, используются для защиты курсовых и дипломных работ и редко становятся доступны более широкому обществу. Red Hat сотрудничает с университетами с целью вовлечения студентов в разработку свободных проектов. Важным критерием успешности работы является ее принятие проектом, а не только достижение учебных задач.

Доклад посвящен опыту разработки свободного программного обеспечения в области безопасности и управления инфраструктурой

предприятий: FreeIPA, SSSD, Samba, MIT Kerberos, BIND, sudo, 389-ds. На протяжении нескольких лет группа инфраструктуры предприятий в Red Hat работает со студентами нескольких университетов в Чехии и США. Практиканты заняты разработкой функционала в различных свободных проектах, параллельно защищая свои курсовые и дипломные работы.

В докладе рассматриваются конкретные примеры проектной работы и прослеживается их судьба после окончания дипломных проектов.

Компания Red Hat разрабатывает многочисленные продукты на основе свободного программного обеспечения. Инженеры компании работают во многих из 80 офисов в разных странах мира, а также удаленно, из своего дома. Два крупнейших инженерных офиса находятся в Бостоне, США, и Брно, Чехия. В этих городах известные университетские традиции в технологической сфере и естественным является развитие контактов с находящимися в них университетами.

Любая группа внутри Red Hat может организовать стажировку. Несмотря на то, что стажировка сама по себе не требует обязательно участия университета в программе, сама концепция стажировки предполагает активное взаимодействие с коллегами в офисе, поэтому с университетами в непосредственной близости от офисов разработки формируются наиболее тесные связи.

Результаты стажировки могут быть использованы для написания курсовых и дипломных работ. В связи с тем, что и в США, и в европейских странах используется двухступенчатая система аттестации, то в реальности дипломные работы делятся на бакалаврские и магистерские, но сложность работ зависит скорее от амбициозности проектов, чем от требований учебных организаций.

На сегодняшний момент в программе по работе с университетами принимают участие 12 чешских и словацких университетов в Европе, MIT, Carnegie Mellon University и несколько других университетов в США.

Для координации работы над различными темами между университетами была написана система управления дипломными работами, сама защищенная как дипломная работа. Увидеть ее в действии можно по адресу `https://thesis-managementsystem.rhcloud.com/`, а исходный текст приложения доступен `https://github.com/jcechace/Thesis-management-system`.

Так как практически вся разработка программного обеспечения Red Hat ведется в рамках публичных проектов, работу, которую ста-

жёр ведет над проектом, видят все — и внутри компании, и за ее пределами. Стажёр выступает в роли публичного лица, его корреспонденция представлена в архивах почтовых рассылок соответствующих проектов, его предложения оцениваются всеми участниками проекта, в том числе и не из Red Hat, он вынужден втягиваться в динамику самого проекта. В результате, оценка результатов стажировки более объективна, но особых скидок на «стажёрность» никто делать не будет. Безусловным плюсом является тот факт, что после окончания университета бывший стажёр может продемонстрировать результаты своего труда потенциальным работодателям.

На данный момент (январь 2014 года) доступно около полусотни вакансий инженерных стажировок, сроком от нескольких месяцев до почти года.

Поскольку один из основных продуктов компании — Red Hat Enterprise Linux — используется для организации инфраструктуры предприятий, Red Hat ведет целенаправленную работу по улучшению свободного программного обеспечения в сфере управления ресурсами. В 2007 году был запущен проект FreeIPA, задачей которого стало создание простого и удобного набора средств для развертывания инфраструктурных объектов, управления информацией о пользователях, группах и машинах, разграничению доступа к ним и обеспечения защищенной инфраструктуры для решения бизнес-критических задач. Основой проекта стало объединение существующих стандартизированных сетевых протоколов в единую среду, улучшение средств управления конфигурациями серверных и клиентских компонент, реализующих эти сетевые протоколы и устранение «болевых точек», мешающих совместному использованию этих компонент.

Несмотря на то, что многие из используемых проектов существуют десятилетия, в них всегда можно найти задачи, за которые никто не берется по различным причинам, зачастую — из-за нехватки времени или необходимости координировать взаимодействие с другими свободными проектами. Такие задачи представляют собой идеальную цель для стажировок, как показывает, в том числе, и деятельность в рамках более масштабного проекта Google Summer of Code.

В рамках работы над проектами Identity Management за последние несколько лет было проведено более десятка стажировок. Вот некоторые из них.

Автоматическая ротация ключей Kerberos для машин и служб

Проект выполнялся Ондреем Косом из Brno University of Technology. Ондрей написал демон `keymonger`, работающий на узлах, входящих в область Kerberos, и следящий за временем жизни ключей служб Kerberos, работающих на этих узлах. По достижении срока жизни ключа, `keymonger` автоматически запрашивает обновление ключа. Таким образом, приложениям не требуется специально заботиться об обороте ключей. Принцип действия взят из существующего демона `certmonger`, который используется во FreeIPA для ротации сертификатов, которыми подписывают свои данные службы, обменивающиеся информацией с использованием SSL и TLS.

К сожалению, работа не была вовремя интегрирована из-за изменений приоритетов. Бакалаврская работа, тем не менее, была успешно защищена. Около полутора лет после защиты работы Ондрей проработал в Red Hat над проектом SSSD, а сейчас продолжает образование и готовится к защите магистерской диссертации в 2015 году.

Учебный курс по использованию библиотеки talloc и документация к библиотеке

Павел Брежина из Masaryk University of Brno выполнил важную работу по документированию библиотеки talloc, активно используемой в проектах Samba, FreeIPA, SSSD, realmd и многих других. Эта библиотека реализует иерархический менеджер памяти, эффективный для работы со структурированными сетевыми протоколами. Из-за того, что библиотека является ключевой для работы указанных проектов, понимание ее устройства и способов использования крайне важно для новых разработчиков. Однако, как это часто и бывает, документация была расчитана на тех, кто и так уже пользуется библиотекой.

Павел написал подробную документацию об устройстве библиотеки и хороший с методической точки зрения курс по использованию talloc. В процессе были обнаружены ошибки, которые Павел исправил и успешно интегрировал в Samba — как исправления, так и саму документацию.

Полный текст диплома доступен на http://is.muni.cz/th/359290/fi_b/?lang=en, а учебный курс — http://talloc.samba.org/talloc/doc/html/libtalloc__tutorial.html

Сейчас Павел работает в Red Hat разработчиком SSSD.

Тестирование производительности MIT Kerberos

Петр Шпачек разработал набор тестов для оценки производительности MIT Kerberos, в частности, в случае хранения базы данных центра распространения ключей в LDAP. В процессе написания магистерской диссертации Петром были обнаружены некоторые проблемы в производительности MIT Kerberos, о них было сообщено в MIT, они были исправлены. Как результат, Петр успешно защитил свой диплом и работает в Red Hat над драйвером LDAP для сервера DNS, bind-dyndb-ldap, https://fedorahosted.org/bind-dyndb-ldap/.

Управление публичными ключами SSH во FreeIPA

Ян Холаста работал над созданием централизованной системы хранения публичных SSH-ключей для пользователей и машин, интегрированной в систему управления FreeIPA. Для новых машин, добавляемых в домен FreeIPA, ключи SSH автоматически загружаются на сервер FreeIPA и регистрируются в DNS. Пользователи могут добавить дополнительно свои ключи через интерфейсы управления FreeIPA, а при попытке установления сессии по SSH на сервера под управлением FreeIPA происходит верификация публичных ключей этих узлов на основании имеющихся в центральном хранилище. Публичные ключи пользователей, таким образом, доступны централизованно и могут быть добавлены администраторами (и другими пользователями) на нужные сервера с использованием средств автоматизации.

Работа была интегрирована во FreeIPA версии 3.0 и доступна в RHEL 6.x. Сейчас Ян работает в Red Hat над проектом FreeIPA.

Проксирование Kerberos поверх HTTP(S)

Робби Харвуд из Carnegie Mellon University летом 2013 года работал над реализацией протокола MS-KKDCP, проксирования Kerberos поверх HTTP(S). Робби разобрался со спецификацией MS-KKDCP,

который еще не реализован нигде, за исключением Windwos Server 2012, и внес необходимые изменения в MIT Kerberos. Дополнительно он создал прототип самого прокси-сервера, который использует модифицированную версию MIT Kerberos.

Код проекта доступен на GitHub:
`https://github.com/frozencemetery/krb5` и
`https://github.com/frozencemetery/krb-proxies`.

Интеграция в MIT Kerberos запланирована на версию 1.13 и позднее.

Дмитрий Костюк, Павел Луцюк
Брест, Брестский государственный технический университет

Применение виртуальных машин в составе иллюстрированных обзоров истории программного обеспечения

Аннотация

Рассматривается опыт использования виртуальных машин в качестве замены копий экрана в иллюстрированной истории графических интерфейсов. Охарактеризована степень наполненности материалов. Рассмотрена проблематика применения вложенной виртуализации с внешней виртуальной машиной, взаимодействующей с документом и активирующей снапшоты. Описано решение проблемы свободного распространения иллюстрированных обзоров с сохранением возможности включения гостевых систем с несвободной лицензией за счет автоматического перестроения итогового документа. Используемое на хост-системе ПО включает гипервизор QEMU, а также свободные HTML-фреймворки для работы с протоколом VNC и отображения интерактивной иллюстрированной хронологии.

Хотя технически для того, чтобы работать с новыми информационными технологиями, не обязательно знать историю их развития, тем не менее специалист, который формулирует либо применяет современную теорию без знания ее истории, рискует лично повторять ошибки предшественников одну за другой.

В русле данного утверждения находится настоящая разработка, предоставляющая пользователю локальной сети либо рабочей станции комплект связанных хронологически HTML-документов, каждый

из которых содержит описание особенностей конкретной графической операционной системы и ее живую иллюстрацию в виде встроенного фрейма с экраном работающей виртуальной машины (благодаря производительности современных ноутбуков и настольных ПК такая задача оказывается достаточно легко выполнимой). Информационный контент разработки основан на лекционном материале по истории графического интерфейса, включающем 40 настольных и 30 мобильных операционных систем и графических оболочек.

Техническая инфраструктура, позволяющая реализовать такие информационные материалы, рассмотрена нами в [1] и включает следующие компоненты:

- виртуальная машина QEMU с аппаратной поддержкой виртуализации;
- VNC-клиент noVNC, написанный на JavaScript и HTML5;
- JavaScript-фреймворк для отображения информационных материалов.

Запуск схемы, показанной рис. 1-а, выполняется стартовым скриптом, сканирующим вложенные подкаталоги в поисках элементов обзора: страниц информационного контента, образов виртуальных машин и скриптов для их запуска. Стартовый скрипт выдает найденным виртуальным машинам номера портов и перестраивает HTML-документ для включения страниц с информационными материалами в хронологию. Такой компонентный подход позволяет разделить информационный обзор на свободно распространяемую часть и дополнения, распространение которых ограничено условиями коммерческих лицензий.

Виртуальная машина используется как полностью изолированный контейнер для хранения мгновенных снимков запущенных ОС [2]. Выбор в пользу QEMU был сделан из-за предельно простого переноса образов виртуальных машин между компьютерами, а также из-за его способности помимо платформы x86 эмулировать процессоры SPARC, PowerPC, Motorola 68k, MIPS и ARM, что необходимо для запуска многих ОС 80-х и 90-х годов. Однако на текущий момент данная многоплатформенность осталась практически незадействованной, т. к. степень поддержки устройств на альтернативных платформах QEMU редко оказывается достаточной для запуска нужной ОС.

В то же время для поддержки практически всех устаревших Intel-несовместимых ОС либо разработаны свободные эмуляторы, либо

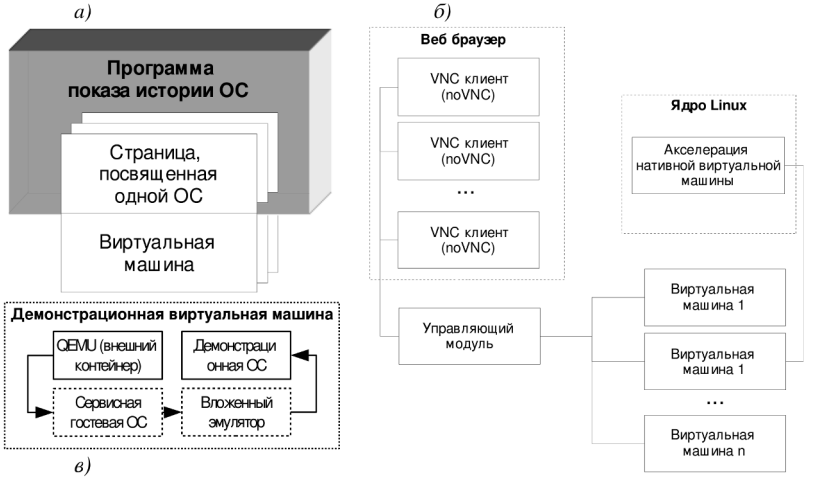


Рис. 1: Схема построения документа (а), взаимодействие компонентов системы (б) и схема вложенной виртуализации (в)

имеются эмуляторы из комплекта SDK. Первый вариант более характерен для настольных ОС, благодаря чему оказалось возможным включение в информационные материалы ОС Xerox Alto, Amiga, RiscOS, Apple Lisa, MacOS версий 1.x, 7.x и X. Проприетарные эмуляторы настольных ОС редки и принадлежат производителю самой ОС, как в случае Xerox GlobalView. В случае ОС мобильных устройств преобладают эмуляторы из состава SDK: Psion EPOC16 и EPOC32, PalmOS, Magic Cap, Windows CE, предрелизные версии Android SDK. Исключением из списка является свободный эмулятор Open Einstein, позволяющий запускать NewtonOS.

Однако перечисленные эмуляторы не поддерживают ни снапшоты, ни необходимый в нашем случае протокол VNC. В результате запуск большой части демонстрируемых ОС осуществляется по схеме вложенной виртуализации (рис. 1-в), где QEMU играет роль внешнего контейнера.

Многие ОС не требуют вложенной виртуализации и потому обходятся без внутреннего эмулятора. В нашем случае это настольные Windows версий 1.x, 2.x, 3.x и 95, GEM от Digital Research, GEOS от Berkeley Softworks, а также ряд мобильных ОС: Pen Windows, Maemo,

Android, WebOS (не в последнюю очередь благодаря тому, что QEMU часто входит в состав SDK).

Предназначенная для свободного распространения часть обзора содержит ОС, являющиеся свободным ПО (изначально, по причине крайнего устаревания, либо будучи свободным клоном). Это различные графические оболочки Unix и Linux для настольных и мобильных компьютеров, а также GEM, Amiga, RiscOS, HaikuOS.

Представляемый материал все еще находится на стадии наполнения; на данный момент в обзоре пропущена часть объектов, играющих важную роль в истории развития графических ОС. Главным образом это новые версии ОС от Microsoft и Apple. Кроме того, DOS-оболочка Visi On, IBM OS/2 всех версий и NeXTSTEP проявили несовместимость с актуальными версиями QEMU. В качестве замены NeXTSTEP в данный момент использован Linux-аналог GNUStep. Проблема с запуском Visi On и OS/2 может быть решена использованием Bochs или VirtualBox, что нежелательно как с точки зрения использования ресурсов, так и в плане портативности информационных материалов.

Число сыгравших важную роль в истории GUI графических оболочек, для которых не существует доступных версий, оказалось весьма невелико: на текущий момент таковыми можно признать многооконный интерфейс Smalltalk конца 70-х, Xerox Star Document Processor, а также мобильные системы PenPoint OS и IBM Simon.

В схеме вложенной виртуализации можно заметить еще один компонент — сервисную гостевую ОС, которая используется для запуска эмулятора. В каждом случае на ее выбор влияли требование минимального потребления памяти, возможность использования циклов бездействия процессора, а также поддержка шины USB для возможности эмуляции позиционирования в абсолютных координатах. Последнее требование важно для комфортного управления мышью в виртуальной машине [1]. В качестве сервисных гостевых ОС помимо нескольких версий Linux использованы FreeDOS и ReactOS. В отношении ReactOS можно дополнительно заметить, что она идеально отвечает всем трем требованиям, и таким образом в нашем собственном опыте это первый случай ее удачного применения.

Литература

[1] *Костюк Д. А.* Особенности использования виртуализованных окружений, внедренных в презентационные материалы // Восьмая конференция «Свободное программное обеспечение высшей школе»: тез. докл. / Переславль, 26–27 января 2012 года. М.: Альт Линукс, 2012. — С. 83–86.

[2] *Костюк Д. А., Дереченник С. С.* Построение прозрачных виртуализованных окружений для изоляции уязвимых программных систем // Комплексная защита информации: матер. XVI научно-практич. конф., Гродно, 17–20 мая 2011 г. Гродно, 2011. — С. 209–212.

Костарев Алексей
Пермь, ООО НЕВОД

Проект: Семейство продуктов контент-репозитория C2R (Apache Hadoop)

Применение онтологического подхода для анализа текстов в облачном контент-репозитории C2R

Аннотация

В докладе обсуждаются основные концепции применения онтологического подхода для анализа текстов в рамках совместных работ ГК ИВС (группы компаний ИВС) и ПГНИУ (Пермский Государственный Научно-Исследовательский Университет). В качестве хранилища текстов и онтологий используется репозиторий C2R, функционирующий в кластере Apache Hadoop и использующий облачные программные продукты HDFS, HBase, Lily, Solr под свободными лицензиями. Создание и ведение онтологий обеспечивается на основе данных Википедии и тематических текстовых корпусов с применением алгоритмов семантического анализа связности слов Serelex и механизмов логического вывода. Все разрабатываемые алгоритмы планируется реализовать в рамках технологий распределенных вычислений MapReduce и Apache Giraph, что обеспечивает неограниченную масштабируемость созданного решения, возможность хранения и обработки в единой базе онтологий для различных тематических областей, обработку за приемлемое время огромных (сотни петабайт) объемов данных.

В последнее время для анализа текстов все чаще используют механизмы, позволяющие учитывать семантические аспекты языка —

синонимы, гиперогимы, гипонимы, когипонимы и т.д. Это позволяет проводить поиск и анализ текстов не только по значениям конкретных слов, но и использовать семантически близкие термины для указанной предметной области. Большое значение в этой связи имеет создание онтологий терминов для различных предметных областей и поддержка их в актуальном состоянии.

Онтологии представляют собой графы различной структуры:

- ориентированные/неориентированные;
- взвешенные/невзвешенные;
- полные/k-полные;
- дерево/лес/сеть/...
- и т. п.

Основная проблема при работе с ними связана как с большим объемом графов онтологии для каждой предметной области, так и с необходимостью поддерживать на одном множестве терминов большое число графов, описывающие различные предметные области.

Другая немаловажная проблема связана с большим объемом вычислительных операций и операций ввода вывода, необходимых как для создания и ведения онтологий, так и для анализа входных текстов с использованием семантических отношений, описанных в онтологиях.

Возможное решение этих проблем лежит в применении технологий распределенного хранения и обработки информации, позволяющих использовать всю вычислительную мощность, оперативную и дисковую память кластера для решения этого класса задач.

Репозиторий C2R, разработанный нами в 2010-2012 годах, использует следующие облачные технологии:

- распределенную файловую системы *Apache Hadoop HDFS*;
- NoSQL базу данных *Apache HBase*;
- репозиторий *Lily*;
- индексатор *Solr*.

Использование данных технологий позволяет организовать облачное распределенное хранение и обработку анализируемых текстов и онтологических данных на кластере серверов Apache Hadoop.

Для оперативной (Online) работы с онтологиями используется как прямой доступ к записям *HBase* по автоматически генерируемому

ключу так и по ключу (списку ключей), получаемому в ответ на запрос к индексатору *Solr*.

Для аналитической работы по созданию и автоматическому ведению онтологий будут использованы алгоритмы распределенной обработки информации:

- *Apache Hadoop MapReduce* — создание и пополнение онтологий из внешних источников (Википедия, Викисловарь, ...)
- *Apache Giraph* (Google Pregel) — обработка распределенных графов для подсчета метрик семантической близости, получения новых знаний и т. п.

Применение данных подходов для хранения и обработки онтологий обеспечивает:

- оптимальное высоконадежное хранение онтологических (графовых) данных в распределенной облачной среде;
- неограниченность объема онтологий и анализируемых текстов (до сотен петабайт);
- автоматическое горизонтальное масштабирование базы;
- распределенная обработка данных с использованием всех мощностей кластера;
- возможность хранения в одной базе (таблице) онтологий для неограниченного числа различных тематических областей с их оптимизацией и получения новых данных.

Разрабатываемую программную платформу предполагается использовать для создания следующего класса программных продуктов:

- *C2R Archivarius* — хранение, обработка и поиск текстовых данных;
- *C2R Mailing* — мониторинг Интернет- и других Online-ресурсов для обнаружения информации релевантной запросам пользователей.
- *C2R DLP* — создание систем уровня DLP (*Data Loss/Leak Prevention*) для контроля трафика и предотвращения утечек.

Создание данных систем планируется производить в два этапа:

1. Создание прототипа с поддержкой Online-режима для хранения и анализа текстов без использования онтологий (2-3 квартал 2014 года);

2. Создание на основе прототипа полнофункционального решения с использованием онтологий (3-4 квартал 2014).

Прототип

На данном этапе (см. рис. 1) обеспечивается хранение, индексация и поиск документов с использованием стандартных опробованных средств:

- создание и хранение документов — репозиторий *Lily*;
- индексация и поиск документов — индексатор *Solr*.

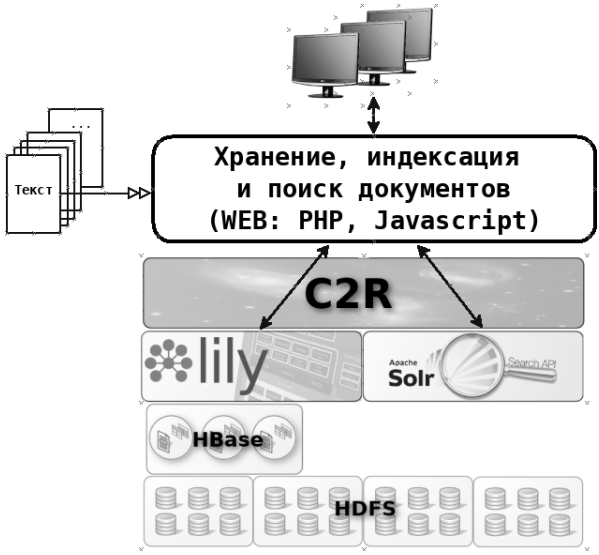


Рис. 1: Информационные потоки прототипа систем

Репозиторий Lily обеспечивает запись, чтение, модификацию, удаление входных документов (CRUD-операции) в нереляционной (NoSQL) базе данных *Apache HBase* и передачу документов для инкрементальной или полной индексации в поисковую систему *Solr*.

Поисковая система Solr, функционирующая в облачном (Cloud) режиме производит индексацию документов репозитория и обеспечивает быстрый поиск записей по их содержимому.

На основе данных сервисов создаются прототипы описанных выше программных продуктов: *C2R Archivarius*, *C2 R Mailing*, *C2R DLP*.

Полнофункциональное решение

В полнофункциональном режиме (см. рис. 2) репозиторий *C2R* обеспечивает кроме хранения и обработки входных анализируемых документов, хранение, пополнение и обогащения онтологических данных, используемых для улучшения качества поиска и более тонкой настройки систем на предметные области клиентов.

Для создания, обновления онтологий планируется использовать существующие онтологии (*Википедия*, *Wordnet*, ...), тематические текстовые корпуса. АРМ онтолога обеспечивает тонкую настройку онтологий на предметные области клиентов и формирование новых знаний по хранимым онтологиям с использованием WEB интерфейса, *Java-framework Jena* и других инструментариев.

Для вычисления семантических расстояний между терминами (source) онтологий и вычислений метрик соответствия анализируемых входных текстов планируется использовать алгоритмы инструментария *Serelex*, реализованные в среде распределенных вычислений по технологиям *Apache MapReduce* и *Apache Giraph*.

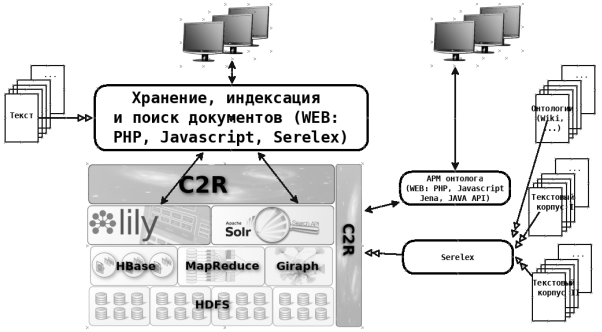


Рис. 2: Информационные потоки полнофункционального решения

Литература

[1] *Костарев А.Ф., Полещук А.Н*, Контент-репозиорий C2R. Свидетельство о государственной регистрации программы для ЭВМ №2011617248/, 2011

[2] *Чуприна С.И.*, Трансформация традиционных информационных систем административного типа в интеллектуальные информационные системы на базе онтологий/, 2011

[3] *Костарев А.Ф., Полещук А.Н.*, Разработка свободно распространяемого контент-репозитория для развёртывания информационных систем на принципах «облачных» вычислений, 2012

[4] *Панченко А., Романов П., Романов А., Филиппович А., Филиппович Ю.*, Серелекс: поиск и визуализация семантически связанных слов., 2013

[5] *Alexander Panchenko*, Similarity Measures for Semantic Relation Extraction, 2013

Михеев Андрей Геннадьевич
Москва, НИТУ МИСиС, МЭСИ, Консалтинговая группа РУНА
Проект: RunaWFE `http://runawfe.org/`

Свободная система управления бизнес-процессами RunaWFE как инструмент для новой парадигмы программирования

Аннотация

В последние годы программистские компании стали использовать системы управления бизнес-процессами (СУБП) при автоматизации предприятий — клиентов. В данном случае использование СУБП является не требованием клиента, а архитектурным решением: Внедрение и сопровождение оказывается быстрее и дешевле традиционной автоматизации. Эти преимущества совпадают с преимуществами парадигмы ООП относительно процедурного программирования. Проводя аналогию, можно утверждать, что через некоторое время потребуется большое количество специалистов с процессным мышлением, заметно отличающимся от традиционного мышления ИТ-специалистов.

Обучение их надо начинать уже сегодня.

В докладе представлен опыт обучения студентов элементам этой технологии на свободном ПО RunaWFE, полученный в НИТУ МИСиС, МЭСИ и УГАТУ.

Преимущества процессной автоматизации

На предприятиях с устойчивыми повторяющимися цепочками операций внедрение, настройка и сопровождение систем на основе СУБП оказывается быстрее и дешевле традиционной автоматизации, при которой для различных задач и подразделений разрабатываются отдельные компоненты приложения.

СУБП позволяют:

- Быстро адаптировать разработку к изменению задач и появлению новых идей за время разработки
- Понизить стоимость разработки за счет:
  - Разработки бизнес-процессов средствами СУБП вместо написания кода
  - Исключение взаимодействия программистов с заказчиком. Бизнес-аналитику и заказчику гораздо более комфортно взаимодействовать друг с другом при совместной разработке основных элементов схемы исполнимого бизнес-процесса, чем заказчику и программисту при обсуждении текста технического задания
  - В этом случае программист освобождается от рутинных задач и может сосредоточиться на разработке сложных графических элементов и коннекторов, что повышает эффективность его труда
- Понизить стоимость технической поддержки
- Существенно понизить стоимость доработок и сопровождения

Эти преимущества (быстрее, дешевле, легче в поддержке и сопровождении) совпадают с преимуществами парадигмы объектно-ориентированного программирования по сравнению с почти вытесненной ей из практики парадигмой процедурного программирования. Понятие парадигма рассматривается в данном случае в терминах концепции парадигм программирования Роберта Флойда [1], которая является расширением концепции парадигм Томаса Куна, предложенной в работе «Структура научных революций» [2].

Потребность в специалистах, обладающих процессным мышлением, и предлагаемые подходы к их обучению

Новая парадигма объектно-ориентированного программирования потребовала новых специалистов, обладающих мышлением, сильно отличающимся от традиционного мышления процедурных программистов. Проводя аналогию с процессной автоматизацией, можно утверждать, что активно развивающаяся в настоящее время автоматизация с использованием СУБП, после достижения некоторого уровня использования в бизнесе, потребует большого количества специалистов — бизнес-аналитиков с процессным мышлением, заметно отличающимся от мышления ИТ-специалистов по традиционной автоматизации предприятий.

Готовить этих специалистов в ВУЗах имеет смысл уже сегодня. По аналогии с обучением программированию, обучение студентов разработке бизнес-процессов можно разделить на две части:

1. изучение нотаций описания бизнес-процессов и обучение работе с конкретными СУБП (аналог обучения синтаксису языков программирования и работе с конкретными компиляторами)
2. изучение различных возможных вариантов реализации в виде исполнимых бизнес-процессов тех или иных типичных ситуаций в бизнесе предприятия (аналог обучения приемам программирования)

Существуют прошедшие апробацию в российских ВУЗах курсы, посвященные первой части обучения (например, в работах [3-5] обобщен опыт обучения студентов разработке исполнимых бизнес-процессов в НИТУ МИСиС, МЭСИ и УГАТУ). В соответствии с программой курса обучения студенты изучают теорию исполнимых бизнес-процессов, графические нотации описания бизнес-процессов, основные компоненты типичных СУБП, а также получают практический опыт разработки и исполнения простейших бизнес-процессов.

В рамках таких курсов обучения изучаются и закрепляются на практике вопросы работы с переменными бизнес-процессов, правилами выбора маршрута движения точек управления, возможности задания сроков выполнения заданий. Разработанные бизнес-процессы студенты исполняют под разными ролями в программной среде.

Учебные курсы, посвященные приемам построения различных решений процессной автоматизации на основе исполнимых бизнес-

процессов, в настоящее время еще только создаются. Рассмотрим приемы процессной реализации нескольких типичных сценариев, которые можно использовать для обучения студентов в рамках курса процессного управления второго типа.

