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

Десятая конференция «Свободное программное обеспечение в высшей школе»

Переславль, 24–25 января 2015 года

Тезисы докладов

Москва,
Альт Линукс,
2015

Десятая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переславль, 24–25 января 2015 года. М.: Альт Линукс, 2015. — 100 с. : ил.

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

ISBN 978-5-905167-18-8

Программа конференции

24 января

11.00-12.00 Регистрация участников и заселение

Дневное заседание
12.00–13.40

12.00 А. Е. Новодворский. Открытие. Информация оргкомитета

12.10 С. М. Абрамов, член кор. РАН. Приветственное слово

12.20–13.00 Н. Н. Непейвода, проф. и др.

Работа с числами в системах счисления с перекрытием и с переносами 7

13.00–13.40 Е. А. Роганов

Система управления учебным процессом и единая образовательная среда Московского государственного индустриального университета 10

13.40–15.00 Перерыв на обед

Вечернее заседание
15.00–18.20

15.00–15.30 Д. А. Костюк и др.

Построение практикумов по программированию встраиваемых систем 14

15.30–16.00 Г. В. Курячий

Кризис UNIX way и фундаментальное IT-образование 18

16.00–16.30 А. Г. Кушниренко

Пять практикумов К. Ю. Полякова по программированию с автоматизированной проверкой в системе КуМир. Результаты переподготовки учителей информатики ... 23

16.30–16.50 Кофе-пауза

16.50–17.20 М. В. Райко, А. Г. Кушниренко

Алгоритмика и программирование для дошкольников и младшеклассников — мировые тенденции и отечественный опыт 26

17.20–17.50 Н. О. Попов, В. В. Яковлев

IDE для изучения Python 30

17.50–18.20 П. Г. Сутырин

Об одном опыте проведения спецкурса по Python с автоматической проверкой домашних заданий 35

18.20 Фуршет по случаю 10-летнего юбилея конференции

25 января

Утреннее заседание
10.00–13.10

10.00–10.15 А. Н. Пустыгин и др.

Построение эквивалентного представления зависимостей в исходном тексте программ с использованием универсального промежуточного представления 39

10.15–10.30 А. Н. Пустыгин и др.

Экспериментальное изучение количественных закономерностей при анализе исходных текстов ПО с открытым кодом по эквивалентным представлениям .. 44

10.30–10.55 И. Ю. Плешкова

Повышение качества поиска в больших объёмах текстовых документов с использованием генетического алгоритма как способ поддержки научных исследований 50

Дневное заседание
14.30–17.40

Вне программы

Н. Н. Непейвода, Е. В. Кочуров, А. А. Демидов, А. Б. Шворин
Переславль-Залесский, ИПС РАН

Работа с числами в системах счисления с перекрытием и с переносами

Аннотация

В связи с посткремниевыми вычислениями актуальной стала задача устранения где возможно «стены памяти»: построение вычислений без хранения промежуточных результатов. Алгоритмы и представления данных, полученные при этом, зачастую могут быть использованы в обычном программировании. Здесь представлены первые результаты работы с принципиально новыми представлениями действительных чисел, разработанными и развитыми за последние полгода группой энтузиастов из ИПС.

Известно, что традиционное позиционное представление действительных чисел некорректно с конструктивной точки зрения (даже сложение невычислимо), а для целых чисел не позволяет ограничить глубину переносов, что требует их запоминания и вызывает эффект стены памяти [1, 2, 3, 4, 5]. В работе [6] предложено представление действительных чисел с перекрытиями, оно развито в [7, 8].

При представлении чисел с перекрытием [8] диапазоны значений соседних цифр перекрываются на долю ε. Таким образом, система характеризуется основанием и долей перекрытия. Традиционные позиционные системы — частный случай с перекрытием 0.

Рис. 1: Разбиение диапазона на цифры

В работе [8] доказано, что оптимальной системой с перекрытием является система «три половинки» $\langle 3, 1/2\rangle$. В ней сложение производится за 1–2 такта независимо от разрядности слагаемых, умножение за $\log_2 n$ тактов, где n — разрядность сомножителей, и тоже может быть реализовано без стены памяти. При сложении достаточно рассмотреть два знака слагаемых для получения одного знака суммы. Приведем для примера формулу А. Б. Шворина для вычисления знака суммы по двум знакам слагаемых, оставив за кадром конкретные алгоритмические детали выбора некоторых параметров.

Пусть даны $\alpha \sim \overline{a_1 a_2 \dots a_N}$, $\beta \sim \overline{b_1 b_2 \dots b_N}$, и требуется найти представление для $\gamma = \frac{\alpha+\beta}{2} = \overline{c_1 c_2 \dots c_N}$. В такой постановке задачи сумма делится пополам, чтобы результат попал в интервал $[0, 1]$.

Пусть $s_i = a_i + b_i$ Тогда $s_i \in \{0, 1, 2, 3, 4\}$. Общее решение для $n \in \{1, \dots N-1\}$ можно записать в виде:

$$c_n = \frac{s_{n+1} - \sigma_n + 2\sigma_{n-1}}{4},$$

$$\text{где } \sigma_0 = s_1$$

$$\sigma_n = \begin{cases} 0 \text{ или } 4, & \text{при } \kappa_n = 0 \\ \kappa_n, & \text{иначе} \end{cases} \qquad (1.1)$$

$$\kappa_n = (2s_n + s_{n+1}) \bmod 4$$

Это не единственная возможность вычислений, другой алгоритм предложен Е. В. Кочуровым. Для системы «три половинки» А. Б. Шворин и Е. В. Кочуров реализовали алгоритмы сложения, вычитания, умножения и перевода из двоичной в эту систему и обратно.

Системы с перекрытием могут работать и для целых чисел, хотя изначально предназначены для действительных. Поскольку целые числа известны точно, в отличие от действительных, это позволяет оптимизировать алгоритмы работы, чем воспользовался Е. В. Кочуров. Далее, он предложил представление целых чисел с переполнением и разработал алгоритмы работы с ними и реализовал программно свои алгоритмы действий над целыми числами. При представлении с переполнением в системе с основанием $n > 2$ в каждом разряде цифрами могут быть числа $0, 1, \dots, n+1$. Е. В. Кочуров доказал, что в системе с переполнением сложение производится без стены памяти, причем для вычисления знака суммы опять-таки достаточно двух знаков слагаемых. Он реализовал арифметические действия над целыми

числами произвольной длины в произвольной системе с переполнением как пакет программ.

А. А. Демидов разработал программу, позволяющую сравнить действия всех этих алгоритмов. Эти программы демонстрируются в ходе доклада и предоставляются для свободного использования под GNU лицензией.

Литература

[1] Непейвода Н. Н. *Конструктивная математика: обзор достоинств, недостатков и уроков I* / Логические исследования. Вып. 17. М.: — Спб: 2011, стр. 191–239.

[2] Brouwer L. E. J. *Besitzt jede reele Zahl eine dezimalberuchentwicklung?* // Proc. Acad. Amsterdam, 23, p 949–954.

[3] Успенский В. А. *Лекции о вычислимых функциях* М.: ГИФМЛ, 1960, 492 с.

[4] Banach S., Mazur S. *Sur les fonctions calculables* // Ann. Soc. Pol. de Math., 16 (1937), p. 223

[5] Mazur S. *Computable Analysis* // Rozprawy Matematyczne, volume 33 Warsaw, 1963.

[6] Непейвода Н. Н. *От алгебр программ к алгебраическим вычислениям* // Технологии информатизации профессиональной деятельности 2014 Ижевск 5–9 ноября стр 45–46.

[7] Непейвода Н. Н., Чаусов Ф. Ф. *Алгебраическая парадигма вычислений* // Суперкомпьютеры, №3 (2014) стр. 53–55.

[8] Непейвода Н. Н., Григоревский И. Н., Лилитко Е. П. *О представлении действительных чисел* // Программные системы: теория и приложения: электрон. научн. журн. 2014. Т. 5, № 4 (22) с. 105–120. URL: http://psta.psiras.ru/read/psta2014_4_105-120.pdf

Евгений Роганов
Москва, МГИУ

Система управления учебным процессом и единая образовательная среда Московского государственного индустриального университета

Аннотация

Более 15 лет в МГИУ на основе СПО создавались действующие сейчас система управления университетом и единая образовательная среда. Доклад посвящён описанию некоторых особенностей этих систем и обсуждению как условий, необходимых для успешной реализации подобных проектов, так и их возможных результатов.

Структура информационной системы

Система управления учебным процессом включает в себя:

- корпоративный портал — веб-среду, предоставляющую доступ всем участникам учебного процесса к различным ресурсам системы в соответствии с их привилегиями;
- систему управления основными образовательными программами (ООП) вуза, включающую учебные планы, график учебного процесса, рабочие программы дисциплин и практик, фонды оценочных средств (ФОСы);
- системы «Приёмная комиссия» и «Деканат»;
- систему управления учебной нагрузкой и штатом профессорско-преподавательского состава (ППС) вуза, включая индивидуальные планы работы преподавателей;
- систему составления расписания учебных занятий и управления аудиторным фондом;
- систему «Отдел кадров»;
- портал открытых данных, содержащий информацию о вузе, публикуемую в соответствии с требованиями Федерального закона «Об образовании в Российской Федерации» и постановлением Правительства РФ.

Единая образовательная среда вуза включает в себя:

- образовательный портал с возможностью реализации дистанционных образовательных технологий (ДОТов), используемый в учебном процессе студентов как очной, так и заочной форм обучения;
- портал поддержки образовательного процесса для разработки учебно-методических материалов (контента) и фондов оценочных средств (ФОСов), содержащий средства проведения и анализа контрольных мероприятий;
- механизмы контроля качества образовательного процесса на основе текущих, промежуточных и итоговых аттестаций, включая систему учёта посещаемости учебных занятий, балльно-рейтинговую систему оценок и систему проверки наличия некорректных заимствований в материалах курсовых и выпускных квалификационных работ.

Технологии, программные продукты, веб-ресурсы

Linux[1] — базовая ОС, GitHub[2] — веб-сервис для хостинга IT-проектов, Ruby[3] — язык программирования, Ruby on Rails[4] — среда разработки веб-приложений, PostgreSQL[5] — реляционная СУБД, MongoDB[6] — документо-ориентированная СУБД, CAS[7] — протокол технологии единого входа, Ace[8] — веб-редактор, Pandoc[9] — конвертер документов, Markdown[10] — простой и удобный язык разметки, LanguageTool[11] — программа проверки грамматики и стиля, TeX Live[12] — современный дистрибутив LaTeX, MathJax[13] — библиотека для визуализации математических формул в браузерах, jQuery[14] — библиотека, обеспечивающая взаимодействие JavaScript и HTML, Reveal.js[15] — одно из средств создания HTML-презентаций.

Особенности систем класса ERP для университетов России

ERP-система на предприятии в идеале позволяет решать весь спектр задач управления и учёта: управление финансами, персоналом, производством, логистикой и цепочками поставок, взаимоотношениями с клиентами и поставщиками (CRM, SCM, HRM, KM, CPM и другие подсистемы). Есть много примеров успешного внедрения подобных систем в России, хотя зачастую возникают проблемы, подобные красочно описанным в этой[16] заметке.

Пример корпорации «Галактика» (http://www.galaktika.ru) показывает востребованность ERP-систем для предприятий и эффективность их внедрения. Есть в этой корпорации и отдел, специализирующийся на IT-решениях для учебных заведений. Успешных внедрений этих продуктов в университеты России, однако, практически нет. И не только у «Галактики».

Причин у этого несколько:

- внутренняя противоречивость законодательства, регламентирующего работу вузов;
- очень частые и лавинообразные изменения требований к системе управления, необходимость сопровождения этих изменений в системе автоматизации;
- наличие в учебном заведении специфических (отраслевых) бизнес-процессов;
- уникальность каждого вуза с точки зрения организации и взаимодействия различных процессов в нём;
- отсутствие чётко заданной целевой функции, которую необходимо минимизировать (максимизировать) в результате управления;
- никогда не прекращающаяся модификация кода системы управления — основная особенность ERP-системы вуза.

Ядро системы должно обеспечивать обработку всех необходимых структур и процессов, специфичных для учебного заведения, — обычная ERP-система для предприятий, к которой «сбоку что-то приделано», не годится! Стандартный цикл сопровождения системы «постановка задачи — составление ТЗ — реализация (программирование) — внедрение» для вузовской ERP не подходит!

Условия успешного создания системы класса ERP и «побочные» эффекты

Для создания и внедрения подобной системы необходимы:

- реальное единоначалие в вузе и наличие достаточно чётко сформулированных правил организации всех бизнес-процессов;
- сознательное и активное участие высшего руководства вуза в процессе создания/внедрения ERP-системы;

- наличие среди сотрудников вуза аналитиков-программистов, занимающихся внедрением и сопровождением приобретённой системы управления или созданием, внедрением и сопровождением ими же разработанной системы.

Опыт МГИУ доказывает эффективность подхода, при котором система разрабатывается, внедряется и поддерживается командой сотрудников университета. Важным «побочным» эффектом, достигаемым при этом, является создание коллектива единомышленников, сочетающих реальную программистскую деятельность с обучением студентов программированию.

МГИУ — единственный университет в Москве, в котором язык Ruby изучается студентами уже более десяти лет, среда разработки Ruby on Rails — пятый год, а количество преподавателей, активно занимающихся реальным программированием с использованием указанного инструментария, превышает 10 человек. Наши выпускники работают не только в российских, но и совместных и зарубежных компаниях, например, в отделе операционных рисков Сбербанка России, Дойче Банке, Внешэкономбанке, отделе логистики информационного центра нефтегазовой компании «Роснефть». Программисты из МГИУ работают в Рамблере, Яндексе, Google ...

Литература

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

[2] GitHub. `https://github.com`

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

[4] Web framework. `http://rubyonrails.org`

[5] PostgresSQL. `http://www.postgresql.org`

[6] MongoDB. `http://www.mongodb.org`

[7] CAS Protocol. `http://jasig.github.io/cas/4.0.x/protocol/CAS-Protocol.html`

[8] Ace Editor. `http://ace.c9.io`

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

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

[11] Proofreading software. `https://www.languagetool.org`

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

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

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

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

[16] Внедрение ERP системы в 3-х актах, с прологом и эпилогом. http://habrahabr.ru/post/102860

Костюк Д. А., Кутень И. С., Пушкин А. Л., Разумейчик В. С.
Брест, Беларусь, Брестский государственный технический университет, Дизайн-центр электроники Promwad

Построение практикумов по программированию встраиваемых систем

Аннотация

Рассматриваются особенности применения процессоров семейства ARM и системы GNU/Linux для обучения студентов программированию встраиваемых электронных систем. Проанализирована направленность и особенности практических курсов, предлагаемых в БГУ и БрГТУ. Обсуждается последовательность практических заданий. Рассмотрены наборы инструментальных программных средств, включая средства разработки, отладки и виртуализации.

Введение

Актуальность изучения студентами-программистами принципов разработки для встраиваемых систем (embedded systems) на основе свободного ПО показывала в последние годы устойчивый рост, связанный с очередным количественным скачком внедрения интеллектуальных электронных систем на базе универсального микроконтроллерного устройства со специализированной прошивкой. Встраиваемая система не привязана к существующей инфраструктуре прикладных программ, а потому разработчикам проще обосновать выбор свободного набора инструментов системного ПО с его модульностью и масштабируемостью.

Этот рост привёл к появлению учебных курсов по программированию встраиваемых систем на основе свободного ПО в Белорусском государственном университете и в Брестском государственном техническом университете. Оба курса имеют прикладную направленность, однако рассматривают предмет на разных уровнях аппаратной абстракции, что делает интересным их комплексное сравнение.

Специфика практикумов

Практикум по разработке встраиваемых систем на факультете прикладной математики БГУ построен на базе опыта многолетней разработки подобных устройств в компании Promwad [1]. Вводная часть касается особенностей использования консоли GNU/Linux, базовых утилит, а также git (последнее важно из практических соображений, т. к. системы контроля версий, при их востребованности, присутствуют в учебном процессе весьма редко). При выполнении работ студенты используют язык C и, в ограниченном объеме, shell. По мере освоения необходимого материала уделяется внимание созданию модулей и взаимодействию с подсистемами ядра.

Учебный курс БрГТУ больше нацелен на особенности системы команд ARM и низкоуровневую организацию встраиваемых систем. Среди причин — исходная ориентация на специальность «Промышленная электроника» (с дополнительно адаптированным вариантом для практико-ориентированной магистратуры), а также уже имеющийся отдельный практикум по разработке модулей ядра Linux. Исходно курс предполагает как запуск ассемблерных программ непосредственно на процессоре ARM, так и выполнение программы под управлением ОС, запущенной на ARM-устройстве.

Выбор аппаратной платформы

При разработке практикумов в качестве платформы были предсказуемо выбраны микроконтроллеры семейства ARM.

В практикуме по разработке встраиваемых систем в БГУ в качестве основной аппаратной платформы планировалось использовать комплекты BeagleBone Black [2] (ARM Cortex A8). Однако в последствии эта идея была отложена в пользу x86. Во-первых, это дало студентам возможность работать самостоятельно, а во-вторых неслож-

ная адаптация заданий практикума сделала исходный код всех работ практически полностью аппаратно-независимым.

Принцип работы с ассемблерной программой в курсе БрГТУ предполагал ее доводку в эмуляторе, а для некоторых работ — тестирование на реальном оборудовании (отладочные платы Atmel на базе ARM9). Однако на практике использование эмулятора также практически вытеснило исходную аппаратную платформу.

Структура практикумов

Курс БГУ можно разделить на следующие этапы:

- практическое ознакомление со средствами разработки и адаптация к предъявляемым требованиям в ходе написания простейшей консольной программы на C;
- ознакомление с особенностями создания bash-скриптов;
- работа с сокетами, процессами и потоками в C;
- написание серии программ по ядру Linux (программирование char-, sys-, dev-, proc-устройств, взаимодействие с таймерами, прерываниями и т. д.).

Требования, предъявляемые к выполнению лабораторного практикума (а также к курсовому проектированию по данной дисциплине, и далее — к дипломным проектам, имеющим отношение к компании Promwad) включают сборку с помощью make и оформление кода по Linux kernel coding style [3]. Код в обязательном порядке размещается студентами на каком-либо публичном git-хостинге (например, `github.com`) и имеет свободную лицензию.

Основа лабораторных работ в курсе БрГТУ — эмулятор QEMU [4], имитирующий одну из нескольких доступных отладочных плат. Для тестирования программ без ОС оттранслированный код располагается по нулевому адресу в файле-образе, соответствующем объёму памяти отладочной платы. Проверка программы в эмуляторе сводится к анализу содержимого регистров в мониторе QEMU, а также в части работ предусмотрен диалог через QEMU serial console.

Практикум включает:

- знакомство с синтаксисом ассемблера ARM и базовыми средствами кросс-компиляции;

- более детальное изучение инструментария (GAS, make) при сборке многофайлового проекта, работу с листингом, а также изучение особенностей доступа к оперативной памяти устройства;
- изучение вычислительных возможностей процессора ARM и доступа к энергонезависимой FLASH-памяти для хранения результатов;
- изучение системы прерываний, совмещение ассемблерного кода и модуля на C;
- знакомство с базовым набором инструментов для компиляции и установки GNU/Linux на ARM-устройство, изучение принципов удалённой отладки с помощью GDB.

Программные инструменты

Оба практикума ориентированы на использование Sourcery CodeBench ARM (одна из существующих сборок GCC для кросс-компиляции). Адаптация курса БГУ к выполнению заданий на настольных ПК была выполнена подстановкой (на уровне make) компилятора из стандартного toolchain дистрибутива, что не повлекло практически никаких изменений в сборку учебных программ по сравнению с их кросс-компиляцией.

В курсе БрГТУ, помимо упоминавшихся инструментов, для заданий с ОС применён пакет автоматизированной сборки BuildRoot [5], упрощающий подготовку образа флеш-памяти с файловой системой, ядром Linux, загрузчиком uBoot и BusyBox в роли минимального набора системных утилит. При этом можно заметить, что сам набор утилит, задействованных на предлагаемых встраиваемых системах, в обоих курсах практически совпадает.

Литература

[1] Promwad: контрактная разработка и изготовление электроники. `http://promwad.ru/`

[2] BeagleBone Black. `http://beagleboard.org/BLACK`

[3] Linux kernel coding style. `https://www.kernel.org/doc/Documentation/CodingStyle`

[4] QEMU open source processor emulator. `http://wiki.qemu.org`

[5] Buildroot: Making Embedded Linux Easy. `http://buildroot.uclibc.org/`

Георгий Курячий
Москва, ВМК МГУ, ООО «Альт Линукс»
`http://uneex.ru`, `http://www.altlinux.ru/products/books/os-unix`

Кризис UNIX way и фундаментальное IT-образование

Аннотация

В докладе делается попытка перечислить и систематизировать кардинальные изменения технологических составляющих таких операционных систем. Большинство таких изменений воспринимаются как отказ от «основополагающих принципов» построения ОС, известных как «путь UNIX». Верно ли, что современная вычислительная система должна быть основана на каких-то других, «более современных», принципах? Более десяти лет назад в своей работе «Операционная система UNIX» ([1]) мы предложили обобщённый подход «проективной системы», формулирующий «основные положения» в терминах человеко-машинного взаимодействия, по возможности без описания реализации. По нашему мнению, этот подход (как и противопоставление «процедурной» организации) совершенно не потерял актуальности. Таким образом, смена технологий ставит проблему новой формулировки инвариантов, зато, по видимости, не обесценивает «путь UNIX» и обеспечивает преподавателя несколькими работающими вариантами реализации.

Призрак бродит

Технологические новшества Linux ⇒ кризис Linux ⇒ кризис UNIX way?

То, что было локальным расширением, становится системообразующим. Примеры:

- ‘rwxrwxrwx‘ → MAC, SELinux, . . . ;
- POSIX IPC → DBus;
- process groups (и не только) → cgropus;
- ‘rwxrwxrwx‘ → Разные виды изоляции;

- текстовое журналирование с профилем → полное бинарное журналирование с поиском;
- архитектура «цветочек» (см. рис.1) → архитектура «динамическая сеть»;
  - вариант «sysvinit → systemd»: (системные утилиты = реализация системных вызовов) + оболочка → системные/прикладные службы + (отслеживание статуса + управление);
- кроссплатфоменная клиент-серверная сетевая графическая среда приложений → локальная 3D подсистема с зависимостью от аппаратуры;
  - например, «X.Org → Wayland»
- shell-scripting → C programming;
- . . . .

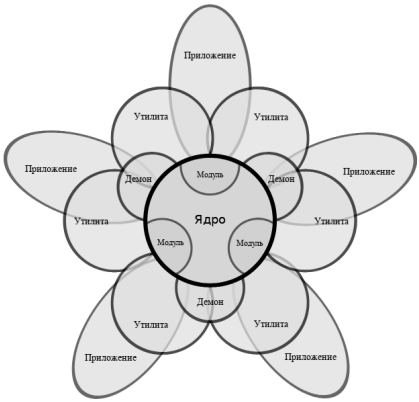


Рис. 1: Архитектура «цветочек».

Тезис: «инварианты»

«Linux — это UNIX сегодня»

«Инвариант» — термин из курса лекций 2011 года (лекция 2) ([3]). Неформально говоря, «что вы встретите в любом Linux-е».

- Консоль: ИКС
- Иерархия файлов
- Процессы, права доступа и суперпользователь
- Текстовые конфигурационные файлы и системная информация
- Оболочка и сценарии
- Пакеты/репозитории
- Свободное сообщество

Антитезис: потеря инвариантности

В действительности от Linux требуется не то, что на самом деле

- Linux $\neq$ ОС $\Rightarrow$ использование только ядра/базового окружения Linux
  - Busybox-based
  - Android
  - Целевые дистрибутивы (прошивки)
  - ...
  - **Симптом**: подавляющая часть данных и управления вытеснена на прикладной уровень
  - **Следствия**:
    - $\Rightarrow$ Интерфейс управления не обязательно CLI (консоль?)
    - $\Rightarrow$ Прикладной интерфейс взаимодействия приложений (оболочка?)
    - $\Rightarrow$ Метаданные не обязаны иметь текстовый вид (конфиги?)
- Профессионализация сообщества разработчиков
  - **Симптом**: роль разработчика/сопровождающего пакет всё дальше от роли пользователя
  - **Следствия**:
    - $\Rightarrow$ «Центры приложений» вместо пакетных диспетчеров (пакеты?)
    - $\Rightarrow$ Депрофессионализация сообщества активных пользователей (сообщество?)

- Динамическое профилирование
  - Временные внешние устройства, в т. ч. «новые»
    - Пример: LKMS, PPD, ...
  - Мобильность: оперативное изменение условий, в т. ч. на «новые»
    - Пример: NM, PolKit, ...
- **Симптом**: не требующая вмешательства пользователя (или минимизирующая таковое) полуавтоматическая адаптация системы к новым условиям
- **Следствия**:
  - ⇒ Не обязательно задавать настройки вручную (конфиги?)
  - ⇒ Недостаточно просто запускать и останавливать задачу, нужно сложное взаимодействие системных и прикладных подсистем (процессы?)
  - ⇒ Множество управляющих действий и совершаются, и обрабатываются приложениями (сценарий?)
- Вопросы масштабирования: множественный деплоймент и обслуживание, непрерывная разработка, ...
  - **Симптом**: ресурсоёмкость разработки становится существенно ниже ресурсоёмкости доставки и сопровождения
  - **Следствия**:
    - ⇒ Скорость работы интегрирующей прослойки имеет значение (оболочка?)
    - ⇒ Проще эффективно переписать, чем комбинировать имеющееся (оболочка?)
  - **Симптом**: упор на сопровождение "окружений" а не отдельных приложений
  - **Следствия**:
    - ⇒ Нет необходимости модифицировать рабочую копию, надо обновлять «прошивку» целиком (пакеты? репозитории?)
    - ⇒ (не только при масштабировании) Сложные уровни изоляции/защиты (процессы? права доступа? root?)

Эту повесть про старушку можно дальше продолжать.

Как возможно фундаментальное образование в области вычислительных систем?

Синтез: инварианты более высокого уровня

«Operating System» переводится как «систематика использования»

Заметим, что все «инварианты linux» — это реализации каких-то более высокоуровневых принципов использования компьютера.

Предложение из 2003-года: принципы организации операционной среды (aka **4U**)

- **И**: принцип информационной открытости
  - для себя
  - для эффективности взаимодействия
  - для свободы ориентации
- **З**: принцип минимизации затрат
  - принятие решений
  - физические действия
  - умственная деятельность
- **У**: принцип умопостижимости контекста
  - human-readable
  - human-writeable
  - 7 +/- 2
- **О**: принцип персональной ответственности
  - захотел — получил
  - достаточность знаний
  - динамическая иерархия

Замечания:

1. Ни один из этих принципов не выглядит устаревшим.
2. Современное состояние ОС оставляет (во всех смыслах слова!) желать лучшего в плане **4U**

Перспективы

Кто виноват? Что делать?

Вариант 1. Рассказывать о различных "реализациях" в надежде на гештальт

Вариант 2. Использовать одну версию ОС в качестве базовой, акцентируя внимание на выполнение (возможно, подновлённой) концепции **4U**

Вариант 3. Написать свою идеальную ОС (не такой уж бессмысленный)

Литература

[1] Курячий Г. В. *Операционная система UNIX* `http://www.altlinux.ru/products/books/os-unix/`

[2] Курячий Г. В., Маслинский К. А. *Операционная система Linux* `http://www.altlinux.ru/products/books/os-linux/`

[3] *Программное обеспечение GNU/Linux* (курс лекций на ВМК МГУ, осенний семестр 2011 года) `https://uneex.ru/LecturesCMC/GnuLinuxSoftware2011`

Кушниренко Анатолий Георгиевич
Москва, Научно-исследовательский институт системных исследований РАН (НИИСИ РАН)

Проект: Учебная система программирования Кумир
`http://www.niisi.ru/kumir/`

Пять практикумов К. Ю. Полякова по программированию с автоматизированной проверкой в системе КуМир. Результаты переподготовки учителей информатики

Аннотация

В докладе будет рассказано о методике подготовки автоматизированных практикумов по программированию в системе КуМир, описана цепочка из 5 практикумов, разработанных К. Ю. Поляковым, и проанализированы отчеты десяти учителей информатики г. Сургута, выполнивших данные практикумы самостоятельно после прослушивания короткого вводного курса. В заключении доклада будут выдвинуты предложения по улучшению методики переподготовки учителей информатики по теме «программирование».

Многоплатформенная свободно распространяемая система программирования КуМир — Комплект Учебных Миров — поддерживает школьный алгоритмический язык с русской лексикой, введенный в 1985 году академиком А. П. Ершовым в момент введения в обязательную программу старшей школы СССР предмета «Основы информатики и вычислительной техники». Система была популярна в 90-е годы прошлого века — годы дефицита вычислительной техники в школах — поскольку с удовлетворительной надежностью и скоростью работала на всех доступных школам аппаратных платформах и позволяла выполнять программы из учебника, изданного к концу века в СССР и России суммарным тиражом около 8 млн. экземпляров. С тех пор много воды утекло, изменились учебные программы, распространенные в школах аппаратные платформы, учебники, сместились акценты в преподавании информатики. Сегодня школьный алгоритмический язык и поддерживающая его система КуМир занимают в отечественном образовании скромную нишу, однако продолжают существовать.

Интерес к языку и системе до последнего времени поддерживался низкими затратами на начальное освоение, богатым набором учебных исполнителей, включая популярного в среде разработчиков вариантов ГИА и ЕГЭ Робота, а также перспективами вхождения языка и системы в список рекомендованных для использования на компьютеризированных государственных аттестациях по информатике.

В последнее время интерес к языку и системе повышает тот факт, что новые государственные стандарты школьного и дошкольного, обязательного и дополнительного образования упоминают задачу развития навыков алгоритмизации, а система КуМир и ее младший брат — ПиктоМир — позволяют построить циклы занятий для развития навыков алгоритмизации продолжительностью от 2–3 до 20–30 часов для самых разных категорий обучаемых. От дошкольников подготовительных групп до старшеклассников, готовящихся к ГИА и ЕГЭ. Еще одна важна категория обучаемых — учителя информатики. От них в новых условиях требуется более высокий, чем прежде, уровень овладения типовыми методами программирования простейших учебных задач в объеме, предусмотренном классификаторами ЕГЭ по информатике последних лет.

Для последних категорий применений система КуМир предлагает педагогам возможности подготовки и эксплуатации практикумов по программированию с автоматизированной проверкой. В этих при-

менениях КуМир выступает как свободно распространяемый инструмент подготовки и использования контента (courseware), права на использование которого определяются разработчиками контента, а не разработчиками КуМира.

В настоящее время в открытом доступе для некоммерческого использования есть «кумировские» практикумы по программированию двух известных педагогов: Д. Кириенко (`http://server.179.ru/wiki/?page=DenisKirienko/Kumir`) и К. Полякова (`http://kpolyakov.narod.ru/school/kumir.htm`).

Идея практикума по программированию в КуМире проста:

Педагог — разработчик практикума — готовит для ученика последовательность заданий, каждое из которых представляет из себя заготовку КуМир-программы. Эта заготовка состоит из фрагментов трех сортов.

Обычные строки — могут редактироваться и удаляться учеником в процессе выполнения задания.

Защищенные абзацы — видимы ученику, но не могут быть им изменены.

Невидимые строки — расположены в конце программы, содержат составленную разработчиком практикума проверяющую программу и вспомогательные подпрограммы, которые разработчик пожелал скрыть от ученика.

Тем самым разработчик практикума вместе с каждым заданием готовит проверяющую программу, которая предназначена для проверки правильности выполнения задания и погружена в тело заготовки задания. Приступая к выполнению очередного задания, ученик работает в системе КуМир, редактируя программу и запуская ее на выполнение обычным порядком. Когда отладка закончена, ученик может проверить правильность выполнения задания, запустив off-line тестирование или послав программу на проверку на сервер учителя.

К. Ю. Поляковым разработан цикл из нескольких практикумов, нацеленный на отработку азов последовательного программирования на процедурных языках: «Робот» — 12 уроков, «Массивы 1» — 6 уроков, «Массивы 2» — 6 уроков, «Строки» — 5 уроков. В каждом уроке предусмотрено несколько заданий разного уровня сложности, начиная с тривиальных упражнений. (Еще один практикум Полякова — «Функции» — в докладе не рассматривается).

Автору доклада представилась возможность ознакомиться с отчетами десяти учителей информатики г. Сургута, выполнивших пере-

численные выше практикумы самостоятельно после прослушивания короткого вводного курса.

Отчеты содержали хронометраж самостоятельного выполнения половины заданий каждого практикума (т.е. от 30 до 40 заданий в каждом практикуме).

По отчетам учителей, выполнение 4-х практикумов занимало от 10 до 28 часов.

По мнению автора доклада, для тех учителей, затраты которых составили 25–30 часов, выбор формы переподготовки в виде самостоятельного прохождения практикумов с автоматической проверкой был ошибочным — для этой категории обучаемых нужна очная форма. Для тех же учителей, затраты которых составили 10–20 часов, решение о проведении переподготовка в форме самостоятельного прохождения практикумов оказалась весьма эффективной. Поскольку затраты на выполнение практикумов легко прогнозировать путем хронометража выполнения первого урока каждого практикума, открывается возможность организации переподготовки учителей информатики по программированию в очно-заочной форме: 6–8 часов знакомства с системой КуМир и практикумами в очной форме и 15–20 часов самостоятельной работы с самопроверкой и сдачей выполненных заданий на официальную проверку.

Кушниренко Анатолий Георгиевич, Райко Миля Вячеславовна

Москва, Научно-исследовательский институт системных исследований РАН (НИИСИ РАН)

Проект: Учебно-игровая среда ПиктоМир `http://www.piktomir.ru/`, `http://www.niisi.ru/kumir/`

Алгоритмика и программирование для дошкольников и младшеклассников — мировые тенденции и отечественный опыт

Аннотация

Понижение возраста освоения элементов алгоритмики детьми стало общемировой тенденцией. Это освоение происходит сегодня во всем

мире: в семье при самостоятельной игре ребенка с бестекстовыми системами программирования или в рамках короткого цикла дополнительных занятий в дошкольном учреждении. В докладе будет рассказано, как накопить в дошкольный период опыт бестекстового «детского» программирования виртуальными роботами в системе ПиктоМир, используя подготовленную авторами свободно распространяемую 100-страничную методичку с «поминутным» планированием и использовать этот опыт в начальной школе, переходя ко «взрослому», текстовому программированию реального робота в системе КуМир.

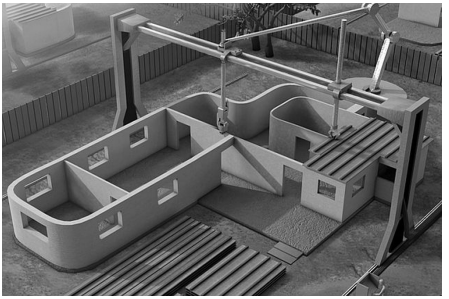
Явления, которые мы будем сегодня обсуждать, должны рассматривать в очень широком контексте освоения нашей цивилизацией результатов развития ИКТ-технологий. Начнем с фактов.

1. Число переключательных элементов в современном микропроцессоре — ключевом элементе ИКТ-технологий — приближается к числу нейронов человеческого мозга.
2. На каждого жителя Земли сегодня приходится до десятка микропроцессоров и микросхем соизмеримой сложности.
3. Компьютеры начинают успешно конкурировать и обгонять человека во многих областях, ранее считавшихся доступными только человеку. Примерами могут служить: игра в шахматы, вождение автомобиля по улицам и дорогам Америки, «гуманитарные» интеллектуальные игры эрудитов типа «Что-где-когда».

Приведенные выше три факта относятся к интеллектуальной деятельности человека. Но и в сфере материального производства компьютеры начинают радикально менять вид технологической цепочки от возникновения замысла до его материального воплощения. Если в конце XX века человечество успешно справилось с задачей «цифровизации» и автоматизации процессов обработки информации, то в начале XXI века начались «цифровизация» и «дискретизация» процессов материального производства. В XXI веке слово *сделать* все чаще будет означать *запрограммировать*. Уже сегодня достаточно составить нужную программу для подходящего 3D-принтера и будет создан (напечатан) нужный физический объект: настенный календарь, шкив электрогенератора автомобиля, пластмассовый пистолет, микропроцессор, или даже жилой дом.