1. Действие должно быть выполнено одновременно двумя исполнителями (например, сотрудник должен передать документ должностному лицу, или расписаться в документе должностного лица). Как правило, студенты пытаются реализовать выполнение такого сценария путем последовательного расположения двух узлов на схеме бизнес-процесса, при этом исполнителем в первом узле является сотрудник, а во втором — должностное лицо (См. Рис. 2).

Однако, практика эксплуатации СУБП на предприятиях показывает, что такое решение является неудачным. Обычно подписание (передача) документа происходит на рабочем месте должностного лица. Если сотрудник отметит выполнение задания до того, как пойдет с документом к должностному лицу, то во многих случаях эта отметка о выполнении задания окажется ложной, так как у сотрудника могут случиться более важные дела и он может изменить решение пойти к должностному лицу прямо сейчас. При этом задание будет удалено из его списка заданий и он легко может забыть, что задание реально не выполнено.

Если же сотрудник сначала пойдет к должностному лицу и подпишет (отдаст) документ, то задание у должностного лица появится только после того, как сотрудник вернется на свое рабочее место и отметит выполнение задания. Это может произойти через длительное время после реального выполнения задания и должностное лицо может уже не помнить, принимал ли он документ у сотрудника. Кроме того, в момент прихода сотрудника, должностное лицо не будет уверено, должно ли оно вообще принимать документ у сотрудника, т. к. у него не будет никакого относящегося к этому задания.

Поэтому практика процессного управления выработала другое решение. На схеме бизнес-процесса узлы, в которых даются задания двум исполнителям, располагаются не последовательно, а параллельно, то есть они находятся в параллельных ветках (см. Рис. 2)

2. Несколько действий подряд должны быть выполнены одновременно двумя исполнителями. — Практика работы с исполнимыми бизнес-процессами на предприятиях показывает, что роли должностных лиц (например, «Бухгалтер», или «Кассир» соответствуют «ответственным» сотрудникам, а роли «Сотрудник» или «Подавший за-

Рис. 1: «Интуитивная» реализация студентами действия, выполняемого одновременно двумя лицами.

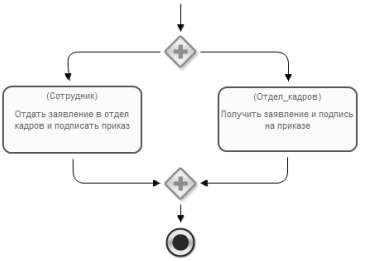


Рис. 2: Правильная реализация действия, выполняемого одновременно двумя лицами.

явку» — гораздо менее «ответственным» сотрудникам, которые могут неделями не отмечать выполнение заданий. Поэтому, в данном случае требуется так составить схему бизнес-процесса, чтобы второстепенные задания, выполняемые сотрудником, не останавливали дальнейшее выполнение бизнес-процесса. То есть, каждое такое задание должно выполняться в параллельной ветке и после него не должно происходить выполнения существенных заданий бизнес-процесса. Пример правильного построения схемы бизнес-процесса представлен на Рис. 3.

3. Схема бизнес-процесса может соответствовать алгоритму решения некоторой задачи. Рассмотрим бизнес-процесс, реализующий игру двух участников (студента и преподавателя). Игра состоит в следующем: есть кучка в 100 камней. Игроки ходят по очереди. За один ход игрок должен взять из кучки не менее одного, но не более 9 камней. Тот, кто возьмет последний камень, является выигравшим. Кроме реализации самой игры бизнес-процесс должен содержать систему поддержки принятия решения, которая на каждом ходе совето-

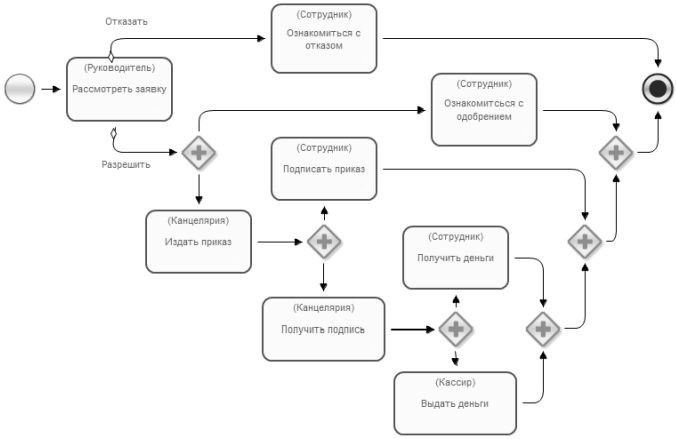


Рис. 3: Пример правильной реализация двух действий, выполняемых подряд одновременно двумя лицами.

вала бы студенту, сколько ему надо взять камешков, чтобы обыграть преподавателя. Пример схемы бизнес-процесса, реализующей игру в камешки представлен на Рис. 4.

4. Бизнес-процесс также может представлять собой решение математической задачи. Например, в качестве задания на разработку бизнес-процесса сильным студентам можно предложить следующую задачу: В поликлинике есть N врачей, каждый из которых выдает справку определенного вида. Для каждого врача есть набор справок, которые нужно получить до того, как прийти к нему на прием за справкой (при обращении эти справки не отдаются врачу, а только предъявляются). На приеме врач может либо выдать справку, либо отказать в ее выдаче. Если врач отказал в выдаче, то при повторных обращениях от тоже будет отказывать. Требуется разработать бизнес-процесс, который будет давать задания пациенту («обратиться к врачу») и врачам («принять пациента») который позволит пациенту получить максимально возможное количество справок в поликлинике.

5. В качестве учебных задач на разработку бизнес-процессов можно формулировать студентам задачи, относящиеся к различным клас-

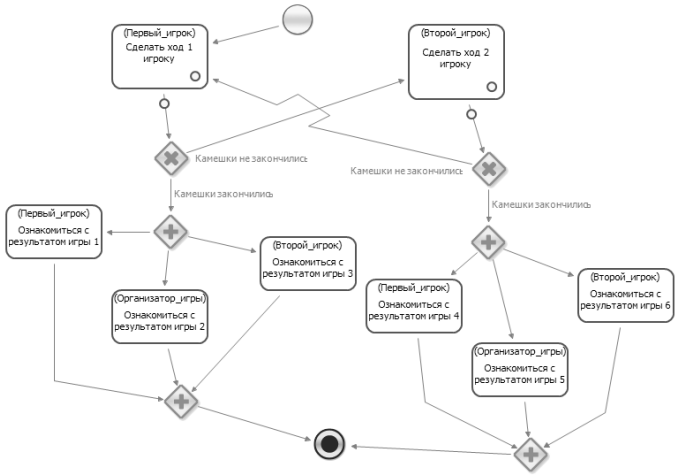


Рис. 4: Студент и преподаватель играют в камешки.

сам математических задач: логистические задачи, задачи динамического программирования, задачи, относящиеся к теории графов и т. п. При этом части схем бизнес-процессов, реализующих эти задачи, будут представлять собой блок-схемы соответствующих математических алгоритмов.

Использование свободного ПО с открытым кодом RunaWFE для обучения специалистов по процессной автоматизации

Для обучения студентов процессной автоматизации в курсах [3-5] применяется свободная система RunaWFE [6]. Использование свободного ПО для обучения обладает многими преимуществами. Использование свободного ПО позволяет легко внедрить курс обучения в учебный процесс любого российского ВУЗа: ПО бесплатно, доступно в интернете на сайте проекта RunaWFE [7], для установки системы RunaWFE не требуется каких-либо ключей или лицензионных файлов. Количество инсталляций не ограничено. Установить ПО, а также выполнять и проверять с его помощью практические работы студен-

тов можно не только в учебных классах, но и на домашних компьютерах. Разработанные бизнес-процессы можно свободно передавать в другие ВУЗы без каких-либо затрат ВУЗов на приобретение ПО. Таким образом, преподаватели могут осуществлять кооперацию при разработке учебных курсов процессного управления, а также обмениваться различными идеями, относящимися к исполнимым бизнес-процессам.

Кроме того, в случае свободного ПО, разработанные в рамках учебной или научной деятельности бизнес-процессы можно внедрить на предприятии без каких-либо расходов предприятия на покупку ПО. Также возможно участие ВУЗов в разработке и тестировании системы RunaWFE, что позволяет полнее учесть в продукте потребности данного ВУЗа.

Литература

[1] Флойд Р. *О парадигмах программирования.* В кн.: Лекции лауреатов премии Тьюринга. М: Мир, 1993

[2] Кун Т. *Структура научных революций.* М.: Прогресс, 1975

[3] Куликов Г.Г., Михеев А.Г., Орлов М.В., Габбасов Р.К., Антонов Д.В. *Изучение методологии BPMN на примере программного продукта RunaWFE. Лабораторный практикум по дисциплине «Автоматизированные информационные системы в производстве» и «Автоматизированные информационные системы в экономике».* — Уфа. УГАТУ. 2010

[4] Пятецкий В.Е, Михеев А.Г., Новичихин В.В. *Система управления бизнес-процессами: основы разработки бизнес-процессов с помощью свободного программного обеспечения: практикум* — М.: Изд. Дом МИСиС, 2013.

[5] Михеев А.Г. *Процессное управление на свободном программном обеспечении»,* — `http://www.intuit.ru/studies/courses/2358/658/info`

[6] Михеев А.Г., Орлов М.В. *Система управления бизнес-процессами и административными регламентами.* // Программные продукты и системы, № 3 2011

[7] Ссылка на сайт проекта RunaWFE: `http://wf.runa.ru/`

Владимир Лукин, проф., Владимир Хиль, асп.,
Лев Чернышов, проф.
Москва, МГППУ, МАИ, Финансовый университет при Правительстве РФ

Распределённая система автоматизированного тестирования

Аннотация

Рассматривается распределенная система хранения тестовых заданий, основанная на web-сервисах, технология её формирования и использования. Цель разработки — объединение усилий преподавателей разных вузов по подготовке и проведению тестирования студентов по дисциплинам информатики.

Год назад мы говорили о нехватке квалифицированных ИТ-специалистов в экономике, что во многом связано с их неудовлетворительной подготовкой [1]. К сожалению, сейчас ситуация только ухудшилась. Помимо прежних проблем, появились новые: перегруженность преподавателей канцелярской работой, увеличение учебной нагрузки и размеров групп. Кроме того, в силу особенности специальности требуется постоянная модификация учебных программ. И если раньше мы говорили об отсутствии времени на дополнительные занятия, сейчас впору говорить об основных, качество которых волей-неволей снижается. В таком положении даже незначительная экономия времени на рутинной работе может принести пользу.

Как и прежде, поиск возможности сэкономить время приводит к сети. Но теперь уже не столько для общения со студентами, сколько с коллегами. Учитывая, что у преподавателей сходные проблемы, можно предположить, что и решаться они будут примерно одинаковыми методами. Действительно, весьма распространённый ныне способ контроля — тестирование — используется практически всеми. Для успешного его проведения требуется набор хороших тестов и, в идеале, автоматизированная система проверки. Чтобы тесты были хоть сколько полезны, тестовые варианты должны быть тщательно подготовлены. Кроме того, вариантов должно быть много: сообразительности студентов, может быть, и не хватает, чтобы хорошо усвоить материал, но чтобы мгновенно распространить правильные варианты ответов, её достаточно. В каждом вузе преподаватели сходных дисциплин вынуждены тратить время на одну и ту же работу: составление тестов.

Возникает естественное предложение — давайте объединим усилия в рамках распределенной системы автоматизированного тестирования.

Технология работы с нею может выглядеть следующим образом. В каждом вузе преподаватели используют свой ресурс (сайт) с подсистемой тестирования. Администратор регистрирует преподавателей, выдавая логин и пароль, сопровождает ПО и оказывает общую поддержку. Преподаватель формирует тесты (задачи), списки групп студентов, проводит очное или дистанционное тестирование. Для организации коллективного использования тестов предлагается доступная извне система, управляющая набором тестов по отдельным дисциплинам, в которую участниками помещаются отлаженные и апробированные наборы тестов. Доступ к ней производится через web-сервисы [2]. Клиентская часть web-сервиса располагается на сайте вуза. Преподаватель при тестировании сможет использовать как полный набор чужих тестов, так и часть, например одну тему, а также сможет смешивать свои и чужие тесты. Кроме того, ему предоставляется возможность копировать и модифицировать тесты. Если проводится рубежный контроль, при задании его параметров указываются адреса web-сервисов, дисциплина, тема, число задач из другого ресурса.

На каждом сайте производится накопление результатов тестирования, которые служат для определения таких параметров проведенных контролей, как время, количество тестов и т.п. Для оценки качества задач полезно знать спектр полученных оценок. На сайтах могут размещаться рабочие программы дисциплин, где указаны темы, их содержание и указания по проведению рубежных контролей с использованием системы тестирования, а также форумы для замечаний и обмена мнениями.

Виды тестов, которые возможно рассматривать в рамках предлагаемой системы, следующие:

- тесты с вариантами ответов (обработка автоматическая);
- тесты с открытыми вопросами (обработка полуавтоматическая [3], контроль преподавателя);
- задачи, ответы на которые проверяются преподавателем;
- генерируемые тесты.

Генерируемые тесты выглядят перспективней, так как они не повторяются и позволяют проводить тренинг, в ходе которого студент заходит на сайт, получает задания, решает и проверяет правильность

решения самостоятельно [4]. Правда, здесь требуется разработка подсистем для каждой дисциплины и даже отдельных тем. Возможно создание фрейворков (мастеров) с типовой архитектурой для создания таких систем.

Поскольку клиентская часть подсистем, выполняемая в браузерах, написана на JavaScript, она открыта и доступна (СПО). Кроме того, предполагается сделать доступными и серверные компоненты.

В настоящее время есть определённый опыт объединения усилий преподавателей МАИ, МГППУ, ФУ, РГУИТП в проведении тестирования по дисциплинам «Операционные системы», «ТВПС», «Специальные разделы программирования», «Функциональное программирование», «Базы данных», «Web-программирование», «Методология и технология проектирования программных систем».

Литература

[1] *Лукин В.Н., Чернышов Л.Н.* О подготовке специалистов в области ПО. VIII конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переславль. — М.: Альт Линукс, 2013.

[2] *Фомин С.С., Чернышов Л.Н.* Web-сервисы в системах дистанционного обучения. Материалы XVI Международной конференции ВМСППС. – М.: Изд-во МАИ-ПРИНТ, 2009.

[3] *Хиль В.А., Чернышов Л.Н.* Обработка открытых ответов в системах тестирования с помощью языка SPARD. II Международная научно-практическая конференция «ИТО-Москва-2013», 2013.

[4] *Чернышов Л.Н.* Программа-тренажер по теории формальных языков и конечных автоматов. Материалы XVIII Международной конференции ВМСППС. — М.: Вузовская книга, 2013.

Денис Силаков
Москва, РОСА, НИУ ВШЭ

Проект: ROSA Linux `http://www.rosalab.ru/`, `http://se.hse.ru/`

РОСА и НИУ ВШЭ — опыт сотрудничества на ниве обучения студентов

Аннотация

В докладе освещается опыт сотрудничества ЗАО «РОСА» и НИУ ВШЭ в области получения студентами навыков участия в реальных промышленных проектах. Рассказывается о темах работ, выполняемых студентами, и уроках, вынесенных обеими сторонами за два года совместной работы.

Одной из проблем современных ВУЗов является «оторванность» их выпускников от реальной жизни. В процессе обучения многие студенты ИТ-специальностей выполняют только задания, предлагаемые преподавателями, и получают лишь поверхностное представление о промышленном процессе разработки.

Для решения этой проблемы, многие ВУЗы идут на сотрудничество с промышленными компаниями, занимающимися разработкой ПО. В случае РОСЫ и ВШЭ такое сотрудничество ведется в трех направлениях:

- студенты отделения программной инженерии ВШЭ проходят в РОСЕ двухнедельную практику, в рамках которой они получают представление о том, как работает компания, а также успевают поработать над небольшими, но реально востребованными задачами;
- сотрудники РОСЫ предлагают темы для курсовых работ, основанные на потребностях компании;
- в рамках дисциплины «Программный проект» команды студентов работают над задачами, предложенными сотрудниками РОСЫ. Данная дисциплина подразумевает изучение полного цикла разработки программного продукта (от сбора требований до введения в эксплуатацию); длится курс около полугода.

Все работы, выполняемые студентами, имеют отношение к тем или иным составляющим дистрибутивов РОСЫ. В рамках практики студенты могут выбирать между двумя направлениями:

- сборка или обновление нескольких программ в дистрибутиве — для этого необходимо ознакомится с самой парадигмой управления ПО в Linux и научиться собирать пакеты;
- разработать автоматизированные тесты для одного из приложений (с использованием инструментов тестирования, используемых в РОСЕ).

В рамках курсовых работ и «Программного проекта» даются задачи посложнее — например, разработка с нуля некоторой программы или инструмента для РОСЫ.

Отметим, что РОСА — далеко не единственная компания, сотрудничающая с ВШЭ. В студентах заинтересованы многие представители рынка и ребятам предлагается выбор из нескольких компаний совершенно разной направленности. Какие преимущества и недостатки есть в таком конкурсе у компании, разрабатывающей СПО?

Основным недостатком следует признать слабое знакомство многих учащихся с Linux. Для прохождения практики или выполнения проекта им придется изучать не только ТЗ, но и новую для них систему, инструменты разработки и так далее.

Перекрыть этот недостаток можно несколькими способами. Во-первых технологически — не все программы пишутся на C/C++, есть и кроссплатформенные инструментарии и библиотеки. Даже студенты, никогда не видевшие Linux, могут оказаться знакомы с Python, Qt, PHP (и другими ориентированными на Web средствами), на худой конец — с Java.

Во-вторых, в РОСЕ мы используем чисто организационные преимущества, которые дает нам природа СПО. Так, мы не требуем, чтобы студенты приходили к нам в офис для выполнения заданий или даже для демонстрации результатов — вся разработка открытая, ведется в нашей системе разработки и сборки пакетов ABF, и результаты можно сразу собирать в репозитории и устанавливать в дистрибутив. Студенты при этом вольны работать над заданиями в любое удобное для них время. А поскольку разработчики РОСЫ есть и в Москве, и во Владивостоке, и в ряде городов между ними, то в любое время суток можно получать оперативные ответы на возникающие вопросы.

Наконец, нелишне напомнить студентам, что результаты их работ будут общедоступны и они смогут ссылаться на них в своих резюме — так что потенциальные работодатели не просто увидят галочку о

прохождении практики в некоторой компании, но смогут изучить и оценить созданный код.

Наш опыт показывает, что указанные преимущества с лихвой перевешивают опасения студентов, связанные с незнакомой им ОС, и недостатка в желающих поучаствовать в проектах РОСЫ мы не испытываем. В результате довольными остаются все стороны — студенты получают реальный опыт работы в команде, ВШЭ повышает квалификацию своих выпускников, а РОСА получает бесплатных помощников. (Конечно, не совсем бесплатных — ведь на общение со студентами уходит время основных разработчиков, но по нашим оценкам, положительная отдача все-таки есть).

Поэтому хотелось бы рекомендовать СПО-разработчикам учесть наш опыт и выходить на контакты в ВУЗами. В конце концов, студенты по крайней мере получат представление о процессе создания СПО. И как знать — возможно им это понравится, и даже если они и не свяжут свою профессиональную деятельность с СПО, то на досуге будут помогать какому-нибудь открытому проекту.

Денис Пынькин, Юрий Адамов
Минск, EPAM Systems
https://github.com/epam-llpd/linux_courses

Linux-образование — симбиоз ВУЗов, коммерческих компаний и LUG

Аннотация

В докладе описываются инициативы IT-компаний, связанные с обучением процессу разработки в среде ОС Linux. Приводятся примеры успешного взаимодействия белорусских коммерческих компаний с ВУЗами и Минским LUG для достижения общих целей. В результате общих усилий создана и развивается открытая и свободная библиотека лекционных материалов по обучению различным аспектам работы в ОС Linux.

Кризис в отрасли

ОС Linux устойчиво набирает популярность в качестве платформы для разработки. Соответственно растет и количество коммерческих

проектов с использованием этой операционной системы. Примерно в 2010-12 годах в стало заметно, что людей, знакомых с полным циклом разработки для Linux-платформы, в особенности для встраиваемых и серверных ее применений, не так уж много.

Хотя более правильным было бы сказать, что таких людей катастрофически не хватает для покрытия нужд белорусских IT компаний.

Основные проблемы обучения в ВУЗах РБ:

– существует направленность на изучение закрытого стека технологий на основе ОС Windows, при этом применение Linux зависит исключительно от отдельных лиц, работающих в учебном заведении[1];

– еще одна проблема прямо заложена в учебных планах, направленных на изучение теории, что приводит к самостоятельному освоению инструментальных средств программирования учащимися, которое, как правило, заканчивается на минимальном уровне владения выбранным (или навязанным) IDE;

– «классическое» обучение практически не затрагивает практики командной разработки с выделением в отдельные процессы собственно самой разработки, тестирования и развертывания ПО;

– преподавателями, в основной своей массе, совершенно игнорируются подходы, принятые в мире, связанном со Свободным ПО[2].

Кроме того отдельно хотелось бы отметить проблему закрытости и кастовости, распространенной в сообществе пользователей и разработчиков Linux, что никак не способствует привлечению новых членов.

Все эти проблемы приводят к тому, что разработчики, тестировщики и, в меньшей степени, администраторы, умеющие работать и знающие ОС Linux «самозарождаются», что является сравнительно медленным процессом и совершенно не подходит для коммерческих компаний.

EPAM, начало

Принципиальное решение о необходимости дополнительного привлечения молодых разработчиков к ОС Linux было принято в рамках департамента «Low Level Programming Department» в начале 2012 го-

да. Учитывая большой опыт компании по работе с учебными заведениями в РБ и острую необходимость в увеличении количества разработчиков для встраиваемых и серверных применений, уже осенью 2012 г., была оборудована совместная лаборатория Epam и БГУИР на базе кафедры ЭВМ КСиС.

Тогда же прошел первый набор (2012-2013) слушателей на курсы по изучению ОС Linux для разработчиков.

Учитывая, что основная целевая аудитория курса — студенты технических специальностей, была разработана программа, рассчитанная на людей, уже умеющих программировать на каком-либо языке, но желающих получить навыки разработки в среде ОС Linux. В программу курса вошли технологии и знания, помогающие адаптироваться к целевой платформе обучения.

В связи с ориентацией департамента на разработку серверных и встраиваемых решений, было принято решение не затрагивать работу и разработку в графическом окружении, а сконцентрироваться на работе в командной строке. Сама программа разделена на 3 отдельных модуля, общей длительностью 66 часов:

— введение в GNU/Linux — минимальный набор знаний об архитектуре и особенностях работы в среде ОС Linux;

— программирование на bash — разработчики рано или поздно сталкиваются с необходимостью разбираться в чужих скриптах, создавать свои, а также автоматизировать свою работу;

— инструментарий разработчика — в этот модуль входят принципы разработки в ОС Linux, методы и навыки работы с классическими инструментами: компилятор, управление сборкой, установкой и распространением приложения, совместная работа с исходным кодом, анализ исполняемого файла и его работы, а также другие средства и подходы применяющиеся при разработке.

Хотелось бы подчеркнуть, что курс читается не профессиональными лекторами, а разработчиками департамента, постоянно применяющими многие из изучаемых технологий на практике.

После успешного окончания курса слушателям были вручены сертификаты о прохождении, а лучшим учащимся предложена работа в Epam LLPD в качестве разработчиков серверных и встраиваемых решений.

EPAM, вторая итерация

В 2013 году курсы были расширены, в результате чего появились еще 2 разработчика, готовых поделиться своими знаниями по следующим направлениям:

— программирование на языке Python;

— программирование на языке C.

Разумеется, также происходят эволюционные изменения и в модулях, созданных в предыдущем году. Сама же библиотека слайдов, наработанная при подготовке первого года, пригодилась для быстрого создания отдельного внутреннего курса Epam по изучению Bash.

Совместная работа

На данный момент существует общая стратегическая задача, Linux сообщества, ВУЗов и корпоративного сектора — это создание широкой и устойчивой экосистемы, связанной с ОС Linux в среде IT-специалистов и желающих стать таковыми.

В процессе роста сообщества, увеличивается не просто количество технических специалистов, кроме того, увеличивается количество людей знающих и разделяющих ценности, заложенные в принципах Свободного ПО. Интерес компаний очевиден, ведь именно из этого сообщества приходят так необходимые для коммерческой разработки профессионалы.

Учитывая специфику области, в профильных департаментах различных компаний, независимо друг от друга было принято решение вести максимально возможно открытую политику обучения.

В первую очередь это привело в неформальной договоренности о создании совместных материалов с открытым доступом для обучения.

Таким образом на github появился открытый проект, содержащий на данном этапе лекционные слайды, помогающие быстро создать презентацию по необходимой теме. Материалы, согласно подходу, принятому при работе с открытыми и свободными проектами, добавляются по мере создания: `https://github.com/epam-llpd/linux_courses`, а по адресу `https://github.com/epam-llpd/linux_courses/network/members/` можно увидеть список лиц и организаций, так или иначе добавивших свой вклад в развитие.

Хотелось бы отметить, что благодаря такому подходу у любого заинтересованного лица — будь то представитель другой компании, либо студент, есть возможность участвовать в процессе обучения, корректировать материалы и создавать новые, а также проводить свои собственные курсы, используя уже созданные материалы.

На данный момент инициативу поддержали и открыли материалы по своим учебным материалам следующие компании:

— EPAM Systems — курс по работе в ОС Linux для разработчиков, курс по Bash;

— SaM Solutions — курс по работе в ОС Linux для тестировщиков;

— Promwad — курс «Программирование встраиваемых систем» на базе ОС Linux.

В своем роде — это уникальный для просторов РБ проект с открытым исходным кодом[3].

Литература

[1] Derechennik S.S., Kostiuk D.A., Pynkin D.A. *PFree/libre software usage in the belarusian system of higher educational institutions* // Друга міжнародна науково-практична конференція FOSS Lviv-2012: Збірник наукових праць/ Львів, 26-28 квітня 2012 р.

[2] Д.А. Пынькин, И.И. Глецевич. *Открытый подход к обучению студентов технической специальности ВУЗа* // 7-я конференция «СПО в высшей школе»: Тезисы докладов. `http://freeschool.altlinux.ru/wp-content/uploads/2012/01/pereslavl-winter-2012.pdf`

[3] Д.А. Пынькин, В.В. Шахов. *Обучение Linux в корпоративном секторе.* Зимняя международная конференция LVEE'2013. Тезисы докладов. `http://lvee.org/en/abstracts/57`

Максим Зубов, Алексей Пустыгин, Евгений Старцев
Челябинск, Челябинский Государственный Университет

Построение универсального представления графа потока управления для статического анализа исходного кода

Аннотация

Анализ потока управления — важный этап статического анализа. Граф потока управления может быть описан в форме универсального высокоуровневого промежуточного представления. Получение такого представления реализовано для языков Java и Python. Для повышения эффективности анализа потока управления предложено связать его с универсальным классовым представлением. Реализована методика уменьшения количества информации в эквивалентном представлении по какому-либо критерию (слайсинг).

В разработке программного обеспечения важную роль играют программные инструменты, позволяющие автоматически выполнять за программиста различную рутинную работу. К таким средствам можно отнести статический анализ исходного текста. Ранее было предложено использовать универсальные промежуточные представления для статического анализа как эффективное решение для двух и более языков [1]. Для анализа исходного кода на объектно-ориентированных языках было предложено универсальное классовое представление, позволяющее эффективно анализировать структуру классов проекта [2]. Интерес представляет анализ потока управления программы, который можно выполнять для всех языков программирования, поддерживающих процедурную парадигму. Основой для такого анализа является граф потока управления [3].

Формат графа потока управления вполне может быть универсальным для нескольких языков; в нем содержатся только блоки линейного текста и ветвления, вызванные циклами или условными переходами. В разных языках имеются синтаксические отличия в части условных операторов, циклов и обработки исключений. Таким образом, одной структурой данных можно описать поток управления для нескольких языков, например Java и Python, для которых уже реализовано получение универсального представления уровня классов.

Представление графа потока управления охватывает только отдельный участок выполнения программы — функцию или метод класса. Сам по себе граф потока управления является ориентированным графом общего вида, допускающим циклы. Предложено хранить дуги графа в виде отдельных элементов, указывая, от какого блока к какому идет дуга, а условные операторы считать отдельными вершинами, в отличие от моделей, где операции перехода отображаются только в виде ребер (дуг) графа [4]. Такой формат позволит хранить и анализировать выражения для условия, по которым выбирается направление выполнения программы.

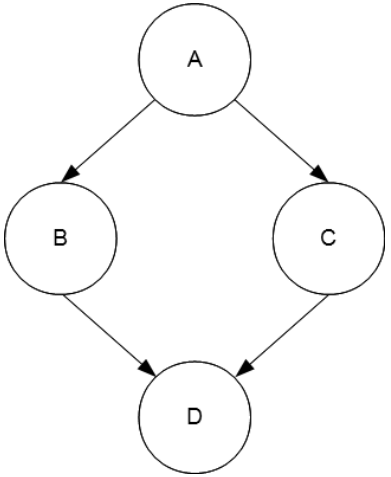


Рис. 1: Пример графа потока управления

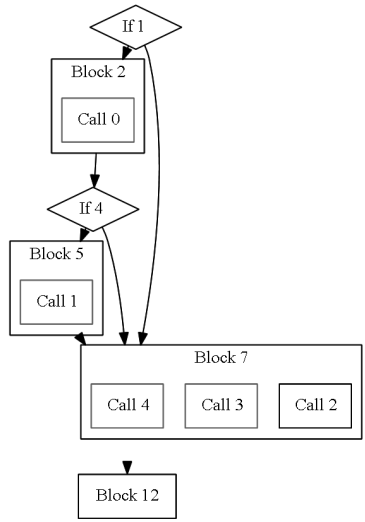


Рис. 2: Пример результата работы визуализатора графа потока управления

Текст промежуточного представления графа потока управления хранится в формате XML, для чего создана схема [7]. На рисунке 1 показан простой случай графа потока управления с 2-мя параллельными ветвями, однако даже в такой простой форме он не является деревом, в листинге 1 показан текст на XML, полученный для него. Как видно, для условной вершины выделен отдельный узел — A_if,

и ветви альтернатив выходят из него. Данный формат базируется на формате GraphML [5] — открытом формате представления графов в виде текста XML.