В свете приведенных фактов меняются и представления землян о задачах системы образования в сфере ИКТ: раз современ-

ное мировое хозяйство все больше «завязано» на информационно-коммуникационные технологии, то без подготовки корпуса кадров, свободно ориентирующихся в процессах информатизации, современное мировое хозяйство не сможет выживать и развиваться.



Первая реакция системы образования на усиление роли ИКТ-технологий — резкое понижение возраста знакомства подрастающего поколения с основными понятиями ИКТ. Подобное понижение возраста освоения наиболее важных для нашей цивилизации понятий в истории уже случалось.

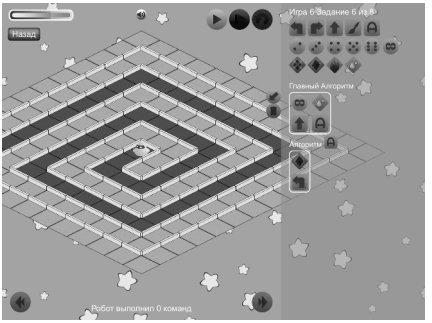
Напомним, что 400 лет назад самая важная отрасль математики — **арифметика** — изучалась в университетах, а сегодня — изучается в первом классе и предварительное знакомство начинается еще до школы. Информатика переместилась из университетов в начальную школу гораздо быстрее. Самая важная отрасль информатики — **алгоритмика** — 40 лет назад изучалась в университетах, сегодня — в

первом классе и предварительное знакомство начинается еще до школы.

Понижение возрастного порога стало возможно после освоения педагогами и программистами бестекстовой методики программирования — представление программы не в виде текстов, а в виде расположенных в пространстве материальных объектов, символизирующих команды, отдаваемые реальному или виртуальному роботу или расположенных на экране компьютера пиктограммах этих команд.

На основе этой методике за последние 5 лет в мире был создан ряд игровых и учебно-игровых систем. Представителем этих систем в России является многоплатформенная, свободно распространяемая учебно-игровая среда ПиктоМир разработки НИИСИ РАН, см. [1–3] и сайт `http://www.piktomir.ru/`.

Наряду с собственно программной системой в НИИСИ РАН была разработана и методика ее использования при проведении цикла из 11 занятий по теме «Алгоритмика» в подготовительных групп ДОУ. Эта методика подробно описана в 100-страничном методическом пособии, размещенном на том же сайте.



В планах НИИСИ РАН на 2015 год — завершение разработки свободно распространяемого методического и программного обеспечения курса «Алгоритмика для второклассников», состоящего из 3 циклов:

- «Алгоритмика 1» — 10 занятий — бестекстовое программирование в системе ПиктоМир;
- «Алгоритмика 2» — 10 занятий — бестекстовое программирование в системе ПиктоМир и текстовое программирование в системе КуМир;

- «Управляем настоящим Роботом» — 6 занятий — программирование простейших алгоритмов управления Лего-роботом в системе КуМир.

Литература

[1] Rogozhkina I. B., Kushnirenko A. G. *PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment.* — // World Conference of Educational Technology and Researches, July, 2011

[2] Кушниренко А. Г., Леонов А. Г. *Программирование для дошкольников и младших школьников.* — // Информатика. — М.: Первое сент., 2011, N 15. — стр.20–23

[3] Кисловская А.Д., Кушниренко А.Г. *Методика обучения алгоритмической грамоте дошкольников и младших школьников* — // Информационные технологии в обеспечении федеральных государственных образовательных стандартов: Материалы Международной научно-практической конференции. 16-17 июня 2014 года. — Елец: ЕГУ им. И. А. Бунина, 2014. — Т. 2. — стр. 3–7.

Н. О. Попов, В. В. Яковлев
Москва, НИУ Высшая школа экономики

Проект: Кумир-Python `http://gitorious.org/kumir2` (branch:python)

IDE для изучения Python

Аннотация

Язык Python все глубже проникает в российскую систему образования. С 2015 года в контрольно-измерительные материалы ЕГЭ включены примеры программ на Python'е, а на факультете компьютерных наук НИУ ВШЭ именно этот язык является первым изучаемым языком программирования. При изучении программирования на языке Python, как впрочем, и на любом другом языке программирования, немаловажной задачей является обеспечение ученика (студента) подходящей средой разработки (IDE). Учебная IDE должна, с одной стороны, предоставлять всю необходимую функциональность, которая потребуется при решении учебных задач, и в то же время, – быть максимально простой, поскольку является лишь инструментом, а не субъектом изучения. Ввиду отсутствия (на момент начала проекта) подобной IDE для

языка программирования Python, мы начали ее создавать на платформе Кумир-2. Целевой аудиторией Кумир-Python являются старшие классы школы и высшие учебные заведения.

Предпосылки разработки

Среда разработки на языке программирования (в дальнейшем — *IDE*) должна предоставлять возможность редактирования и проверки программ, их выполнения и пошаговой отладки. В качестве IDE можно использовать связку из обычного текстового редактора и терминала, в котором запускаются интерпретатор и инструменты для проверки синтаксиса. Эта комбинация достаточно часто используется при написании небольших программ, однако у этого решения, помимо удобства использования, есть существенный недостаток: нет возможности наглядной пошаговой отладки программ. С другой стороны, существуют полноценные IDE, например PyCharm[1], созданная на базе IntelliJ IDEA, которая обладает всей необходимой функциональностью. Использование подобных сред разработки оправдано в производственных целях, но для учебных задач их возможности не просто избыточны, но и создают существенные неудобства, такие как необходимость освоения среды, или создание и настройка проекта перед тем, как написать даже самую простую программу.

Существуют также специализированные системы, разработанные специально для образовательных целей. Классическими примером такой IDE можно считать систему с открытым исходным кодом Кумир, разрабатываемую в НИИСИ РАН, которая использует учебный Алгоритмический Язык и предназначена для использования в школьном курсе программирования. Эта среда разработки имеет простой интерфейс, редактор кода, совмещенный с анализатором ошибок и возможность пошагового выполнения программ с отображением текущих значений переменных как на полях редактора, так и в отдельном окне. Мы решили использовать наработки и исходный код[2] системы Кумир для создания учебной IDE для языка Python 3.x[3].

Система Кумир спроектирована таким образом, что позволяет заменить русский алгоритмический язык программирования на любой другой[4]. В то же время, можно использовать такие особенности системы, как отображение синтаксических ошибок на полях редактора, или систему курсов[5].

Работа над созданием среды Кумир-Python началась в 2013 году, когда в свободном доступе не было ни одной полнофункциональной среды разработки для Python, которую можно рекомендовать студентам. В конце 2013 года были появилась бесплатная редакция PyCharm Community Edition, а совсем недавно — специализированная редакция PyCharm Educational Edition[6] для образовательных учреждений. Основной особенностью этой редакции является поддержка курсов с возможностью автоматизированной проверки заданий, как и в Кумир. Тем не менее, для использования в школах, даже эта редакция PyCharm является достаточно сложной средой.

Интеграция интерпретатора

В системе Кумир-Python интерпретатор используется в адресном пространстве IDE, а не отдельным процессом. Это позволяет добиться очень тесной интеграции интерпретатора со средой разработки, что, в частности, позволяет реализовать не только полнофункциональный отладчик, но и вмешиваться в работу интерпретатора во время выполнения программ. Это необходимо, например, для написания тестирующих алгоритмов при реализации автоматизированной проверки заданий.

С другой стороны, использование такого подхода накладывает ряд ограничений на возможность использования некоторых сторонних библиотек. Например, становится невозможным использование библиотеки PyQt (PySide) в программах, которые запускаются из IDE, а также появляется риск падения системы при использовании сторонних модулей Python, реализованных на C/C++. Мы считаем, что для учебных задач эти ограничения не существенны, поскольку как использование Qt, так и написание C/C++ модулей для Python, предполагает более высокую квалификацию, чем та, на которую ориентировано начальное обучение.

Отметим, что отдельный экземпляр интерпретатора, именуемый «Песочница» (*от англ.* Sandbox), доступен для использования в качестве калькулятора, или быстрой проверки конструкций языка, без написания программ.

Проверка программ во время редактирования

Язык Python является динамическим языком программирования, поэтому семантический и синтаксический анализ программ для этого языка затруднен. Тем не менее, значительную часть ошибок в программе (но не все ошибки) можно найти с помощью специализированных анализаторов кода.

Для проверки программ мы используем упорядоченный по приоритетам набор различных анализаторов:

1. Анализатор ошибок PyLint[7].
2. Анализатор ошибок PyFlakes[8].
3. Анализатора стиля кода PEP-8[9], использование которого является опциональным (пункт в настройках), поскольку, вывод результатов его работы может испугать любого неподготовленного пользователя. Но в некоторых случаях его использование может быть обязательным формальным требованием.
4. Предполагается реализация дополнительного анализатора, который будет проверять совместимость типов величин в вычисляемых выражениях и аргументах функций. Помимо обнаружения ошибок, результаты работы этого анализатора необходимы для реализации семантического автодополнения кода.

Если несколько анализаторов нашли ошибку, которая относится к одному и тому же участку кода, то принимается та из них, которая найдена анализатором с более высоким приоритетом. Экспериментально было установлено, что наиболее адекватную диагностику выдает анализатор PyLint, поэтому он имеет наивысший приоритет.

Дополнительные возможности

Система Кумир-Python наследует важные для организации процесса обучения возможности системы Кумир. Выделим две такие возможности.

Во-первых, возможно использование графических исполнителей системы Кумир в школьном курсе информатики, как обычных Python-модулей. Единственное отличие от системы программирования Кумир — это использование англоязычных имен модулей и функций. В настоящее время реализованы исполнители Робот (модуль

robot) и Рисователь (модуль painter), работа с которыми описана в школьных учебниках[10, 11, 12].

Во-вторых, реализована поддержка курсов [5]. Автоматическая проверка заданий реализуется функциями __pre_test__, __pre_run__ и __post_run__, которые находятся в не отображаемой системой Кумир «скрытой» части программы. Она отделяется в текстовом файле программы от основной части Python-комментарием специального вида. В процессе проверки заданий можно вмешиваться в работу интерпретатора, например, принудительно изменять значения переменных, или имитировать ввод данных из стандартного потока ввода.

Выводы

В настоящее время система Кумир-Python позволяет редактировать и выполнять программы, использовать пошаговый отладчик и входящие в поставку «исполнители» системы Кумир.

Сообщения используемых анализаторов об ошибках в редактируемых программах отображаются на поля, но пока только на исходном (английском) языке. В дальнейшем предполагается выполнить как перевод этих сообщений на русский язык, так и уточнить их классификацию на предмет того, какие из сообщений являются действительно ошибками, а какие — предупреждения о потенциальных ошибках.

Незавершенными на данный момент остаются части синтаксического анализатора, которые выполняют подсветку синтаксиса, построение списка имен для семантического автодополнения, и проверку совместимости типов величин.

Исходные тексты доступны в git-ветке python репозитория разработки системы Кумир[2]. В комплект поставки входит пример реализации курса по языку Python.

Опытную эксплуатацию среды предполагается начать в 2015–2016 учебном году. Целевой аудиторией Кумир-Python являются старшие классы школы. Ввиду появления свободно доступных редакций PyCharm, целесообразность использования Кумир-Python в высших учебных заведениях остается под вопросом, ответ на который может быть получен по результатам опытной эксплуатации.

Литература

[1] http://jetbrains.ru/products/pycharm/

[2] http://gitorious.org/kumir2

[3] https://www.python.org/

[4] 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.

[5] Д. В. Хачко, Д. П. Кириенко, А. Г. Кушниренко, А. Г. Леонов, М. А. Ройтберг. *Поддержка курсов в системе КуМир.* VI Конференция «Свободное программное обеспечение в высшей школе», М.: Альт Линукс, 2011.

[6] https://www.jetbrains.com/pycharm-educational/

[7] http://www.pylint.org/

[8] https://pypi.python.org/pypi/pyflakes

[9] https://www.python.org/dev/peps/pep-0008/

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

[11] *Информатика. 10 класс. Углубленный уровень. В 2 частях (комплект из 2 книг).* К. Ю. Поляков, Е. А. Еремин. М.: Бином, 2014. 648 с.

[12] *Информатика. 11 класс. Углубленный уровень. Части 1–2 (комплект).* К. Ю. Поляков, Е. А. Еремин. М.: Бином, 2014. 532 с.

Павел Сутырин
Москва, МГУ имени М.В. Ломоносова
http://cs.msu.ru, http://python.cs.msu.ru

Об одном опыте проведения спецкурса по Python с автоматической проверкой домашних заданий

Аннотация

В докладе описывается опыт проведения семестрового спецкурса начального уровня по языку программирования Python (2.7) для студентов (50 чел.) факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова. Применено разделение труда между преподавателями: один читает собственно лекции и готовит домашние задания, другой отвечает за сопровождение тестирующей системы

и подготовку видеозаписей (скринкастов) лекций. Описывается опыт применения олимпиадной системы автоматического тестирования программ (ejudge) для проверки домашних заданий, выполняемых слушателями в течение семестра (100 пользователей, 30 заданий, 8 000 посылок заданий на проверку). Дается обзор задач, которые использовались в курсе, а также методики тестирования программ, использованные для проверки домашних заданий. Проанализированы особенности применения этой системы в контексте спецкурса.

О спецкурсе

В осеннем семестре 2014-2015 учебного года на факультете ВМК МГУ проводился экспериментальный спецкурс по языку программирования Python 2.7. Аудитория спецкурса — студенты 1-5 курсов факультета и несколько слушателей не из МГУ.

Лектор и автор заданий: Георгий Курячий, ассистент: Павел Сутырин.

Лекции проводились в режиме непрерывной демонстрации экрана, на котором большую часть времени велась работа с интерактивным интерпретатором Python: опробование обсуждаемых возможностей языка, иногда написание более крупных программ в виде файлов. Запись экрана со звуком (screencast) через короткое время выкладывалась в открытый доступ.

Для каждой лекции (кроме двух последних) был подготовлен набор из 3-4 заданий на программирование на языке Python, требующих как эффективной алгоритмизации, так и применения инструментария языка.

Для проверки домашних заданий и проведения предэкзаменационного тестирования была использована свободная программная система ejudge (автор: Александр Чернов, ВМК МГУ), широко используемая для проведения олимпиад по программированию (в т.ч. региональных этапов ACM ICPC), для поддержки основного факультетского практикума по программированию (некоторые группы 2 курса: системное программирование под Unix и объектно-ориентированное программирование), а также на других образовательных веб-ресурсах по программированию.

Система была новой для авторов курса, и в качестве одной из метазадач было задумано практическое испытание этой системы для целей преподавания языка программирования.

Автоматическая проверка заданий

Начальный этап освоения системы был облегчен наличием штатного образа виртуальной машины VirtualBox.

На основном факультетском экземпляре системы (`ejudge.cs.msu.ru`), администрируемом самим автором системы, для спецкурса был создан отдельный турнир и выданы ограниченные права доступа для его администрирования через веб-панель (на уровне ejudge), а также заведены ssh-пользователи с доступом в каталоги конфигураций турнира (на уровне ОС). Таким образом, можно назвать эту конфигурацию прикладным shared-хостингом (ejudge-хостингом).

Система ejudge позволяет аутентифицированным (и авторизованным на участие в турнире) пользователям турнира (или спецкурса) отправлять на проверку файлы через веб-интерфейс посредством HTTPS-запросов. Каждое полученное решение запускается в защищенном режиме выполнения (для этих целей автором ejudge также разработан и поддерживается патч к ядру Linux, позволяющий существенно ограничить работу запущенной программы с файловой системой, процессами и т.п.), и его правильность проверяется на наборе формальных тестов. В системе сохраняются все варианты программ, отправленные пользователями на тестирование (т.н. «посылки») вместе с протоколами их тестирования.

Система ejudge позволяет задавать тесты вида «вход-выход» (входные данные подаются в текстовом формате на stdin программе, выходные данные ожидаются на stdout), а также по нашей просьбе автором любезно было добавлено косвенное тестирование: запуск отдельной тестирующей программы, которой в качестве параметра передается имя файла, отправленного на проверку. Второй способ позволил реализовать полноценные модульные тесты, в рамках которых файл решения импортируется как модуль Python и его содержимое (например, классы) эксплуатируется тестирующим кодом на Python.

По итогам семестра работа пользователей с системой оценивалась по следующим критериям:

1. число полностью сданных задач (прошедших все тесты);
2. число полностью и вовремя сданных задач;
3. общее количество ошибочных посылок (не прошедших хотя бы один тест);

Статистика по заданиям

Всего задач:	31
Всего тестов:	124
Зарегистрировалось пользователей:	285
Решило хотя бы одну задачу:	156
Решило 50% задач:	92
Решило 80% задач:	73
Решило 90% задач:	63
Решило 95% задач:	56
Решило 100% задач:	29
Всего посылок	8 430

4. количество посылок, ошибочных на первом тесте (заведомо известном!);
5. количество посылок, превысивших время выполнения (неэффективный алгоритм).

В соответствии с этими критериями были предложены оценки «автоматом».

Таким образом было отсеяно значительное количество слушателей, своевременно и плодотворно трудившихся в семестре. Для остальных был назначен экзамен, которому предшествовало тестирование — еще один турнир в ejudge из 5 задач, подобных семестровым, на 4 часа времени.

Таким образом, из двух наборов студентов («ранних», прослушавших курс в реальном времени, и «поздних» — хорошо знающих Python и желающих получить тематический спецкурс «малой кровью», или же пропустивших лекции) к очному экзамену остались лишь те, чьи знания требуют более детальной беседы, для каковой уже подготовлен рабочий материал — решенные задачи, отправленные программы (вместе с их динамикой развития от посылки к посылке, отражающей ход мысли решающего).

Литература

[1] Г. Курячий, П. Сутырин. *Язык программирования Python (страничка спецкурса)* (`https://uneex.ru/LecturesCMC/PythonIntro2014`)

[2] А. Чернов. *Ejudge contest management system.* (`https://ejudge.ru/`)

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

Построение эквивалентного представления зависимостей в исходном тексте программ с использованием универсального промежуточного представления

Аннотация

Использование высокоуровневого промежуточного представления часто может оказаться полезным в процессе автоматического извлечения знаний из исходных текстов программ. Примером одного из таких анализов является анализ всех существующих зависимостей в программах на различных языках программирования. Описывается эквивалентное представление зависимостей и процесс его построения на основе универсального промежуточного представления.