Листинг 1 — Текст XML-представления графа потока управления

```
<Method>
  <Block id=''A''/>
  <Flow from_id=''A'' to_id=''A_if''/>
  <If id=''A_if''/>
  <Flow from_id=''A_if'' to_id=''B''/>
  <Block id=''B''/>
  <Flow from_id=''A_if'' to_id=''C''/>
  <Block id=''C''/>
  <Flow from_id=''B'' to_id=''D''/>
  <Flow from_id=''C'' to_id=''D''/>
  <Block type=''Exit'' id=''D''/>
</Method>
```

Получение универсального представления графа потока управления было реализовано для языков Java и Python на основе разработанного ранее генератора абстрактного синтаксического дерева (AST) [6], который также используется для генерации UCR. Следует отметить, что в данном случае само AST не является промежуточным представлением, так как над ним не выполняется никакого анализа. Оно является просто набором данных,получаемым при разборе исходного текста.

Совместное использование двух универсальных представлений: графа потока управления (далее CFG) и диаграммы классов (далее UCR) открывает новые возможности анализа. Для объектно-ориентированных языков каждый отдельный участок потока управления — метод класса, поэтому методу сопоставляется атрибут класса, к которому метод принадлежит. В языке Python разрешены функции, не принадлежащие какому-либо классу; указания на класс функции не будет. Однако это не вызывает ошибок анализа, так как такие функции не связаны с диаграммой классов.

На рисунке схематично изображена работа всей программной системы, в том числе с анализаторами, основанных на совместном использовании представлений.

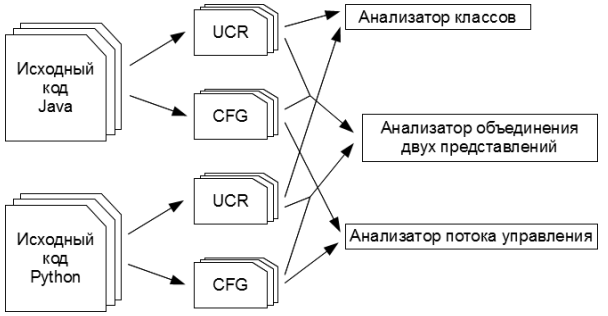


Рис. 3: Схематичное изображение работы анализаторов универсальных промежуточных представлений

Визуализация потока управления производится с помощью утилиты dot из пакета graphviz [8]. Отметим, что внутри блоков графа выделяются вызовы функций/методов (именуемые с использованием префикса «Call»). Вызовы функций/методов безусловно представляют основной интерес, так как обеспечивают связи по управлению между отдельными функциями /методами (связность графа). Пример результата визуализации приведен на рисунке 2. На нем представлен метод set_color_formatter() класса logilab.common.logging_ext.ColorFormatter из пакета Logilab Common [9] (листинг 2).

Также интерес представляет слайсинг-преобразование — метод уменьшения количества данных в представлении по какому-либо критерию [10]. Наиболее интересен совместный слайсинг двух представлений. Особый интерес представляет слайсинг одного из представлений по критерию, относящемуся к другому. Так, например, для CFG можно выделять те методы классов, в которых создается указанный класс (вызывается его метод-конструктор). Для UCR можно получать срезы тех классов, которые создаются в указанном блоке потока управления, или получать срез тех классов, в методах которых создается указанный класс.

Листинг 2 — Исходный текст метода *set_color_formatter*() класса *logilab.common.logging_ext.ColorFormatter* из пакета Logilab Common

```
def set_color_formatter(logger=None, **kw):
```

```
if logger is None:
    logger = logging.getLogger()
    if not logger.handlers:
        logging.basicConfig()
format_msg = logger.handlers[0].formatter._fmt
fmt = ColorFormatter(format_msg, **kw)
fmt.colorfilters.append(xxx_cyan)
logger.handlers[0].setFormatter(fmt)
```

Поскольку формат слайсинг-преобразованного графа не отличается от формата исходного представления графа(до преобразования), то полученные срезы можно снова анализировать теми же инструментами анализа, которые использовались для представления предыдущего уровня, уточняя срезы по дополнительным критериям. Кроме того, постоянство формата при преобразованиях графа обеспечивает и единообразную визуализацию результатов слайсинга на любом этапе.

Литература

[1] Зубов М.В., Пустыгин А.Н., Старцев Е.В. *Подходы к статическому анализу открытого исходного кода* — Брест: Альтернатива, 2012. — 158с. — С. 36–40.

[2] Зубов М.В., Пустыгин А.Н., Старцев Е.В. *Применение универсальных промежуточных представлений для статического анализа исходного программного кода.* — Томск: Издательство ТУСУР, 2013. — 142 с. — С 64-69.

[3] Граф потока управления — Википедия. [Электронный документ] URL: `http://ru.wikipedia.org/wiki/`Граф_потока_управления (дата обращения 13.01.2014)

[4] Control Flow — GNU Compiler Collection (GCC) Internals. [Электронный документ] `http://gcc.gnu.org/onlinedocs/gccint/Control-Flow.html` (дата обращения 13.01.2014)

[5] The GraphML File Format. [Электронный документ] URL: `http://graphml.graphdrawing.org/` (дата обращения 13.01.2014)

[6] А. Ахо, М. Лам, Р. Сети, Д. Ульман. *Компиляторы: принципы, технологии и инструментарий* — М.: Вильямс, 2010. — 1184 с.

[7] XML-схема универсального представления графа потока управления. [Электронный документ] URL: `https://github.com/exbluesbreaker/csu-code-analysis/blob/master/data/cfg.xsd` (дата обращения 13.01.2014)

[8] Graphviz — Graph Vizualization software. [Электронный документ] URL: `http://www.graphviz.org/` (дата обращения 13.01.2014)

[9] Logilab-Common. [Электронный документ] URL: `http://www.logilab.org/project/logilab-common` (дата обращения 13.01.2014)

[10] Program slicing — Википедия. [Электронный документ] URL: `http://en.wikipedia.org/wiki/Program_slicing` (дата обращения 13.01.2014)

Алексей Пустыгин, Николай Ошнуров, Александр Ковалевский
Челябинск, Челябинский Государственный Университет

Проект технологии извлечения знаний из исходных текстов на языках C++ и C# с использованием общего промежуточного представления

Использование высокоуровневого промежуточного представления часто может оказаться полезным в процессе автоматического извлечения знаний из исходных текстов программ. Особенно это заметно, когда необходимо анализировать исходные тексты на нескольких языках программирования, поскольку формат такого представления может быть расширен для описания нескольких языков. Это упрощает процесс создания утилит для автоматического извлечения знаний из исходных текстов.

Извлечение знаний из исходных кодов программ является задачей обработки текстов. Для её решения нужен набор программных инструментов для каждого типа анализа или получения эквивалентных представлений. Поскольку в каждом случае осуществляется разбор исходного текста, данную функцию целесообразно выделить в самостоятельный функционал — генератор промежуточного представления.

В случае анализа гетерогенных исходных текстов, то есть текстов на нескольких языках программирования, задача становится еще

сложнее. Простейшим решением данной проблемы является создание генераторов промежуточного представления, специфичных для каждого языка, однако, данный подход не является эффективным. Это связано с тем, что создание таких инструментов — сама по себе трудоёмкая задача.

В предыдущих работах [2] предлагалось решение этой проблемы путём введения единого универсального промежуточного представления исходных текстов на различных ЯП.

Для извлечения информации из исходных текстов на ЯП C, C++, C# были поставлены задачи:

1. Предложить формат для построения промежуточного представления гетерогенных текстов на C, C++ и C#,
2. Спроектировать инструменты для получения такого представления.

Формат промежуточного представления должен отвечать некоторому набору критериев, среди них можно выделить:

1. Простота реализации инструментов для автоматической обработки,
2. Расширяемость — простота добавления новых языковых конструкций,
3. Читабельность — удобство восприятния человеком.

Среди распространённых форматов представления структурированных данных этому набору критериев удовлетворяют XML [11], JSON [13] и YAML [21].

Стек технологий XML предоставляет инструменты, которые можно использовать для решения задачи извлечения знаний из исходных текстов программ путём обработки их промежуточного представления. Из них можно выделить следующие:

1. XPath [18] — язык запросов для выборки и навигации по узлам XML документа, вычисления некоторых его метрик, что позволяет эффективно его использовать для задачи извлечения знаний. Является стандартом консорциума W3C.
2. XQuery [19] — функциональный язык запросов, разработанный для выполнения запросов и трансформации коллекций структурированных и неструктурированных данных. Является официальной рекомендацией W3C.

3. XSLT [20] — язык для трансформации XML документов в другие XML документы, объекты HTML или XSL, которые затем могут быть преобразованы в PDF, PostScript, SVG или PNG. Т.е. позволяет получить, например, визуальное эквивалентное представление, пригодное для анализа. Является официальной рекомендацией W3C.
4. XML-базы данных, например, Sedna [4] (лицензия `http://www.apache.org/licenses/LICENSE-2.0.html` Apache License 2.0) или BaseX (лицензия `http://en.wikipedia.org/wiki/BSD_license` BSD) [6], `http://basex.org/` которые можно использовать в качестве единого хранилища знаний об исходного коде одного или нескольких проектов.

Таким образом, выбор формата XML как основы для промежуточного представления позволяет сократить затраты на создание инструментов анализа и построения эквивалентных представлений.

JSON и YAML, в свою очередь не обеспечивают возможностей XML, в силу отсутствия для них подобного инструментария.

Для получения промежуточного представления исходного кода предпола- гается использовать абстрактное синтаксическое дерево (AST) [1], дополненное семантической информацией. Среди существующих форматов представления в контексте задачи извлечения знаний можно выделить следующие:

— SrcML (Source Markup Language) [3] — открытое документоориентированное представление исходных текстов, поддерживающее языки C, C++ (CppML), Java (JavaML), частично C# (SrcML.Net). Основным его недостатком является отсутствие важных атрибутов в узлах AST, что не позволяет проводить полный анализ кода.

— Специфические для языков представления, такие как GCC-XML [12] (основанное на внутреннем представлении компилятора GCC), cpp2xml [16], не обладающие свойством универсальности.

Для поиска возможных альтернативных решений был произведен поиск инструментов для получения AST. В таблице 1 приведен их обзор.

Таблица 1: Обзор открытых инструментов для получения AST

Инструмент	**Язык**	**Возможности**	**Недостатки**
ROSE Compiler [15]	C++	`http://clang.llvm.org/` Поддержка C, C++11	Поддержка только C++
Eclipse CDT [10]	C++	IDE для статического анализа. Поддержка C++	Отсутствие поддержки C++11.
Mozilla Dehydra & Treehydra (GCC GIMPLE) [9]	C++	Извлечение информации о некоторых узлах AST	Разработка приостановлена в 2010, неполное и неэквивалентное исходному тексту AST
Mono C# Compiler [7]	C#	Кроссплатформенность решения (Windows, GNU/Linux, MacOS, iOS, Android)	Несовместимость API разных версий компилятора.

Главным требованием к формату промежуточного представления было включение в состав атрибутов узлов такого набора тегов, который позволяет удовлетворить требованиям спецификации языков при разборе всех синтаксических конструкций, описанных в стандартах.

Источниками для создания формата промежуточного представления послужили следующие документы:

1. Спецификация языка C# [8]
2. Проект стандарта C++ [17].

Для генерации промежуточного представления по C++ и C# по спроектированному формату были выбраны следующие инструменты, удовлетворяющие заданным критериям (полнота, актуальность инструмента, предполагаемые затраты на разработку промежуточного представления):

— Clang и его библиотека libclang (для языков C/C++), которые обеспечивают поддержку C++11, Objective-C, поддерживаются активным сообществом, крупными компанями (например Apple, Google), включают SDK и отдельную библиотеку для получения AST. Библиотека libclang — неотъемлемая часть инструментария clang, которая ппрпредназначена для статического анализа и автозавершения кода и. используется в IDE Xcode от Apple.

— ICSharpCode.NRefactory [14] (для языка C#) — библиотека, основанная на исходном коде компилятора Mono, предоставляющая инструментарий для преобразования исходного кода и поддерживает последнюю версию спецификации языка C# 5.0.

В результате проделанной работы был разработан формат универсального промежуточного представления для языков C, C++ и C# [5], а также составлен эскизный проект технологии извлечения знаний из исходных текстов, использующий данное представление, которое соответствует необходимому набору критериев и может быть расширено для поддержки новых синтаксических конструкций, в том числе других языков.

Литература

[1] А. Ахо, М. Лам, Р. Сети, Д. Ульман. *Компиляторы: принципы, технологии и инструментарий.* — М.: Вильямс, 2010. — 1184 с.

[2] Зубов М.В. *Статический анализ ПО с помощью его промежуточных представлений и технологий с открытым исходным кодом* / М.В. Зубов, А.Н. Пустыгин, Е.В. Старцев // Материалы 2-й Междунар. конф. «FOSS. Lviv–2012», Львов. — Львів: Сорока, 2012. — С. 165–168.

[3] Collard M.L., Kagdi H.H., Maletic, J.I. *An XML-Based Lightweight C++ Fact Extractor* // Program Comprehension, 2003. 11th IEEE International Workshop. May 10 — 11, 2003. P. 134-143.

[4] Fomichev A., Grinev M., Kuznetsov S. Sedna: *A Native XML DBMS* // SOFSEM 2006: Theory and Practice of Computer Science. Lecture Notes in Computer Science. 2006. Vol. 3831. P 272-281.

[5] Формат универсального промежуточного представления. Электронный ресурс: URL: `https://raw2.github.com/ifanatic/CodeAnalysis/master/doc/IntermidiateRepresentation.txt` (дата обращения 13.01.2014).

[6] BaseX. The XML Database. Электронный ресурс: URL: `http://www.basex.org/` (дата обращения 13.01.2014).

[7] Crossplatform, open source .NET development framework. Электронный ресурс: URL: `http://www.mono-project.com/` (дата обращения 13.01.2014).

[8] C# Language Specification 5.0. Электронный ресурс: URL: `http://www.microsoft.com/en-us/download/details.aspx?id=7029` (дата обращения 13.01.2014).

[9] Dehydra. Электронный ресурс: URL: `https://developer.mozilla.org/en-US/docs/Dehydra` (дата обращения 13.01.2014).

[10] Eclipse CDT (C/C++ Development Tooling). Электронный ресурс: URL: `http://www.eclipse.org/cdt/` (дата обращения 13.01.2014).

[11] Extensible Markup Language (XML). Электронный ресурс: URL: `http://www.w3.org/XML/` (дата обращения 13.01.2014).

[12] GCC-XML. Электронный ресурс: URL: `https://github.com/gccxml/gccxml` (дата обращения 13.01.2014).

[13] JSON (JavaScript Object Notation). Электронный ресурс: URL: `http://www.json.org/` (дата обращения 13.01.2014).

[14] NRefactory library. Электронный ресурс:URL: `https://www.github.com/icsharpcode/NRefactory` (дата обращения 13.01.2014).

[15] ROSE compiler infrastructure. Электронный ресурс: `http://rosecompiler.org/` (дата обращения 13.01.2014).

[16] The C++2XML page. Электронный ресурс: URL: `http://scl.csd.uwo.ca/Projects/cpp2xml/index.html` (дата обращения 13.01.2014).

[17] Working Draft, Standard for Programming Language C++. Электронный ресурс:URL: `https://www.github.com/google/cxx-std-draft/blob/master/papers/n3337.pdf` (дата обращения 13.01.2014).

[18] XML Path Language (XPath). W3C Recommendation, 16 November 1999. Электронный ресурс: URL: `http://www.w3.org/TR/xpath/` (дата обращения 13.01.2014).

[19] XQuery 1.0: An XML Query Language (Second Edition). W3C Recommendation, 14 December 2010. Электронный ресурс: URL: `http://www.w3.org/TR/xquery/` (дата обращения 13.01.2014).

[20] XSL Transformations (XSLT). W3C Recommendation, 16 November 1999. Электронный ресурс: URL: `http://www.w3.org/TR/xslt` (дата обращения 13.01.2014).

[21] YAML: YAML Ain't Markup Language. Электронный ресурс: URL: `http://www.yaml.org/` (дата обращения 13.01.2014).

Александр Боковой
Эспоо, Финляндия, Red Hat

Проект: FreeIPA `http://freeipa.org`

Управление инфраструктурой предприятия с FreeIPA

Аннотация

Доклад посвящен проекту FreeIPA, задачей которого является создание средств управления инфраструктурой предприятия на основе свободного программного обеспечения. FreeIPA объединяет в единую систему LDAP сервер 389-ds, MIT Kerberos, Samba, BIND и ряд других компонент, дополняя их системой управления и настройки, которая позволяет разворачивать инфраструктуру предприятия за несколько десятков минут в автоматическом режиме. FreeIPA позволяет интегрировать различные POSIX-совместимые системы, но наибольший эффект достигается при использовании Linux и клиентских компонент FreeIPA, разрабатываемых в рамках проекта SSSD, System Security Services Daemon. При использовании SSSD становится возможной централизация принятия решения по ограничению доступа к различным ресурсам и службам, работающим на машинах под управлением SSSD, централизованному назначению контекстов SELinux пользователям и сквозная авторизация средствами Kerberos. FreeIPA также позволяет интегрировать существующие системы на основе Active Directory, решая проблему эффективного управления разнородными средами.

Станислав Фомин
Москва, ЗАО РОСА, ИСПРАН

Магия пера или эффективная свобода преподавания со стилусом

Лекции и семинары, конференции и бизнес-тренинги — называться это может по-разному, но суть в любом случае одна: автор-лектор-тренер, пытается передать аудитории смысл голосом, жестами и, самом важное, что-то показывая — формулы ли это на доске, ватман с картинками в руках, или стеклянные колбы с живыми опытами.

Ранее, кроме живых демонстраций, вариантов визуализации почти не было — только унылая доска в меловых разводах. Затем с Запада

пришли флипчарты и «смерть от PowerPoint-а», и в целом, сейчас актуален спор между любителями слайдов и «старой школой доски».

Плюсы слайдов и других заранее заготовленных материалов, не важно, скучные ли это рожденные из документов слайдоменты, веселые картинки или «адов матан» — это большая читаемость текста, формул, диаграмм и визуальная эффективность, даже если там будут только «картинки с котиками». Минусы — линейность и детерминированность, очень трудно отойти от накатанного сценария с заготовленным набором слайдов. Это еще более-менее терпимо в коротком докладе на конференции, но неудобно для семинара-тренинга.

Этого недостатка лишены мастера флипчартов и модных маркерных досок — они могут быстро импровизировать, набросать поясняющую диаграмму или простую иллюстрацию, написать нужную формулу и анимировать зависимости и переходы... с другой стороны — качество этих рисунков, текста и формул — ужасное, и если писать только на доске, без заготовок, то это чистый проигрыш даже простым «слайдоментам».

Что делать? Ну, теоретически, можно совмещать — показывать и заготовленные слайды, и при этом импровизировать и дорисовывать пропущенное на доске. Тут возникает куча проблем с переключением, как внимания аудитории, так и инструментов лектора, не говоря уже о необходимости дополнительной «недвижимости» досок. Ведь проектор уже стал обязательным оборудованием для лекториев, портативный проектор и ноут легко взять с собой на какой-нибудь выездной тренинг, где, проецируя на стену можно обойтись без специального экрана. А если у слушателей есть лептопы или планшеты, то можно просто транслировать им экран.

Так что же делать? Тут на помощь приходят стилусные ноутбуки, т.е. лептопы, в которых можно рисовать пером напрямую по экрану. Сейчас мощный тренд тотальной сенсоризации дисплеев, но даже если ваш ноутбук не такой, то можно специально для выступлений купить простой pentablet с ebay — например, б.у. HP 2730p, которые когда-то стоили $4000, сейчас можно найти за $50. Это позволительно даже небогатому преподавателю, а на худой конец можно взять простой планшет за $10. Ну, а если бюджет позволяет, можно поставить в класс тачскрин-моноблок типа HP TouchSmart 610.

Темой нашего доклада будет эффективный open-source софт, который позволит достичь синергии между заготовленными материалами и импровизацией на месте, позволит вам, если и не превратится в

звезду TED, то по крайней мере, эффективно объяснять то, что вы знаете.

Сначала мы посмотрим живьем класс программ-журналов, заменяющих флипчарты — journal, jarnal, xournal и др. У них есть свои плюсы и минусы, местами даже уникальные возможности. В любом случае, освоив хотя бы одну из них, вы уже не будете нуждаться в досках и флипчартах.

Но кроме досок и слайдов, самой лучшей визуализацией ведь были «живые опыты», а в современных условиях на лептопах это будут живые демонстрации работы с софтом и сервисами, показ видеофрагментов, управление внешними устройствами, — и возникает очевидный вопрос: нельзя ли совместить «магию пера» с «живой демонстрацией»?

В мире Windows давно была маленькая суперпрограмма ZoomIt, написанная гениальным Марком Руссиновичем, с минималистичным оптимизированным интерфейсом. С ней можно было просто и удобно в любой момент рисовать поверх всего на экране разными цветами и увеличивать любой его фрагмент.

Но ZoomIT, хоть и бесплатен, но закрыт и работает только для Windows, а в мире open source и Linux с этим было сложно. Аналогичные программы вроде бы были, но сожалению, совершенно unusable.

- Ardesia — перегруженная и глючная;
- GromIT — неудобный, глючный, и заброшенный в 2004;
- GromIT-MPX — неудачная попытка его реанимировать;
- ... ну ничего хоть близко лежащего с ZoomIt.

Good news, everyone! Наша команда реализовала ScreenPen — аналог ZoomIT, и его возможности мы покажем и опишем в этом докладе.

Итак, установив и овладев упомянутыми в докладе программами вы станете мастером ballroom & conferenceroom style выступлений, но это еще не предел!

Мы покажем, как неограниченно распространить лучи ваших знаний, ведь теперь, когда вся визуальная часть сконцентрирована на экране вашего лептопа, остается только научиться эффективно транслировать ваш экран и голос, получая от аудитории комфортную и оперативную обратную связь.

Так вот, это можно сделать обойдясь бесплатными сервисами и open-source программами, и, если останется время, мы расскажем и о том, как просто и эффективно сделать и это.

Сергей Таугер, Татьяна Смирнова, Анна Творогова, Иван Воробьев
Москва, МГУ имени М.В.Ломоносова

Проект: Micro-Manager, ImageJ
http://micro-manager.org/,http://rsbweb.nih.gov/ij

Опыт использования СПО в микроскопии: моторизованный микроскоп и Micro-Manager

Аннотация

На кафедре Клеточной биологии и гистологии Биологического факультета МГУ мы ведем исследовательскую и учебную работу с использованием моторизованных микроскопов. Использование моторизованных периферических устройств расширяет область применения микроскопа, а также снижает затраты на получение единичной микрофотографии вплоть до полной автоматизации съёмки. В качестве ПО для управление моторизованными частями микроскопа нами выбрана свободная программа Micro-Manager, которая обеспечивает полную поддержку всех имеющихся компонентов оборудования. Micro-Manager запускается как плагин к свободной программе ImageJ, широко используемой при обработке научных изображений и имеющей большой набор готовых модулей обработки и анализа изображений. Связка из Micro-Manager и ImageJ предоставляет фреймворк для автоматизированного получения и анализа микрофотографий с низким порогом вхождения. Возможности программ демонстрируются на примере двух учебных задач, предлагаемых студентам нашей кафедры.

Современные исследовательские микроскопы оснащены цифровыми камерами и моторизированы, что расширяет возможности их применения. Так, моторизованный затвор, закрывающий ход света на образец между кадрами, препятствует выгоранию флуоресцентной метки и делает возможной продолжительную съемку. Управляемое смещение вдоль оптической оси микроскопа позволяет делать набор кадров по глубине с постоянным шагом, чтобы реконструировать пространственную структуру. Моторизация смены наборов светофильтров дает возможность последовательно получать изображение объекта в разных каналах флуоресценции, предметного столика — снимать несколько полей зрения в рамках одного эксперимента. При та-

кой степени моторизации возможна полностью автоматическая съемка с высокой скоростью в 5 измерениях (пространство, время, длина волны испускания флуоресцентной метки). Такие возможности являются рутинными в современной микроскопии.

При покупке моторизованного микроскопа «под ключ» предлагают приобрести проприетарное ПО от производителя микроскопа (ZEN от Zeiss; NIS Elements от Nikon, cellSens от Olympus и LAS AF от Leica), но опытные пользователи предпочитают использовать иные программы, особенно если используются части от 4+ вендоров или используются кастомные решения. Среди наиболее распространенных программных пакетов — MetaMorph (Molecular Devices), СПО MicroManager (UCSF, Vale Lab), SlideBook (Intelligent Imaging Innovations), ImagePro (Media Cybernetics).

Для управления нашей системой используется программа Micro-Manager (MM) [1]. MM запускается как плагин к ImageJ (NIH Image) [1]. MM поддерживает большое количество микроскопов и периферии [3], а развитое сообщество пользователей готово помочь при разработке новых или нестандартных систем. Пользователю предоставляется интерфейс, удобный для решения базовых задач (рис. 3). Если функций базовой версии недостаточно, MM расширяется плагинами или пользовательскими скриптами.

Система на базе моторизованного микроскопа может расширяться для специализированных методов исследования системами для конфокальной микроскопии; микроскопии со сверхвысоким разрешением; управления температурой, газовым и жидкостным составом в объеме наблюдения, и др. Управление этими модулями доступно и пользователям MM.

Наша система (рис. 2) построена на базе моторизованного микроскопа Nikon TiE и 14 битной высокочувствительной камеры Photometrics CoolSnap HQ2 и расширена колесом со светофильтрами возбуждающего света, заслонками проходящего и возбуждающего света на базе ATmega32 и трёхосным предметным столиком (Mad City Labs).

Дружественный интерфейс позволяет новичку легко сконфигурировать систему, при сохранении возможности глубокой настройки оборудования (рис. 1).

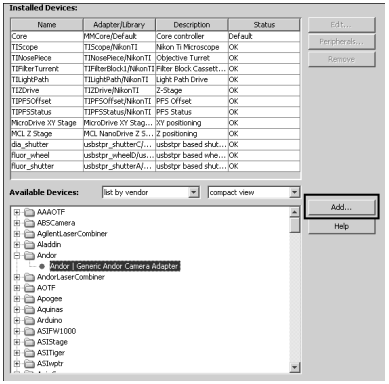


Рис. 1: GUI менеджера конфигурации оборудования. Добавление поддерживаемых устройств к текущей конфигурации осуществляется нажатием одной кнопки.

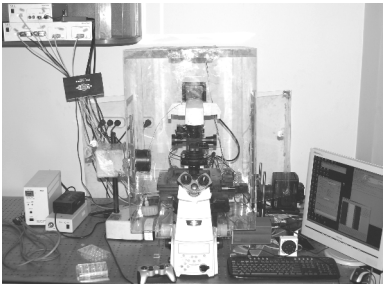


Рис. 2: Микроскоп под управлением MicroManager в нашей лаборатории

Студентам мы предлагаем две задачи, направленные в числе прочего на освоение исследовательского микроскопа под управлением ММ:

Задача 1: Настроить микроскоп, определить параметры съемки, снять серию снимков, посчитать скорость движения и длину треков гранул во время перемещения на большие расстояния в цитоплазме клеток, составить отчет.

Задача 2: Получить цветной фильм клетки с двумя флуоресцентными метками. В «зеленом» канале интервал между кадрами 5 минут; в «красном» канале — 3с, сериями по 40кадров, серии начинать раз в 5 минут; длительность съемки 15 минут.

В обеих задачах после настройки вся съёмка ведётся в автоматическом режиме.

Получение изображений в ММ

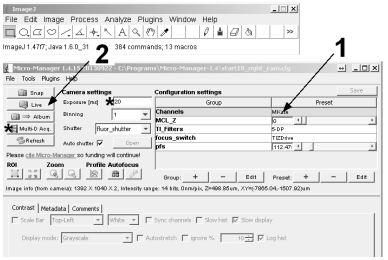


Рис. 3: Базовая настройка цейтраферной съёмки для задачи 1 в главном окне ММ

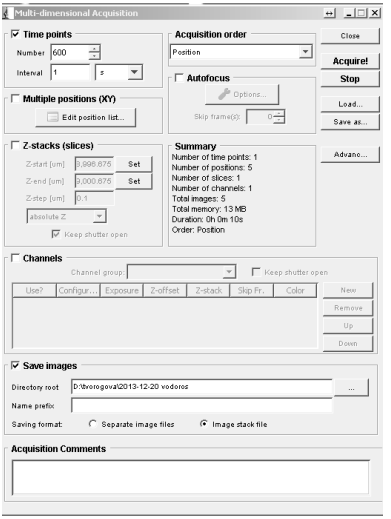


Рис. 4: Настройка многомерной цейтраферной съёмки для задачи 1

Задача 1.

Управление ведется из главного окна ММ. Выбираем из предустановленных конфигураций каналов «Phase» — съемку в проходящем свете (Рис. 3, метка 1), включаем Live режим (Рис. 3, метка 2), выставляем экспозицию камеры. Управляя предметным столиком джойстиком или через плагин Stage Control, устанавливаем область съёмки и фокус. Нам нужен фильм, поэтому переходим в меню получения множественных изображений (MultiD Acquisition).

Выставляем Time points: 600 кадров с интервалом в 1 секунду. Выбираем настройки сохранения. Acquire!

Задача 2.

Подбираем минимальную достаточную экспозицию в режиме «Live» для каждого канала флуоресценции. В меню «MultiD Acquisition» добавляем нужные каналы — «зеленый» и «красный» (рис. 5).

Настраиваем время между кадрами: переходим в «Advanced acquisition options», добавляем 40 временных точек с интервалом 3с, следующую — через 180с, и так далее (рис. 6). В настройках «зеленого» канала и проходящего света указываем пропуск 39 кадров при съемке — «Skip Frame»; «красный» канал снимается без пропуска кадров. Acquire! При еще более сложных настройках разумнее использовать скрипты.

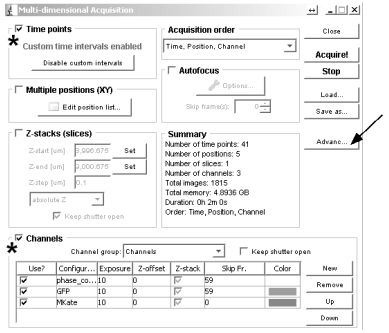


Рис. 5: Выбор необходимых каналов (проходящий свет, зелёный и красный каналы) и временных промежутков.

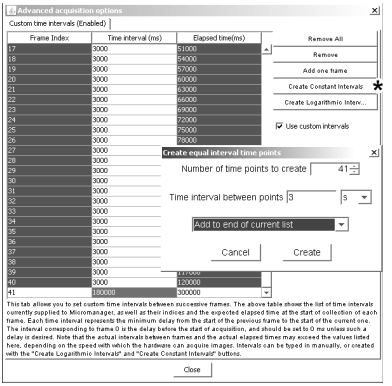


Рис. 6: Задание сложных временных интервалов.

Мы советуем студентам воспользоваться ImageJ для обработки и анализа полученных изображений, но они свободны в выборе инструментов. Так, одной студентке хватило прозрачной бумаги, приложенной к монитору, и линейки для выполнения задачи 1.

Литература

[1] *Arthur Edelstein, Nenad Amodaj, Karl Hoover, Ron Vale, and Nico Stuurman*, Computer Control of Microscopes Using μManager, 2010, Computer Control of Microscopes Using μManager. Current Protocols in Molecular Biology 14.20.1-14.20.17

[2] Rasband, W.S., ImageJ U. S. National Institutes of Health, Bethesda, Maryland, USA, `http://imagej.nih.gov/ij/`, 1997-2012.

[3] http://micro-manager.org/wiki/DeviceSupport

Татьяна Смирнова, Сергей Таугер, Анна Творогова, Иван Воробьев
Москва, МГУ имени М.В.Ломоносова

Проект: Micro-Manager, ImageJ
http://micro-manager.org/,http://rsbweb.nih.gov/ij

Опыт использования СПО в микроскопии: Возможности ImageJ и его плагинов на примере решения двух учебных задач

ImageJ [1] [2] [3] — популярная свободная кроссплатформенная программа, написанная на языке Java, для работы с растровыми изображениями. Она изначально разрабатывалась для научных изображений и получила широкое распространение в медицине, биологии и астрономии. Большое число приложений для специфических задач из этих областей сделаны как плагины к ImageJ. Платформа поддерживает все основные форматы (в том числе 32-битный TIFF, avi), форматы отраслевых стандартов (например, медицинский DICOM, HDF, форматы вендорных программ для управления микроскопическим оборудованием), имеет широкие возможности экспорта. Поддерживает многослойные изображения и виртуальные наборы серий изображений в нескольких цветовых каналах, временных точках и координатах.

ImageJ поддерживает внешние вызовы и может быть встроен или вызываться из сторонних программ. Автоматизацию работы доступна пользователю без навыков программирования за счёт упрощённого языка написания макросов (BeanShell) и встроенной функции записи произведённых действий в виде строк макроса. Плагины обычно устанавливаются размещением исполняемых файлов в папке `/ImageJ/Plugins`, и после перезапуска программы они появятся во вкладке Plugins. К недостаткам можно отнести скудность функций для рисования.

Для выполнения наших задач предлагается следующий порядок действий.

Задача 1 (применение плагинов).

- Открыть фильм при помощи команды Import (рис. 1, 5).
- Устранить смещение столика между кадрами (если есть) с помощью плагинов Images Stabilizer[4] или StackReg [5]
- Для трассировки гранул используем плагин Manual Tracking [7] (рис. 2).

Обязательное выделение гранул вручную обусловлено необходимостью игнорировать перемещения, вызванные броуновским движением.

В настройках окна «tracking» вводим параметры фильма, на котором проводятся измерения — интервал между кадрами (1с) и размер пикселя в плоскости изображения в мкм (9,5 пикс/мкм). Для каждой гранулы строится собственный трек.

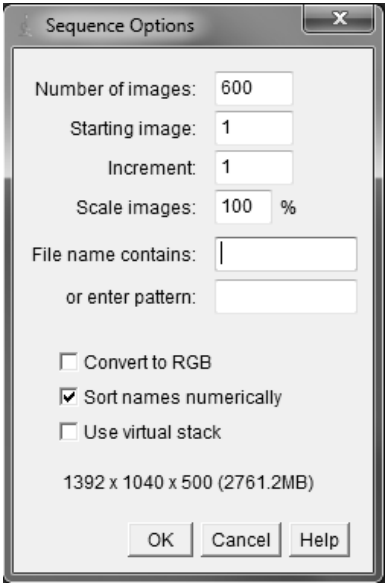


Рис. 1: Импорт серий изображений

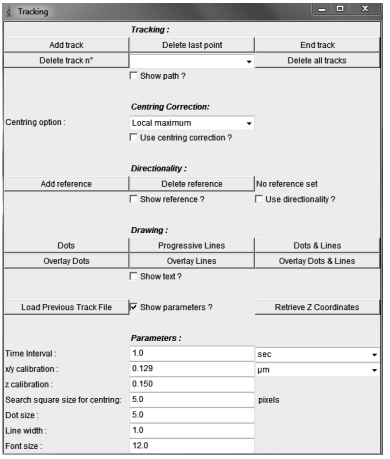


Рис. 2: GUI плагина Manual Tracking

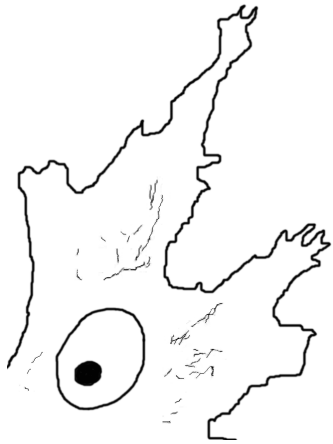


Рис. 3: Треки гранул в цитоплазме, наложенные на рисунок границы клетки.

Данные по трекам экспортируются в текстовый файл, где указаны координаты каждой точки трека, яркость пикселей в этой области, расстояние между последующими отмеченными точками и мгновенная скорость между соседними точками трека. Плагин позволяет сохранить и загрузить наборы треков, а также создать последовательность кадров, с наложением треков на фон или на последовательность фотографий (рис. 3).

Сохраненные данные по трекам легко импортируются в табличный редактор или другие программы для получения статистических данных и построения гистограмм.

Задача 2 (визуализация данных базовыми функциями ImageJ, рис. 4)

Изображение открывается как многомерное (hyperstack). Для редактирования его необходимо разделить на серии (Image > Hyperstacks > Hyperstack to Stack). Теперь можно выровнять смещение между кадрами, разделить серию на 3 части по каналам (Image>Stacks>Tools>Make Substack) и по отдельности обрабатывать «красный» и «зелёный» каналы. Простейшим способом уменьшить шум является фильтрация, например Гауссово размывание (Process>Filter>Gaussian blur) или медианная фильтрация (Pro-

cess>Filter>Median); детали изображения делаются более контрастными с помощью unsharp mask (Process>Filter>Unsharp mask).

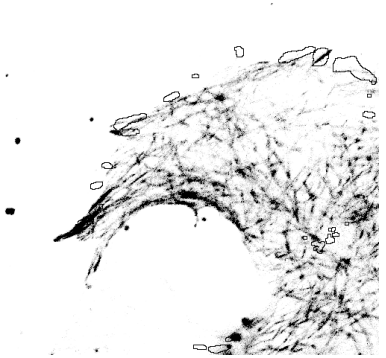


Рис. 4: Визуализация данных базовыми функциями ImageJ: «красный» канал в градациях серого; «зелёный» — контуром бинаризованного изображения. Соответствует локализации в живой клетке белков Eb3-RFP и paxillin-GFP, соответственно.

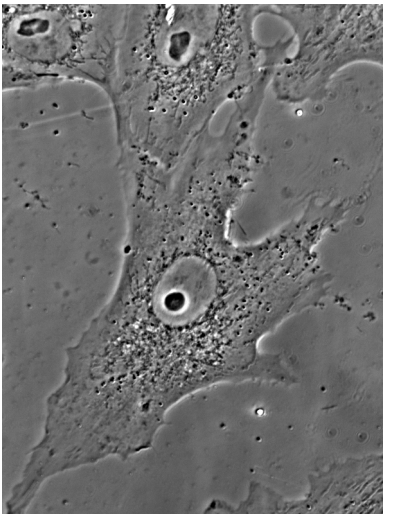


Рис. 5: Кадр из фильма (задача 1), та же клетка на рис. 3. Прижизненная съемка культуры клеток.

Обработанные серии градиентных снимков следует преобразовать в одну серию цветных. Для подготовки цветной серии снимков применяются следующие шаги: каждой серии снимков назначаем свою карту псевдоцветов (Image>Lookup Tables), после этого регулируем яркость и контраст (Image>Adjust>Brightness/Contrast, Process>Math>Gamma) и подставляем псевдоцвета как значения цвета (Image>Type>RGB color). Полученные раскрашенные каналы поточечно суммируются в финальную серию (Process>Image Calculator); эта серия может быть сохранена или как многослойный RGB TIFF,

или как анимированный GIF, или как фильм (AVI) с настраиваемой частотой кадров.

Таким образом, свободная программа ImageJ с богатейшим набором плагинов обеспечивает единую среду, пригодную для широкого круга задач микроскопии — от управления микроскопом до получения численных данных и создания иллюстраций.

Литература

[1] *Rasband, W.S.*, ImageJ U. S. National Institutes of Health, Bethesda, Maryland, USA, `http://imagej.nih.gov/ij/`, 1997-2012.

[2] *Schneider, C.A., Rasband, W.S., Eliceiri, K.W.* «NIH Image to ImageJ: 25 years of image analysis». Nature Methods 9, 671-675, 2012. (This article is available online.)

[3] *Abramoff, M.D., Magalhaes, P.J., Ram, S.J.* «Image Processing with ImageJ». Biophotonics International, volume 11, issue 7, pp. 36-42, 2004. (This article is available as a PDF.)

[4] *K. Li*, «The image stabilizer plugin for ImageJ», `http://www.cs.cmu.edu/~kangli/code/Image_Stabilizer.html`, February, 2008.

[5] `http://bigwww.epfl.ch/thevenaz/stackreg/`

[6] *P. Thévenaz, U.E. Ruttimann, M. Unser*, A Pyramid Approach to Subpixel Registration Based on Intensity, IEEE Transactions on Image Processing, vol. 7, no. 1, pp. 27-41, January 1998.

[7] Fabrice P. Cordelières `http://rsb.info.nih.gov/ij/plugins/track/track.html`

Алексей Хорошилов
Москва, ИСП РАН
http://linuxtesting.org/

Обучение принципам построения ядра операционных систем на практике

Аннотация

В докладе представлен опыт проведения лекций и лабораторных работ по программированию ядра операционной системы, целью которых является знакомство студентов на практике с основными принципами построения операционных систем. Обучение построено на основе материалов курса Operating System Engineering, который развивается в Массачусетском технологическом институте уже более 10 лет. В качестве основы для работы студентов используется учебное «экзо»-ядро операционной системы JOS, распространяемое под свободной лицензией. В докладе представлена структура курса, включая его отличия от курса МТИ, и особенности использования свободного программного обеспечения при проведении лабораторных работ. Курс «Конструирование ядра операционной системы» проводится для студентов кафедры Системного программирования факультета ВМиК Московского Государственного Университета имени М.В. Ломоносова.

Свободное программное обеспечение представляет массу материалов для обучения студентов в области информационных технологий и для развития их компетенций по различным направлениям [1]. Не является исключением и обучение принципам построения ядра операционной системы (ОС). Существует целый ряд свободных реализаций ядер ОС. Особое место среди них отводится ядру ОС Linux, которое занимает передовые позиции в целом наборе сегментов рынка, таких как суперкомпьютеры, сервера, встраиваемые устройства и мобильные устройства. Ядро ОС Linux поддерживает множество аппаратных платформ, реализует множество возможностей и при этом удерживает накладные расходы на собственную работу на приемлемой уровне. Неизбежным следствием этого является сложность внутреннего устройства ключевых компонентов ядра и достаточно высокий барьер для понимания их реализации. В результате ядро ОС Linux является отличным иллюстративным материалом для демонстрации передовых решений, проблем и тенденций в развитии и применении

ядер ОС, но мало подходит для первых шагов в программировании ядра на практике.

Для этих целей обычно используют учебные ядра, такие как JOS [2] и Pintos [3], микроядерные ОС, такие как MINIX3 [4], или даже модельные ядра, которые работают внутри обычных пользовательских процессов совместно с соответствующей моделью аппаратуры, как, например, Nachos [5], разработанная изначально на языке программирования C++ , а в последствии портированная на Java.

В рамках курса «Конструирование ядра операционной системы» для построения лабораторных работ было выбрано «экзо»-ядро JOS, которое уже более 10 лет используется для проведения курса Operating System Engineering в Массачусетском технологическом институте [2]. JOS работает в эмуляторе Qemu в режиме эмуляции архитектуры x86 и может быть запущена на не самых современных x86-совместимых компьютерах. JOS поддерживает простейшее управление задачами, управление виртуальной памятью, межпроцессное взаимодействие и файловую систему. В рамках курса Operating System Engineering части ядра JOS выдаются студентам постепенно и в неполнофункциональном виде, чтобы студенты самостоятельно доводили отдельные компоненты до работоспособного состояния.

По схожим принципам организован и курс «Конструирование ядра операционной системы». Его структура базируется на принципах проблемно-ориентированного обучения [6,7], согласно которым процесс обучения строится как последовательное решение поставленных проблем. Формирование курса заключается в построении цепочки «проблем», таким образом, чтобы для разрешения каждой следующей «проблемы» нужно было использовать как можно больше навыков, полученных при решении уже «пройденных». Однако, чтобы обучение происходило наиболее эффективно, необходимо обеспечить баланс между знанием, то есть тем, что студенту уже известно при попадании в проблемную ситуацию, и незнанием, то есть тем, что необходимо знать для ее успешного разрешения, но студенту ещё не известно.

Последовательность рассмотрения ключевых проблем построения ОС следующая:

- Как запустить вычислительную программу на компьютере?
- Как запустить несколько программ одновременно?
- Как работать с разделяемыми данными?

- Как защитить ядро от программы и одну программу от другой?
- Как обмениваться данными между программой и ядром?
- Как организовать обмен данных между программами?
- Как организовать работу с внешними устройствами хранения?
- Как организовать поддержку периферийных устройств?

На практических занятиях с целью получения ответов на поставленные проблемы выполняется постепенная разработка маленькой операционной системы, основаной на JOS. Разработка ведётся на языке Си с небольшими количеством вставок на ассемблере x86. Запуск и отладка выполняются на эмуляторе ЭВМ Qemu. Каркас операционной системы постепенно выдаётся студентам, наполнение каркаса осуществляется совместно на семинарских занятиях и в ходе самостоятельной работы.

В дополнение к совместному обсуждению проблем и выполнения небольших домашних заданий, во второй половине курса слушатели получают индивидуальные практические задания, в рамках которых требуется разработка той или иной возможности ядра ОС, как правило связанной с управлением памятью, управлением процессами или файловой системой.

Лекции состоят из двух потоков изложения. Первый поток предваряет материал, рассматриваемый на семинарах, давая вводную теоретическую информацию для последующих практических занятий. Второй поток изложения следует по рассматриваемым темам за семинарскими занятиями, охватывая проблематику каждой темы гораздо шире за счет рассмотрения альтернативных способов решения проблем, обсуждения истории вопроса, рассмотрения решений, принятых в современных промышленных ОС, в первую очередь в ОС Linux, а также за счет рассмотрения смежной проблематики.

Литература

[1] О.Л. Петренко, А.В. Хорошилов. *Профессиональное самоопределение студентов в контекстной среде обучения* в монографии «Образовательная среда вуза как фактор профессионального самоопределения студентов». ISBN 978-5-91940-066-0. Москва, 2011.

[2] Курс Operating System Engineering Массачусетского технологического института. `http://pdos.csail.mit.edu/6.828`

[3] Сайт учебного ядра ОС Pintos. http://pintos-os.org/

[4] Сайт микроядерной ОС MINIX3. http://www.minix3.org/

[5] Сайт ОС Nachos. http://homes.cs.washington.edu/~tom/nachos/

[6] А.М.Матюшкин. *Проблемные ситуации в мышлении и обучении.* Москва, 1972.

[7] О.Л.Петренко, А.В.Хорошилов. *Создание контекстной среды для подготовки ИТ-специалистов.* Сборник докладов 7-ой всероссийской конференции «Преподавание ИТ в России», Йошкар-Ола, 18–19 мая 2009 г.

Иван Хахаев
Санкт-Петербург, Открытое акционерное общество
«Научно-исследовательский институт программных средств»

Особенности моделирования морских течений в OpenFOAM

Аннотация

Рассматривается предварительные результаты моделирования турбулентных морских течений, выполненного в OpenFOAM. Обсуждаются параметры модели бассейна моря, возможности проверки ее адекватности, детали и проблемы препроцессинга и постпроцессинга. Оцениваются ресурсы, требуемые для проведения детальных расчетов, а также существующая методическая база для использования пакета OpenFOAM в учебных и исследовательских целях.

В связи со строительством космодрома «Восточный» возникла задача минимизации экологических рисков, связанных с загрязнением районов падения ступеней ракет-носителей остатками ракетного топлива и другими токсичными материалами. Часть запланированных районов падения находится в Охотском море, которое является одним из самых богатых в мире по своим природным ресурсам.

Важным инструментом прогнозирования распространения загрязнений и выработки оптимальных сценариев их локализации и ликвидации может быть компьютерное моделирование гидродинамики, в том числе с учетом сосуществования воды и льда. Однако первым этапом такого моделирования является построение модели и проверка ее адекватности.

В простейшем случае необходимо убедиться, что возможно построить модель, в которой реализуется близкая к реальности картина течений, учитываются реальные размеры акватории и рельеф дна.

В качестве инструмента моделирования выбран пакет OpenFOAM, обеспечивающий численное решение широкого класса задач механики сплошных сред (МСС). Для многих часто встречающихся классов таких задач в OpenFOAM существует готовый «решатель» — реализованный алгоритм моделирования методом конечных элементов (объемов). В рассматриваемом случае задача состоит в расчете турбулентного течения несжимаемой жидкости. В пакете OpenFOAM соответствующий решатель называется simpleFoam.

Первый шаг — моделирование свободного течения в условном квадрате шельфовой зоны с размерами 10 x 10 x 0.8 км (модель 1) — позволил подобрать начальные и граничные условия, обеспечивающие картину течений, сходную с опубликованными результатами измерений.

Второй шаг — моделирование течений для реальных размеров акватории — привел к значительным трудностям препроцессинга (построения сетки). Эти трудности связаны со сложной формой акватории, сложным рельефом дна и соотношением характерного размера акватории к максимальной глубине. При этом оказалось, что автоматизация препроцессинга (в частности, использование пакета Salome) не позволяет получить сетку, удовлетворяющую требованиям решателя. Для сетки оказываются критичными асимметрия ячеек, неортогональность ребер ячеек, а также соотношение длин ребер. Кроме того, возможно получить расходящееся решение (величина невязки становится больше максимально возможного вещественного числа) или решение с неадекватными значениями скорости течения (около 300 м/с при начальной скорости 1 м/с). Поэтому на втором шаге расчеты проводились по максимально упрощенной модели части акватории размером 2500 x 500 км и с глубинами от 2000 до 50 м (модель 2). Для такой модели удалось получить достаточно адекватную картину крупномасштабных течений.

Все вычисления выполнялись в виртуальной машине Xubuntu 12.04 в VirtualBox под ALT Linux p6. Для модели 1 просчитывалась сетка 50x50x40 ячеек, размер расчетной ячейки составлял 200x200x20 м (общее количество ячеек – 100000) при 40 шагах по времени. Общее время вычислений составило 30 минут, объем результатов вычислений составил около 1 Гб. Для модели 2 размер расчетной ячейки со-

ставлял 4000x2000 метров в плоскости X-Y при переменном размере по оси Z (глубине), общее количество ячеек в расчетном объеме составило 1425000 при максимальном соотношении длин граней 766,9. При 40 шагах по времени расчеты занимали около 7 часов, объем числовых данных, полученных в результате расчета, составил около 7.7 Гбайт. Работа с такими объемами данных уже существенно снижает скорость рендеринга результатов при поспроцессинге в ParaView. Отсюда понятно, что детализация модели и уменьшение соотношения длин граней приводит к необходимости распараллеливания вычислений в многопроцессорных конфигурациях и наличия высокопроизводительной графической подсистемы.

Методическое обеспечение работы с OpenFOAM представлено в виде документации на сайте разработчика, примерами для каждого решателя, а в русскоязычном сегменте — материалами на `https://unihub.ru/resources/teachingmaterials`. Документация на сайте разработчика носит справочный характер (и содержит некоторые ошибки) и полезна только для тех, кто уже хорошо понимает возможности пакета и специфику его применения. Примеры для решателей не имеют текстовых описаний (кроме самых общих), поэтому они позволяют только убедится в работоспособности решателя. Материалы на unihub.ru либо добавляют описания задач к примерам для решателей, либо затрагивают отдельные узкие вопросы. Нужно однако отметить, что на unihub.ru имеется русский перевод документации по препроцессингу (созданию сеток).

Таким образом, для эффективного решения задач МСС средствами OpenFOAM требуется существенное развитие методической поддержки. Один из способов такого развития — активное использование пакета в научных исследованиях с открытой публикацией конкретных задач (use cases) и способов их решения.

Евгений Алексеев, Оксана Чеснокова, Татьяна Кучер
Донецк, Донецкий национальный технический университет
www.teacher.dn-ua.com

Использование компилятора gcc и библиотеки MathGL в курсе «Вычислительная техника и алгоритмические языки»

Аннотация

Рассмотрена возможность внедрения библиотеки Mathgl в учебный процесс. Кратко описаны возможности использования библиотек при визуализации инженерных задач.

Авторы многие годы преподают программирование на C(C++) студентам электротехнического факультета. Современный курс программирования включает знакомство с методами составления алгоритмов, синтаксисом языка C(C++). На первом этапе студенты учатся разрабатывать линейные, разветвляющие и циклические программы. Далее будущие инженеры изучают функции, указатели, массивы, матрицы. Завершается курс «Вычислительная техника и алгоритмические языки» знакомством с объектно-ориентированным программированием. Для разработки приложений многие студенты используют компилятор gcc. По окончании теоретической части курса будущие инженеры в курсовой работе разрабатывают реальную программу решения электротехнической задачи. Во многих задачах иллюстрации полученных результатов необходимо построить график. Для этого можно сохранить полученные данные в файл, а затем использовать различные приложения (GNU Plot, GNU Octave, Scilab, LibreOffice Calc и др.) для построения графиков.

Однако, для построения различных графических объектов может использоваться и свободная кроссплатформенная библиотека MathGL [1], которая предназначена для создания широкого спектра графиков. С помощью этой библиотеки можно получить качественное изображение (на экране и в файле) нескольких десятков различных друх- и трёхмерных графиков. Библиотека предназначена для работы с языками программирования C(C++), Fortran, Python, Octave. В состав библиотеки также входят скриптовый язык Mathgl и утилита udav. Официальный сайт разработчика `http://mathgl.sourceforge.`

net/doc_ru/Main.html. Последнюю версию программы для различных ОС можно скачать на странице загрузки http://mathgl.sourceforge.net/doc_ru/Download.html. Группа в Google — https://groups.google.com/forum/\#!forum/mathgl. Русскоязычная страница с описанием библиотеки — http://mathgl.sourceforge.net/doc_ru/index.html, англоязычная — http://mathgl.sourceforge.net/doc_en/index.html. Синтаксис, используемый при построении графиков в MathGL, очень похож, на синтаксис Matlab, Scilab, GNU Octave. Для построения несложных графиков библиотека может быть освоена студентами первого курса, что позволит при выполнении курсовой работы получить законченный программный продукт с графической частью. Разрабатывая программу для своей курсовой работы студентам достаточно использовать Geany, gcc и MathGL. На старших курсах и в НИРС при программировании своих задач студенты могут также использовать библиотеку MathGL.

Библиотека MathGL может не только использоваться в учебном процессе, но она широко применяется и в научных исследованиях.

Литература

[1] MathGL 2.2: Main. URL: http://mathgl.sourceforge.net/doc_ru/Main.html (дата обращения: 14.01.2014)

Сергей Мартишин, Владимир Симонов, Марина Храпченко
Москва, Институт системного программирования РАН, Московский городской педагогический университет

Использование фреймворка Kohana для разработки студенческих проектов на СПО

Аннотация

Рассмотрены программные средства свободного и свободно распространяемого программного обеспечения при выполнении студенческих проектов (курсовое, дипломное проектирование) для направления подготовки «Информационные системы и технологии», профиль «Информационные технологии в образовании». Показана эффективность использования веб-фреймворков, например Kohana, совместимого с

LAMP. В проектах также используются язык PHP, СУБД MySQL и другие средства разработки информационных систем (MySQL Workbench, библиотека jQuery (JavaScript))

При обучении студентов направления «Информационные системы и технологии», помимо изучения базовых теоретических основ разработки приложений, необходимо практически подготовить студентов для последующей работы в качестве IT-специалистов. Таким образом, через разработку актуальных курсовых и дипломных проектов, студенты должны овладеть практическими навыками самостоятельного проектирования и реализации программного продукта.

При создании программного продукта в качестве вспомогательного инструмента часто используются программные средства, облегчающие разработку большого проекта.

Существует достаточно большой инструментарий, упрощающий и делающий более наглядным процесс разработки приложений. Свободное программное обеспечение здесь завоевывает все большую популярность, поскольку используемые для обучения студентов программные средства должны быть, во-первых, доступны, во-вторых, обладать необходимой функциональностью. В этом смысле доступность свободного и свободно распространяемого программного обеспечения (СПО) очевидна.

Что касается функциональных возможностей, то среди множества средств проектирования и разработки программного обеспечения (ПО) в последнее время появились средства, являющиеся СПО и позволяющие существенно автоматизировать весь процесс создания ПО.

Если программное средство, являющееся инструментом разработки, строится по каркасному принципу, то есть в нем существует постоянная часть (каркас, который остается неизменным) и изменяемая часть — модули, то такое программное средство называется фреймворк (framework).

Kohana — это веб-фреймворк с открытым кодом на PHP5, использующий архитектуру HMVC (Hierarchical Model View Controller — Иерахические Модель-Вид-Контроллер).

Важным достоинством является совместимость Kohana и традиционного набора серверного программного обеспечения LAMP (ОС Linux, web-сервер Apache, СУБД MySQL и PHP), который не предъявляет высоких требований к ресурсам (достаточно обычного офисно-

го компьютера с процессором, начиная с Pentium IV или аналогичного от AMD и 1 Gb оперативной памяти) и прост в установке.

Установка фрейворка осуществляется скачиванием с сайта `http://kohanaframework.org` актуальной версии продукта в директорию, в которой будет находится разрабатываемые модули. На данном же сайте имеется вся необходимая документация. Для корректной работы с Kohana также необходима дополнительная настройка статуса в SELinux.

При помощи Kohana создаются необходимые программные модули, выполняющие требуемые функции, и пользовательский интерфейс. Для разработки модулей используется язык PHP, который является языком программирования общего назначения с открытым исходным кодом. PHP позволяет использовать объектно-ориентированный подход.

Достоинства PHP — поддержка регулярных выражений, большое число стандартных функций, поддержка работы с различными базами данных (в том числе и MySQL), поддержка XML (Extensible Markup Language) и ODBC (Open Database Connectivity).

Одним из основных отличий использования Kohana по сравнению с CMS является управляемость и переносимость кода, поскольку в Kohana логика проекта находится в Контроллерах, а представление в Видах и перенос ПО осуществляется простым копированием. Для передачи данных из Контроллеров в файлы Вида в Kohana имеются удобные функции для работы с глобальными переменными set_global и bind_global. Также следует отметить гибкость роутинга (процесса определения маршрута внутри приложения после поступления запроса), позволяющего за счет использования регулярных выражений обращаться к нужному файлу и скрывать от пользователя пути к файлам. Также имеются функции для работы с базами данных и электронной почтой. В Kohana имеются инструменты для проверки входных данных из форм и защита баз данных от SQL-инъекций.

Фреймворк Kohana позволяет быстро разрабатывать проекты, беря на себя большую часть рутинной работы программиста. С учетом того, что приложение «тонкий клиент» для web-интерфейса становится все более востребованным, Kohana значительно упрощает разработку таких приложений. Идеология Kohana позволяет группам разработчиков взаимодействовать между собой, обеспечивая достаточную инкапсуляцию при разработке кода. Необходимо отметить, что в

отдельных случаях студентами использовались и другие фреймворки, например Yii.

Таким образом, изучение веб-фреймворка Kohana позволит будущим педагогам и специалистам (выпускникам-бакалаврам) овладеть практическими навыками разработки и использования информационных систем с веб-доступом, актуальных для образовательного процесса. Практика курсового, дипломного проектирования, а также разработка студенческих проектов на конкурсы и по заказам показала высокую эффективность использования веб-фреймворка Kohana.

Литература

[1] Kohana [Электронный ресурс]. Режим доступа: `http://kohanaframework.org`, свободный. — Загл. с экрана. — Яз. англ.

[2] Kohana [Электронный ресурс]. Режим доступа: `http://kohanaframework.su`, свободный. — Загл. с экрана. — Яз. русск.

[3] PHP [Электронный ресурс]. Режим доступа: `http://www.php.net`, свободный. — Загл. с экрана. — Яз. англ.