Введение

В процессе статического анализа программных систем, написанных на нескольких языках программирования, зачастую возникают задачи различного рода и уровней сложности, которые необходимо решать для каждого из используемых в системе языков. Одним из подходов, упрощающих проведение подобных работ, является использование универсального промежуточного представления (УПП) как средства для создания эквивалентных представлений программного кода [1][2]. Примером такого представления является представление зависимостей в исходном тексте, которое уже сейчас находит эффективное применение в средах разработки как один из ключевых элементов рефакторинга кода.

Представление зависимостей

Граф зависимостей — ориентированный граф представляющий собой зависимости нескольких объектов по отношению друг к другу[3].

В задачах статического анализа исходных текстов на объектно-ориентированных языках в качестве объектов выступают следующие сущности:

1. Логические единицы, содержащие скомпилированный код (Исполняемые файлы, разделяемые библиотеки, сборки)
2. Объединения классов (например, пространства имен C++, C#, пакеты Java)
3. Типы данных (классы, интерфейсы)
4. Операции над типами данных (методы)

Данное представление описывает связи между двумя сущностями.

Между ними устанавливаются следующие связи:

1. Наследование — механизм языка, позволяющий описать новый класс на основе уже существующего (родительского, базового) класса. Класс-потомок может добавить собственные методы и свойства, а также пользоваться родительскими методами и свойствами.
2. Агрегация — отношение «часть-целое» между двумя равноправными объектами, когда один объект имеет ссылку на другой объект.
3. Метод — принадлежность метода определённому типу.
4. Передача типа в качестве параметра методу. Ее можно рассматривать как частный случай агрегации, в которой в качестве источника зависимости выступает тип, передаваемый методу в качестве параметра.
5. Вызов метода.

Зависимости в данном представлении могут быть описаны как явные и неявные. Под явными зависимостями понимаются зависимости, которые указаны напрямую в исходном тексте. Пример такой зависимости можно увидеть на листинге 1.

```
class Employee
{
...
\ \ Department m_department;
...
}
```

Листинг 1. Пример явной зависимости в исходном тексте

Под неявными зависимостями понимаются зависимости, которые не представлены явно в исходном тексте, но могут возникнуть в процессе его развития или описаны извне. Например, при передаче интерфейса к качестве параметра методу, возникает неявная зависимость между этим методам и всеми реализациями переданного интерфейса.

Формальное описание зависимостей

Упрощённая модель УПП

Пусть ID — множество всех идентификаторов описанных в УПП. $Names$- множество всех имен узлов из УПП. $Types$- множество всех типов узлов УПП(теги). N — множество всех узлов из УПП исходного текста, $n_i \in (ID_{n_i}, Name_{n_i}, Type_{n_i}, Parent_{n_i})$, где $ID_{n_i} \in ID \cup \emptyset, Name_{n_i} \in Names \cup \emptyset, Type_{n_i} \in Types, Parent_{n_i} \in N \cup \emptyset$. Будем называть узел n_c потомком узла n_p ($n_c descend n_p$), если существует набор $S = \{n_k = n_c, n_{k+1}, ..., n_s = n_p\}$, такой, что $n_{k+1} = Parent_{n_k}, ..., n_s = Parent_{n_{s-1}}$.

Множество $A = \{a_i \in N | Type_{a_i} = Arg\}$, назовем множеством всех аргументов методов. Множество $M = \{m_i \in N | Type_{m_i} = MethodDecl\}$ — множество все методов, используемых в исходном тексте программы. Множество $C = \{c_i \in N | Type_{c_i} = Class\}$- множество всех узлов, классов описанных в анализируемом исходном тексте, $C_{ext} = \{c_i \in N | Type_{c_i} = Type, c_i \notin C\}$- множество всех внешних типов, не встречающихся в исходном тексте, но не имеющих в нем объявлений. Примерами таких типов, можно считать типы, объявленные во внешних библиотеках, которые используются в программе.

Модель представления зависимостей

Данная модель не затрагивает уровни зависимостей между логическими единицами и объединениями классов и описывает только лишь взаимодействие типов и операций над ними.

Множеством сущностей, между которыми устанавливаются зависимости, назовем множество $D = M \cup C \cup C_{ext}$.

Отношение наследования в УПП

Тип $c_c \in C$ наследуется от типа $C_p \in C \cup C_{ext}$ ($c_c inherit c_p$) если выполняется следующее условие: $\exists c_k, c_{k+1} : Parent_{c_k} = c_p. Type_{c_k} =$ Inherits, $Type_{c_{k+1}} = Inherit, Parent_{c_{k+1}} = c_k, Parent_{c_c} = c_{k+1}$.

Если узлы Inherits и Inherit такие, что узел наследника является родительским узлом для Inherits, Inherits — родительским узлом для Inherit, который, в свою очередь, родительский для ссылки на узел базового класса.

```
<Class id="53458261" name="Nancy.DefaultNancyContextFactory" kind="class">
```

```
...
     <Inherits>
       <Inherit type="public">
         <Type ref="1" name="object" kind="internal" />
       </Inherit>
       <Inherit type="public">
         <Type ref="44305076" name="Nancy.INancyContextFactory" kind="interface" />
       </Inherit>
     </Inherits>
...
<Class>
```

Листинг 2. Пример отношения наследования

Отношение агрегации в УПП

Тип $c_a \in C \cup c_{ext}$ агрегирован в $c_c \in C \cup M($ $c_c aggregate c_a)$, если выполняются следующие условия: $c_a descend c_c, Type_{c_a} = Type \nexists c_k : Type_{c_k} = Arg, c_a descend c_k$.

Тип является аггрегируемым в классе или методе, если ссылка на этот тип является потомком узла этого класса или метода.

```
<MethodDecl id="19982180" name="Invoke" kind="method" argc="1" line="61" pos="8">
...
<Type ref="28400769" name="System.Threading.CancellationToken" kind="struct" />
...
</MethodDecl>
```

Листинг 3. Пример отношения агрегации

Отношение действия в УПП

Метод $m_n \in M method c_c \in C$, если .

Метод относится к некоторому классу, если его узел является непосредственным наследником узла этого класса.

```
<MethodDecl id="56030827" kind="constructor" line="11" pos="8">
    <Block line="12" pos="8">
        ...
    </Block>
 </MethodDecl>
```

Листинг 4. Пример отношения действия

Результаты

Был разработан формат XML представления графа зависимостей [4]. Так же были проанализированы два проекта с открытым исходным кодом на языке C#. Первый из них — NancyFx.Core представляет собой библиотеку для построения веб приложений на платформе .Net [5]. Второй проект Newtonsoft.Json — библиотека для работы с файлами формата JSON [6].

	NancyFx.Core	**Newtonsoft.Json**
Количество классов	411	204
Среднее количество методов на класс	4	9
Количество связей типа “Агрегация”	4153	4962
Количество связей типа “Вызов”	2681	2769
Количество связей типа “Передача параметра методу”	3134	3137
Количество связей типа “Наследование”	356	196

Литература

[1] Зубов М. В. *Применение универсальны хпромежуточных представлений для статического анализа исходногопрограммного кода* / Зубов М. В., Пустыгин А. Н., Старцев Е. В. // Доклады Томского государственного университета систем управления и радиоэлектроники. — 2013. — Т. 27, № 1. — С. 64–68.

[2] Ошнуров Н. А. *Построение универсального промежуточного представления исходных текстов на языках C++ и C#* / Н. А. Ошнуров, А. Н. Пустыгин, А. А. Ковалевский // Доклады Томского государственного университета систем управления и радиоэлектроники. — 2014. — Т. 33, № 3. — С. 135–139.

[3] Dependency Graph. Электронный ресурс: `https://en.wikipedia.org/wiki/Dependency_graph` (дата обращения 07.01.2015).

[4] dependencygraph.xsd. Электронный ресурс: `https://github.com/ifanatic/CodeAnalysis/blob/master/doc/dependencygraph.xsd` (дата обращения 07.01.2015).

[5] NancyFx.Core. Электронный ресурс: `https://github.com/NancyFx/Nancy/tree/master/src/Nancy` (дата обращения 07.01.2015).

[6] Newtonsoft.Json. Электронный ресурс: `https://github.com/JamesNK/Newtonsoft.Json` (дата обращения 07.01.2015).

Пустыгин А.Н., Зубов М.В. Старцев Е.В.
Челябинск, Челябинский государственный университет

Экспериментальное изучение количественных закономерностей при анализе исходных текстов ПО с открытым кодом по эквивалентным представлениям

Аннотация

Использование эквивалентных представлений исходных текстов программ для извлечения знаний разработчика дает широкие возможности для анализа . Обычно результаты такого анализа представлены в текстовой или графической форме. Применение эквивалентных представлений позволяет получать также и количественные оценки, которые характеризуют исходный код ПО. Рассматриваются результаты исследований количественных закономерностей, извлеченных из эквивалентного представления.

В течение некоторого времени разрабатывались инструменты для систематического извлечения информации из исходных текстов [1], построенные по предложенной схеме рис 1.

Рис. 1: Обобщенная схема анализа исходного текста с помощью унифицированного пакета инструментов

Главной конструктивной идеей этой схемы является использование универсальных форматов промежуточного (ПП) и эквивалентных представлений (ЭП), что позволяет применять единый инструментарий для решения одинаковых задач анализа текстов на разных языках. На настоящий момент предложены форматы универсальных промежуточных представлений для пар java/pynhon, c++/C#, построена и протестирована линейка генераторов таких наборов данных по исходному тексту, что позволило разработать и протестировать прототипы некоторых функций анализаторов эквивалентных представлений. Использование универсальных ПП позволяет вводить универ-

сальные эквивалентные представления, предназначенные для решения конкретных задач извлечения информации или анализа исходного текста ПО.

В том числе разработаны прототипы:

1. Анализатор использования «жадных» методов,
2. Анализатор положения создаваемых объектов,
3. Построитель срезов представления потока управления,
4. Анализатор недостижимых функциональных блоков,
5. Построитель срезов диаграмм классов,
6. Построитель трасс между заданными функциональными блоками.

Примеры выполнения тестов функционала приводятся в [8]. Использовавшиеся в качестве тестовых проектов разработки взяты из открытых хранилищ с открытым исходным текстом и являлись поддерживаемыми проектами на момент их выбора. Их размер был признан достаточным для массовых испытаний, а поддержка разработчика в какой-то мере гарантировала «практически допустимый» уровень алгоритмических ошибок. Для получения практических результатов были выбраны несколько Python и Java проектов. Используемые промежуточные представления — универсальные, поэтому все равно, на каком языке написан анализируемый проект. Python-проекты: logilab[2], pylint[3], numpy [5], twisted[6] и bazaar[4]. Java-проекты: собственные java-разработки для генерации представлений UCR(jclassgen) и UCFR(jcfgen)[8], а так же компилятор javac из пакета OpenJDK [7].

Проведение экспериментов над полученными эквивалентными представлениями позволило обнаружить и исследовать в численном эксперименте над текстовыми проектами [2,3,6] ряд количественных характеристик, сопровождающих извлечение информации из исходного текста ПО, описанные в [9].

Нарушение монотонности при анализе динамической типизации по исходному тексту

В [1] был предложено решение проблемы характерной для статического анализа динамических языков программирования на примере Python. Информация о типах данных не может быть получена в

явном виде из исходного кода, так как атрибут типа необязателен. Известны некоторые алгоритмы вывода типов, нами использовалась «утиная типизация» подбора классов-кандидатов для полей классов, называемых в методике утиными. Под утиным полем понимается поле класса, об агрегируемом классе которого (объект какого класса хранится в поле) делается предположение. Агрегация — отношение «часть-целое» между двумя равноправными объектами, когда один объект имеет ссылку на другой объект. Вводились сигнатуры утиного поля и характеристической сигнатуры возможного класса-кандидата. Характеристическая сигнатура класса-кандидата представляла собой кортеж, содержащий множества его полей и методов, а сигнатура утиного поля — полей и методов к которым происходило обращение в методах класса, которому принадлежит утиное поле.

Основные результаты дает проверка методики на основе динамического анализа на типовых сценариях исполнения для проектов. Детали динамической методики были затронуты ранее в статье [1], Суть способа в том, что благодаря динамическому анализу получается корректная информация о типах данных полей классов. Эта информация сравнивается с теми результатами, которые получились при статическом анализе. Сценарии исполнения для трех типовых проектов — logilab[2], pylint[3] и bazaar[4] аналогичны тем, что использовались при проверке оригинальной методики и здесь не затрагиваются.

Отметим, что особый интерес при динамической проверке представляет исследование влияния используемого значения порога для функции при подборе классов-кандидатов, что и было выполнено в ходе динамической проверки. Результаты представлены ниже. Использовалось 10 различных значений порога от 0.1 до 0.95 для обеих пороговых функций. Значения порога 1.0 не исследовались, ведь для получения значения пороговой функции, равного 1.0, обязательным условием является то, что у класса-кандидата имеются все поля и методы из сигнатуры утиного поля.

Как видно из приведённых графиков, в диапазоне пороговых значений 0.4 . . . 0.6 наблюдается нарушение монотонности полученной зависимости для всех исследовавшихся проектов, что отражает общие закономерности исходных текстов тестовых проектов.

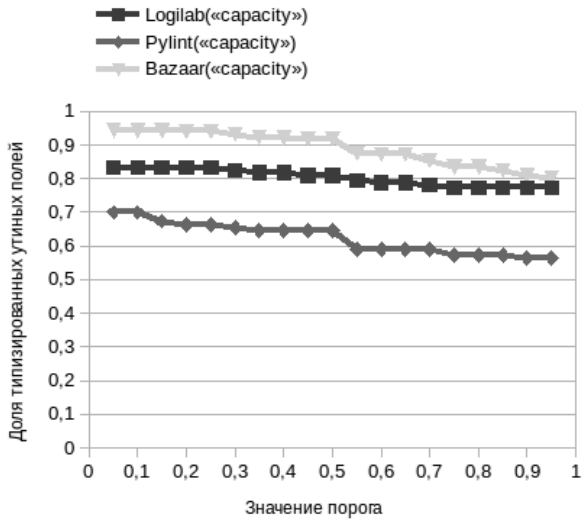


Рис. 2: Зависимость доли типизированных утиных полей от порога.

Нарушение монотонности при анализе «жадных» методов по исходному тексту

Модель данного анализа описана в [9]. Жадными считаются функции, которые часто обращаются к методам объекта другого класса. От таких функций/методов следует избавляться путем переноса их в другой класс. Для выполнения поиска таких методов необходимо определять количественную характеристику, значение которой можно было сравнивать с некоторым пороговым уровнем, устанавливающим границу между «жадным» и допустимым. Для определения оптимальной величины порога k воспользуемся методами численного имитационного моделирования. Использовались перечисленные выше тестовые проекты с открытым исходным кодом, которые представлены в левой колонке таблицы 1. Рассмотрим различные целые значения $k \in [1, 10]$. Для каждого метода каждого проекта посчитаем его k_m — количество обращений к методам объекта другого класса. Далее по всему проекту подсчитывается число методов, таких что $k_m > k$. Для каждого проекта подсчитывается доля жадных методов от обще-

Проект	Процент найденных методов в зависимости от порога, %									
	1	2	3	4	5	6	7	8	9	10
jclassgen	23,08	9,62	3,21	1,28	1,28	0,64	0,00	0,00	0,00	0,00
jcfgen	14,62	4,98	2,66	1,66	1,33	1,00	0,66	0,66	0,66	0,00
pylint	19,45	4,56	1,82	0,30	0,30	0,30	0,30	0,30	0,30	0,30
logilab	19,67	6,14	2,69	1,25	0,86	0,48	0,38	0,29	0,29	0,19
numpy	26,72	10,23	5,15	2,93	1,98	1,19	0,87	0,56	0,40	0,32
javac	18,22	7,57	4,42	2,88	2,13	1,49	1,11	0,87	0,62	0,49
twisted	27,16	9,42	3,84	1,80	1,09	0,62	0,35	0,19	0,14	0,14
bazaar	43,89	19,21	10,44	5,83	3,57	2,21	1,51	1,00	0,63	0,48

Таблица 1: Процентное соотношение найденных методов от общего числа

го количества методов, и результаты для различных значений порога приведены в таблице 1.

Таким образом, из таблицы видно, что при значениях порога от 1 до 4 наблюдается резкое падение процента искомых функций (практически в 10 раз). Это говорит о том, что такое значение порога недостаточно велико, и на нем имеется большое количество ложных срабатываний, которые сильно зависят от значения порога. То есть в этой области большую часть составляет случайная величина.

При значении k от 5 и выше наблюдается практически линейная зависимость. Для выбранных проектов при значении порога выше 8 наблюдается отсутствие зависимости или же 0 значение. Это говорит о том, что данные значение слишком строгие и отбрасывает практически все варианты. Выбор более низкого порога покроет и срабатывания на этих значениях.

Величина k влияет на строгость анализа, так как она указывает, к какому количеству полей и методов переменной должен обратиться метод, чтобы его можно было счесть жадным. Эта величина является входной для анализатора, и исследователь проекта сам может ее определить. Как показало моделирование анализа на выбранных проектах, величины 6 и 7 вполне подходят для значений, с которых стоит начинать исследование. Такое количество обращений к методам объекта другого класса вполне можно счесть жадным.

Заключение

Полученные численные оценки выполнения тестовых экспериментов позволяют обнаружить эмпирические закономерности, существу-

ющие в данной части предметной области. Важнейшим следует признать обнаружение диапазонов резкого изменения графиков зависимостей, характерных для всей совокупности тестовых проектов. Если считать использованный набора тестовых проектов представительной выборкой, можно рекомендовать полученные количественные оценки к использованию в качестве эмпирических ориентиров для построения алгоритмов извлечения в частных случаях, то есть при выполнении практических исследований исходных текстов ПО.