[4] Linux Online [Электронный ресурс]. Режим доступа: `http://www.linux.org`, свободный. — Загл. с экрана. — Яз. Англ.

[5] *Мартишин С.А., Симонов В. Л., Храпченко М.В.* Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench, учебное пособие, М: ИД Форум — Инфра-М, 2012, 160 с. -ил.

[6] *Мартишин С.А., Симонов В.Л., Храпченко М.В.* Дипломное проектирование на СПО // Сб. тезисов Восьмой конференции «Свободное программное обеспечение в высшей школе», Переславль, 26-27 января 2013, стр. 32-35.

[7] *Мартишин С.А., Храпченко М.В.* Использование LAMP и MySQL Workbench в процессе обучения студентов // Сб. тезисов Седьмой конференции «Свободное программное обеспечение в высшей школе», Переславль, 28-29 января 2012, стр. 108-110.

Евгений Чичкарев, Ольга Феодори, Мария Пономарева
Мариуполь, Донецкая обл. Украина, Приазовский государственный технический университет
http://www.pstu.edu

Сравнительное исследование производительности математических пакетов и библиотек на многоядерных процессорах

Аннотация

Выполнено сравнение производительности основных численных алгоритмов, реализованных с использованием открытых и проприетарных математических пакетов. Проанализированы причины провалов производительности открытых математических пакетов (по сравнению с MatLab), показаны условия, при которых достигается выигрыш в производительности открытых пакетов (также по сравнению с MatLab). Выполнено исследование производительности вычислительных программ на одно- и многоядерных процессорах, реализованных на интерпретируемых языках, а также откомпилированных в байт-код или двоичный код, проанализированы преимущества и недостатки открытых пакетов и библиотек.

Введение

Разработка сложных вычислительных приложений обычно требует больших усилий по отладке и профилированию алгоритмов, поэтому широкое распространение получили математические компьютерные системы, основанные на численных матричных вычислениях — Matlab и его клоны — Octave, Scilab, FreeMat и более далекие аналоги — python/NumPy/SciPy и интегрированная среда Sage. Для многих технических и научных задач использование этих пакетов позволяет получить необходимые результаты вычислений при значительном сокращении затрат труда на разработку интерфейса, построение иллюстраций, отладку численных методов, т.к. все необходимые компоненты имеются в составе всех упомянутых пакетов.

Однако Matlab и его аналоги относится к интерпретируемым языкам, что осложняет их использование для крупномасштабных научных вычислений. Все потенциальные возможности языка реализуются благодаря обширному набору библиотек и наличию широких воз-

можностей векторизации операций с матрицами и векторами. Понятие концепции векторизации занимает центральное место в понимании того, как следует писать эффективный код на языке Matlab и его аналогах. Все данные хранятся в оперативной памяти в виде матриц, поэтому и скорость работы численных алгоритмов в Matlab напрямую зависит от использования операции векторизации.

Повышение производительности программ на языке Матлаб требует соблюдения ряда правил, обеспечивающих достаточно высокую производительность разработанных программ, а именно:

- следует везде, где это возможно, использовать векторные операции, а не циклы;
- следует предварительно выделять память для векторов и матриц;
- следует использовать анонимные или инлайн-функции.

В последних версиях Матлаб появилась новая возможность создавать высокопроизводительные приложения — JIT-компилятор, позволяющий существенно ускорить выполнение кода с использование циклов for или while. При этом скорость выполнения кода с использованием векторизации операций с массивами и кода, в котором операции с массивами выполняются поэлементно с использованием вложенных циклов, оказывается вполне сопоставимой.

Другой важной возможностью повышения производительности вычислительных приложений является распараллеливание. За счет автоматического распараллеливания и многопоточности можно существенно ускорить выполнение кода встроенных библиотек на компьютерах с многоядерными процессорами (пересобрать open source с использованием Atlas и OpenMP или использовать встроенные средства Матлаб).

Постановка задачи исследования

Цель данной работы — провести сравнительно исследование производительности различных математических пакетов, проанализировать влияние на скорость решения сопоставимых классов задач наличие/отсутствие JIT-компилятора и производительности низкоуровневых функций различных пакетов (выполняющих непосредственно манипуляции с матрицами и векторами, операции обращения матриц, решение систем уравнений и т.п.).

Сопоставление различных тестовых и реальных задач выполнялось для пакетов Matlab, Octave, Scilab, python/SciPy/NumPy при работе в ОС Windows и Linux. Учитывая широкий охват различных версий пакетов, ОС, компьютеров, указание точной версии пакета и используемых условий (ОС, процессор, объем памяти) производилось непосредственно.

Большинство тестов выполнялось на компьютере с процессором Intel Core 2 Duo 2,2 GHz, оперативная память 4 GB DDR2, ОС Debian 7.2 x64. Некоторые тесты проводились на компьютере под управлением ОС Ubuntu 13.10, оборудованном 4 ГБ памяти с процессором Intel Core 2 Duo E7500 (2,94 GHz).

Производительность математических пакетов при манипулировании матрицами

Для сравнения производительности проведены и тесты с матричными операциями на Java и C++. Для тестирования параллельных вычислений с помощью математических пакетов и библиотек использовано умножение матриц, как одна из базовых и часто используемых операций над матрицами (матрицы заполнялись случайными числами). Для оценки производительности пакета Octave тест (см. ниже) выполнялся для пакета Octave 3.6.2 (поддержка параллельных вычислений — пакет octave-parallel 2.1.1). Для оценки производительности матричных операций на Python использовался Python 2.7.3 (NumPy 1.8.0 с поддержкой библиотеки линейной алгебры BLAS). Для оценки производительности MatLab использовался пакет Matlab 8.01 (встроенные средства параллелизации). Для сравнения производительности тот же тест проведен и с использованием Java (для расчетов использовалась Java SE-1.7 с библиотекой линейной алгебры JBLAS 1.2.3).

Лучшую производительность показал Matlab, причем с наименьшим использованием оперативной памяти: для умножения двух матриц 5000x5000 потребовалось 500 Мб. При выполнении теста в Octave и Python использовано 750 Мб и 700 Мб соответственно, для Java — 1200 Мб. Java показывает лучшую производительность в сравнении с интерпретируемым Python и пакетом Octave. Язык MATLAB, используемый в Octave и Matlab также является интерпретируемым, но производительность одного и того же кода в Matlab в разы выше (в данном случае — за счет JIt-компилятора).

В некоторых случаях производительность Python можно существенно повысить за счет изменения типа индексирования массива. Пакет Numpy по умолчанию использует развертывание массива по строкам (как язык C), Matlab — развертывание массива по столбцам (как Fortran). При развертывании массива по столбцам время доступа к элементам массива в Python уменьшается в 5 раз, так как для операций над массивами в Numpy используются Fortran-библиотеки ATLAS / BLAS. Например, при выполнении манипуляций с трехмерными массивами для Numpy с использованием C-массивов время выполнения теста составило 3,173 с, с использованием Fortran — массивов — 0,645 с (для сравнения на Matlab время выполнения теста составило 1,681 с). Таким образом, за счет рационального использования типов массивов даже при использовании JIT-компилятора python/NumPy демонстрирует более высокую производительность, чем Матлаб.

Анализ производительности на вычислительных задачах

В последних версиях Matlab и в экспериментальных версиях Scilab и Octave появилась возможность использования JIT компиляции. Встроенный JIT-ускоритель Matlab в первую очередь направлен на улучшение скорости циклов и операций линейной алгебры. Если строка M-кода была обработана раньше, то весьма вероятно, что переменные имеют те же типы, которые они имели в прошлый раз. При первом выполнении строки кода система анализирует переменные и генерирует специфические коды для типов данных, которые были найдены. При последующих вызовах этой строки сгенерированный код может использоваться повторно до тех пор, пока типы переменных и размеры не изменились.

Достаточно наглядный пример эффективности JIT-компилятора представлен решением задачи Дирихле для уравнения теплопроводности по явной разностной схеме. Проводилось сопоставление кода с использованием/без использования векторизации для решения одной и той же задачи. Решение выполнялось на Матлаб и Питоне. Время выполнения (в секундах) составило: Matlab 8.01 (без использования JIT-ускорителя) — 78.3628 с; Python 2.7.3 — 61.8283 с; Matlab 8.01 (с JIT-ускорителем) 0.5337 с. Казалось бы, Matlab демонстрирует чудеса производительности. Однако эффект ускорения выполнения теста за счет использования JIT-компилятора свелся к компенсации потерь

производительности при использовании откровенно неэффективной (для Matlab) операции поэлементного доступа к элементам массива во вложенный циклах for. При выполнении того же расчета с использованием векторизованного, а не поэлементного доступа "провал" производительности Octave, Scilab не наблюдается. Вывод подтверждается аналогичным тестом решения уравнения Лапласа методом сеток с использованием векторизации. Расчеты выполнялись на компьютере под управлением Ubuntu 13.10, оборудованном 4 ГБ памяти с процессором Intel Core 2 Duo E7500 (2,94 GHz) с использованием пакетов Octave 3.6.4 и Matlab 7.12. Время вычислений составило 2,34 с (Octave), 1,08 с (Matlab), т.е. выигрыш производительности Matlab на данном тесте — примерно в 2 раза. Для сравнения использование для решения системы линейных уравнений различной размерности метода сопряженных градиентов показало примерно двукратный выигрыш в производительности Octave (по сравнению с Matlab, причем на задачах большой размерности). Аналогичный тест для встроенного солвера Scilab (на базе UMFPack) показал еще больший прирост производительности.

Как язык Матлаб, так и язык Питон — языки с динамической типизацией, и JIT-компиляция для них достаточно сложный процесс. Проведение и интерпретация испытаний производительности для динамически компилируемых языков программирования, таких как Java, является намного более трудной задачей, чем для статически компилируемых языков, например C или C++. JIT-компилятор периодически перекомпилирует элементы программы в процессе её выполнения, и эта перекомпиляция может выполняться в неожиданное время после накопления определенного количества данных для анализа, при загрузке новых переменных или синтаксических конструкций. Тестирование откровенно сырого JIT-компилятора, появившегося в Octave-3.8, показало, что измерения времени при постоянной перекомпиляции могут быть совершенно не точными и обманчивыми, и эффект его использования проявляется через довольно продолжительное время. Одной из проблем написания хорошей тестовой программы является удаление оптимизирующим компилятором части конструкций, не влияющих на результат выполнения программы. Программы тестов производительности часто не выполняют вывод каких-либо результатов на экран; это означает, что некоторый или весь ваш код может быть оптимизирован и удален без вашего ведома, и с этого момента вы будете тестировать не то что думаете. В част-

ности, многие микротесты для программ на Java работают намного лучше, будучи запущенными с параметром -server вместо -client, не потому что серверный компилятор быстрее (хотя чаще всего так и есть), а потому что серверный компилятор более приспособлен к оптимизации фрагментов кода.

Учитывая наличие широкого набора встроенных функций и солверов, выполнено и сопоставление производительности Matlab, Octave, Scilab, python на комплексном тесте Scimark2. Несмотря на отсутствие в текущих версиях рассматриваемых пакетов JIT-компиляции, производительность на комплексном тесте вполне сопоставима с проприетарным (и достаточно дорогим) Matlab, а по некоторым тестам и превосходит его.

Достоинством открытых пакетов является возможность тонкой настройки программного обесчпечения на имеющийся вычислительный комплекс. В частности, выполнение аналогичного теста для python/numpy/scipy, пересобранного с использованием библиотеки Atlas, оптимизированной для работы с OpenMP на 2-х или 4-х ядерном процессоре показало прирост производительности в 1,5 -2 раза (что полностью перекрывает проигрыш по производительности Matlab с включенным Jit-компилятором).

Выводы

1. Сравнение производительности на различных тестовых задачах показала весьма неоднозначный результат:
   - производительность встроенных функций, реализованных в виде библиотек или с использованием векторизации, практически одинакова (в ряде случаев с выигрышем в сторону аналогов Matlab);
   - производительность программ с вложенными циклами в десятки раз выше с использованием JIT-компиляции.

   При этом, чтобы цикл for был ускорен с помощью JIT, он должен отвечать следующим условиям:
   - Индексы цикла являются диапазоном скалярных значений;
   - Код в цикле использует только поддерживаемые типы данных и формы массивов;
   - Любые функции, вызываемые в цикле, являются встроенными функциями Matlab.

2. Сопоставление производительности вычислений на Python, Java, Matlab, Octave показало, что полноценный компилятор в байт-код в отдельных случаях (при наличии поэлементного доступа к массивам во вложенных циклах) обеспечивает более высокую производительность.
3. Наиболее эффективное решение для разработки методов решения уравнений математической физики — использование векторизованных алгоритмов со встроенными солверами, что позволит в полном объеме использовать потенциал как открытых, так и проприетарных программ.

Алексей Дьяченко
Москва, ООО «Открытые технологии»
Проект: Moodle http://www.opentechnology.ru/

Технические и организационные аспекты внедрения СДО Moodle в образовательной организации

Аннотация

Современное образовательное учреждение уже трудно представить без электронных средств поддержки образовательного процесса: электронного тестирования, интернет-публикации учебных материалов, средств коммуникации, инструментов управления и контроля образовательного процесса. Они широко применимо не только в дистанционном образовании, но и в качестве поддержки очного образовательного процесса. Все перечисленные инструменты реализованы в свободно-распространяемой среде дистанционного обучения Moodle.

Как и у любого корпоративного ПО, внедрение СДО Moodle начинается с постановки целей и назначения ответственных лиц. Назначенному сотруднику предстоит решить вопрос физического расположения и администрирования системы, организации жизненного цикла учетных записей пользователей, обучения пользователей, наполнения системы учебными материалами и др.

Принято считать, что среды дистанционного обучения (СДО)[1] применимы, в первую очередь, в заочном образовании. Действитель-

[1]Не вдаваясь в терминологические подробности, в данной статье используется распространенный в России термин «среда дистанционного обучения» (СДО), являющийся собирательным аналогом зарубежных:VLE, CMS и LMS. Термин VLE

но, данный класс программных продуктов относятся к дистанционным образовательным технологиям. Однако, и с юридической и с практической стороны, дистанционные образовательные технологии применимы и эффективны в любой форме обучения: очной, очно-заочной, заочной, как в официальных образовательных учреждениях, так и в самообразовании, семейном или корпоративном образовании.

Обычно, выделяют следующие сервисы СДО:

- Хранение и доставка слушателям статических образовательных материалов: текстов и мультимедийных материалов.
- Тестирование слушателей.
- Поддержка стандартов обмена образовательными материалами: Scorm, IMS, различные форматы выгрузки тестов в файл. Как правило, речь идет о воспроизведении или импорте. В некоторых СДО эти инструменты дополняют, а в некоторых заменяют собой первые два пункта.
- Сбор, оценка и рецензирование работ слушателей, таких как эссе, рефераты, домашние задания и др.
- Неструктурированное взаимодействие: форум, чат, обмен сообщениями.
- Инструменты групповой работы: wiki, планировщики образовательных проектов.
- Поддержка других типов учебных материалов и образовательной деятельности. Например, глоссарий, автопроставление ссылок между материалами, перекрестное оценивание работ самими учащимися.
- Организация процесса изучения учебных материалов: отображение структуры курса, сценарии предъявления учебных материалов и заданий для прохождения, журналирование действий слушателей, текущее оценивание успехов.

можно считать более общим, хотя он более распространен в Европе и Азии, тогда как аналогичные системы в Северной Америке предпочитают называть LMS. Концепция «систем управления курсами» (CMS) является более узкой, сосредоточенной на хранении и доставке образовательных материалов слушателям и организации их образовательной активности, такой как выполнение заданий, тестов, образовательных проектов и др.

- Редактирование учебных материалов. Может заменяться импортом и воспроизведением материалов, созданных в авторинговом ПО.
- Управление списками пользователей, полномочиями доступа в системе. Импорт или API-для автоматического обмена этими данными.
- Вебинар, видеоконференция и Online-трансляция.
- Управление образовательным процессом: траектория обучения, учебный план и история прохождения дисциплин, академические группы, журналы успеваемости и посещаемости, дневники, зачетки, ведомости и другие инструменты, в зависимости от организации, уровня и формы образования.

Свободный программный продукт Moodle поддерживает все перечисленные инструменты, кроме последних двух, поддержка которых добавляется установкой не входящих в состав базовой версии модулей[2]

Определение целей и ожидаемых выгод внедрения СДО, необходимо для обоснования предстоящих затрат и поддержания интереса руководства к проекту. Успешное внедрение СДО практически невозможно без личной поддержки руководителя, уполномоченного распределять ресурсы в организации.

Назначение исполнителя, ответственного за организацию учебного процесса в СДО Moodle. Эту роль можно назвать «диспетчер-администратор», часто это сотрудник в должности руководителя или заместителя руководителя Центра Дистанционного Обучения т.п. Важно, чтобы внедрение и последующее функционирование СДО стали основной обязанностью данного сотрудника в организации, иначе проект будет пробуксовывать из-за нехватки времени.

Развертывание прототипа: на этом этапе можно не задумываться о вопросах нагрузки на систему, интеграции с другими системами. Можно воспользоваться сервисом SaaS или временной площадкой. Важно только представлять механизм переноса разработанных учебных материалов в промышленную СДО. Прототип позволит синхронизировать видение результата у всех заинтересованных лиц, начать обучение сотрудников и разработку учебных материалов.

Тестовая эксплуатация: необходимо выбрать наиболее показательную, с точки зрения целей внедрения, категорию слушателей,

[2]Присутствуют в комплекте «Русский Moodle» от ООО «Открытые технологии» `http://www.opentechnology.ru/news/russian_moodle_131222.html`.

отобрать сотрудников, участвующих в тестовой эксплуатации, составить список требуемых электронных учебных материалов.

Существует 3 подхода к созданию учебных материалов:

- Приобретение готовых учебных материалов, если удастся их найти.
- Профессиональная разработка учебных материалов. Качественный, но медленный и затратный путь.
- Разработка учебных материалов силами преподавателей. Практически неизбежны кустарный вид, неоднородное качество и проблемы с авторскими правами.

До окончания тестовой эксплуатации рекомендуется обходиться базовым функционалом СДО Moodle и избегать любых программных модификаций или установки нестандартных модулей.

Важно проработать жизненный цикл учетных записей пользователей СДО Moodle: регистрация и удаление, назначение и отписка с учебных курсов, передача итогов обучения, переводы, отчисления, академические отпуска и др. Если в организации уже внедрены информационные системы со списками слушателей и сотрудников, то интеграция с ними является единственным корректным путем.

К работе над этим и последующими этапами следует привлечь специалиста по СДО Moodle и технологиям LAMP, на которых она построена. Совместно с ним проектируется интеграция, требуемые модификации и дополнительные модули, развертывание СДО для тестовой и промышленной эксплуатации.

В. В. Яковлев, Д. В.,Хачко, А. Г. Кушниренко, М. А. Ройтберг
Москва, Пущино, НИИСИ РАН

Проект: Кумир `http://niisi.ru/kumir`, `http://gitorious.org/kumir2`

Ускорение выполнения Кумир-программ с помощью LLVM

Аннотация

Система [1] программирования Кумир, использующая Школьный Алгоритмический Язык [2], предназначена для первичного обучения программированию. Поэтому в версиях Кумир 1.x не было уделено внимание скорости выполнения программ. В версии Кумир 2.0 [3] удалось

на порядок увеличить быстродействие интерпретатора Кумир, в частности, за счет использования более простого внутреннего представления интерпретируемой программы. В версии Кумир 2.1 для Linux были предприняты усилия для дальнейшего увеличения скорости выполнение программ. Это было достигнуто, используя промежуточное представление программы в виде биткода LLVM [4].

Общие сведения

Существует особый класс задач, которые могут решаться с помощью Кумир, но при этом предъявляют высокие требования к производительности: задачи школьных олимпиад по программированию. Правильные решения олимпиадных задач, как правило, имеют алгоритмическую сложность не выше $O(N^k)$, где N — размер исходных данных; k зависит от конкретной задачи. Возможно правильные, но неэффективные решения имеют экспоненциальную сложность либо полиномиальную сложность степени $k' > k$.

При автоматизированной проверке решений олимпиадных задач, задаются ограничения на время выполнения. На вход программы подаются тестовые входные данные, размер которых достаточен для того, чтобы выявить неэффективные решения. Нормативное время выполнения программ, как правило, подбирается эмпирически в расчете на использование компилируемых (Си, Паскаль) языков программирования. Таким образом, школьники, которые используют интерпретируемые языки программирования, оказываются в невыгодном положении.

Способы генерации машинного кода

Существует два подхода для генерации машинного кода: AOT (*A*head *O*f *T*ime) и JIT (*J*ust *I*n *T*ime). В первом случае генератор кода выполняется ровно один раз во время компиляции программы, либо инсталляции на целевом компьютере; во втором — генерация кода выполняется непосредственно перед выполнением программы. Преимуществом AOT является, как правило (но не всегда), более высокая производительность полученного кода, а преимуществом подхода JIT – компактность выполняемого кода, хранимого в промежуточном представлении, и более высокая переносимость.

При проектировании системы Кумир рассматривалось несколько вариантов существующих систем генерации кода с помощью промежуточного представления.

1. **ECMA-335**[5]**,** более известный как Microsoft Intermediate Language. Помимо Microsoft, используется системами программирования PascalABC.NET и Delphi, которые работают только в ОС Windows. Имеет возможность генерации как AOT так и JIT. Стандарт представления является открытым и хорошо документированным, однако существующие реализации среды выполнения .NET и MONO используют патенты, принадлежащие корпорации Microsoft.
2. **Байткод виртуальной машины Java (JSR-000924)**[6]**,** Используется не только компилятором с языка Java, но и со многими другими языками программирования: Scheme, Clojure, Groovy и т. д. Помимо эталонной реализации Sun/Oracle среды выполнения, существует несколько открытых реализаций: IcedTea, OpenJDK, Apache Harmony, благодаря чему байткод Java выгодно отличается от MSIL в плане переносимости.
3. **Промежуточное представление в виде Си или C++ программы.** Этот способ был апробирован[7] в одной из ранних реализаций системы Кумир 1.x и был одной из первых реализаций в прототипе Кумир 2.x. Были выявлены два существенных (особенно в среде Windows) недостатка: необходимость включения в поставку полного комплекта инструментов для компиляции C/C++, и медленная работа самого компилятора C++. Преимуществом генерации в C/C++, помимо высокой скорости выполнения (используется подход AOT), является полная унификация кода runtime-библиотеки с обычной интерпретируемой версией системы Кумир.
4. **Биткод LLVM[8].** Этот способ является дальнейшим развитием метода использования существующего C/C++ компилятора, при этом из последовательности перевода «Кумир → C/C++ →Промежуточное представление → Машинный код» исключается перевод «C/C++ → Промежуточное представление», который является наиболее медленной стадией трансляции. Полученное промежуточное LLVM-представление может быть как выполнено в режиме JIT-компиляции, так и предварительно скомпилировано в машинный код.

Реализация в системе Кумир

В системе Кумир версии 2.1 основным методом выполнения программ в GUI-режиме является интерпретатор. Этот же интерпретатор доступен в виде инструмента командной строки (`kumir2-run`) в поставках для всех поддерживаемых ОС. Реализация компилятора в системе Кумир 2.x является модульной и состоит из двух обособленный частей: анализатор программ (frontend) и генератор выполняемого кода (backend).

Анализатор программ выполняет разбор текста программы и строит дерево ее разбора. Генератор кода обходит это дерево и формирует выполняемую программу. Версия 2.0 включала в себя только один генератор, который строил байткод для входящего в поставку интерпретатора. Версия Кумир 2.1 (пока только для в варианте для Linux) включает в поставку еще один генератор – генератор биткода LLVM. Входящий в поставку инструмент `kumir2-llvm` создает либо машинный код, либо, с помощью задания ключей командной строки, исходный биткод LLVM. Для генерации машинного кода, полученный LLVM-биткод передается стороннему инструменту `clang`, который является интерфейсом для запуска цепочки процессов: `llc`→`as`→`ld`.

Особенностью генератора кода в Кумир является полная эквивалентность выполнения сгенерированных программ соответствующим программам, выполняемым с помощью интерпретатора. Это достигается за счет двух факторов:

1. Стандартная библиотека языка Кумир (точнее, ее расширение, поддерживающее операции ввода-вывода и некоторые другие функциональности) реализована на C++ без использования сторонних библиотек. Один и тот же исходный код стандартной библиотеки компилируется либо как часть интерпретатора, либо, с помощью компилятора CLang, – в биткод LLVM, который затем включается в генерируемый код программ.

2. Все операции языка Кумир, которые требуют контроля во время выполнения (численное переполнение, обращение к элементам массивов, передача и массивов в виде аргументов и т. д.), реализованы единообразно как для интерпретатора, так и для LLVM-биткода – в виде функций C++, предварительно скомпилированных CLang, а затем включаемых в генерируемую программу.

Такая реализация, с одной стороны, почти даром обеспечивает полную идентичность выполнения программ по сравнению с эталонным интерпретатором. С другой стороны, вызов внешних, по отношению к LLVM-программе, C++-функций существенно снижает производительность: такие функции нельзя реализовать как встроенные (inline) фрагменты кода, поэтому программа выполняет много лишних вызовов через стек.

Для увеличения производительности, в последующих реализациях код на C++ постепенно будет заменен на генерацию соответствующих LLVM-фрагментов там, где это представляется рациональным, а для контроля эквивалентности реализации будет использоваться тестирование в процессе непрерывной интеграции.

Сравнение производительности

Для сравнения производительности на различных языках программирования были реализованы три тестовые программы:

1. Алгоритм Флойда-Уоршелла поиска кратчайших путей между парами вершин. Этот алгоритм имеет сложность $O(n^3)$. В качестве входных данных использовалась целочисленная матрица 256×256 весов, предварительно сформированная случайным образом; измерялось время выполнения 100 вызовов алгоритма.

2. Поиск числа перестановок в целочисленном массиве, используя метод сортировки слиянием. Этот тест, помимо оценки производительности работы с массивами, является тестом на эффективность многократного рекурсивного вызова функций. В качестве входных данных использовался предварительно сформированный набор случайных чисел; измерялось время выполнения 100 вызовов алгоритма.

3. Вычисление коэффициентов ряда Фурье для полинома. Вещественные коэффициенты вычисляются в цикле и нигде не сохраняются для того, чтобы можно было оценить производительность только самих вычислений. Поскольку оптимизатор компилятора Си расценивал этот код как не обязательный к выполнению, то вычислялась простая сумма вычисленных коэффициентов. Рассчитывался ряд из 10^7 пар коэффициентов разложения квадратичного полинома $x^2 + 5x + 3$; интегралы вычислялись с точностью 0.1.

При сравнении производительности использовались следующие комбинации языков, компиляторов и их опций:

1. FreePascal с включенной оптимизацией по времени выполнения (опция `-OG`).
2. FreePascal с контролем целочисленного переполнения (опция `-Co`), переполнения стека (опция `-Ct`) и выхода за границы массива (опция `-Cr`). В отличии от языка Кумир, FreePascal не имеет возможности контроля наличия значения у переменной или элемента массива.
3. Язык ANSI C90, компилятор CLang 3.3, без оптимизации (опция `-O0`). Использование языка Си с отключенной оптимизацией позволяет оценить, что именно выполняется процессором, и тем самым – делать теоретическую оценку скорости выполнения отдельных инструкций программы.
4. Язык ANSI C90, компилятор GCC 4.8.1, общеупотребительная оптимизация (опция -O2).
5. Язык Java в реализации от Oracle (версия 1.7.0) с автоматическим использованием JIT-компилятора (опция `-Xmixed`). Эта реализация интересна с точки зрения оценки производительности JIT по сравнению с AOT.
6. Язык Python в реализации CPython версии 2.7.5. С его помощью оценивалась производительность хорошо оптимизированного, но тем не менее – интерпретатора, со свойственными всем интерпретаторам недостатками.
7. Язык Кумир в реализации, описанной выше.

Производились измерения самих алгоритмов, исключая операции ввода-вывода. Результаты измерений приведены в таблице 1.

Результаты

1. Реализован компилятор с языка программирования Кумир в машинный код. Корректность работы этого компилятора подтверждена соответствием результатов выполнения полученных с его помощью программ, соответствующим эталонам, входящим в корпус тестов [9], который предназначен для интерпретатора Кумир.

Язык (опции)	Алг. Флойда		Сорт. слиянием		Коэфф. Фурье	
	Вр., мс	×C, раз	Вр., мс	×C, раз	Вр., мс	×C, раз
Pascal (`fpc -OG`)	10 540	0.96	8 210	0.69	44 880	0.91
Pascal (`fpc -Cr -Co -Ct`)	26 200	2.38	13 280	1.12	45 500	0.93
ANSI C (`clang -O0`)	11 030	1.00	11 889	1.00	49 067	1.00
ANSI C (`gcc -O2`)	2 402	0.22	7 830	0.66	48 257	0.98
Java (`java -Xmixed`)	3 798	0.34	6 878	0.58	394 218	8.03
Python (`python2.7`)	439 955	39.89	435 906	36.66	479 834	9.78
Kumir (`kumir2-llvm`)	186 362	16.90	175 447	14.76	142 493	2.90

Таблица 1: Результаты сравнения производительности различных языков программирования. Для каждой реализации приведено абсолютное время работы тестовых программ на ПК с процессором Xeon E3-1240@3.4GHz, и относительное значение, показывающее во сколько раз реализация медленнее по сравнению с не оптимизированной ANSI C-реализацией.

2. Производительность Кумир-программ, создаваемых с помощью нового компилятора, все еще уступает производительности программ, написанных на языке Паскаль. Тем не менее, этого уровня уже достаточно для успешного использования языка программирования Кумир в школьных олимпиадах муниципального уровня для учеников основной школы – наиболее массовых олимпиад, в которых могут принимать участие ученики, использующие Кумир.
   Относительно низкая производительность обусловлена особенностями текущей реализации, в которой существуют резервы для ее дальнейшего улучшения с 3–17-кратного замедления по сравнению с неоптимизированным Си до уровня 2–5-кратного.

3. Компилятор основан на инфраструктуре LLVM, поэтому на данный момент доступен только для Unix-подобных систем. Использование в среде Windows пока невозможно ввиду отсутствия (по состоянию на декабрь 2013 г., версия LLVM 3.3) работоспособного инструмента для связывания (линковки) полученного с помощью LLVM кода со стандартными библиотеками операционной системы. Недостающий инструмент в настоящее время находится в стадии активной разработки [10].

Литература

[1] `http://gitorious.org/kumir2`

[2] *Информатика: 7–9 кл.: Учеб. для общеобразоват. учр.* А. Г. Кушниренко, Г. В. Лебедев, Я. Н. Зайдельман. М.: Дрофа, 2003. 335 с.

[3] А. Г. Кушниренко, М. А. Ройтберг, Д. В. Хачко, В. В. Яковлев *Кумир 2.0: комплилятор и среда выполнения.* VIII Конференция «Свободное программное обеспечение в высшей школе», М.: Альт Линукс, 2013.

[4] "Introduction to the LLVM Compiler System" Chris Lattner Plenary Talk, ACAT 2008: Advanced Computing and Analysis Techniques in Physics Research, Erice, Sicily, Italy, Nov. 2008.

[5] `http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf`

[6] `http://docs.oracle.com/javase/specs/jvms/se7/html/index.html`

[7] М. А. Ройтберг, В. В. Яковлев *Конвертор КУМИР – C++: поддержка перехода от учебных к профессиональным системам программирования.* Третья конференция «Свободное программное обеспечение в высшей школе». М.: AltLinux, 2008.

[8] "LLVA: A Low-level Virtual Instruction Set Architecture", Vikram Adve, Chris Lattner, Michael Brukman, Anand Shukla, and Brian Gaeke. Proceedings of the 36th annual ACM/IEEE international symposium on Microarchitecture (MICRO-36), San Diego, California, Dec. 2003.

[9] А. В. Карпов, Е. А. Святушенко, Н. М. Субоч, М. А. Ройтберг *Методы тестирования в разработке системы обучени япрограммированию КуМир.* IV конференция «Свободное программное обеспечение в высшей школе». М.: AltLinux, 2009.

[10] `http://lld.llvm.org/`

Михаил Быков
Москва, diglossa.ru
https://github.com/component

Компоненты — unix way в веб-программировании

Аннотация

Доклад состоит из серии статей, тексты доступны по адресу https://github.com/mbykov/articles

1. Первое знакомство

Компоненты — это способ модульной разработки браузерного кода. Компонента может включать в себя и css, и html, и шрифты, и все вместе. Формат модулей следует стандарту CommonJS. Для Каждой компоненты можно организовать свои юнит-тесты, в том числе и в консоли.

2. Пример создания компоненты

В качестве примера создадим простую компоненту, выпадающий dropdown список. Обратите внимание, в коде нет jquery.

3. Юнит-тестирование — консоль

Пример тестирования в консоли

4. Юнит-тестирование — браузер

Тестирование в браузере можно выполнять вручную, и с помощью phantom.js, и автоматизировать для интеграции с saucelabs, travis, testem, etc, etc

5. Z's dead, baby

Если в вашем коде используется лишь часть функциональности большой библиотеки, например, jquery.js, разумно именно ее и использовать, подключив нужную компоненту. Вместо _.map используйте component/map, etc

6. make vs. grunt vs. gulp, watch

Make сохраняет свое значение среди современных task runners, включая новейшие grunt.js и gulp.js. Их популярность объясняется самоочевидностью кода, а работа Makefile усложняется отсутствием современного руководства. Тут я попытаюсь восполнить пробел.

Утилиту `github.com/visionmedia/watch` следовало бы собрать для Альт Линукс.

Павел Силаев, Николай Гарбуз
Орёл, НИЛ Спец.ПО при ФГБОУ ВПО «Госуниверситет — УНПК»

Облачный сервис управления доступом в сеть университета неограниченного круга лиц

Аннотация

В данном проекте рассматривается современный университет в разрезе построения сети с мобильным доступом. Предлагаемое решение позволяет осуществлять доступ в сеть, учитывая потребности различных групп пользователей. Система подразумевает наличие удалённого сервера авторизации и подключаемых к нему специализированных беспроводных маршрутизаторов, обеспечивающих непосредственный доступ в сеть. Взаимодействие элементов системы поддерживается приложениями с открытым исходным кодом.

Современный университет — это сложная организационно техническая инфраструктура: множество зданий и сооружений в пределах города, несколько филиалов в пределах региона (рисунок 1).

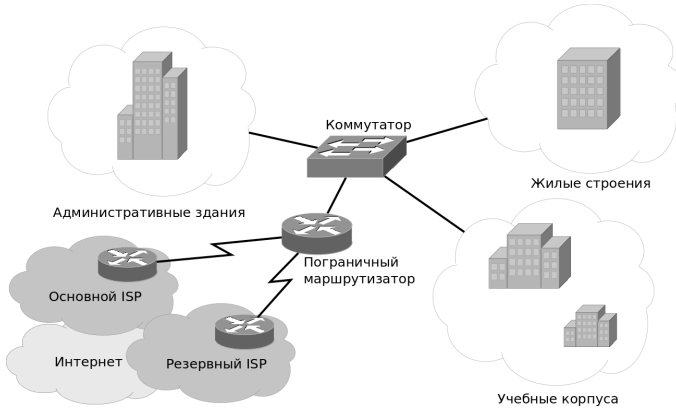


Рис. 1: Современный университет

Современный университет это:

- Административные здания;

- Социальные строения (гостиницы, общежития, спорткомплексы);
- Учебные, производственные и лабораторные корпуса.

Современный университет предъявляет особые требования к сети:

- Мобильность пользователей;
- Гарантированная связь;
- Публичный доступ;
- Простота развёртывания/расширения адресного пространства.

Удовлетворение требований к сети университета основано на трёх главных группах пользователей:

- Администрация;
- Студенты;
- Анонимные пользователи.

Таблица 1: Права групп пользователей в сети

	Доступ в Интернет	**Доступ в Интранет**	**Гарантированное качество связи**
Администрация	+	+	+
Студенты	+	+	±
Анонимные пользователи	+	-	-

Все пользователи сети всегда принадлежат к одной из этих групп. Пользователи первых двух групп обладают персональными атрибутами доступа в сеть (login/pass). Последние имеют доступ в сеть без регистрации.

Существуют и другие группы пользователей в сети, однако каждый пользователь принадлежит к одной из главных всегда.

Наибольший интерес представляет организация доступа в сеть в местах присутствия неопределённого круга лиц (вестибюли, спортивные сооружения, актовые залы, буфеты и столовые и кафе, зоны отдыха и т.п.), где подключаются мобильные пользователи по протоколу WiFi . Как правило таких местах всегда присутствуют стационарные рабочие станции административных служб, требующие гарантированной скорости соединения не зависимо от числа подключенных мобильных пользователей. Ограничение прав пользователей осуществляется на WiFi маршрутизаторе, гарантированные соединения административных рабочих мест осуществляются через

проводной Ethernet, привилегированные и анонимные — через воздушные линии связи. Такой механизм реализуется, применяя новую топологию сети современного университета (рисунок 2).

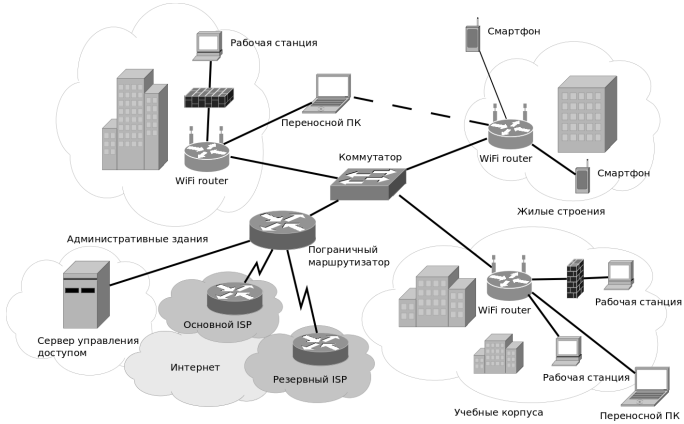


Рис. 2: Топология сети с механизмом разграничения доступа

Для организации точек доступа использован маршрутизатор NetGear N300 с прошивкой DD-WRT, распространяемой под лицензией GNU GPL, с модулем WiFi dog, организующий взаимодействие с сервером доступа по специальному протоколу для авторизации. Сервер авторизации функционирует под управлением AuthPuppy CC BY-NC-SA на платформе Debian.

В предлагаемом решении заложены условия для быстрого и гибкого развёртывания точек доступа мобильных клиентов в удалённых подразделениях и филиалах, реализуемые по двум независимым стратегиям (рисунок 3):

1. Администратору нового удалённого сегмента сети передаётся готовое к использованию устройство с установленной прошивкой, интегрирующей новый сегмент в университетскую сеть. Для инициации нового сегмента достаточно подключить питание и сеть.
2. Администратору нового удалённого сегмента сети передаётся прошивка в виде файла с инструкцией по установке, а так же спецификация оборудования, поддерживающего данную прошивку. Администратор самостоятельно устанавливает прошивку на маршрутизатор, далее новый сегмент автоматически инициализируется после подключения питания и сети.

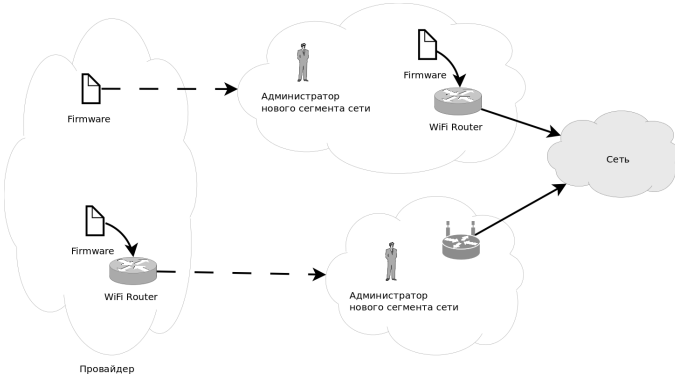


Рис. 3: Стратегии развёртывания

Дмитрий Казаков
Москва, Krita Foundation

Проект: Krita, GSoC `http://google-melange.com`

Программа Google Summer of Code как способ привлечения студентов к разработке СПО проектов

Аннотация

В докладе рассказывается о стипендиальной программе Google Summer of Code (GSoC), позволяющей студентам получить практический опыт разработки ПО, участвуя в свободном проекте. На примере проекта Krita, участвовавшего в GSoC с 2006 по 2013 годы, разобраны вопросы, возникающие при работе со студентами и их отборе. Представлены примерные темы проектов Krita на GSoC 2014. В завершение рассказывается об опыте участия Krita в программах Google Code In и Summer of KDE.

О Google Summer of Code

Google Summer Of Code (GSoC) — это стипендиальная программа для студентов, позволяющая им в течение лета работать над проектами с открытым исходным кодом. За историю существования программы через нее

прошли более 7500 студентов из 440 открытых проектов. Все вместе они написали более 50 млн строк кода. Первый GSoC состоялся в 2005 году и нынешнем году он будет проходить уже в десятый раз.

Многие из прошлых студентов никогда не участвовали в СПО проектах до GSoC; другие же использовали GSoC, чтобы целиком и полностью сконцентрироваться на уже существующем проекте в течение всего лета. Большинство «выпускников» программы становятся впоследствии менторами и помогают вновь прибывшим студентам освоиться в мире СПО.

Проект Krita принимает участие в GSoC ежегодно с 2006-го года. Все основные разработчики проекта в свое время прошли через эту программу.

Цели программы

1. Создавать и распространять свободное ПО во благо всех
2. Мотивировать молодых разработчиков к участию в свободных программных проектах
3. Помогать свободным проектам в поиске и привлечении новых разработчиков и комиттеров.
4. Предоставлять студентам возможность «работать по специальности» (читай, «жонглировать битами, а не гамбургерами»)
5. Помогать студентам получать опыт работы в реальных проектах (вопросы распределенной разработки, лицензирования ПО, этикета в коммьюнити)

Google Summer of Code 2014

Для того, чтобы принять участие в программе, студент должен выбрать организацию, с которой хочет работать, решить, какой проект он хочет выполнить и составить в срок заявку на этот проект. Несмотря на то, что условия программы напрямую не обязывают студента вступать в контакт с выбранной им организацией до даты оглашения решения о принятии/отклонения его заявки, большинство организаций предпочитают работать только со студентами, которые уже привнесли какой-то вклад в проект (коммиты, патчи, багфиксы). Только таким образом организация может удостовериться в серьезности намерений студента и оценить его знания/способности в программировании.

В 2014-м году прием заявок от студентов начнется **10 марта** и окончится **21 марта** в **19:00 UTC**[1].

[1] 19:00 UTC соответствует 23:00 по Московскому времени

Опыт нашего участия в GSoC показал, что ожидать серьезного результата можно только от студентов, пришедших в проект в конце января – начале февраля.

После окончания приема заявок организации в течение месяца их рассматривают. **21 апреля** их решения публикуется на официальном сайте программы GSoC 2014.

С 21-го апреля по 19-е мая наступает т.н. Community Bounding Period, во время которого студент знакомится со своим ментором, коммьюнити и исходным кодом проекта. При наличии договоренности с ментором студент может начать работу над своим проектом раньше срока.

В период с **19-го мая**[2] по **22-е августа** студент работает над своим проектом. Сколько времени он должен на него тратить, определяется правилами конкретной организации. В KDE, самой крупной организации из участвующих в GSoC, считается, что студент должен работать полный рабочий день, т.е. 40 часов в неделю. Конечно, никто не сможет определить, сколько на самом деле студент потратит времени, однако если ментор заметит, что студент отлынивает от работы и/или не справляется с поставленными задачами, то его проект будет прерван раньше времени и признан неудачным.

В случае успешного окончания проекта студент получит стипендию в размере 5500 долларов США. Сумма будет переведена студенту на банковскую карту тремя частями: в начале ($500), середине ($2250) и конце ($2750) проекта.

Проблемы участия в GSoC

Основной проблемой для российских студентов является летняя экзаменационная сессия, время проведения которой пересекается со сроками выполнения работ по проекту. Этой проблемы можно избежать, если договориться со своим ментором и перенести срок начала работ на начало мая. Обычно большинство организаций идут на встречу своим студентам.

Основной проблемой для организаций является отбор кандидатов. Для организации важным является не только то, чтобы студент выполнил свой проект, но и чтобы он остался в коммьюнити после окончания лета.

Чтобы решить эту проблему применяют два метода:

- принимают только тех студентов, которые давно присутствуют в проекте и сделали некоторый вклад в виде патчей и багфиксов;
- для каждой идеи проекта назначают предварительное задание, которое студент должен выполнить, чтобы быть зачисленным на GSoC.

[2] период работы пересекается с экзаменационной сессией большинства российских ВУЗов, возможные пути решения проблемы описаны ниже

Идеи проектов для GSoC 2014 в Krita

- Рисование на нескольких слоях одновременно для создания 3D-текстур
- Симуляция эффекта масляной краски
- Оптимизация пересчета дерева слоев изображения
- Пересчет дерева слоев на GPU

Другие программы для студентов

- Summer of KDE — поощрительная программа, организованная KDE для студентов, не прошедших отбор GSoC.
- Google Code In — стипендиальная программа для школьников моложе 18 лет. Проходит зимой.
- Open Academy — программа менторской поддержки студенческих команд. Проходит весной.

Дмитрий Казаков
Москва, Krita Foundation

Проект: Krita `http://krita.org`

Krita — графический редактор для художников

Аннотация

Krita — свободный графический редактор для художников, поддерживающий все популярные цветовые пространства: RGB, CMYK, Lab и Grayscale. Пользовательский интерфейс редактора заточен специально под нужды художников: наряду с поворотами и зеркалированием холста имеется функция динамического расширения границ изображения. В Krita также имеется широкий выбор кистей для рисования, в том числе имитирующих неровности поверхности бумаги и размазывание краски. Редактор может использоваться для подготовки студентов художественных специальностей к работе с графическими планшетами и рисованию на компьютере.

Krita (Крита) — это проект свободного программного обеспечения для художников. Его основной задачей является создание изображений (рисунков) с нуля. В настоящий момент Krita ориентируется на три наиболее востребованных рабочих процесса:

- создание *эскизов* для фильмов и компьютерных игр. Этот процесс включает в себя прототипирование персонажей, раскадровку сюжета фильма или окружения компьютерной игры. Основными требованиями в данном случае являются наличие разнообразных кистей и простота изменения размера холста «на лету», чтобы художник мог продолжать рисовать во всех направлениях без дополнительных манипуляций;

- создание *комиксов*. В данном случае для нормальной работы художника приложение должно обеспечивать качественный рендеринг тонких линий, как на экране монитора, так и на печати. Оно также должно предоставлять возможность быстрого добавления штриховок и заливок;

- подготовка *текстур* для 3D моделей компьютерных игр. Для обеспечения этого процесса Krita имеет два режима работы *Режим зацикливания (Wraparound Mode)* и *Массив Клонов (Clones Array)*.

Для упрощения навигации по изображению и выполнения наиболее часто используемых операций в Krita предусмотрены специальные жесты:

- *прокрутка* — Space + клик на холсте
- *масштабирование* — Ctrl + [Alt] + Space + клик на холсте
- *поворот холста* — Shift + [Alt] + Space + клик на холсте
- *изменение размера кисти* — Shift + клик на холсте
- *инструмент «пипетка»* — Ctrl + [Alt] + клик на холсте

Помимо продвинутого функционала для художников в Krita присутствуют и вполне ожидаемые инструменты:

- поддерживаются все популярные цветовые пространства (RGB, Grayscale, CMYK, Lab)[1] в произвольной разрядности (8/16 bit integer, 16/32 bit float);

- имеются инструменты преобразования изображения: поддерживаются как афинные преобразования, так и «деформации», описываемые с помощью сплайнов.

[1] В результате работы с художниками мы выяснили, что многие непопулярные в полиграфии цветовые пространства, такие как Lab и scRGB, находят свое применение у художников: из-за различий в их математическом определении привычные алгоритмы смешения цвета дают другой (обычно более мягкий) результат.

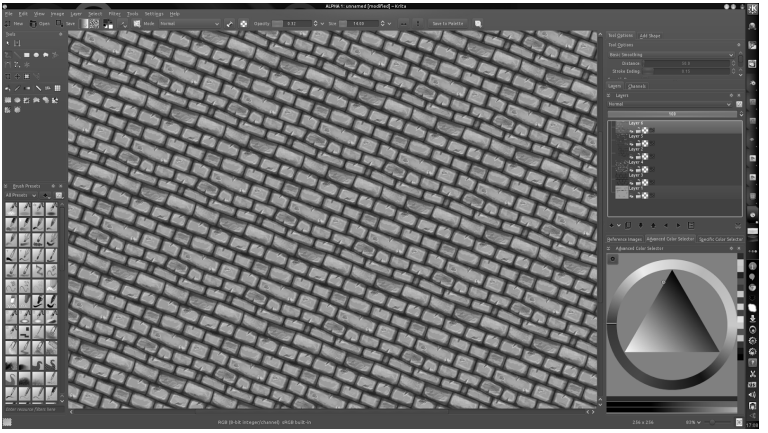


Рис. 1: Режим зацикливания (Wraparound Mode). Изображение 256 × 256 пикселей зацикливается на экране. Рисование возможно в любой части холста

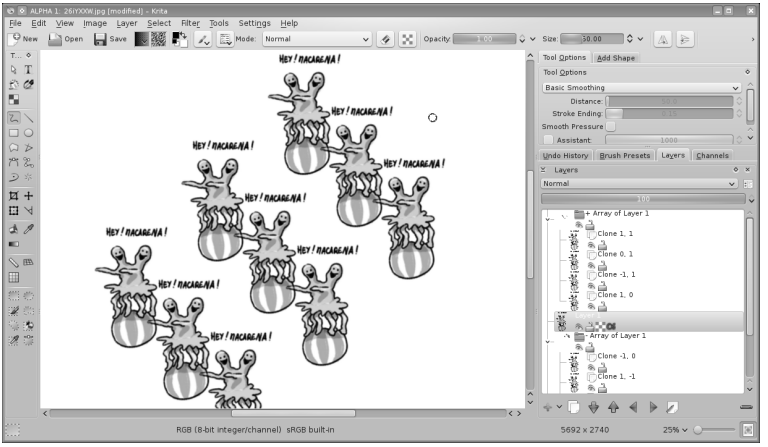


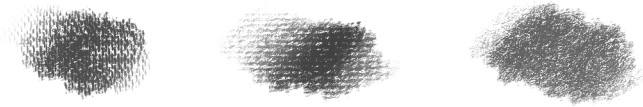
Рис. 2: Массив клонов (Clones Array). Изменение оригинала повторит изменение во всех клонах

Основным отличием Krita от аналогов является широкий выбор движков кистей (brush engines):

- *пиксельная кисть* (pixel brush) — простейший движок, соответствующий кистям других редакторов;

- *пачкающая кисть* (color smudge brush) кроме подачи краски на холст также размазывает текущее содержимое слоя, что моделирует процесс рисования сыпучими/жидкими материалами (пастель, краска);

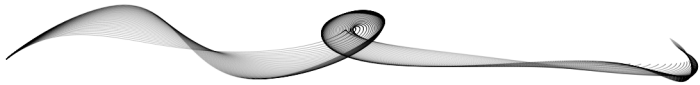
- к каждой кисти можно добавить *текстуру*, которая будет моделировать шероховатость бумаги;



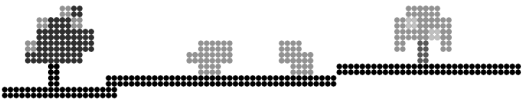
- *кисть для набросков* (sketch brush) позволяет создавать/заполнять объемы штриховкой всего лишь одним движением руки;

- *кисть с щетиной* (hairy brush)

- *«частицы»* (particle brush)



- *«решетка»* (grid brush)



В данный момент проект Krita активно развивается. За 2013 год в репозитарий было добавлено более трех тысяч коммитов от более чем шестидесяти контрибьюторов. Проект в очередной, восьмой, раз принял участие в программе Google Summer Of Code и подготовил два релиза.

Дмитрий Жильцов
Обнинск, ИАТЭ НИЯУ МИФИ

Проект: Agda `http://wiki.portal.chalmers.se/agda/pmwiki.php`

Язык Agda и его использование

Аннотация

Язык Agda — это язык функционального программирования, основанный на интуиционистской теории типов. Благодаря чрезвычайно выразительной системе типов его можно использовать как систему проверки доказательств. В докладе описываются основные его особенности и примеры использования при исследованиях в области теории вычислимости, языков программирования и формальной верификации. Кроме того описаны его текущие ограничения и приведены оценки перспективы его использования в преподавании математики и программирования.

Agda — это язык функционального программирования с зависимыми типами. Он основан на интуиционистской теории типов Мартин-Лёфа [13]. Текущая версия разработана Ульфом Норелом в своей диссертации [18]. Теория Мартин-Лёфа была значительно расширена. В частности, Agda поддерживает индуктивные семейства [7] и индуктивно-рекурсивные типы [8].

Компилятор Agda написан на языке Haskell, его исходный код свободно доступен. Кроме того, Agda имеет специальный режим для редактора Emacs, обеспечивающий интерактивную разработку, поэтапное уточнение и запуск не полностью определённых программ.

Agda поддерживает параметрический полиморфизм, индуктивные типы и определение рекурсивных функции над ними посредством сопоставления с образцом (pattern matching). В этой связи он схож с языками Haskell и

ML. Однако определения индуктивных типов должны удовлетворять условию строгой положительности, при определении рекурсивных функций допускается использование только структурной рекурсии, кроме того определение должно покрывать все возможные случаи. Эти условия гарантируют то, что любая программа на языке Agda завершается. Ограничение на рекурсию не является слишком жестким, поскольку Agda поддерживает гибкий механизм видов (views), определённый в работе [15].

На язык Agda распространяется соответствие Карри-Говарда: его можно рассматривать как формальную систему интуиционистской логики. При этом типам соответствуют высказывания, а типизированным выражениям – доказательства этих высказываний. Наличие зависимых типов позволяет определять предикаты. Это (а также разрешимость процедуры проверки принадлежности выражения к типу) делает язык Agda достаточно выразительным, чтобы рассматривать его как систему проверки доказательств.

Выразительные свойства Agda (в частности возможность описывать синтаксис и семантику других языков) делают его популярным инструментом исследования в области языков программирования, формальной верификации и т.д. В докладе основное внимание уделяется двум взаимосвязанным проблемам. Первая это конструирование универсумов для нужд политипического программирования [1, 10, 12, 16, 17]. Вторая это исследование связей между семействами типов, при которых один тип можно рассматривать как уточнение другого, и реализация механизмов переносов определений и доказательств между такими типами [3, 6, 14].

Кроме того обсуждаются текущие ограничения и проблемы, связанные с использованием языка Agda и родственных языков с зависимыми типами (Epigram, Idris). В частности, упоминаются проблемы, лежащей в основе теории [2]; автоматизации процесса доказательств и создания языка тактик доказательств [9, 11]; компиляции и оптимизации [4, 5]. Рассматриваются перспективы использования языка Agda в образовательном процессе.

Литература

[1] Thorsten Altenkirch and Conor McBride. Generic programming within dependently typed programming. In *Generic Programming*, pages 1–20. Kluwer, 2003. Proceedings of the IFIP TC2 Working Conference on Generic Programming, Schloss Dagstuhl, July 2002.

[2] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In *PLPV '07: Proceedings of the 2007 workshop on Programming languages meets program verification*, pages 57–68, New York, NY, USA, 2007. ACM.

[3] Robert Atkey, Patricia Johann, and Neil Ghani. When is a type refinement an inductive type? In *Proceedings of the 14th international conference on Foundations of software science and computational structures*, FOSSACS'11/ETAPS'11, pages 72–87. Springer-Verlag, 2011.

[4] Edwin Brady. Idris, a general-purpose dependently typed programming language: Design and implementation. *J. Funct. Program.*, 23(5):552–593, 2013.

[5] Edwin Brady, Conor McBride, and James McKinna. Inductive families need not store their indices. *Types for Proofs and Programs, Torino, 2003*, volume 3085 of *LNCS*, pages 115–129. Springer-Verlag, 2004.

[6] Pierre-Évariste Dagand and Conor McBride. Transporting functions across ornaments. In *ICFP 2012*, pages 103–114. ACM, 2012.

[7] Peter Dybjer. Inductive families. *Formal Asp. Comput.*, 6(4):440–465, 1994.

[8] Peter Dybjer and Anton Setzer. Indexed induction-recursion. In *Proof Theory in Computer Science*, volume 2183 of *Lecture Notes in Computer Science*, pages 93–113. Springer, 2001.

[9] Simon Foster and Georg Struth. Integrating an automated theorem prover into Agda. In *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2011.

[10] Daniel R. Licata and Robert Harper. A universe of binding and computation. In *ACM SIGPLAN International Conference on Functional Programming*, 2009.

[11] Fredrik Lindblad and Marcin Benke. A tool for automated theorem proving in Agda. In *TYPES*, volume 3839 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2004.

[12] Andres Löh and José Pedro Magalhães. Generic programming with indexed functors. In *Proceedings of the 7th ACM SIGPLAN Workshop on Generic Programming*, WGP '11, pages 1–12, New York, NY, USA, 2011. ACM.

[13] Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1980.

[14] Conor McBride. Ornamental algebras, algebraic ornaments. Manuscript, available online, 2011. `http://personal.cis.strath.ac.uk/~conor/pub/OAAO/LitOrn.pdf`.

[15] Conor McBride and James McKinna. The view from the left. *Journal of Functional Programming*, 14(1):69–111, 2004.

[16] Peter Morris, Thorsten Altenkirch, and Neil Ghani. A universe of strictly positive families. *International Journal of Foundations of Computer Science*, 20(1):83–107, 2009.

[17] Peter Morris, Thorsten Altenkirch, and Conor McBride. Exploring the regular tree types. In *Types for Proofs and Programs (TYPES 2004)*, Lecture Notes in Computer Science, 2006.

[18] Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology and Göteborg University, 2007.

Игорь Воронин, Вероника Воронина
Московская обл., Шатура, Нижегородская обл., г. Павлово,
ИПЛИТ РАН, МБОУ СОШ №7 г.Павлово

Проект: Образовательный проект УМКИ, Фестиваль РоботоБУМ
`http://umki.vinforika.ru/`, `http://robotobum.ru`

Роботы в образовании или что такое «РоботоБУМ»

Аннотация

Существует множество важных проблем, на которые никто не хочет обращать внимания, до тех пор, пока ситуация не становится катастрофической.

Одной из таких проблем в России становится её недостаточная обеспеченность инженерными кадрами. Все чаще падают космические ракеты и спутники, происходят техногенные катастрофы, обусловленные недостаточным профессионализмом обслуживающего персонала, разработчиков и проектировщиков.

Это вызвано, конечно, целым рядом причин. Однако, все, связанные с образовательной средой единодушно отмечают, что в последние несколько лет наблюдается снижение интереса учащихся к изучению точных наук, и как следствие, падение качества образования в целом. Как выйти из этого тупика разбирается в данной статье.

Робототехника в образовании

Работу по мотивации детей к занятиям серьезной наукой нужно начинать как можно раньше, желательно в начальной школе! Откуда такой вывод? При анкетировании детей, на предмет, желают ли они заниматься в кружках технической направленности определилась следующая картина:

в девятых и более старших классах практически никакого интереса, в 6-8-х классах интерес проявился в основном у тех детей, которые самостоятельно дома, или в организациях дополнительного образования занимаются легоконструированием, радиоэлектроникой, программированием. А вот у учащихся четвертого класса интерес оказался просто огромен. То есть, если дети до 11-12 лет не касались технического творчества, то с возрастом у них интерес к этому занятию возбудить достаточно сложно. Поэтому, работу по пропедевтике робототехники, физики, знакомству с началами программирования необходимо проводить в начальной школе и пятых классах. В результате, в среднюю школу придут дети, у которых прилично развиты конструкторские навыки, сформировано алгоритмическое мышление, привит интерес к экспериментированию.