Литература

[1] Зубов М. В., Пустыгин А. Н., Старцев Е. В., *Получение типов данных в языках с динамической типизацией для статического анализа исходного кода с помощью универсального классового представления* // Вестн. Астрахан. гос. ун-та. Сер.: Управление, вычислительная техника и информатика. — 2013. — № 2. — с.66–74

[2] ndex (Logilab.org) [Электронный ресурс]. — Режим доступа: `http://www.logilab.org`, свободный (дата обращения: 15.06.2014)

[3] Pylint — code analysis for Python | [Электронный ресурс]. — Режим доступа: `http://www.pylint.org/`, свободный (дата обращения: 15.06.2014)

[4] Bazaar[Электронный ресурс]. — Режим доступа: `http://bazaar.canonical.com/en/`, свободный (дата обращения: 15.06.2014)

[5] NumPy — Numpy [Электронный ресурс]: `http://www.numpy.org/` (дата обращения 15.06.2014)

[6] Twisted [Электронный ресурс]: `https://twistedmatrix.com` (дата обращения 15.06.2014)

[7] OpenJDK [Электронный ресурс]: `http://openjdk.java.net/` (дата обращения 16.06.2014)

[8] csu-code-analysis GitHub [Электронный ресурс]: `https://github.com/exbluesbreaker/csu-code-analysis` (дата обращения 15.06.2014)

[9] Зубов М. В., Пустыгин А. Н., Старцев Е. В. *Численное моделирование анализа исходного кода с использованием промежуточных представлений* // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. — 2014. — №4. — Астрахань: Издательство АСТГУ. — С. 55–66

Ирина Плешкова
Пермь, Пермский государственный национальный исследовательский университет (ПГНИУ)

Повышение качества поиска в больших объёмах текстовых документов с использованием генетического алгоритма как способ поддержки научных исследований

Аннотация

В докладе рассматривается проблема работы с большими объёмами текстовых документов. Существующие подходы к синтаксическому поиску имеют недостаточно высокое качество, а семантические неприменимы к большим объёмам данных. В докладе описывается новый подход к эффективной реализации семантического поиска, применимый к большим объёмам документов, с использованием генетического алгоритма. Предлагаются эвристики, учитывающие традиционную структуру научной публикации и таким образом позволяющие улучшить качество поисковых результатов.

В мире, согласно исследованию IBM, каждый день уже в 2012 году появлялось 2.5 экзабайта данных. Считается, что закон Мура применим не только к вычислительным мощностям, но и к объёмам данных, т.е. можно ожидать, по крайней мере, удвоение объёмов данных каждые 2 года. Согласно статистике IDC, 90% данных хранятся в неструктурированном, в том числе в текстовом виде. Когда данных много, в них сложно что-то найти. А если человек не знает, какие конкретно слова используются в нужных ему документах, а знает только предметную область, то ему не подойдёт традиционный поиск по подстроке.

В частности, у некоторых компаний за время работы накапливается много документов в текстовом виде. Это могут быть нормативные акты, контракты, инструкции, технические задания, заказы и т.д., которые не хранятся в информационной системе, даже если она внедрена.

В организациях, занимающихся научно-исследовательской деятельностью, например, НИИ, может быть своя электронная библиотека, состоящая как из публикаций, написанных работниками, так и

научных материалов, которые были куплены для проведения исследований. Интеллектуальный поиск по ним поможет сэкономить как время работников, поскольку в выдаче будут присутствовать только релевантные документы, так и средства в случае, если необходимая информация уже была в библиотеке.

Говоря о библиотеках вообще, можно заметить тенденцию к цифровому формату книг и журналов. Интеллектуальный поиск позволит реализовать нечто вроде рекомендаций по тематике, а не по конкретным словам, использующимся в тексте. Тот же интеллектуальный подход можно применить к новостным сайтам.

Все эти примеры сводятся к тому, что зачастую бывает полезен поиск по смыслу, а не по используемым в тексте словам.

Предлагается новый подход к семантическому поиску по текстовым данным, и, конкретно, новый способ семантической индексации документов с помощью генетического алгоритма и онтологий. Генетические алгоритмы быстры и достаточно точны при правильной настройке параметров, а также при использовании совместно с онтологиями они позволяют индексировать понятия, которых нет в тексте в явном виде.

На основе общей модели генетического алгоритма была построена модель индексации текстовых документов. В ней каждому понятию приписывается вес, вычисляемый на основе множества критериев. Правила вычисления веса задаются декларативно и определяются в процессе исследования. В частности, было выяснено, что для научных публикаций при вычислении веса понятий, помимо стандартных статистических характеристик, таких как, например, частота встречаемости слов, можно использовать данные о структуре текста, ввиду стандартизованности этой структуры. Например, предлагается повышать вес понятий, встречающихся в аннотации либо в списке ключевых слов.

Разрабатываемая система семантического поиска работает с хранилищем данных, где содержатся документы и онтологии. Индексация происходит в режиме оффлайн с помощью генетического алгоритма. Система поиска получает на вход запрос пользователя, производит поиск в реальном времени по уже построенному семантическому индексу и возвращает релевантные документы. Заметим, что эти документы могут не содержать ни одного слова из запроса, но, тем не менее, удовлетворять информационные потребности пользователя.

Основные критерии качества поисковой системы — скорость и качество поиска. Традиционные поисковые системы выполняют поиск быстро, но результаты могут быть неточны из-за того, что они не учитывать семантически связанные понятия и контекст. Системы семантического поиска обычно предоставляют более полные и точные результаты, но работают медленнее из-за того, что семантика учитывается в процессе поиска, а не индексации. В предлагаемом подходе с одной стороны, учитываются и связанные по смыслу понятия и контекст запроса за счёт использования онтологий и индексации с помощью генетического алгоритма, а с другой стороны, скорость поиска будет высокой, поскольку на этапе поиска будет лишь обращение к семантическому индексу, который построен в оффлайн режиме.

Работа выполнена при поддержке гранта фонда содействия развитию малых форм предприятий в научно-технической сфере «УМНИК».

Литература

[1] Плешкова И.Ю. *Разработка системы семантического поиска по текстовым документам* // Материалы X Студенческого регионального конкурса научных проектов по программе УМНИК 27-28 ноября 2014 г. Пермь. 2014. pp. 35-38.

Игорь Постаногов
Пермь, Пермский государственный национальный исследовательский университет (ПГНИУ)

Проектирование и реализация решений интеллектуального анализа BigData с использованием стека технологий Apache Spark и методов онтологического инжиниринга

Представлен краткий отчёт о преподавании курса по обработке больших объемов данных на 2 курсе магистратуры специальности «математическое и программное обеспечение вычислительных систем» для решения задач онтологического инжиниринга. Для демонстрации современного уровня возможностей по обработке BigData

предложено использовать платформу Apache Spark. Рассмотрены основные компоненты платформы, их преимущества и простота совместного использования. Преподавание курса также позволило восполнить пробел магистрантов в знаниях в части функционального программирования. Рассмотрен компонент Apache Spark GraphX как средство поддержки разрабатываемой системы, основанной на методах онтологического инжиниринга.

Согласно «циклу зрелости технологий» компании Gartner от июля 2014 года, концепция BigData перешагнула «пик чрезмерных ожиданий», а область Data Science крайне близка к его вершине. В ближайшие годы ожидается усиление уже сформировавшийся нехватки кадров, разбирающихся в задачах анализа больших данных.

Для того чтобы студенты магистратуры были более востребованы по её окончании и овладели навыками, необходимыми работодателям, с 2013 года в рамках учебной дисциплины «Современные Internet-технологии» введено изучение парадигм анализа больших данных, а также практическое использование существующих их реализаций.

Стоит отметить, что год таких изменений выбран достаточно удачно, поскольку в 2013 году вышла версия Apache Hadoop 2.0, в которой впервые представлен модуль YARN, отвечающий за управление ресурсами кластеров и планирование заданий. Новый планировщик ресурсов — значительный скачок в части расширения множества приложений, которые можно исполнять в рамках указанной платформы. YARN поддерживает множество моделей обработки *вдобавок* к MapReduce — парадигме, успешно применяемой в большом классе задач, однако имеющей известные проблемы при решении итеративных и интерактивных задач. Для снятия ограничений и сложностей парадигмы, для конкретных типов задач создавались свои фреймворки — Pregel для обработки графов, Twister для итеративных задач и Storm для задач, в которых входные данные поступают в режиме реального времени. Среди недостатков обходных решений выделяют:

- Необходимость затрат на обучение, поскольку для новой задачи может потребоваться принципиально новый фреймворк.
- Ограничение типов задач, решаемых отдельным фреймворком (например, Dryad и MR Merge не поддерживают циклический поток данных). В то же время, возникают сложности в интеграции при решении задач на стыке возможностей нескольких фреймворков.

- Неадекватность среде вычислений (Twister не отказоустойчив).

Актуальность задач обработки BigData не обошла стороной и академические круги. AMPLab в UC Berkeley уже в 2009 году начали разрабатывать Spark — новую парадигму вычислений. Её реализация имела десятикратный прирост в скорости в сравнении с MapReduce для задач итеративного машинного обучения (согласно статье 2010 года). Говоря о современных успехах реализации стоит отметить, что осенью 2014 года Spark поставил новый рекорд в Daytona Gray Sort 100TB Benchmark, побив предыдущий рекорд, установленный Hadoop MapReduce, закончив свою работу в 3 раза быстрее и использовав при этом 10 раз меньше вычислительных узлов.

На момент написания статьи, Spark является проектом верхнего уровня Apache и распространяющимся под соответствующей лицензией. Есть сведения об использовании Apache Spark такими компаниями как IBM, SAP, Oracle, DataStax и другими, для решения таких задач как

- In-memory аналитика и поиск аномалий (Conviva)
- Интерактивные запросы к потокам данных (Quatifind)
- Анализ логов (Foursquare)
- Оценка загруженности дорог по информации с мобильных GPS (Mobile Millennium)
- Классификация спама в Twitter (Monarch)

Такая популярность Spark'а объясняется отсутствием основного ограничения парадигмы MapReduce — необходимости разбиения задачи на этапы Map и Reduce. Spark решил задачу обеспечения параллелизма и отказоустойчивости, введя концепцию Resilient Distributed Dataset (RDD) — устойчивого распространенного набора данных, являющегося абстракцией распределённой памяти. Отказоустойчивость реализуется сохранением «родословной» для каждого экземпляра RDD, что позволяет восстановить утерянные (например, вследствие отказа одного из вычислительных узлов) его элементы. RDD, по сути, это неизменяемая коллекция элементов, которую можно трансформировать.

В качестве операторов трансформации выступают традиционные для функциональных языков программирования операции *map*, *filter*, *union*, *reduce*, *cartesian* и проч. Возможность их распределённого выполнения гарантируется способностью к сериализации функ-

ций под оператором трансформации в существующей реализации Apache Spark на Java-совместимом языке программирования Scala. Spark предоставляет API для языков программирования Scala, Java & Python, однако именно на Scala его использование выглядит наиболее естественно. Реализация подсчёта слов на Scala с использованием Apache Spark требует лишь кода
spark.textFile("hdfs://...").flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey((a, b) => a + b).

Apache Spark содержит встроенные библиотеки, позволяющие решать различные типы задач:

- Spark SQL, предоставляющий унифицированный доступ к структурированным данным.
- Spark Streaming, содержащий абстракции, позволяющие реализовывать отказоустойчивые приложения обработки данных, поступающих в режиме реального времени.
- MLlib, реализующий алгоритмы машинного обучения.
- GraphX, предоставляющий возможности обработки графов, а также являющийся Pregel-совместимым.

Заданием по описываемой учебной дисциплине «Современные Internet-технологии» стала модификация одного из задания дисциплины «Добыча знаний: теоретические основы и инструментальные средства Data mining», изучаемой в том же семестре. По «Data mining» в том числе требовалось реализовать поиск по словам и метаданным вебстраниц (своего рода поисковая система), однако не было никаких требований к средствам реализации. В задании по «Современным Internet-технологиям» были выделены две части.

Непроверяемая часть задания:

- Выбрать web-сайт и скачать его содержимое (частично).
- Извлечь текстовую информацию из каждой страницы.
- Извлечь метаинформацию из каждой страницы.

Проверяемая часть задания:

- Построить индекс типа RDD[(A, B)], в котором ключом является слово, а значением — список ссылок, на страницах которых встречается это слово.
- Построить временную Spark SQL таблицу, в которой есть 2 столбца: (значение метаданного; ссылка на страницу, имеющую это значение метаданного).

- Реализовать поиск по словам на странице.
- Реализовать поиск по метаданным страниц.
- Реализовать запись/чтение индекса. Индекс не должен строится каждый раз при запуске программы.
- «Слова» и «Значения метаданных» должны передаваться как HTTP GET параметры результирующего приложения. В ответ необходимо возвращать список ссылок на ресурсы, удовлетворяющие запросу.

Второе задание для магистрантов было групповым (группы по 2 человека). Первой группе было необходимо «написать программу MapReduce сканирования и переименование записей Lily». Задание было направлено на обучение работы с традиционным MapReduce подходом. Второй группе предлагалось переходное MapReduce-Spark задание, предполагающее «реализацию умножения матриц в парадигме MapReduce и Spark для сравнения их эффективности и простоты написания». Третьей группе было дано задание на «реализацию в GraphX поиска кратчайшего пути в графе сформированном на основе Википедии для разрабатываемой системы, основанной на методах онтологического инжиниринга», выбранное с учётом темы магистерской диссертации.

В рамках учебной дисциплины также были рассмотрены технологии HDFS, Giraph, YARN, HBase, Lily, Solr, Cascading и прочие, однако описание опыта их преподавания выходит за рамки доклада.

Литература

[1] Русаков С. В., Хеннер Е. К., Чуприна С. И. *Интеграция базовой университетской подготовки специалистов по информатике и информационным технологиям* // Университетское управление: практика и анализ, № 3, 2014. С. 119–125.

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

[3] *Spark: Cluster Computing with Working Sets.* Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. HotCloud 2010. June 2010.

[4] Официальный сайт Apache Spark. `https://spark.apache.org/`

[5] Spark News *Spark wins Daytona Gray Sort 100TB Benchmark.* https://spark.apache.org/news/spark-wins-daytona-gray-sort-100tb-benchmark.html

[6] Paco Nathan *How Apache Spark fits in the Big Data landscape.* http://www.slideshare.net/pacoid/how-spark-fits-into-the-big-data-landscape

[7] М. Тим Джонс *Spark, альтернатива для быстрого анализа данных.* http://www.ibm.com/developerworks/ru/library/os-spark/

Евгений Алексеев, Пётр Дёмин
Киров, Вятский государственный университет
Проект: www.1aer.ru

Свободные и бесплатные программы для создания математических сайтов

Аннотация

Приведен обзор современного состояния программных средств, позволяющих создавать сайты, содержащие большое количество формул и графиков.

При разработке сайтов математической направленности одной из серьёзных проблем является публикация формул и графиков.

Публикация формул на web-страницах

Для публикации формул можно использовать три подхода.

1. Использование MathML.
2. Использование приложений для генерации графических изображений с формулами.
3. Использование библиотек JavaScript для вставки в html-код формул LaTeX.

MathML [1] (http://www.w3.org/Math/) — язык разметки для представления математических символов и формул в html документах. Однако на сегодняшний день MathML полностью поддерживается только в FireFox (частично — в Google Chrome с плагином MatJax).

Использовать MathML можно непосредственно, вводя код формулы в html код страницы. Однако для пользователя, который не знаком

с языком математической разметки, набор формул может оказаться не совсем простой задачей. Для упрощения получения формулы в формате MathML можно использовать следующие свободные инструменты:

- LibreOffice Math — набираем формулу и сохраняем её в формате MathML;
- расширение FireFox — FireMath позволяет вводить формулы с помощью кнопок панелей инструментов и сохранять его как в формате MathML, так и в виде графического изображения (PNG, JPEG).

Вторым подходом является использование программ генерации изображений с формулами. В последнее время появилось множество online приложений для генерации формул. Среди них можно отметить:

- расширение Google Chrome — Daum Equation Editor, которое позволяет сохранить формулу в формате LaTeX и в виде графического png файла;
- визуальный on-line редактор формул `http://primat.org/editor/redaktor.html`[1], который генерирует формулу в формате LaTeXи графический файл в формате gif;
- Использование математических приложений, которые генерируют математические формулы (WolframAlpa, SmathStudio).

Существует приложение на Perl TextoGif (`http://www.fourmilab.ch/webtools/textogif/textogif.html`), которое преобразовывает формулу в формате Tex в графический файл (png, gif).

Однако все рассмотренные приложения не позволяют автоматизировать процесс вставки формул в код html-страницы. На взгляд авторов, лучшим решением является подключение библиотек на JavaScript, позволяющих вставлять код формул на LaTeX непосредственно в web страницу. Среди них можно выделить:

- библиотеку MathJax (`http://www.mathjax.org/`);
- библиотеку KaTex (`http://khan.github.io/KaTeX/`);
- библиотеку jsMath (`http://www.math.union.edu/~dpvc/jsmath/`);

[1] Аналогичные сайты `http://www.codecogs.com/latex/eqneditor.php`, `http://www.astronet.ru/db/latex2gif/`, `http://ru.numberempire.com/texequationeditor/equationeditor.php`.

- сервис Google для формирования изображения формулы из формулы LaTeX (`https://google-developers.appspot.com/chart/infographics/docs/formulas`).

Программное обеспечение для формирования графиков

На научных сайтах очень часто встречаются графики различных функций. Для рисования графиков можно использовать Canvas — растровый холст HTML5, предназначенный для рисования. В этом случае для изображения графика функции надо его запрограммировать. Альтернативным подходом для встраивания графиков в html-страницу является использование онлайнового физико-математического пакета ГРАФ `http://physics.nad.ru/graph.html`. С помощью этого пакета можно строить и форматировать графики. Пакет ГРАФ генерирует графический файл в формате png, который можно встроить в web страницу.

Таким образом, современные программные средства позволяют генерировать формулы и графики для вставки их в html-страницы.

Кроме того, JavaScript позволяет создавать on-line визуальные приложения для решения математических и инженерных задач различной сложности.

Это позволит разработать новое поколение инженерных и математических сайтов.

Литература

[1] Елизаров А. М., Липачев Е. К., Малахальцев М. А. *Веб-технологии для математика. Основы MathML.* М., Физматлит, 2010. — 194с.

Денис Силаков, Елена Силакова
Москва, ООО «НТЦ ИТ РОСА», ЦКП им. Д.И. Менделеева
Проект: ROSA Desktop Fresh http://www.rosalab.ru

ROSA Desktop Fresh в химической лаборатории

Аннотация

В докладе освещается использования дистрибутива ROSA Desktop Fresh для обработки экспериментальных результатов, получаемых на оборудовании Центра коллективного пользования (ЦКП) им. Д.И. Менделеева. Рассказывается об опыте замещения ряда проприетарных приложений свободными аналогами и о трудностях, с которыми приходится сталкиваться в работе.