Вывод: таким образом необходимо активно начинать пробуждение интереса к точным наукам и массовую популяризацию профессии инженера, причем предпринимать такие шаги, необходимо для детей с достаточно раннего возраста. Необходимо вернуть в общество массовый интерес к научно-техническому творчеству.

На настоящий момент, существует достаточное количество образовательных технологий которые способствуют развитию критического мышления и умения решать задачи, однако, в образовательных средах, вдохновляющих к новаторству через науку, технологию, математику, способствующих творчеству, умению анализировать ситуацию, применить теоретические познания для решения проблем реального мира, сегодня наблюдается определенный дефицит.

Наиболее перспективный путь в этом направлении — это робототехника, позволяющая в игровой форме знакомить детей с наукой. Робототехника, которая является эффективным методом для изучения важных областей науки, технологии, конструирования, математики и входит в новую международную парадигму: STEM-образование (Science, Technology, Engineering, Mathmatics).

Организация лаборатории робототехники в школе или учреждении дополнительного образования — это:

- внедрение современных научно-практических технологий в образовательный процесс;
- содействие развитию детского научно-технического творчества;
- популяризация профессии инженера и достижений в области робототехники;
- новые формы работы с одаренными детьми;
- эффективные формы работы с проблемными детьми;
- возможности инновационного обучения;

- игровые технологии в обучении;
- популяризация профессий научно-технического направления.

Роботизированные платформы УМКИ

В настоящее время, мало кого можно удивить радиоуправляемой машинкой. Но если необходимо управлять не одним устройством, а двумя, тремя, сотней?..

Как договориться с устройством, чтобы радиосигнал от одного пульта не заставлял сразу все машинки двигаться в одну сторону? Как добиться, чтобы затраты энергии на управление были бы минимальны, а сеть таких машинок просуществовала максимально долго? Как организовать взаимодействие между несколькими операторами устройств?

Очевидно, что в ситуации где присутствует много управляемых элементов, люди могут очень легко запутаться. Поэтому необходимо, чтобы команды управления по радиоканалу адресовались на конкретное устройство, точно в нужный момент времени. Даже если робототехническая платформа находится вне зоны прямой видимости базовой станции, нужно уметь позиционировать, местонахождение каждого устройства, с тем, чтобы, например, зная в какой точке роботом обнаружен устройством очаг задымления, вторая платформа, оборудованная средствами тушения, могла подъехать точно в это место и потушить пожар.

Вот так, чтобы легко и просто — в процессе игры, научиться управлять подобными устройствами, был разработан образовательный конструктор: УМКИ — Управляемый по радио каналу Машинный Конструктор Инновационный — разработка российских ученых, в отличие от большинства предлагаемых сегодня в образовании зарубежных образовательных робототехнических комплектов. Достоинство конструктора в том, что он комплектуется роботизированными платформами, которые связываются между собой в единую управляемую сенсорную сеть, на основе протокола ZigBee.

В процессе работы с этим конструктором, ученики получают базовые знания по управлению сначала одним устройством, потом группой устройств объединенных в распределенную беспроводную сенсорную сеть. Каждая роботизированная платформа оснащается либо набором сенсоров-датчиков для различных физических величин, либо исполнительными механизмами. Машинку можно заставить двигаться по программе, ориентироваться на местности и выполнять разнообразные задания. Курс обучения составлен таким образом, чтобы на каждом этапе детям было максимально интересно получать знания. Выполняя шаг за шагом задания к занятиям — учащиеся проходят разнообразные миссии: осваивают далекие планеты, занимаются охраной окружающей среды, тушат пожары и многое-многое

другое. Занимаясь с конструктором УМКИ, в игровой форме дети получают основы серьезных инженерных знаний, воспитывается их информационная, техническая и исследовательская культура, происходит формирование навыков коллективного труда.

В базовой комплектации текущей версии «Лаборатории УМКИ» роботизированный комплекс-конструктор УМКИ состоит из:

1. Четырехколесного вездехода, иначе говоря, передвижной платформы (элементы «Электронного конструктора «Знаток»» — радиоуправляемый вездеход «Лидер») с модулем zigbee, который позволяет связываться множеству платформ в единую, распределенную самоорганизующуюся сенсорную сеть.

2. Радио-шлюза — соединяющегося по USB с персональным компьютером, который служит для отправки команд по радиоканалу и приема ответов о выполненных процедурах.

3. Программного обеспечения для управления передвижной роботизированной платформой.

Вместе с тем, конструктор УМКИ может быть доукомплектован набором различных датчиков, унифицированых таким образом, что они легко и просто подключаются к базовой платформе и становятся доступными для выполнения дополнительных миссий. Также в комплект Лаборатории входят электронные конструкторы «Школа 999» и «Альтернативные источники энергии», наборы конструкторов механических роботов и управляемый летающий робот-мультикоптер.

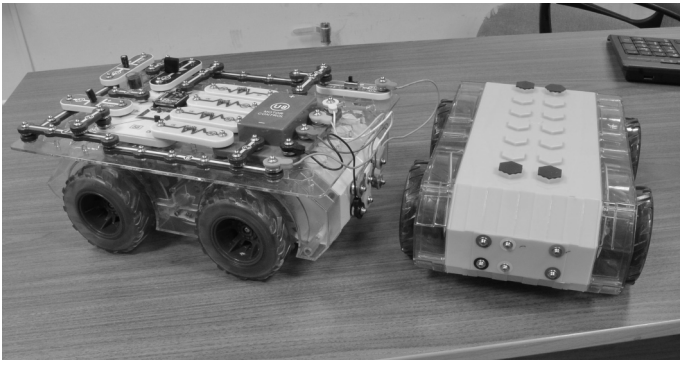


Рис. 1: Вездеход «Лидер» из конструктора «Знаток» с модулем zigbee

Разработана программа курса и методические материалы для преподавателей (учителя начальных классов, педагоги дополнительного образования, родители, руководители кружков технического творчества), наборы заданий для детей и подробное руководство пользователя (автор программы Воронина В.В.). В материалах курса «УМКИ» раздел ориентированный на детей наполнен научно-популярной информацией, задания представлены в игровой форме — как процесс изучения окружающего нас мира. Все миссии курса представлены в разнообразных вариантах, допускающих изменения и связаны между собой логической составляющей и при необходимости каждый из учителей сможет, без особых временных затрат сформировать собственную рабочую программу.

После первого обучающего этапа на котором происходит начальное знакомство с управлением умной машинки (SmartCar) — УМКИ, на котором дети учатся создавать простые управляющее программы и знакомятся с различными сенсорными устройствами, происходит непосредственное изучение возможностей конструкторов «Лаборатории УМКИ», реализованное в форме миссий.

Что такое РоботоБУМ

Это Всероссийский фестиваль РОБОТОтехники — Будущее Умных Машин. Сейчас все большую популярность набирают различные соревнования роботов.

Чем же отличается РоботоБУМ от прочих фестивалей и состязаний роботов?

Состязания роботов сейчас становятся все популярнее, однако, все подобные состязания имеют спортивную направленность: роботы сражаются друг с другом, соревнуются кто быстрее и точнее пройдет заданным маршрутом, и т. д., РоботоБУМ же имеет научный характер. Хотя фестиваль РоботоБУМ включал множество различных «эпизодов», где происходили мастер-классы по управлению и сборке роботов и так же демонстрировались возможности сконструированных роботов, но в то же время, дети создавали макеты роботов и антураж «миссий», сочиняли стихи и участвовали в интеллектуальны играх, и главное основным событием — стержнем мероприятия РоботоБУМ, стала Всероссийская научно-практическая конференция, на которой съехавшиеся с разных концов страны ребята, представили свои научно-исследовательские и практические работы, работы, презентовали авторские модели роботов, определили перспективы своей дальнейшей работы.

Выбор роботизированных платформ фестиваля РоботоБУМ, не ограничивался, как в большинстве случаев, зарубежными поставщиками лего

Рис. 2: Участники фестиваля РоботоБУМ

и др, а в большей мере был ориентирован на разработки отечественных ученых.

В этом мероприятии принимали участие ребята, которые занимаются в учреждениях дополнительного образования, победители и призеры республиканских, краевых и областных мероприятий технической направленности из различных субъектов Российской Федерации.

География участников была представлена очень широко — ребята съехались из различных уголков нашей страны. Например Екатеринбургскую область области представляли команды из Екатеринбурга и Ирбита, Нижегородскую — ребята из Павлово (Команда УМКИ МБОУ СОШ №7) и Выксы, были команды из Омска, из Кабардино-Балкарии и других субъектов федерации.

И подводя итог, можно резюмировать: робототехника в школе предоставляет учащимся технологии XXI века, способствует развитию их коммуникативных способностей, развивает навыки взаимодействия, самостоятельности при принятии решений, раскрывает их творческий потенциал.

Рис. 3: Победители и призеры фестиваля РоботоБУМ

Григорий Злобин, Петр Рыковский, Роман Шувар
Львов, Львовский национальный университет имени Ивана Франко

Использование свободного программного обеспечения на факультете электроники ЛНУ имени Ивана Франко. Перезагрузка. Год спустя

Аннотация

В докладе рассмотрена ситуция с использованием СПО на факультете электроники ЛНУ им. И. Франко через год после принятия решения о переходе на использование СПО в учебных лабораториях

Свободное программное обеспечение на факультете электроники ЛНУ имени Ивана Франко используется уже больше двадцати лет [1], [2], [3], [4], [5], [6]. В [1] перечислены направления использования СПО:

- серверы — Linux (Debian, Open SuSE), Unix FBSD;

- обучение — операционная система (Debian, Open SuSE), офисный пакет (OpenOffice), средства программирования (gcc, Kuzya IDE, Qt Creator), математические пакеты (Octave, SciLab, Maxima, Labplot), технология терминал-сервер;
- студенческая научная работа — операционная система (Debian, Open SuSE), офисный пакет (OpenOffice), средства программирования (gcc, Kuzya IDE, Qt Creator), математические пакеты (Octave, SciLab, Maxima, Labplot), системы управления базами данных (MySQL), эмуляторы аппаратных средств и операционных систем;
- научные исследования — операционная система (Debian, Open SuSE), офисный пакет (OpenOffice), средства программирования (gcc, Kuzya IDE, Qt Creator), математические пакеты (Octave, SciLab, Maxima, Labplot), организация вычислительных кластеров (Scientific Linux, Kickstarter, Webmin, SGE, Ganglia, OpenMPI , MPICH2, BLAS, FFTW, NorduGrid ARC, Condor, CUDA 5.0 production release).

До 2012 г. вопрос выбора программного обеспечения для использования в учебном процессе и научных исследованиях решался исключительно преподавателем либо исследователем. К сожалению, лицензионность программного обеспечения при этом не учитывалась. Поэтому на рабочих местах в учебных и научных лабораториях накопилось большое количество нелицензионного программного обеспечения для ОС Microsoft Windows. Ситуация с нелицензионным программным обеспечением на факультете электроники обострилась после получения ректором университета в ноябре 2012 г. письма от представительства фирмы Microsoft в Украине с предложением приобрести в 2012/2013 гг необходимое количество лицензий на ОС Microsoft Windows на все рабочие места в университете. Очевидно, что средств, которые были выделены факультету электроники на 2013 г., было недостаточно для обеспечения лицензионности программного обеспечения на рабочих местах в учебных и научных лабораториях. Поэтому комиссией, которая была создана на факультете, было принято решение о переводе большей части учебных лабораторий на свободное программное обеспечения. На факультете электроники для обеспечения учебного процесса функционирует семь общефакультетских учебных лабораторий ОЛ 1, ОЛ 2, ОЛ 3, ОЛ 4, ОЛ 6, ОЛ7.

В лаборатории ОЛ1 развернуто 15 ПЭВМ (Intel Pentium IV 3 ГГц, ОЗУ 512 МБ) с операционными системами Microsoft Windows XP и Debian GNU/Linux.

В таблице 1 представлена информация об учебных курсах и программном обеспечении, которое используется для проведения лабораторных работ в ОЛ1.

Таблица 1:

Название курса	**ОС**	**Прикладное программное обеспечение**
Микропроцессорная техника	Windows/ Wine+Linux	Keil mVision
Проектирование реконфигурируемых микропроцессорных систем	Windows/ Linux	пакет программ фирмы Cypress
Цифровые сигналы	Windows/ Wine+Linux	DIP (собственная разработка в Delphi)
Графическое программирование	Windows/ Linux	язык G в среде LabVIEW
Микропроцессорные системи автома-тического управления	Windows/ Linux	SciLab

В лаборатории ОЛ3 развернуто 14 ПЭВМ (Intel Pentium II 366 МГц, ОЗУ 512 МБ), которые выполняют функцию терминальных клиентов для учебного сервера терминалов AMD® Phenom® CPU 4x с частотой 2,6 ГГц, 8 Гб ОЗУ, 250 Гб HDD и операционной системой Open SuSE. В таблице 2 представлена информация об учебных курсах и программном обеспечении, которое используется для проведения лабораторных работ в ОЛ3.

Таблица 2:

Название курса	**ОС**	**Прикладное программное обеспечение**
Теория алгоритмов	Linux	Free Pascal, Codeblocks
Системный анализ	Linux	Geany, СУБД Oracle
Компьютерные сети	Linux	netmon, Firefox, PacketTracer
Инженерная компьютерная графика	Linux	Gimp, Labplot
Персональные компьютеры	Linux	Прикладное программное обеспечение для Linux
Моделирование в электронике	Linux	KiCad, Lucid Electronics Workbench
Проектирование систем моделиро-вания	Linux	Qtiplot
Основы системного программиро-вания	Linux	gcc, Codeblocks

В лаборатории ОЛ4 развернуто 12 ПЭВМ (Intel Celeron 2.66ГГц, ОЗПр 512 МБ) с операционной системой Open SuSE. В таблице 3 представлена информация об учебных курсах и программном обеспечении, которое используется для проведения лабораторных работ в ОЛ4.

Таблица 3:

Название курса	**ОС**	**Прикладное программное обеспечение**
Проектирование информационных систем	Linux	Dia, star-UML umbrello
Вычислительная техника и автомати-зация эксперимента	Linux	nasm, Labview
Моделирование систем	Linux	Altera Quartus
Web-технологии	Linux	Eclipse, Gimp, Веб-переглядач
Операционные системы	Linux	gcc, nasm
Компьютерная графика	Linux	Gimp, Labplot
Проектирование аналоговых и цифровых систем	Linux	Scilab, Proteus
Современная теория управления	Linux	SMathStudio
Апаратное и программное обеспе-чение нейронных сетей	Linux	Codeblocks, Geany
Методы распознавания образов	Linux	н.д.
Нейронные сети	Linux	Codeblocks, Geany

В лаборатории ОЛ5 развернуто 14 ПЭВМ AMD® Athlon II 2x с частотой 2,9 ГГц, 8 Гб ОЗУ, 500 Гб HDD, GPU — Nvidia GTS450 (CUDA 2.1), операционная система: Scientific Linux 6.2 ядро 3.6.6, которые являются узлами учебного вычислительного класса и одновременно выпоняют функцию терминальных клиентов для учебного сервера терминалов Intel® Xeon® CPU X3440 4x с частотой 2,53 ГГц, 16 Гб ОЗУ, 500 Гб HDD и операционной системой Linux (SUSE 13.2), ядро 3.7. В таблице 4 представлена информация об учебных курсах и программном обеспечении, которое используется для проведения лабораторных работ в ОЛ5.

Таблица 4:

Название курса	**ОС**	**Прикладное программное обеспечение**
Числовые методы	Linux	Free Pascal, Geany, GCC, Code::Blocks, LabPlot
Математические методы исследова-ния операций	Linux	Free Pascal, Geany, GCC, Code::Blocks, LabPlot
Организация баз данных	Linux	LibreOffice Base, MariaDB, PostgreSQL
Инженерная компьютерная графика	Linux	GIMP, InkScape, KiCAD, FreeCAD
Цифровая обработка информации	Linux	SciLab, Maxima, FreePascal, GCC, FreeImage, Geany, Code::Blocks, GIMP, Audacity
Программирование и алгоритмичес-кие языки	Linux	Free Pascal, Geany, GCC, CodeBlocks
Компьютерная схемотехника и архи-тектура компьютеров	Linux	Ktechlab, .QUCS, Eschema, KiCad
Проектирование Веб-систем	Linux	Firefox, Kate, JavaScript, PHP
Веб-технологии	Linux	Apache, PHP, Perl, Firefox, Kate, JavaScript
Основы программных средств информацион-ных технологий	Linux	LibreOffice
Системы мультимедиа	Linux	SciLab, Maxima, Geany, GCC, Code::Blocks, GIMP, Audacity
Аналоговая и цифровая обработка информации	Linux	gcc, CodeBlocks, Eclipse, GIMP, Audacity
Проектирование информационных систем	Linux	Dia, Umbrello
Управление проектами	Linux	LibreProject, OpenProj

В лаборатории ОЛ6 развернуто 15 ПЭВМ (AMD Athlon II X2 240 2,8 ГГц, ОЗУ 4 ГБ) с операционными системами Microsoft Windows XP и Debian GNU/Linux. В таблице 5 представлена информация об учебных курсах и программном обеспечении, которое используется для проведения лабораторных работ в ОЛ6.

Таблица 5:

Название курса	**ОС**	**Прикладное программное обеспечение**
Алгоритмизация и программирование	Windows/ Linux	Embarcadero RAD Studio может использоваться Free Pascal, CodeBlocks
Объектно-орієнтированое программирование	Windows/ Linux	Embarcadero RAD Studio может использоваться Free Pascal, CodeBlocks, Qt Creator
Кроссплатформное программирова-ние	Windows/ Linux	Microsoft Visual C# может использоваться Mono Develop
Технологии компьютерного модели-рования	Linux	LibreOffice и любой язык програм-мирования
Системы компьютерной математики в научных исследованиях	Linux	Scilab, Maxima
Цифровая обработка изображений	Windows/ Linux	собственные разработки в Delphi
Цифровые сигналы	Windows/ Linux	собственные разработки в Delphi

В лаборатории ОЛ7 развернуто 8 бездисковых ПЭВМ Intel® Pentium® II/III с частотами 400 — 1000 МГц, 256 МБ ОЗУ и 4 ПЭВМ Intel® Pentium® II/III с частотой 2 ГГц, 256 МБ ОЗУ, которые выполняют функцию терминальных клиентов для учебного сервера терминалов AMD® Phenom® 4x с частотой 2,6 ГГц, 8 ГБ ОЗУ, 250 ГБ ЖМД, операционая система Open SuSe 13.2. В таблице 6 представлена информация об учебных курсах и программном обеспечении, которое используется для проведения лабораторных работ в ОЛ7.

Таблица 6:

Название курса	**ОС**	**Прикладное программное обеспечение**
Основы компьютерной лингвистики	Linux	Python, Qt Translate, ISpell, ESpell
Теормя принятия решений	Linux	GCC, Geany, CodeBlocks
САПР ОЕС	Linux	Sprint-layout 5 (учебная версия)

Таблица 6 — продолжение

Название курса	**ОС**	**Прикладное программное обеспечение**
Компьютерное проектирование оптоэлектронных систем	Linux	Zemax (пробная версия)
Управление проектами	Linux	LibreProject, OpenProj, LibreOffice, Umbrello, UML
Математические методы исследова-ния операций	Linux	Free Pascal, Geany, GCC, Code::Blocks

Подводя итоги можно константировать, что:

1. Несмотря на неожиданный характер перехода на свободное программное обеспечение переход в целом оказался удачным;

2. Широкий спектр доступного свободного программного обеспечения показал возможность полного обеспечения лабораторных работ по нормативным и выборочным дисциплинам. Это ставит под сомнение целесообразность дальнейших трат денежных средств на приобретение платных программных продуктов. В первую очередь следует направлять денежные средства на своевременное обновление и модернизацию компьютеров, которые используются в учебном процессе.

3. Лабораторные работы по некоторым курсам выполняются в MS Windows не через необходимость, а только потому, что лекторы, которые читают эти лекционные курсы, не позаботились о поиске свободных аналогов используемых программ либо о запуске нужных им программ в системе Wine.

Литература

[1] Апуневич С. Є., Злобін Г. Г., Рикалюк Р. Є., Шувар Р., *Використання вільного програмного забезпечення у навчанні і наукових дослідженнях у Львівському національному університеті імені Івана Франка*, 2011

[2] Батюк А.Я., Злобін Г.Г., *Використання ВПЗ для тестування апаратного забезпечення ПЕОМ в навчальному процесі факультету електроніки ЛНУ імені Івана Франка*, 2012

[3] Бойко Я. , Ванькевич Д., Злобін Г., *Використання технології віртуалізації в навчальному процесі факультету електроніки ЛНУ імені Івана Франка*, 2012

[4] Рудий М.Ф., *Використання крос-платформного інструментарію розробки програмного забезпечення Qt для створення навчальних програм*, 2012

[5] Шийка Ю.А., Шувар Р.Я., *Виконання завдань розподіленої обробки зображення під управлінням системи CONDOR*, 2012

[6] Столярчук О.В., Шувар Р.Я., Продивус А.М., *Виконання завдань розподіленої обробки зображення під управлінням системи CONDOR*, 2012

Владислав Величко
Луганск, Украина, Луганский национальный университет имени Тараса Шевченко

Построение информационно-образовательной среды университета на базе свободного программного обеспечения

Аннотация

За компонентами информационно-образовательной среды университета рассматривается возможность её построения с использованием свободного программного обеспечения.

Использование свободного программного обеспечения в учебном процессе не носит ни систематический, ни последовательный характер. Причин такой ситуации, на наш взгляд несколько. Во-первых, это психологическая проблема, основанная на мифе о том, что свободное программное обеспечение предназначено для специалистов в области информационных технологий и её использование будет непосильным рядовому пользователю. Данное опасение не лишено смысла, ведь если углубиться в историю свободного программного обеспечения, то следует признать, что оно создавалось именно для узкого окружения, но в данное время свободное программное обеспечение предназначено и может быть использовано для широкого круга пользователей. Во-вторых, методическая проблема, поскольку до настоящего времени нет целенаправленной ориентации относительно сопровождения учебных материалов которые бы опирались на свободное программное обеспечение, или как вариант не опиралось бы на конкретное программное обеспечение. Действующие учебные программы, учебники, методические разработки опираются именно на проприетарное программное обеспечение. В-третьих, как это ни парадоксально, финансовая проблема. Для внедрения свободного программного обеспечения, которое зачастую является бесплатным, иногда одно понятие подменяют другим, необходимы средства для его внедрения. Эти средства необходимы для курсов повышения квалификации, разработку, адаптацию и внедрение новых учебных программ

и их методического сопровождения, также средства необходимы для популяризации свободного программного обеспечения.

Чтобы преодолеть вышеперечисленные проблемы следует целенаправленно и систематизировано использовать свободное программное обеспечение в образовательной деятельности, провести своего рода сопоставление существующего программного обеспечения к существующим запросам. Современное университетское образование с точки зрения информационной составляющей описывается с помощью понятия информационно-образовательной среды (ИОС).

Информационно-образовательная среда университета как сложная многомерная реальность изучалась во многих аспектах. Как способ развития самостоятельной работы при изучении того или иного учебного предмета, как фактор развития информационной культуры будущего учителя, как средство повышения эффективности обучения в университете (О. Зайцева, Н. Сизинцева, О. Ардеев и др.). Исследовались педагогические основы развития информационной среды вуза, теоретические основы становления педагога в информационной среде, система формирование готовности учителей к конструированию ИОС предметного обучения, дидактические условия развития дистанционного образования в информационном пространстве университета, дистанционная поддержка образовательной деятельности педагогов (О. Соколова, Т. Абрамян, Е. Кулик и др.). Изучались отдельные элементы среды, информационно-правовой и информационно-логический компоненты среды телекоммуникационного обучения (Т. Еременко, О. Зимина, В. Мозолин и др.). Информационно-образовательная среда изучалась как объект проектирования и социального измерения (К. Кречетников, И. Захарова, О. Казанская, М. Нежурина, Н. Мельникова и др.), как основа для формирования научно-информационного пространства университета (В. Богословский, В. Потёмкин). Без внимания не были оставлены и технологические аспекты построения информационно-образовательной среды университета (О. Казанцева, В. Гужов, С. Астанин, М. Нежурина и др.), но не было акцентировано внимание не том, может ли свободное программное обеспечение полностью удовлетворить спрос на программное обеспечение для информационно-образовательной среды.

В структуре информационно-образовательной среды присутствуют четыре компоненты, в которых в той или иной степени используется программное обеспечение. Для *пространственно-семантического компонента,* где упор делается на создание информационного окружения со стороны нашего исследования необходимо программное обеспечение для создания и поддержки всевозможных информационных порталов, сайтов, электронных библиотек, виртуальных музеев и т.п. Для выполнения этих задач в свободном программном обеспечении имеется целый арсенал необходимых средств. *Технологический компонент* ИОС нуждается в средствах

организации обучения, в том числе и дистанционном, удаленном. В частности, в проведении всевозможных виртуальных лекций, вебинаров, тестирований, контрольных, ведение журналов учета и посещаемости и т.п. *Компетентностный компонент* ИОС нуждается в программном обеспечении способном создавать базы знаний, информационно-поисковые системы, интеллект-карты и т.п. *Коммуникативный компонент* ИОС нуждается в программном обеспечении виртуального взаимодействия между субъектами среды в он-лайн и офф-лайн режимах.

Анализ запросов информационно-образовательной среды проведенный по структуре её компонент на программное обеспечение показывает, что современное свободное программное обеспечение полностью удовлетворяет как в количественном так и качественном плане все возрастающие запросы университетского образования. И соответственно переход на использование свободного программного обеспечения для построения информационно-образовательной среды университета не только возможен, но и позволит подготовить будущих специалистов, которые в скором времени и будут формировать общественное представление о свободном программном обеспечении решая возникшие проблемы в его использовании.

Литература

[1] . Панченко Л.Ф. *Інформаційно-освітнє середовище сучасного університету* : монографія / Л.Ф. Панченко ; Держ. закл. «Луган. нац. ун-т імені Тараса Шевченка». – Луганськ : Вид-во ДЗ «ЛНУ імені Тараса Шевченка», 2010. – 280 с. [на укр. языке]

Андрей Батюк, Василий Рабык
Львов, Львовский национальный университет имени Ивана Франко

Проектирование аналоговых и цифровых систем в ОС Linux

Аннотация

Рассмотрены особенности инсталляции и работы программ проектирования аналоговых и цифровых систем под управлением операционной системы Linux и особенности подключения лабораторных стендов к компьютерам.

Ряд дисциплин, читаемых на факультете электроники Львовского национального университета имени Ивана Франко, связанные с проектированием аналоговых, цифровых и микропроцессорных систем. К ним относятся: «Микропроцессорная техника», «Микропроцессорные системы», «Аппаратное и программное обеспечение нейронных сетей», «Проектирование аналоговых и цифровых систем», «Микропроцессорные системы автоматического управления». При выполнении лабораторного практикума по этим дисциплинам используются соответствующие программные среды разработки и проектирования.

Подавляющее большинство программ проектирования аналоговых и цифровых систем работают под управлением ОС Windows. Сегодня часто возникает необходимость работы этих программ под управлением ОС Linux. Для их установки в ОС Linux был использован эмулятор Windows API–WINE.

Разработка программ и их тестирование для микропроцессорных систем на основе микроконтроллеров семейства MCS51 выполняется с помощью среды Keil μVision4 на лабораторном стенде EV8031/AVR [1]. Для пересылки кода программы (расширение *.hex, *.bin) из компьютера в микроконтроллер стенда или внешнюю память стенда используется программа EvalGUI, разработанная производителем лабораторных стендов через интерфейс RS232. Подключение стенда к компьютеру осуществляется через преобразователь USB — RS232, реализованный на микросхеме FT232RL:

```
$ ln -s /dev/ttyUSB0 ~/.wine/dosdevices/com1
```

Лабораторные работы по «Микропроцессорной технике» и «Микропроцессорным системам» связанные с изучением схем подключения периферийных устройств к микроконтроллерам семейства MCS51, реализацией алгоритмов и программ ввода с этих устройств в микроконтроллер и вывода из микроконтроллера во внешние устройства как дискретных, так и аналоговых сигналов. В частности, изучаются интерфейсы обмена информацией 1Wire, I2C; вывод информации на светодиодную «линейку», статическую и динамическую индикацию, знакосинтезирующий индикатор; жидкокристаллический индикатор; обмен информацией между стендом и компьютером с помощью интерфейса RS232; формирование сигналов заданной формы с помощью ЦАП; измерение постоянного напряжения с помощью АЦП; измерение периода и частоты последовательности прямоугольных импульсов с использованием лабораторного стенда.

Для проектирования цифровых устройств и систем на основе ПЛИС используется среда IDE Quartus II Ver. 9.1 Web Edition фирмы Altera, которая имеет поддержку ОС Linux и лабораторный стенд DE0 [2] с установленной на нем ПЛИС EP3C16F484C6 семейства Cyclone III фирмы Altera. Эта ПЛИС имеет FPGA структуру. В её состав входят: 15408 логических

блоков, 56 блоков памяти объёмом 9K, 56 блоков умножения 18x18, 20 глобальных цепей синхронизации. Связь стенда DE0 с ПК осуществляется через USB интерфейс. В состав стенда входит интегрированный USB Blaster, через который выполняется программирование конфигурации ПЛИС. Для полноценного его функционирования необходимо осуществить настройку режима работы на компьютере с ОС Linux [3] и выбрать USB Blaster в настройках Quartus II.

Для инсталляции Linux версии IDE Quartus II необходимо загрузить с сайта производителя установочный пакет (упакованные бинарные файлы *.tar.gz), распаковать и установить:

```
$ gzip -d 12.1sp1_243_quartus_free_linux.tar.gz
$ tar -xf 12.1sp1_243_quartus_free_linux.tar
$ cd ./12.1sp1_243_quartus_free_linux
$ ./setup --standalone --install=quartus_free --temp=
```

Для 64–битных систем следует заменить libz.so.1 в 12.1sp1_243_quartus_free_linux/ altera_installer/bin из соответствующего пакета для 32–битного дистрибутива.

Для удобства работы в .bashrc добавляем следующую строку:

```
PATH=${HOME}/altera/12.1sp1/quartus/bin:$PATH
```

После этого для запуску IDE Quartus II можно использовать команду:

```
$ quartus
```

или для 64–битных систем:

```
$ quartus -64bit
```

Для корректной работы программатора необходимо добавить правило udev:

```
# cat /etc/udev/rules.d/52-usbblaster.rules
SUBSYSTEMS=="usb", ATTR{idVendor}=="09fb",
     ATTR{idProduct}=="6001", MODE="0666",
     GROUP="adm", OWNER="andrew"
```

Лабораторные практикум по курсу «Аппаратное и программное обеспечение нейронных сетей» связан с изучением лабораторного стенда DE0, интерфейса САПР Quartus II, методов описания цифровых устройств на основе ПЛИС. Выполняются работы, связанные с выводом информации

на индикацию (светодиодная линейка, семисегментный индикатор, жидкокристаллический индикатор), исследованием мегафункций арифметических операций, проектированием АЛУ, реализацией и моделированием основных узлов нейронной сети на ПЛИС.

Для выполнения лабораторных работ по курсу «Проектирование аналоговых и цифровых систем» используются IDE Quartus II и SCILAB. С помощью пакета программ SCILAB выполняются работы по преобразованию сигналов (дискретное и быстрое преобразование Фурье), проектированию аналоговых и цифровых фильтров. В IDE Quartus II осуществляется реализация цифровых систем, связанных с обменом информации между стендом DE0 и компьютером, выводом графической информации (VGA), реализация рекурсивных и нерекурсивных цифровых фильтров.

Литература

[1] Учебный отладочный стенд «Ev8031/AVR», `http://opensys.com.ua/Stend/Ev8031`

[2] Altera DE0 Board, `http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=364`

[3] USB-Blaster Driver for Linux, `http://www.altera.com/download/drivers/dri-usb_b-lnx.html`

Сергей Мартишин, Марина Храпченко
Москва, Институт системного программирования РАН

Анализ способов защиты информации в базах данных

Аннотация

Анализируются способы защиты информации в системах управления базами данных (СУБД) свободного и свободно распространяемого программного обеспечения (СПО). Показана необходимость обучения администрированию СУБД при выполнении студенческих проектов (курсовое, дипломное проектирование) для направления подготовки «Информационные системы и технологии», профиль «Информационные технологии в образовании». Даны рекомендации по дальнейшему повышению Безопасности хранения данных

В предыдущих докладах [2,3] авторы рассматривали вопросы использования свободного и свободно распространяемого программного обеспечения (СПО) в процессе обучения студентов, а также при выполнении дипломного проектирования. Однако помимо механического освоения программных продуктов, например, входящих в LAMP, студентам необходимо хорошо понимать наличие требований к конфиденциальности хранимой информации. Нет никаких проблем с установкой СПО и началом работы с ним, но обратной стороной такого подхода является необходимость обеспечения достаточного уровня безопасности при хранении и обработке данных.

Поскольку база данных является ядром практически любой информационной системы (ИС), то ее настройки, поддержки и администрирования напрямую зависит качество работы ИС.

В последние годы появилось значительное количество систем управления баз данных СПО, которые обладают высокой производительностью, бесплатны и хорошо документированы. Кроме того, многие из них предназначены для использования в ОС Linux с широко распространенным сервером Apache и многие языки программирования (PHP, Python, Ruby) имеют необходимые драйверы и функции, позволяющие достаточно просто наладить работу с этими СУБД.

Заметим также, что не все эти СУБД поддерживают реляционную модель данных, все большее распространение начали приобретать базы данных, основанные на модели NoSQL («не только SQL»). Также следует заметить, что чаще всего СУБД имеют клиент-серверную архитектуру и являются распределенными. То есть подавляющее большинство пользователей имеют интерфейс «тонкий клиент», поэтому основная часть задач по обработке информации переносится на некоторый ресурс, например, сервер или облако. Очевидно, что данные пользователя, при выполнении им запросов, подвергаются различного рода угрозам, которые связаны с атаками на программное обеспечение и данные пользователей. В связи с этим особенно актуальным становится вопрос о безопасности информации, причем не только самих данных, но и результаты запросов и вычислений, а также информации, которая может быть получена в результате такого рода запросов.

Практически во всех СУБД имеются специальные средства защиты данных, как от их несанкционированного использования, так и от сбоев и отказов аппаратуры. Также имеются кроссплатформенные аппаратно-программные средства защиты информации при ее распределенной обработке. Таким образом для защиты информации используются:

- аутентификация, авторизация и контроль прав доступа пользователей, а также назначение им привилегий,
- брандмауэр (сетевой экран) для фильтрации пакетов,

- передача по сети зашифрованных данных, использование VPN,
- в случае хранения конфиденциальных данных, используется их шифрование (например, хеширование паролей или криптопротоколы для данных внутри СУБД),
- регулярное обновление ПО для поддержки системы безопасности в актуальном состоянии,
- доверенные вычислений на базе криптосерверов, например, Trusted Platform Module (TPM).

Заметим, что значительная часть средств защиты, относящаяся непосредственно к СУБД, требует всестороннего изучения студентами, поскольку безопасность на этом уровне определяется качеством администрирования базы данных, в частности, назначением привилегий.

Однако не следует забывать, что имеется опасность сговора администратора и похищения данных. Кроме того, количество разрешенных функций для того или иного пользователя может быть ошибочно расширено. Использование защищенных модулей и брандмауэров также полностью не решает проблему, поскольку они уязвимы для хакерских атак.

Таким образом, очевидно, что отсутствие шифрования данных на стороне клиента может привести к похищению или модификации их инсайдером. Поэтому помимо администрирования, большое значение имеет изучение студентами методов работы с зашифрованными данными (например, полностью гомоморфное шифрование [5]).

Литература

[1] Мартишин С.А., Симонов В. Л., Храпченко М.В. *Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench*, учебное пособие, М: ИД Форум — Инфра-М, 2012, 160 с. -ил.

[2] Мартишин С.А., Симонов В.Л., Храпченко М.В. *Дипломное проектирование на СПО* // Сб. тезисов Восьмой конференции «Свободное программное обеспечение в высшей школе», Переславль, 26-27 января 2013, стр. 32-35.

[3] Мартишин С.А., Храпченко М.В. *Использование LAMP и MySQL Workbench в процессе обучения студентов* // Сб. тезисов Седьмой конференции «Свободное программное обеспечение в высшей школе», Переславль, 28-29 января 2012, стр. 108-110.

[4] Варновский Н.П., Мартишин С.А., Шокуров А.В., Храпченко М.В., *Методы пороговой криптографии для защиты облачных вычислений*, Труды института системного программирования РАН, сдана в печать.

[5] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

Дочкин Сергей
Кемерово, ФГБОУ ВПО «Кузбасский государственный технический университет»

Особенности внедрения свободных продуктов в профессиональное образование: региональный аспект

Аннотация

В статье рассматриваются особенности внедрения программного обеспечения на основе свободных лицензий в систему профессионального образования региона, связанные с этим трудности и результаты проведенных исследований. Данные вопросы рассматриваются в аспекте использования СПО в образовательных организациях среднего и высшего профессионального образования, как основных элементов системы образования решающих задачи подготовки кадров для экономики региона. Отмечается что особую роль в этом должны сыграть образовательные организации дополнительного профессионального образования, решающие задачи повышения квалификации и профессиональной переподготовки педагогов.

Несмотря на высокие темпы развития информационно-коммуникационных технологий (ИКТ), проблема отставания России по показателям, характеризующим состояние информационного общества и степень информатизации остается нерешенной. Одна из причин такого отставания — недостаточное внимание к российским разработкам в области ИКТ, в том числе ориентированным на свободное программное обеспечение (СПО). Не следует забывать, что согласно Плана реализации Стратегии развития информационного общества в РФ, уже к концу 2010 года 25% общеобразовательных учреждений каждого региона РФ должны были использовать СПО не менее, чем на 50% компьютерах. Причем предварительный анализ показал, что к началу 2014 года данный показатель все еще не достигнут. Вторая причина — все еще недостаточно высокий уровень подготовки российских пользователей (и работников образования в том числе) в области ИКТ, который в настоящее время сводится к уровню пользования конкретным продуктом: операционная система Windows, программы из MS Office, и практически все.

Один из возможных вариантов изменения ситуации — освоение и широкое использование свободных программных продуктов, в том числе и отечественных. К решению проблемы следует приступать с организаций образования, обучая как преподавателей, так и самих обучающихся. И значимость работы организаций дополнительного профессионального образования (ДПО) в этом направлении достаточно велика.

Применительно к профессиональному образованию (ПО) вообще, и формирования ИКТ-компетентности педагогов в области СПО в частности следует отметить следующее. Задача модернизации профессиональной школы — преодоление замкнутости и обеспечения открытости ее внешним воздействиям, и одно из важнейших направлений реализации этой задачи — формирование и развитие региональной информационной образовательной среды. Однако, обучение учащихся в образовательных организациях ПО информатике, формирование их ИКТ-компетентности на основе коммерческих программ (обычно иностранных) ограничивает возможности взаимодействия, интеграции и открытости. А также связано с большими финансовыми затратами. До сих пор коммерческое программное обеспечение остается недоступным для многих ОУ, особенного среднего профессионального образования, что не позволяет использовать весь диапазон возможностей средств ИКТ в образовательном процессе, требует постоянного технического обновления оборудования из-за возрастающих требований разрабатываемых программ. Анализ показал, что только около 10% образовательных организаций ПО пользуются операционные системы на основе GNU/Linux. Свободные приложения Firefox, OpenOffice.org, GIMP и прочие уже освоены и установлены не менее чем на 7–8% компьютеров (в том числе и под ОС Windows), и их доля увеличивается вместе с ужесточением законодательства об авторских правах и постоянные тратами на продление лицензий. В свою очередь подключение к Интернету открывает возможность получения свободно распространяемых программ и их использования, оставаясь в правовом поле, выбора того перечня программ которые максимально соответствуют имеющемуся оборудованию и потребностям. Это если коллектив педагогов и руководителей готовы к такой деятельности.

К примеру, на данный момент СПО только частично вводится в воспитательно-образовательный процесс образовательных организаций ПО Кемеровской области, при этом наши исследования показали, что уже 70% учреждений готовы внедрять СПО, 76% педагогов, ответственных за информатизацию в своих учреждениях, имеют опыт использования СПО; ведется работа по адаптации существующих Windows-ориентированных электронных изданий учебного назначения, накопленного контента на свободную платформу. Но большинство учреждений ПО не используют свободное программное обеспечение в своей деятельности, хотя имеют огромную по-

требность из-за ограничений в финансировании, по причине отсутствия педагогов, способных использовать новые продукты.

К настоящему времени за счет принятых мер усилиями в Кемеровской области (целенаправленная подготовка, дистанционное обучение, разработка электронных изданий учебного назначения) удалось существенно повысить уровень ИКТ-компетентности педагогов в области коммерческих продуктов. В среднем 75–80% преподавателей образовательных организаций ПО активно используют средства ИКТ и программные продукты в своей профессиональной деятельности. Однако СПО используют не более 10% педагогов. Соответственно в настоящее время наиболее актуальной становится проблема подготовки (переподготовки) ППР к деятельности в новой среде.

Необходимость решения данной проблемы была подтверждена на федеральном уровне соответствующим постановлением Правительства РФ № 2299 от 17.12.2010 года по переходу федеральных органов исполнительной власти и федеральных бюджетных учреждений на использование свободного программного обеспечения к 2015 году. План, в частности, предусматривает формирование пакета базового свободного программного обеспечения для решения типовых задач деятельности федеральных органов исполнительной власти и внедрение его в госорганах и подведомственных бюджетных учреждениях, ежеквартальное обновление, создание единого репозитория СПО, используемого в федеральных органах власти. Сложность решения данной задачи в регионе обусловлена следующим.

1. Вопросы использования СПО в общеобразовательных учреждениях уже проработаны по ряду направлениям: педагогическом, экономическом, психологическом, методическом, организационном, что позволяет школам использовать данные продукты, хотя бы в рамках преподавания информатики. Но использование СПО в учреждениях ПО (в первую очередь среднего профессионального) проработано крайне слабо. С другой стороны это отставание не критичное — чаще всего СПО в школьной среде используется эпизодически, в рамках изучения информатики. При этом деятельность руководящего и педагогического состава вновь основана на Windows-ориентированных программах. Долгое время образовательные, обучающие и тестовые программы создавались исключительно для платформы Windows с использованием несвободных компонентов. Некоторая часть данных продуктов пригодна для использования с пакетом СПО, однако в целом необходимо проведение работ по переносу накопленного контента на свободную платформу, анализ лицензионного отягощения имеющихся программ.

2.Программы повышения квалификации и переподготовки педагогов учреждений ПО также ориентированы на использование Windows-ориентированных продуктов и технологий. Данные программы, как и раз-

личные организационные структуры, их продвигающие, прочно заняли место на рынке образовательных и консультационных услуг по всей территории страны. Более того, как показал опыт первых занятий в среде Linux, аудитория слушателей не активно идет по пути их освоения.

3. Отсутствует механизм, способный обеспечить внедрение СПО учреждения ПО, и методологии оценки компетенций педагогов, административных и технических работников в отношении внедрения и использования СПО.

4. Сложность заключается и в том, что СПО не означает, что оно является полностью бесплатным. Часть продуктов придется все равно приобретать. Кроме того базовый предмет «Информатика и ИКТ», входящий в состав типовой школьной программы, пока основывается на изучении, как правило, Windows-ориентированных программ. Внедряемые и рекомендуемые электронные УМК, также чаще всего предполагают использование коммерческой платформы. Подобное решение напрашивается когда заходит речь и о специфических (например, бухгалтерских программ, или САПР). Значит, закупку проприентарного программного обеспечения придется все равно производить. В тоже время имеются и положительные моменты — поставляемые информационных системы на основе «1С-Колледж» уже являются кроссплатформенными. Уже имеется опыт установки и использования данной системы на основе GNU/Linux

5.Имеются и некоторые объективные недостатки, к которым следует отнести ограниченность некоторых продуктов СПО в функциональном плане, отсутствие отдельных классов программ или их малая распространенность, отсутствие поддержки данных продуктов со стороны крупных производителей программного обеспечения. Но на наш взгляд, в первую очередь требуется преодолеть психологический барьер и определенный консерватизм пользователей, а также не высокий уровень информационной грамотности, характерный для субъектов образовательного пространства.

Ведь вся проблема в том, что до сих пор в ОУ учат не пользоваться компьютером, а пользоваться одной конкретной операционной системой. И оказываясь один на один с любой другой операционной системой, обучающийся, рабочий, специалист понимает, что он ничего не умеет, кроме как нажимать определенную последовательность кнопок в определенном интерфейсе. Именно в этой области необходимо приложение определенных усилий и представителей образования, и представителей органов власти — следует решить, хотим ли мы чтобы наши выпускники умели обращаться с вычислительной техникой, управлять ею, или чтобы они были придатком к определенному набору программного обеспечения, без которого они не смогут управлять ПК (особенно если это программное обеспечение иностранного производства). И начинать необходимо с педагогов.

Итак, не смотря на то, что переход профессионального образования на СПО преждевременен, мероприятия обеспечения перехода следует проводить, особо активизировав направления: обеспечения доступа педагогического сообщества к объективной и полной информации о существующем СПО и возможностях его использования; тиражирование и распространение опыта по его использованию учреждениями ПО; разработку программных продуктов, позволяющих в среде GNU/Linux создавать электронных образовательные ресурсы (электронные пособия, тесты, обучающие модули). Сделать предстоит еще очень много, достаточно вспомнить, сколько лет и сил ушло на то, чтобы научить педагогов использованию ИКТ на основе Windows-программ. В сложившейся ситуации необходимо в первую очередь продолжать повышение квалификации работников образования в области ИКТ, продумать механизм информирования о существующих разработках, а также разработать ряд модульных методик, программ и курсов, направленных на более активное использование свободно распространяемого ПО.

П.Перцев, М.Губин
Елец, Центр СПО ЕГУ им. И.А.Бунина
http://fosscenter.elsu.ru

Опыт использования расширений языка Python для решения математических задач

Аннотация

При решении математических задач на ЭВМ обычно используются так называемые системы компьютерной математики. Однако, в своем большинстве это громоздкие и не совсем бесплатные программы (конечно, и среди математических пакетов есть прекрасные программные продукты, такие как, Maxima, Octave и ряд других). Авторы, наряду с такого рода программными пакетами, используют в ряде математических дисциплин и язык программирования Python, который практически ни в чем не уступает специализированным программам.

Python — высокоуровневый язык программирования общего назначения с акцентом на производительность разработчика и читаемость кода. Многие авторы [3], отмечая его достоинства, заметили, что язык *Python* свободный, интерпретируемый, расширяемый, встраиваемый, объектно-ориентированный и очень высокого уровня.

Одной из сильнейших сторон языка *Python* является обширная стандартная библиотека и огромное количество сторонних модулей, в том числе и для решения математических задач. От стандартных модулей *math*

и *cmath*, содержащих функции действительных и комплексных аргументов соответственно, до, например, сторонней библиотеки *NumPy*, позиционируемой как свободный и более мощный эквивалент *MATLAB* [2].

Модуль *math* подробно рассматривался многими авторами, например [2,3], однако можно отметить, что *Python* работает с комплексными числами «из коробки», и мнимая часть записывается через j, вместо i в математике.

Уже упомянутая библиотека *NumPy* имеет инструменты для работы с многомерными массивами и полиномами, реализует преобразование Фурье, функции линейной алгебры и множество других возможностей.

Например, вычисление значения некоторого определенного интеграла будет выглядеть так:

```
from scipy import integrate
a, b = -4, 4
x2 = lambda x: x**2
print integrate.quad(x2, a, b)
```

Результатом будет значение определенного интеграла и абсолютной погрешности: (42.666666666666664, 4.736951571734001e-13)

Также следует отметить пакеты, дополняющие и расширяющие возможности *NumPy* — *Matplotlib* и *SciPy*, в частности *Matplotlib*, — это библиотека для построения графиков и визуализации данных [1]. Кроме большого количества типов графиков, которые можно построить с помощью этого пакета, приятной особенностью *Matplotlib* является то, что функции для построения графиков напоминают функции *MATLAB*. Например, нижеследующий код производит отображение простого графика:

```
from matplotlib import pyplot
from numpy import arange
x = arange(-4, 4, 0.1)
x2 =[z**2 for z in x]
pyplot.plot(x, x2)
pyplot.show()
```

SciPy — целая экосистема программ для математики, естественных наук и инженерии. Она включает в себя непосредственно библиотеку *SciPy*, содержащую инструменты для интегрирования, оптимизации, решения дифференциальных уравнений и других научных вычислений, так и уже упомянутые *NumPy* и *Matplotlib*, расширяющие её возможности. Помимо этого *SciPy* включает в себя пакеты символьных вычислений *SymPy*, анализа данных *pandas*, расширенную интерактивную оболочку *IPython* и ряд других инструментов, делающих *SciPy* мощным продуктом для высокопроизводительных математических вычислений.

Например, интегрирование функции из первого примера

```
from sympy import *
x = symbols('x')
print integrate(x**2, x)
```

вернет первообразную `x**3/3`

Или возможно решение алгебраического уравнения:

$$x + y = 3,$$
$$x^2 + 2y^2 - xy + 2x - 3y = 3$$

```
from sympy import *
x, y = symbols('x y')
print solve([x + y - 3,x**2 + 2*y**2 - x*y + 2*x - 3*y - 3],[x, y])
```

Знакомая пользователям *MATLAB* функция вернет список с решениями системы: [(1, 2), (3/2, 3/2)]

Следует отметить, что все рассматриваемые библиотеки распространяются свободно — *NumPy* и *SciPy* имеют лицензию *BSD*, а *Matplotlib BSD*–подобную лицензию. Кроме того, данное программное обеспечение является кроссплатформенным, что является немаловажным при их использовании в вузах и школах, где до сих пор нет точной определенности по использованию конкретных операционных систем и программных пакетов в образовательном процессе.

В целом, *Python* с перечисленными библиотеками можно относительно легко применять в учебных заведениях: на внеклассных занятиях по математическим дисциплинам в школах или при ведении дисциплин математическго профиля в высших учебных заведениях, например, такой дисциплины как «Компьютерная математика», где авторы и использовали данные наработки.

Однако, на наш взгляд, вряд ли стоит рекомендовать *Python* для изучения в качестве первого языка программирования — именно в качестве среды для проектирования программ. Язык с виду кажется простым, но требует некоторого опыта для четкого понимания принципов работы некоторых конструкций [4]. Но, следует признать и тот факт, что *Python* идеально подходит для решения многих прикладных задач, поэтому его вполне можно использовать после ознакомления с основами программирования на более «легких» языках.

Литература

[1] Ильин Е.В. *Математический Python.* `http://jenyay.net/Programming/PyMath`

[2] Лутц М. *Изучаем Python*, 3-е издание. — СПб.: Символ–Плюс, 2009. — 848 с.

[3] Россум Г. и др. *Язык программирования Python*. — 2001. — 454 с.

[4] Сукин И.А. *Python, проглатывающий слона*. — Информатика, 2012, № 2. — С.22–42.

Татьяна Губина
Елец, ФГБОУ ВПО «Елецкий государственный университет им.И.А.Бунина»

Проект: Виртуальная распределенная кафедра «Информационные технологии на основе программного обеспечения с открытым кодом» в составе национального виртуального университета IT-образования(совместный проект с ВМК МГУ им.М.В. Ломоносова) http://dist.elsu.ru/

Программно-методическое обеспечение курса по изучению пакета LibreOffice

Аннотация

В настоящее время актуальным является вопрос подготовки будущих учителей к использованию не столько проприетарного программного обеспечения, а сколько их способность и готовность к использованию свободно распространяемого программного обеспечения. Как показывает опыт, большую часть профессиональных задач учитель может решить с помощью обычного офисного пакета. В статье приводится опыт организации учебного процесса по формированию компетентности бакалавров педагогического образования в области офисных технологий на примере пакета LibreOffice.

В последние годы многие школы страны переходят на использование некоммерческого программного обеспечения. Это вызвано несколькими причинами.

Во-первых, в настоящее время наблюдается тенденция к сокращению бюджетного финансирования сферы образования, в том числе и в целях приобретения лицензионного программного обеспечения. Большинство региональных менеджеров-управленцев образования и руководителей образовательных учреждений, в свою очередь, стараются сэкономить на этом деньги, то есть осуществляется некое давление «сверху».

Во-вторых, во многих вузах страны эффективно внедряется некоммерческое ПО. Среди выпускников вузов появляется все больше учителей, готовых и способных использовать свободное программное обеспечение в современной информационно-образовательной среде школы.

В Елецком государственном университете накоплен положительный опыт организации учебного процесса на базе некоммерческого программного обеспечения, в том числе и свободно распространяемого. В составе национального виртуального университета IT-образования (совместный проект с ВМК МГУ им. М.В. Ломоносова) работает виртуальная распределенная кафедра «Информационные технологии на основе программного обеспечения с открытым кодом» на базе Центра свободного программного обеспечения. Автором статьи проведена апробация курсов «Работа в офисных пакетах», «Информационные технологии», «Информационные технологии в математике», базирующихся на использовании свободного программного обеспечения для решения профессионально-ориентированных задач обучающимися.

Курс «Работа в офисных пакетах» является факультативным и преподаётся бакалаврам педагогического образования на первом курсе. Базовым офисным пакетом является бесплатный и имеющий открытый исходный код пакет LibreOffice, разрабатываемый некоммерческой организацией The Document Foundation.

Для организации учебного процесса на сайте `http://dist.elsu.ru/` имеется дистанционный курс, включающий в себя:

- требования к уровню освоения содержания курса;
- темы и их содержание;
- понятийно-терминологический аппарат;
- лекционный материал;
- теоретические задания;
- содержание лабораторных работ;
- задания для самостоятельной работы к лабораторным занятиям;
- вопросы для самоконтроля;
- промежуточные и итоговый тесты;
- дополнительные материалы (подготовленные файлы для выполнения практических заданий, изображения и пр.).

Курс разбит на 7 модулей.

Модуль 1. Общие сведения о пакете офисных приложений LibreOffice

При изучении данного модуля большое внимание уделяется настройке графического интерфейса пакета в целом и его отдельных компонент, а также установке пакетов расширений.

Модуль 2. Текстовый процессор Writer

Основной акцент делается на автоматизацию работы по созданию интегрированных текстовых документов: добавление колонтитулов, оглавлений, списка библиографии, нумерованных таблиц, иллюстраций, схем, сносок, ссылок и пр., а также работе с составными документами.

Модуль 3. Редактор математических формул Math

Большое внимание уделяется изменению атрибутов форматирования внутри формулы в ручном режиме и правилам набора сложных математических формул. Кроме того, дополнительно рассматривается пакет расширений Dmaths как дополнение к возможностям Math для набора различных математических объектов, в том числе и формул.

Модуль 4. Электронная таблица Calc

При изучении данного модуля большое внимание уделяется решению типовых задач: вычислениям по заданным формулам, построению диаграмм и графиков функций, работе с массивами, работе с функциями «Подбор параметра» и «Поиск решения», фильтрации и сортировке данных, работе со сводными таблицами, а также работе с макросами.

Модуль 5. Система управления базами данных Base

Большое внимание уделяется основным этапам создания баз данных, связыванию данных, конструированию запросов в режиме мастера и с использованием языка SQL.

Модуль 6. Пакет подготовки компьютерных презентаций Impress

Основной акцент делается на создании динамических презентаций, а также экспорту готовой презентации в форматы pdf и swf, вопросам сжатия размеров презентации.

Модуль 7. Редактор векторной графики Draw

Основное внимание уделяется технологии создания сложных графических изображений на основе примитивов, рисованию функциональных схем и чертежей.

Каждая лабораторная работа и задание для самостоятельной работы, включая теоретическое, сопровождаются организационно-методическими указаниями по их выполнению: цель задания, сроки выполнения задания, максимальное количество баллов за задание и разбалловка, основные требования к результатам работы, время, необходимое для выполнения задания.

Каждое выполненное задание — аудиторное и внеаудиторное — оценивается баллами, которые суммируются в общем зачете в процессе изучения офисного пакета. На каждом текущем этапе студент может контролировать сумму набранных баллов и процент выполненных заданий, что позволяет ему самостоятельно осуществлять оценку усвоенных знаний, сформированных умений и навыков.

В настоящее время опубликовано учебно-методическое пособие «Пакет офисных приложений LibreOffice: задания для самостоятельной работы» [1], готовится к изданию пособие «Пакет офисных приложений LibreOffice: лабораторный практикум».

Литература

[1] Губина Т.Н., Карасева Е.И., Пакет офисных приложений LibreOffice: задания для самостоятельной работы: учебно-методическое пособие. — Елец: ЕГУ им.И.А. Бунина, 2013.

Юрий Лебедев
Санкт-Петербург, Институт международного бизнеса и права

Создание свободного ПО для подготовки приложений к дипломам о высшем образовании

Аннотация

Обзор проблемы создания приложений к дипломам о высшем образовании, анализ существующих решений, выявление основных недостатков, формирование требований к разрабатываемому приложению на основе сделанного анализа с учетом использования свободного программного обеспечения.

Высшие учебные заведения часто сталкиваются с тем, что такая простая операция как заполнение и формирование приложения к диплому превращается в трудновыполнимую, а зачастую и дорогостоящую задачу.

Проблемы, возникающие при формировании приложения, часто связаны с высокой стоимостью решений и с трудностью выбора того или иного программного обеспечения. Большинство предлагаемых продуктов предполагают приобретение лицензии только на один компьютер, а это может вызывать некоторые трудности, если вдруг оборудование вышло из строя или отсутствует необходимый доступ.

Одним из способов решения данной проблемы может стать разработка приложения данного направления на основе свободного программного обеспечения. Такое приложение позволило бы получать доступ к дистрибутиву неограниченному количеству пользователей, а также предоставило бы необходимую гибкость настройки под каждое конкретное высшее учебное заведение.

Для того чтобы полностью понять ситуацию, которая существует в данном сегменте, достаточно рассмотреть некоторые предлагаемые решения.

Такими решениями стали «КиберДиплом», «Бюротика: Диплом», ПО «Диплом».

Основными недостатками, которые можно выделить в данных решениях стали:

- Невозможность создания шаблонов приложений;
- Невозможность заполнения однотипными данными целой группы студентов (список дисциплин);
- Высокая стоимость лицензии;
- Невозможность дополнения или изменения существующего кода программы.

Основываясь на сделанном анализе и выявленных недостатках, было принято решение о создании программного обеспечения для подготовки приложений к дипломам, удовлетворяющего следующим требованиям:

- Использование свободного программного обеспечения, такого как OpenOffice.org. и PostgreSql;
- Обеспечить создание сообщества поддержки пользователей и секретарей высших учебных заведений;
- Обеспечить возможность гибкой настройки приложения под нужды каждого из пользователей;
- Реализовать возможность создания и редактирования шаблонов приложений к дипломам;
- Возможность экспорта данных в популярные текстовые форматы;
- Возможность импорта данных из текстовых форматов или базы данных;
- Реализовать возможность быстрой обработки однотипных данных для группы объектов;
- Использование средств и языка программирования, позволяющих использовать приложение в среде любой операционной системы;
- Строгое соблюдение необходимых правил оформления документов, в соответствии с действующим законодательством;
- Удобство понимания кода сторонними разработчиками, множество комментариев;
- Бесплатность.

Таким образом, разработанное приложение должно стать достойной бесплатной альтернативой существующим платным продуктам. Оно обладает равными им, а местами даже превосходящими их возможностями создания, подготовки и формирования приложений к дипломам о высшем образовании.