Центры коллективного пользования (ЦКП) призваны дать возможность ученым и студентам пользоваться дорогостоящим оборудованием, приобретение которого конкретным институтом или учебным заведением с финансовой точки зрения не оправдано или вовсе невозможно. В ЦКП им. Менделеева используются приборы, позволяющие проводить анализ состава образцов — хроматограф, спектрофотометр, атомно-абсорбционные и ИК-Фурье спектрометры, масс-спектрометр с индуктивно связанной плазмой, электронный микроскоп и другие.

Каждый из приборов, предполагающих программную обработку получаемых данных, обычно поставляется с собственным ПО. Более того, нередко с прибором поставляется непосредственно компьютер с уже установленными программами — при стоимости прибора в несколько миллионов рублей, компьютер можно рассматривать как мелкую периферию.

Предлагаемое поставщиками приборов ПО (по крайней мере, в нашем случае) является проприетарным и работает только в ОС Windows. Заменить ОС на компьютере, непосредственно работающим с прибором, проблематично — ряду приборов требуются специфичные драйвера, альтернативное ПО вряд ли будет одобрено инженерами, осуществляющими сервисное обслуживание и так далее. А поскольку стоимость всех этих программ фактически вошла в стоимость прибора, то и финансовых стимулов для этого нет.

Неудобства, конечно, имеются — например, для предотвращения заражения критически важных машин вирусами, эти машины не под-

ключены к интернету, а результаты измерений передаются пользователям на неперезаписываемых CD-дисках (никаких флешек!). Однако на данный момент сотрудники готовы мириться с этими неудобствами.

Тем не менее, есть другой фактор, открывающий дистрибутивам Linux дорогу в ЦКП и схожие организации. Дело в том, что исследовательская работа заключается не только в снятии замеров с помощью некоторого прибора, но и в последующем анализе результатов.

Поставщики приборов предоставляют приложения для проведения такого анализа, однако не всегда они удобные, а обычно еще и небесплатные. Поэтому законопослушный ученый должен либо обрабатывать результаты на той же машине, где он их получил (и занимать время дорогостоящего оборудования), либо покупать себе отдельную лицензию. Однако результаты измерений обычно представляют собой кучу цифр либо картинок, которые можно перенести на другую машину и обрабатывать, чем душе угодно. На нашем личном примере мы попробовали использовать для обработки данных дистрибутив ROSA Desktop Fresh и в этом докладе рассмотрим приглянувшиеся нам приложения из его репозиториев.

Базовые программы

Начнем с программ и утилит, которые в наших условиях не являются принципиальными, но сильно упрощают работу. К таковым следует отнести:

- Представления периодической таблицы Менделеева — gchemtable и gperiodic.
- «Химический калькулятор» Gchemcalc (из набора Gnome Chemistry Utils) для анализа веществ сложной структуры.
- Программы для рисования химических структур, трехмерных моделей молекул и кристаллов и тому подобного.

Среди программ последнего вида есть немало свободных — rasmol, gcrystal, gchem3d и другие. Эти приложения хороши для относительно несложных действий (например, нарисовать красивую картинку для статьи), однако могут спасовать при необходимости моделирования и анализа сложных структур. На такие случаи мы установили Marvin Beans от ChemAxon — закрытый инструментарий, имеющий

вполне функциональную бесплатную версию. Написан инструментарий на Java и в Linux чувствует себя отлично.

Обработка данных

Многие задачи по обработке данных, получаемых от различных спектрометров и схожих приборов, достаточно рутинны. Достаточно взять полученный набор значений, построить по ним графики спектров, вычислить положения пиков, отклонения и тому подобные характеристики.

В первом приближении, с такими задачами справляются редакторы электронных таблиц. На этом поле LibreOffice Calc оказался достойной заменой MS Excel.

Для более сложных расчетов используются «продвинутые» приложения, коих на первый взгляд хватает и в Linux — Veusz, OpenDX, или даже система статистических вычислений R. В нашем случае оказалось, что основной кандидат на замену проприетарным аналогам — это QtiPlot, построенный по образу и подобию Origin, ибо последний являлся в организации стандартом де-факто.

Недостаток всех этих программ в том, что они не учитывают специфики обрабатываемых данных. Например, если масс-спектрометр разделяет элементы в зависимости от отношения массы к заряду ядра, то в результирующем спектре может произойти наложение сигналов от двух разных ионов с одинаковым отношением заряда к массе. Современные приборы являются достаточно «интеллектуальными» и способны автоматически корректировать и учитывать многие подобные аспекты (естественно, с подсказками оператора), однако всегда выдавать идеальный результат они еще не в состоянии.

Как следствие, анализ полученных результатов — задача отнюдь не тривиальная, и посильная помощь со стороны ПО здесь очень пригодиться. Например, программа может вывести подсказки о соответствии определенных пиков в спектре тем или иным веществам, помочь рассчитать характеристики сложных полимеров и так далее. Единственной свободной программой такого рода, которую нам удалось найти, является massXpert. Приложение нацелено на анализ спектров полимеров и при этом обладает изрядной универсальностью, не ограничиваясь каким-то одним их видом.

Заключение

Главный итог наших усилий — это возможность использовать машину с Linux как рабочее место химика-аналитика. Однако отметим, что обойтись исключительно свободными программами не получилось (несмотря на относительную простоту решаемых задач), и текущие тенденции разработки свободного ПО пока не позволяют утверждать, что в скором будущем ситуация изменится.

Андрей Черепанов
Москва, ООО «Альт Линукс»
http://altlinux.ru

Что новое ожидается в Восьмой платформе Альт Линукс

Аннотация

Доклад посвящён основным направлениям развития и возможностям, адресованных учреждениям и организациям образования, в рамках Восьмой платформы Альт Линукс.

Альт Линукс периодически, раз в 2 года, выпускает программную платформу, на базе которой выпускаются различные решения (например, дистрибутивы).

Технически платформа представляет собой актуальную пакетную базу (срез Sisyphus) с заданными параметрами. После выпуска платформа доступна в виде обновляемого репозитория. Сразу после выпуска создаются дистрибутивы: универсальный Кентавр, дистрибутивы для рабочих станций Simply Linux и KDesktop и образовательный комплект.

Очередная платформа Альт Линукс будет восьмой и ожидается весной 2015 года.

Что ожидается в Восьмой платформе?

- В тренде импортозамещения интересно внедрение Samba в режиме AD DC как замены Active Directory. Это позволит не только управлять машинами с Linux, но и с Microsoft Windows. Следующим шагом станет применение Zarafa как замены Microsoft

Exchange и подготовки миграции с другими инфраструктурными службами Microsoft (например, SharePoint).

- Вся оснастка традиционного ALT-домена будет адаптирована для использования как Samba AD DC (вместо обычной Samba), так и 389-ds (вместо OpenLDAP).
- Появится возможность настройки входа в Active Directory/ Samba AD DC и домен RELS штатным способом в Центре управления.
- Ожидается интеграция в Alterator средств массового управления на основе Puppet, Ansible или Salt.
- ALT-домен должен уметь работать не на одном сервере, а на группе серверов с возможностью распределения по ним предоставляемых служб.
- Автоматическая установка получит возможность указания дополнительных действий после установки для постустановочной кастомизации.
- Появится Центр приложений Альт Линукс, выбор и установка программного обеспечения переходит от концепции пакетов к более привычным пользователям концепциям приложений и наборов.
- В части виртуализации ожидается свежее ядро с поддержкой OpenVZ.
- Доработка системы конфигурирования Alterator в части отдельного самодокументированного вызова бэкендов без фронтенда и упрощения отладки.
- Поддержка перспективных архитектур ARM: Baikal и AArch64.
- Расширение поддержки UEFI (RAID1, PXE boot, подписанное ядро, модули, ключи OEM/пользователя).
- Поддержка опций SSD при установке.

Что интересного ожидается для образовательных учреждение и организаций?

- Ребрендинг. Теперь образовательный комплект даже по названию должен подходить не только для школ, а для всего

спектра образовательных организаций: детские сады, школы, СУЗы, ВУЗы, учреждения управления образованием. Состав программного обеспечения будет также пересмотрен.

- Для удобства будет подготовлен один образ, с которого можно установить профиль для той или иной области применения.
- Централизованное управление рабочими станциями (управление программным обеспечением, параметрами настройки и их принудительным обновлением).
- Ограничения изменения параметров настройки (режим киоска, когда никакие изменения учащегося не сохраняются).
- Комбинирование функциональных комплектов программного обеспечения (например, развертывание сетевых служб на компьютере преподавателя).

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

Обучение процессному управлению на свободном ПО

Аннотация

Разработка исполнимых бизнес-процессов требует нового мышления, отличающегося от традиционного мышления программистов или бизнес-аналитиков. Для обучения такому мышлению предлагается давать студентам в виде заданий на разработку исполнимых бизнес-процессов (и объяснять основные подходы к их решению) задачи, архитектурно соответствующие распространенным ситуациям в бизнесе предприятий.

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

Потребность в специалистах, обладающих процессным мышлением

Процессная автоматизация позволяет:

- Быстро адаптировать разработку к изменению задач за время разработки
- Понизить стоимость разработки за счет:
  - Разработки бизнес-процессов средствами СУБП вместо написания кода
  - Исключения взаимодействия программистов с заказчиком
  - Освобождения программиста от рутинных задач
- Понизить стоимость технической поддержки
- Понизить стоимость доработок и сопровождения

Эти преимущества начинает оценивать бизнес. Однако разработка процессных решений требует новых специалистов — бизнес-аналитиков с процессным мышлением, отличающимся от мышления традиционных ИТ-специалистов.

Этих специалистов надо обучать приемам построения различных решений процессной автоматизации.

Для обучения такому мышлению предлагается давать студентам в виде заданий на разработку исполнимых бизнес-процессов задачи, архитектурно соответствующие распространенным ситуациям в бизнесе предприятий. Рассмотрим типичные сценарии.

Если в бизнес-процессе несколько действий подряд должны быть выполнены одновременно двумя исполнителями, то 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 неправильного построения схемы такого бизнес-процесса представлен на Рис. 1, а правильного — на Рис. 2.

В схемах бизнес-процессов можно использовать элементы разделения без парных им элементов — слияний. В этом случае для удаления выполнивших свою задачу точек управления используется элемент — событие завершения потока управления. На Рис. 3 приведен пример такой схемы, эквивалентной схеме, представленной на Рис. 2. Однако, предпочтительной схемой в данном случае является схема с парными разделениями и слияниями (Рис. 2.), так как такие схемы, несмотря на большее число содержащихся в них элементов, являются более понятными пользователям: в таких схемах участок между

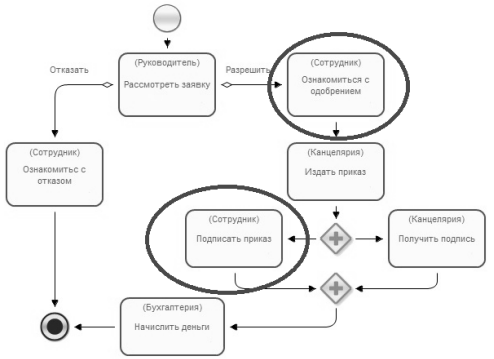


Рис. 1: Неправильная реализация бизнес-процесса со второстепенными действиями (выделены блокирующие второстепенные задания).

разделением и парным ему слиянием можно мысленно декомпозировать и таким образом разделить схему бизнес-процесса на две более простых. При наличии некоторой практики бизнес-аналитик может быстро «читать» такие схемы. В случае же больших схем с непарными элементами бизнес-аналитику требуется схему «расшифровывать» что требует гораздо больше времени и усилий.

В качестве учебных задач на разработку бизнес-процессов имеет смысл давать студентам логистические задачи, задачи динамического программирования, задачи, задачи теории операций и т. п. При этом части схем бизнес-процессов, реализующих эти задачи, будут представлять собой блок-схемы соответствующих математических алгоритмов.

Примеры учебных задач.

- Игры
- Быки и коровы
- Морской бой
- Студент и преподаватель проводят эксперимент
- Дилемма узника
- Задача о дуэли
- Задача о разборчивой невесте

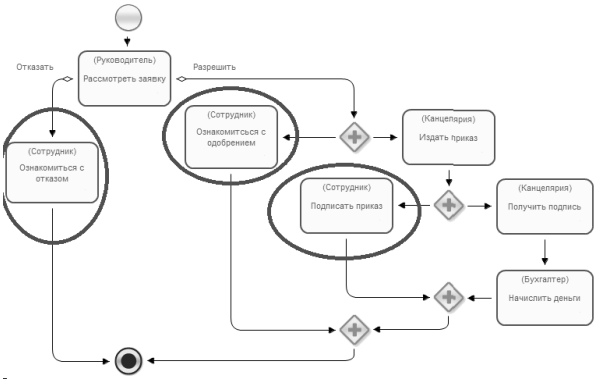


Рис. 2: Правильная реализация бизнес-процесса со второстепенными действиями (выделены второстепенные задания, не являющиеся блокирующими).

Задача Мартина Гарднера о разборчивой невесте

В качестве примера приведем построение бизнес-процесса выбора жениха, соответствующего задаче о разборчивой невесте.

Описание бизнес-процесса: Пришло время принцессе выбирать себе жениха. В назначенный день явились N царевичей. Их построили в очередь в случайном порядке и стали по одному приглашать к принцессе. Про любых двух претендентов принцесса, познакомившись с ними, может сказать, какой из них лучше. Познакомившись с претендентом, принцесса может либо принять предложение (и тогда выбор сделан навсегда), либо отвергнуть его (и тогда претендент потерян: царевичи гордые и не возвращаются). Если невеста отвергла всех женихов, то она уходит в монастырь.

Пример реализации (основной бизнес-процесс):

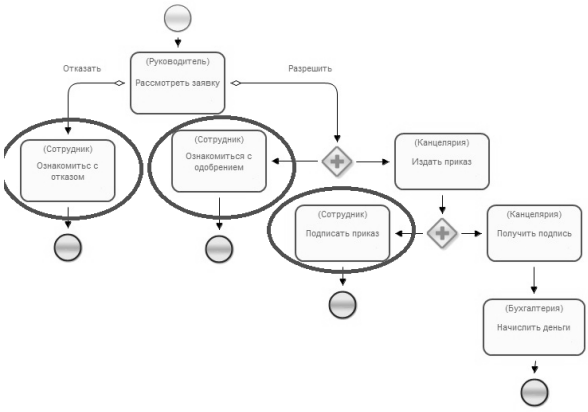


Рис. 3: Альтернативная реализация бизнес-процесса со второстепенными действиями (выделены второстепенные задания, не являющиеся блокирующими).

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

На основе согласованных наборов специально подобранных бизнес-ситуаций и вариантов процессных решений для них можно составлять учебные курсы, посвященные приемам построения эффективных решений процессной автоматизации. В докладе представлен опыт обучения студентов элементам этой технологии, полученный в НИТУ МИСиС и МЭСИ [1-3]. Методику обучения легко перенести в другие ВУЗы, так как она построена на использовании свободного программного обеспечения (система RunaWFE [4]), доступного через интернет, не требующего оплаты или регистрации.

Свободное ПО позволяет выполнять и проверять с его помощью практические работы студентов не только в учебных классах, но и на домашних компьютерах студентов и преподавателей. Разработанные бизнес-процессы можно свободно передавать другим лицам без каких-либо затрат на ПО, регистраций, получения лицензий и т.п. Таким образом, преподаватели могут осуществлять кооперацию при разра-

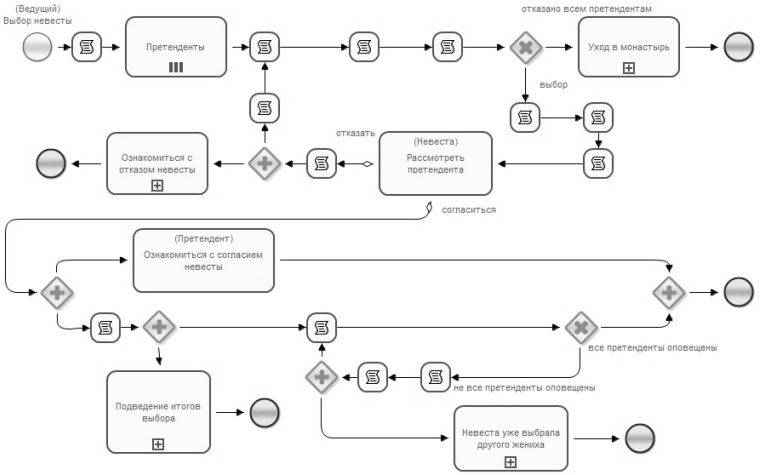


Рис. 4: Реализация основного бизнес-процесса выбора жениха из задачи М. Гарднера о разборчивой невесте (подпроцессы простые и не приведены).

ботке учебных курсов, а также обмениваться различными идеями, относящимися к процессному управлению. Разработанные в рамках учебной или научной деятельности бизнес-процессы можно внедрить на предприятии без дополнительных расходов предприятия на покупку ПО. Также возможно участие студентов и преподавателей ВУЗов в разработке и тестировании системы RunaWFE, что позволяет полнее учесть в продукте потребности данного ВУЗа.

Литература

[1] 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

[2] 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.

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

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

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

Михаил Быков
Москва, http://diglossa.org
http://morph.diglossa.org/sa

Практический морфоанализатор санскрита — Морфей

Аннотация

Создан практический морфоанализатор санскрита, то есть программа, определяющая словарное значение и морфологическое описание по словоформе (обратная задача). Практический — потому что (прямая) задача порождения всех словоформ по словарному значению (гораздо более сложная) — не решалась. Вместо этого использовались тесты (словарное значение — словоформа), скопированные из сети. Программа имеет более 10 тыс. тестов

На примере санскрита (который считается сложным языком) показано, что задача создания морфоанализатора для флективного языка не так уж и страшна, не боги горшки обжигают. В v.1 анализатора Морфей обрабатываются отдельные слова. Тесты проходят для более чем 98% частотного словаря, без учета составных слов. v2 будет уметь обрабатывать и составные слова. Что даст 98% на реальном частотном словаре.

Для разработки выбран nodejs, вместо ruby. Это сделано из соображений удобства — один язык в браузере, в couchDB и собственно в анализаторе. Модули программы используются во всех трех средах без изменения. Кроме того, переход с руби ускорил работу программы в 3–4 раза (оценка на глазок).

Недостатком выбранного подхода является то, что программа находит правильное словарное значение по словофрме в данном тесте. Но кроме того она находит еще множество иных словарных значений. Также правильных — в большинстве случаев. Однако тест не позволяет определить, нет ли в результате лишних значений. Но определяет только то, что среди найденных значений привильное — есть.

Для того, чтобы решить эту проблему, нужно уметь решать прямую задачу — порождать все словоформы по словарному значению. Для санскрита это возможно с помощью использования алгоритма, описанного в книге «Аштадхяйи», написаной Панини ок IV в. ВС. Я планирую сделать это в v.3, если Бог даст.

Аналогичные прграммы (использованные мной для порождения тестов) можно посмотреть здесь:
https://sites.google.com/site/sanskritcode/
http://sanskrit.inria.fr/DICO/grammar.html
http://spokensanskrit.de/
http://sanskrit.jnu.ac.in/kridanta/ktag.jsp
http://sanskrit.uohyd.ernet.in/scl/skt_gen/kqw/kqw_gen.html
http://tdil-dc.in/san/skt_gen/generators.html
http://sanskrit1.ccv.brown.edu/tomcat/sl/VerbalMorphology

Сергей Мартишин, Владимир Симонов, Марина Храпченко
Москва, Институт системного программирования РАН, Российский государственный социальный университет, кафедра моделирования информационных систем и сетей факультета информационных технологий
http://www.ispras.ru

Разработка информационных систем с использованием СПО NoSQL СУБД MongoDB

Аннотация

Описаны принципы функционирования систем управления базами данных (СУБД) NoSQL. В качестве примера рассмотрены особенности и представлено описание возможностей NoSQL СУБД MongoDB документоориентированной СУБД с открытым кодом.

В процессе подготовки студентов и магистрантов по направлениям «Информационные системы и технологии», «Информатика и вычислительная техника», «Программная инженерия» и ряду других,

необходимо ориентировать будущих специалистов на использование технологий, наилучшим образом подходящих для выполнения стоящих перед ними задач [5]. Поскольку приоритетным направлением их деятельности будет разработка и поддержка различного рода информационных систем (ИС), необходимо, чтобы студенты были знакомы со всем многообразием средств разработки и проектирования ИС. В этой связи представляется важным изучение студентами помимо реляционных СУБД также СУБД NoSQL («не только SQL»).

Появление СУБД NoSQL связано с тем, что за последние 15-20 лет значительно изменился характер информации, подлежащий обработке и, соответственно, изменились интерактивные приложения.

Во-первых, в значительной степени возросло количество пользователей, использующих приложения, причем часто одновременно, в том числе и мультмедийную информацию на мобильных устройствах. Во-вторых, значительно увеличился объем данных, подлежащих хранению и обработке. В-третьих, значительное количество хранимой информации сложно структурировать, используя технологию систем управления реляционными базами данных (реляционных СУБД).

Таким образом, необходимость быстрой обработки информации в условиях работы в многопользовательском режиме стала еще актуальнее, а традиционные системы с вертикальной масштабируемостью, такие как реляционные СУБД, перестали справляться с данной задачей. Горизонтальное масштабирование, основанное на нескольких независимых серверах, потребовало организации иной структуры данных — NoSQL, которые за счет отсутствия жесткой структуры обеспечивают более быстрый доступ к данным [1].

Заметим также, что в последние годы появилось значительное количество СУБД NoSQL 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.

Однако следует заметить, что СУБД NoSQL не гарантируют выполнения требований ACID (Atomicity (Атомарности), Consistency (Согласованности), Isolation (Изолированности), Durability (Долговечности)), а также не имеют аналогов команд BEGIN TRANSACTION, COMMIT и ROLLBACK, в связи с чем их нежелательно использовать, например, для финансовых систем, где транзакции необ-

ходимы. Тем не менее, для систем со значительным количеством клиентов, в которых приоритетными свойствами являются доступность и устойчивость к разделению (теорема CAP), использование NoSQL имеет значительные преимущества как для пользователей, так и для разработчиков: снижение времени отклика, использование различных типов хранилищ (в том числе облачных), возможность обойтись без создания схемы данных, за счет чего сокращается требуемое время разработки.

Согласно последним данным, MongoDB является наиболее популярной NoSQL базой данных, которая строится на JSON (JavaScript Object Notation) — подобных документах, хранящими данные в виде «ключ — значение» (документо-ориентированное хранение). MongoDB, кроме того, поддерживает такие возможности, как гибкую поддержку индекса, автоматический шардинг, большое количество запросов, встроенные репликации, поддержку спецификации хранения GridFS для хранения и извлечения файлов, которые превышают предельный размер BSON-документ 16 Мб, и многое другое [2].

Необходимо обратить внимание, что MongoDB не поддерживает объединение коллекций, то есть оператор JOIN в привычном смысле не функционирует. Для возможности масштабирования MongoDB информация хранится в виде коллекций *документов* с соответствующими данными в документах, таким образом, чтобы устранить необходимость в соединениях с другими коллекциями. Тем не менее, в некоторых случаях имеет смысл хранить соответствующую информацию в отдельных документах, как правило, в разных коллекциях или базах данных. В MongoDB, например, можно сохранить значения поля **_id** одного документа в другом документе в качестве ссылки. Тогда приложение может выполнять запрос данных из второго документа. Эти ссылки являются простыми, и для большинства случаев их использования достаточно. Для использования документов из разных коллекций применяются ссылки из одного документа в другой с указанием значения поля первого документа **_id**, названия коллекции, и, при необходимости, имени базы данных [2].

Заметим, что в плане работы с документами — их созданием, изменением, удалением, а также для работы с запросами в MongoDB, существуют методы, которые принципиально не отличаются от известных операторов языка SQL. Индексирование в MongoDB в основном аналогично индексированию в реляционных базах данных. Таким образом, первоначальное освоение студентами MongoDB не должно

вызвать сложности. Однако дальнейшее изучение ее возможностей позволит студентам понять преимущества и недостатки использования MongoDB применительно к конкретной разрабатываемой ИС [3].

Литература

[1] Фаулер М., Садаладж П. Дж. *NoSQL: новая методология разработки нереляционных баз данных — NoSQL Distilled.* — М.: «Вильямс», 2013. — 192 с. — ISBN 978-5-8459-1829-1

[2] MongoDB. [Электронный ресурс]. Режим доступа: URL: `http://www.mongodb.org` — Яз. англ. Дата обращения: 30.11.2014.

[3] С. Д. Кузнецов, А. В. Посконин. *Распределенные горизонтально масштабируемые решения для управления данными.* Труды института системного программирования РАН, том 24, 2013, с. 327-358. ISSN 2220-6426 (Online), ISSN 2079-8156 (Print).

[4] 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.

[5] 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.

Стас Фомин
Москва, ИСП РАН, НТЦ ИТ «РОСА»
`http://wiki.4intra.net/User:StasFomin`

MediaWikiQuizzer или ВикиЭкзамены — тесты, удобные и для преподавателя и для студента

Аннотация

«Тесты с вариантами для проверки знаний» — классический подход для дистанционного и очного обучения. Однако все это работает, когда их легко создавать и поддерживать, когда удобно и преподавателю и студенту. Иначе получается взаимное мучение. Автору доклада, в свое время не удалось найти такой системы, и пришлось изобрести свое, на основе широко известной открытой вики-системы MediaWiki, и ряде собственных доработок MediaWiki, также с открытым исходным кодом.

Расширенная аннотация

Технологии современного образования для достижения эффективности и масштабируемости, должны максимально разгружать ключевой ресурс — преподавателя, и при этом быть привлекать и увлекать студентов. Одним из аспектов современных подходов является автоматическая и дистанционная проверка учащимся своих знаний, даже если сам курс не является дистанционным. Такая проверка нужна не только для «выставления оценок», но и для самостоятельной тренировки ученика, поиска «дыр» в своих знаниях. Классическим, наиболее автоматизируемым подходом для решения таких задач является «автоматическое тестирование с вариантами ответов», которое используется во всех системах дистанционного обучения.

Однако, практически во всех системах, с которыми приходилось сталкиваться автору, при разработке таких систем, почему-то делается упор на что угодно — совместимость с стандартами SCORM, интеграцией с учетными системами ВУЗов, и даже межвузовским взаимодействием [2,3] ... на всем, кроме самого важного — удобства создания и обновления этих самых тестов. Обычно это очень неудобный интерфейс, где путем HTML-форм, медленно и уныло забиваются вопросы, ограниченные унылым плоским текстом, или наоборот, замусоренные HTML-разметкой.

В результате очень сложно:

- обновлять тесты (с хранением истории);
- вести совместное редактирование;
- использовать продвинутое форматирование — формулы, графы, изображения;
- отслеживать эффективность и автоматический поиск ошибок в тестах — ибо тесты с ошибками, или просто неудачно сформулированные вопросы чудовищно демотивируют обучающегося («ну вас к черту, я же ответил правильно!»);
- строить целостные варианты (например, комбинировать одни и те же вопросы в различные блоки, в зависимости от содержания курсов).

В целом, проблема в неудобности и недоделанности CMS (Content Management Systems) для ведения тестов, в результате:

- разработкой тестов обычно занимаются не преподаватели, а студенты, которых не жалко заставить заниматься трудоемким забиванием текста в формы;
- вопросы страдают примитивизмом, часто составляются по букоедскому принципу, накрывая без понимания предложения из текста курса (в стиле «Мать грозит ему в окно... — А кто грозит ему в окно? А мать грозит ему куда? А мать что делает ему в окно?»), и а затем тесты не изменяются, что приводит к мучению учащихся, (см. отзывы к курсам на сайте intuit.ru);
- от всего этого страдают студенты («ответы неправильные, вопросы дуратские, этот тест невозможно пройти правильно, ну все это нафиг»).

Решение автора — взять одну из самых удобных CMS, MediaWiki, и расширить ее функциональность средствами контроля представленных знаний. Наиболее распространенная IT-технология контроля знаний — это тесты, являющие собой набор сформулированных вопросов, снабженных вариантами ответа, часть из которых являются правильными, а часть — нет. Это дает как максимальную простоту тестирования, так и точность проверки.

Проектировщику обучающих материалов нужно только сформулировать проверочную структуру — набор вопросов, каждый из которых имеет варианты правильных и неправильных ответов, и возможно разьяснение.

Базовые технологии MediaWiki немедленно дают возможности:

- Удобного совместного редактирования с историей — тесты (вопросы+варианты ответа) ведутся в «плоских» статьях (а не в неудобных веб-формах), в результате их очень легко редактировать, перенося блоки текста из одного вопроса в другой, заводя варианты одного и того же вопроса, или модифицируя «засвеченный» студентами вопрос.
- MediaWiki-разметка является прекрасным балансом между возможностями форматирования, и незамусоренной читаемостью.
- Можно, используя механизмы шаблонов MediaWiki, включать в композитные тесты блоки вопросов, и создавать разные тесты для разных курсов, в зависимости от рассмотренных тем, и при этом не терять целостность — т.е. исправление какого-либо теста немедленно исправит ошибки и во всех включающих его тестах.

Дополнительные возможности, с учетом расширений проекта MediaWiki4Intranet [4] делают MediaWiki идеальной технологией для ведения вопросов, и даже целиком дистанционных курсов. В MediaWiki4Intranet можно использовать и удобную для ручного ввода Wiki-разметку, и HTML-код, и даже LaTeX-верстку, которая дает неограниченные возможности в представлении математики. Также можно размещать различные диаграммы (посредством Graphviz, Gnuplot, PlantUML), видеоролики, а также просто изображения, включая векторные. Доработаны возможности экспорта статей, так, что в результате очень легко экспортировать-импортировать блоки вопросов (включая необходимую графику и т.п.) между различными системами.

И самое главное, с помощью расширения IntraACL из проекта MediaWiki4IntraNet, реализовано самое необходимое — разграничение прав, чтобы «спрятать» вопросы от студентов, ну или даже ввести сложное дискреционное разграничение прав, позволяя часть вопросов редактировать даже студентам, а часть прятать...

Остается дело за малым — сделать саму систему тестирования.

Автором давно (с 2006 года) разработано расширение MediawikiQuizzer [5], развитие которого с тех пор непрерывно продолжается сообществом.

Оно позволяет описывать, тесты в виде вики-статей, синтаксис которых примерно понятен из примера:

```
;Title: Эффективные алгоритмы
;Intro: Тест по курсу «Эффективные алгоритмы»
;Shuffle choices: yes
;Shuffle questions:  yes
;Limit questions: 3

== Question: Полиномиальный в среднем для упаковки  ==
Если алгоритму из темы про полиномиальный в среднем алгоритм упаковки
подать на вход единичную матрицу инцидентности, он, если считать от
длины входа, затратит время ...

=== Answers ===
* Correct answer: экспоненциальное
* линейное
* квадратичное
* полином, но степени больше 2
* <m>np^2 \ln m</m>

== Вопрос: Complexity-P-NP-ZPP-PP-OK ==
Это правильная схема вложенности классов?
<graph>
digraph G{
  P -> ZPP;
  ZPP->RP;
```

```
  ZPP->coNP;
  ZPP->PP;
  RP->PP;
  coNP->PP;
}
</graph>

=== Ответы ===
* Правильный ответ: Да
* Нет
```

и получать:

- Автоматическую систему тестирования — с рандомизация порядка вопросов и вариантов ответа, с счетчиком времени и автоматическим подсчетом очков в зависимости от сложности случайного прохождения.
- Систему дистанционного обучения — работа в режиме «тьютора», показывающая и объясняющая учащемуся его ошибки.
- Возможность быстро подготовить бумажные варианты тестовых заданий, для проведения экзамена там, где нет интернета.
- Выборку подмножества вопросов из полной базы, что удобно, предоставляя студентам для тренировки тестовые наборы вопросов (например, три или пять случайных вопросов) — это побуждает их отвечать, ибо быстро и несложно.
- Ведение журнала сдачи тестов — очень интересно наблюдать за активностью студентов, и ростом активности в ночь перед экзаменом.
- Расчет статистики правильных ответов для всех вопросов, чтобы ориентируясь на них, либо исправлять неудачные вопросы с низким процентом правильных ответов, либо усложнять легкие.
- Автоматическое исключение вопросов либо большим процентом неверных ответов, либо более легких — т.е. нет смысла следить за курсом, неудачные по статистике вопросы будут автоматически исключаться, и экзаменационные вопросы можно отбирать из тех вопросов, которые прошли уже обкатку в тестировочных раундах.
- Геймификацию — прошедшим тесты студенты могут получить электронное подтверждение/сертификат — «графический диплом» или даже QR-код с подтвержающей ссылкой, который они смогут считать мобильным телефоном.

И наоборот, некоторые распространенные «фичи» дистанционных тестов мы не стали реализовывать специально. Например, мы принципиально против тестов вида «выбери M правильных вариантов из N» — т.е. экспоненциальный взрыв сложности вопроса, вместо нормальной линейной, при одном правильном варианте. Только тесты с одним правильным ответом можно нормально сбалансировать по сложности ответа, и если кто не в курсе — классические западные тесты (GRE, GMAT...) не только содержат единственный правильный ответ, но и фиксированное (обычно ровно пять) число вариантов.

Также, тесты можно проходить даже не зарегистрировавшись, и в целом, нет специализированной интеграции с классовым учетом студентов — и это даже хорошо, ибо снимает барьер для студентов, мотивируя их «играться», не стесняясь неправильных ответов.

Система уже почти десять лет использовалась и для академического и для внутрикорпоративного обучения, упоминалась и в докладах конференций [1], т.е. система в стабильна, и при этом продолжает развиваться.

Живьем систему можно посмотреть на сайте курсов по алгоритмам Института Системного Программирования РАН [6].

Разумеется, автор не призывает использовать тесты, как единственный возможный источник оценки знаний, и тем более, автоматически конвертировать оценки за тесты в оценку на экзамене. Такой подход методологически вреден — ведь при этом, как известно, студенты будут «зубрить» ответы на тесты, игнорируя глубокое понимание темы. Нужны и задачи для совместного решения и лабораторные работы (что автором также реализовано на MediaWiki4IntraNet), и устное общение на экзамене...

Но. В любом случае,

- даже «заученные тесты» приносят пользу — запоминаются определения, хотя бы на уровне «вспомню, где это было, чтобы потом посмотреть».
- на экзамене, тесты мгновенно позволяют
  - увидеть «дыры» в знаниях («не успел прочитать эту тему»), которые потом можно проверить задачами или устным общением.
  - отсеять совершенно неготовых студентов, и сэкономить временной и ментальный ресурс преподавателя — «Видишь, на тест ты ответил хуже, чем случайный выбор. Ты не просто

ничего не знаешь, более того — у тебя проблема с кармой, если ты отвечаешь хуже игрового кубика».

- легкость ведения базы вопросов делает возможным полное покрытие тестами каждой темы любого обучающего материала, практически, как написание тестов для программного кода.

Литература

[1] *«Все блюда для интранета из MediaWiki: ВикиБлоги, ВикиПрезентации, ВикиЭкзамены и ВикиЗакладки»*, `http://lib.custis.ru/312-all-from-mediawiki-add-2010`

[2] *«Распределённая система автоматизированного тестирования»*, `http://talks.rosalab.com/20140125-8`

[3] *«Технические и организационные аспекты внедрения СДО Moodle в образовательной организации»*, `http://talks.rosalab.com/20140126-B`

[4] *Проект MediaWiki4Intranet*, `http://wiki.4intra.net/MediaWiki4Intranet`

[5] *Проект MediawikiQuizzer* `http://wiki.4intra.net/MediawikiQuizzer`

[6] *Курсы по алгоритмам ИСПРАН и МФТИ*, `http://discopal.ispras.ru`

Воронин Игорь Вадимович
Шатура, ИПЛИТ РАН

Проект: Образовательный проект — УМКИ `http://umki.vinforika.ru/`

Разработки образовательных роботов по проекту УМКИ

В данной работе обсуждаются вопросы обучения основам робототехники на основе использования среды КУМИР и языка Скретч базовым алгоритмам управления и дистанционной связи множества разнообразных устройств в единую сенсорную сеть. Обсуждаются возможные варианты образовательных роботов для разных классов средней школы. Проект УМКИ основан на СПО, использование в нем программной среды КУМИР увеличивает наглядность и повышает удобство в обучении программированию передвижных роботизированных платформ.

В настоящее время в Российской Федерации сформирован и реализуется комплекс стратегических задач, направленных на развитие технического образования. Так министр образования и науки РФ Дмитрий Ливанов в рамках проходящего в 2014 году в Сочи международного форума «Дни робототехники» заявл, что основы робототехники будут включены в образовательную программу российских школ. Спецкурс войдет в уже имеющийся предмет «Технология». По его словам, так называемым урокам технического творчества, которые уже проводятся для всех учащихся с пятого по девятый класс, будет добавлен курс по созданию роботов. Это принципиально важно для конкурентоспособности нашей страны, так или иначе по этому пути идут все. И сегодня у нас есть все возможности, чтобы развивать такое направление как образовательная робототехника, — сказал там министр. Отрасль позволит школьникам на практике реализовывать знания и навыки, полученные в ходе изучения физики, математики и черчении. Конечно, не каждый в будущем станет Королевым или Джобсом, но у каждого школьника должен быть шанс попробовать свои силы. Доработка имеющихся и разработка новых экземпляров роботизированных платформ подразумевает работу по двум направлениям — аппаратному и программному обеспечению для управления устройствами.

Для импортозамещения предлагаются три варианта платформ SmartCar. Которые различаются по возможностям и по целевой аудитории.

Вариант 1. Для начальной школы 1-4 класс и дошкольного образования. Создание робота конструктора — **CAR-start**. Который умеет сканировать программу с кубиков или с пазлов, которые раскладываются на столе или на магнитной доске и ездить по этой программе по заданным промежуточным точкам. При этом происходит подготовка ребят к соревнованию — езда по точкам. Вместе с автоматическим управлением по программе этот вариант платформы может управляться в ручную с мобильника по блютусу. В этом случае, он может ездить вперед-назад, вправо-влево.

Состав:

- Контроллер который включает в себя - чип ATmega и чип Bluetooth,
- Корпус. Пока фанерный, обработанный, в дальнейшем возможно акрил вырезанный лазером или литой полистирол.

- Драйвер движков,
- Датчик фото-оптический для сканирования кубиков программы,
- Аккумулятор.

Аудитория:

- Сельские и малокомплектные школы.
- Младшие классы общеобразовательной школы
- классы коррекционной школы
- дошкольные учреждения

Программное обеспечение: C++ для программирования процессора, Андроид для управления устройством

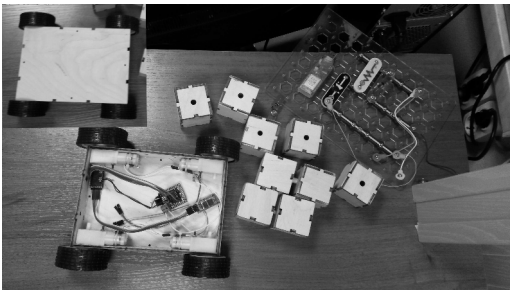


Рис. 1: На фото эскизный образец CAR-start с крышкой и без. Кубики и корпус пока еще не обработаны

Вариант 2. Средняя общеобразовательная школа 4-8 класс для занятий на уроках технологии, в дополнительном образовании, информатики и кружках экологии и природоведения. Платформа CAR-base, управляется из Исполнителей программы Кумир и Скретч. Так же может программироваться и управляться вручную. Имеет набор датчиков встроенных на борту. Объединяется в группы по технологии MESH ZigBee, понимает прием данных по радиоканалу формати API фреймов Состав:

- Контроллер который включает в себя — чип ATmega и чип ZigBee,

- Корпус. Пока акрил вырезанный лазером, в дальнейшем возможен литой полистирол.
- Драйвер движков.
- Датчики: Влажность, освещенность, задымления, температуры, давления, остаточной энергии батарей, датчики оборотов - энкодеры
- Батарейки

Аудитория:

- Общеобразовательные школы, средние классы.
- Уроки Технологии, кружки робототехники, подготовка к ГИА и соревнованиям.
- Лагеря отдыха детей.
- Учреждения дополнительного образования.
- Подготовка на базе набора к участию в конференции РоботоБУМ.

Программное обеспечение: gnu c++ 4.2, QT4, Кумир 2, Alt Linux 7 Gnome

Вариант 3. Роботизированный набора CAR-super, подразумевает использование в старших классах образовательной школы. Его контроллер состоит либо из ТРИК — либо из Расберипай. Связь с этой платыорма либо по протоколу ZigBee. Либо для доступа внутрь контроллера по WiFi. Плата датчиков и драйверы движков разнесены от платы контроллера. Дополнительные датчики могут присоединяться отдельно. Платформа комплектуется базовой конфигурацией и может дополняться навесными опциями, таким как видеокамера, рука — хват, манипулятор, маркер с системой поднятия-опускания, платформой для баскет бота. Из ПО — система распознавания контура объектов, система распознавания цвета объектов, управление четырьмя колесами разнонаправленного движения.

Состав:

- Контроллер который включает в себя: либо ТРИК, либо Raspberry Pi
- Корпус. Пока фанерный, обработанный, в дальнейшем возможно акрил вырезанный лазером или литой полистирол.
- Драйвер движков.

- Колеса либо стандартные, либо всенаправленные Omni Wheel Nylon
- Датчики: Влажность, освещенность, задымления, температуры, давления, остаточной энергии батарей, датчики оборотов - энкодеры
- 12 вольтовой аккумулятор

Аудитория:

- Общеобразовательные школы, страшие классы.
- Средне специальные учреждения
- Уроки Технологии, кружки робототехники, подготовка к ЕГЭ, олипиадам, соревнованиям.
- Лагеря отдыха детей.
- Учреждения дополнительного образования.
- Подготовка на базе набора к участию в конференции РоботоБУМ.

Программное обеспечение: gnu c++ 4.2, QT4, Кумир 2, Alt Linux 7 Gnome

Литература

[1] Хачко Д. В., Яковлев В. В, Субоч Н. М., Джелядин Т. Р., Ройтберг М. А., Кушниренко А. Г., Леонов А. Г. *Кумир 1.9. Практикумы и исполнители. Средства интенсификации обучения.* // Седьмая конференция «Свободное программное обеспечение в высшей школе». М.: Институт логики, 2012. с. 36.

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

[3] Электронный ресурс: КуМир `http://www.niisi.ru/kumir/`

[4] Электронный ресурс: Концепция ФЦП развития образования на 2016–2020 годы `http://government.ru/docs/16479/`

[5] Электронный ресурс: Проект KIWI Robotics `http://www.youtube.com/watch?v=dg7eWKvL3Rk`

Чернышов Л. Н., Махлай В. С.
Москва, Финансовый университет при правительстве РФ, Московский авиационный институт

Web-приложение для проведения контрольных и практических работ по программированию с автоматической генерацией заданий

Аннотация

Для преподавания дисциплин по различным разделам программирования актуальна автоматизация учебного процесса. Традиционные системы тестирования имеет ряд существенных недостатков. В работе предлагается подход, связанный с генерацией как тестовых заданий, так и обычных заданий на программирование. Задача, которую обучающийся должен запрограммировать на языке Phyton, формируется по определенному шаблону. При обработке результаты программы, выполняемой на сервере, сравниваются с правильными ответами, и формируется информация об успешности решения. Архитектура приложения позволяет подключать модули генерации задач по разным темам (например, по языку SQL).

Контроль знаний реализуется в большинство существующих обучающих систем. Эти системы обладают удобными средствами создания учебных курсов и наборов заданий. Однако вопросы формирования заданий, механизмы и их реализация, как правило, являются слабым местом таких систем. Автоматизация разработки заданий, генерация большого числа различных вариантов — актуальный вопрос, важность которого возрастает в современной системе образования [1–3]. Дисциплины программирования имеют ряд особенностей, которые требуют особого подхода к системам генерации заданий.

Обычное компьютерное тестирование имеет ряд недостатков, а именно: негативные психологические реакции обучающихся на компьютерное представление тестов, воздействие на результаты выполнения работ предшествующего опыта работы с компьютером, влияние интерфейса образовательной системы на результаты тестирования. Суждение об уровне знаний только по ответам проигрывает очной форме собеседования. Преимущество систем с генерацией практических заданий вполне очевидна.

Разработка преподавателем практических заданий производится предполагает следующие основные шаги:

- структурирование учебного материала в соответствии с поставленными целями, выделение ключевых понятий, входящих в область контроля;
- составление практических заданий и бланка ответов на них;
- экспериментальная проверка с целью предварительной оценки качества заданий;
- модификация заданий по результатам проверки, изменение шкалы и способа оценки, коррекция параметров генерации вариантов.

Наиболее трудоемким из перечисленных этапов является оставление вопросов для тестовых заданий и формирование индивидуальных вариантов для каждого обучаемого. Этот этап облегчается с переходом от фиксированных формулировок к параметризированным шаблонам, по которым формируются уникальные задания для каждого конкретного обращения к генератору. Схема работы генератора показана на рисунке 1.

Рис. 1: Общая схема работы генератора практических заданий.

База знаний содержит полную информацию относительно предметной области вопросов и их структуры, шаблоны формулировок задач, области изменения параметров вопросов, алгоритм нахождения решения или проверки наличия решения, алгоритм изменения значения параметров.

Параметры генерации заданий задают начальные значения параметров генерации и ограничения на тип генерируемых заданий. Начальные значения параметров генерации могут быть основаны на индивидуальных параметрах студентов, датчиках случайных чисел, на основании неких унифицированных величин (например, номера зачетки).

Базы параметров выданных практических работ и заданий предназначаются для идентификации задания, выполненное студентом, чтобы избежать мошенничества и недобросовестной работы со стороны испытуемых. Также база необходима для оперативного анализа и исправления ошибок.

Конечной целью работы генератора является сформированный уникальный набор заданий. Дальнейшее его предоставление пользователю для выполнения и формат отображения зависят от специфики используемого программного решения и его интерфейса, в качестве оного может выступать: web-решение, настольное приложение или мобильное приложени. В общем случае достаточно выгружать сгенерированное содержимое на общепринятом языке разметки для дальнейшего использования (например, xml).

Прототип размещен по адресу `http://testgeneration.evilejik.webfactional.com/`, репозиторий исходного кода — на GitHub: `https://github.com/evilejik/test_generation`. Приложение апробировано на трех видах задач: теоретические вопросы, разработка Python-программ на матрицы, задачи на язык SQL.

Литература

[1] Левинская М. А. *Автоматизированная генерация заданий по математике для контроля знаний учащихся.* — Educational Technology & Society 5(4), 2002.

[2] Кручинин В. В. *Генераторы в компьютерных учебных программах* / — Томск: Изд-во Том. Ун-та, 2003.

[3] Братчиков И. Л. *Генерация тестовых заданий в экспертно-обучающих системах* / — Журнал «Вестник РУДН» серия «Информатизация образования», 2012, №2.

Александр Гороховский
Киев, Украина, Национальный технический университет Украины «КПИ»
Проект: LaTeX4students http://kpi.ua/ru, http://donntu.org/

Опыт преподавания LaTeX студентам технических университетов

Аннотация

Рассмотрены особенности обучения студентов химических и экологических специальностей компьютерной вёрстке текстовых материалов (квалификационных, курсовых работ или проектов, лабораторных и практических отчетов, рефератов) совместно с рисунками, таблицами, спецификациями в соответствии с существующими ГОСТами, отечественными полиграфическими нормами в издательской системе LaTeX, в рамках дисциплины «Информационные технологии».

Начиная с 2007 года и по настоящее время, в рамках дисциплины «Информационные технологии», мной читается небольшой курс лекций и практических занятий, посвященных издательской системе LaTeX, для студентов химических и экологических специальностей, сначала в Донецком национальном техническом университете («ДПИ», г. Донецк), а затем в Национальном техническом университете Украины «КПИ» (г. Киев). Конечно, это непосредственно связано с практической возможностью использования LaTeX студентами в учебном процессе. Программы подобраны кроссплатформенные, т.е. в расчете на ОС GNU/Linux (Alt Linux, Ubunta, Debian) так и MS Windows.

Наверное, не является секретом то, что результатом выполнения студентами лабораторных и практических работ, сбора информации при прохождении практики, поиска технического решения или проведения любого научного исследования являются материалы, обработка и дальнейшее использование которых сегодня требует квалифицированного применения компьютерных программ. В большей степени это касается набора и редактирования текста, создания формул (математических и химических), таблиц, чертежей, схем, рисунков, ведения библиографии и просмотра всего макета перед получением окончательной печатной версии: реферата, отчета по лабораторным и практическим занятиям, дипломной, курсовой работы или проекта.

В настоящее время стандартными системами подготовки различных работ в электронном виде являются разнообразные реализации издательской системы TEX-LATEX-BibTeX (TeX Live, MiKTeX)[1, 2, 3] и WYSIWYG офисные программы Libre (Open), Microsoft Office.

Не останавливаясь на плюсах и минусах LATEX, хотелось бы рассмотреть логическую схему курса и его ключевые темы. Поскольку студенты — химики и экологи, не специализируются в области программирования, важным является показать процесс создания готовых документов с разнообразием формул, таблиц и рисунков, как достаточно простой, гибко настраиваемый и универсальный, к тому же не обременённый командами программирования.

Вначале студенты знакомятся как происходит работа с LATEX, а именно:

1. С помощью обычного или специального (`TexWorks`, `Kile`) текстового редактора подготавливается файл с текстом и минимальными командами LATEX.
2. Выполняется проверка орфографии встроенными в редактор средствами или с помощью внешних программ (ispell, aspell).
3. С помощью программы-транслятора `pdflatex` выполняют вёрстку этого документа в pdf-файл, который можно распечатать.
4. В случае неудовлетворительного результата — вносятся изменения в исходный текстовый файл, повторяя предыдущие шаги (1-3).

Концептуально TEX-нический документ рассматривается, состоящим из двух частей:

- *преамбулы-заголовка*, подключаемого `\input{head}`;
- *основной области* документа с текстом, TEX-командами, таблицами, рисунками и другими необходимыми элементами. Эта область начинается командой `\input{begin}` и заканчивается командой `\input{end}`[1].

```
\input{head}
\input{begin}
... текст документа ...
\input{end}
```

[1] В классическом исполнении должны быть заданы соответствующие им команды `\begin{document}` и `\end{document}`.

Между этими областями могут быть вставлены дополнительные пакеты командой `\userpackage{Пакет}`.

Фактически все необходимые настройки титульного листа документа (автор и руководитель работы, группа, дисциплина, тема работы, кафедра, ВУЗ, язык и пр.) спрятаны в файле `head.tex`[2] и они могут быть скорректированы. Файл `begin.tex` выполняет функцию: дополнительных настроек оформления всего документа, подключения листов титульного, реферата, ключевых слов и содержания работы. Он в корректировке не нуждается. Файл `end.tex` необходим для включения в документ, при необходимости, спецификаций к чертежам. Опциональными являются файлы реферата `ref.tex` и ключевых слов `kws.tex`. Все эти файлы студенты получают во время практических занятий.

Символы, пробелы, TEX-нические команды, блоки, декларации, окружения и символы-исключения, имеющие особый статус — следующая тема. Также внимание студентов обращается на автоматическое формирование переносов в словах, выравнивание текста абзацев по ширине страниц и их выделение с помощью пустых строк.

Следующая важная тема — **создание структурных элементов документа** с различным уровнем подчинения при помощи команд, приведенных в таблице 1.

TEX-команда	Аналог	Описание элемента
`\vstup`		Введение
`\razdel{Название}`	`\section{}`	Название раздела
`\prazdel{Название}`	`\subsection{}`	Название подраздела
`\pprazdel{Название}`	`\subsubsection{}`	Название пункта
`\vivod`		Выводы
`\bibliography{bib-файл(ы)}`		Список литературы
`\dodatok{Название}`		Название приложения

Таблица 1: Структурные элементы документа

Подчёркивается, что нумерация структурных элементов[3] и *создание оглавления* документа происходит автоматически при вёрстке,

[2] Основа — пакет `eskdx`, версия 0.97 / Константин Кориков `http://eskdx.org.ua/downloads/1`.

[3] За исключением *Введение*, *Выводы*, *Список литературы*.

поэтому в *Названии* не указывают ни каких номеров. Независимо от реального начертания *Название*, для раздела — оно автоматически формируется БОЛЬШИМИ буквами, а в оглавлении будет таким каким указано в команде. Приложениям автоматически присваиваются буквенные обозначения по ГОСТу.

Кроме этого, рассматривается **создание списков**: простых (не нумерованных), нумерованных, тематических и компактных[4] в окружениях `itemize`, `enumerate`, `description`, `compactenum`, `compactitem`; принудительных разрывов, переносов и выравниваний (по центру, левому или правому краю).

В следующей теме студенты знакомятся с понятием метка (`label`) и **созданием перекрёстных ссылок** на страницы документа, его разделы, подразделы, приложения, литературные источники, таблицы, рисунки, формулы, элементы списка и другие нумерованные элементы или перечисления. Здесь обращается внимание на классические команды `\pageref`, `\ref`, `\eqref`, `\cite` и „умные“[5] — `\cref`, `\Cref`, `\labelcref`.

Отмечается особенность формирования библиографических записей и меток в программе `JabRef`[6] и то, что порядок их следования в команде `\cite` не важен — в свёрстанном документе они будут напечатаны в квадратных скобках, в порядке возрастания номеров, а диапазон номеров — через короткое тире.

Далее рассматриваются наиболее часто встречающиеся в технических работах специальные символы: двойные кавычки («елочки», „лапки“), верхние и нижние индексы, №, °C, ∼, ≈, ±, ⩾, ⩽, ≠, корни, дроби и др., а также использование единиц длины (размерностей) — абсолютные (mm, cm, in, pt) и относительные (em, ex, `\textwidth`, `\textheight`)

Следующая важная тема — **создание математических формул** внутри текста и выделенных в отдельную строку. Здесь студенты знакомятся с понятием математического режима (моды), реализуемый внутри математических окружений и команд.

Отмечается, что в математической моде TEX во время вёрстки игнорирует все пробелы между символами в исходном тексте и расставляет промежутки сам. Формулы не должны содержать пустых

[4] Пакет `paralist` / Bernd Schandl `http://www.ctan.org/pkg/paralist` .
[5] Пакет `cleveref` / Toby Cubitt `http://www.ctan.org/pkg/cleveref`.
[6] `http://jabref.sourceforge.net`

строк. Математические символы, а также греческие буквы, которых нет на клавиатуре, печатаются специальными командами или с помощью специальных программ, например, TeXaide.

Для формул внутри текста их заключают в знаки `$ формула $`. Если формула не помещается на строке она автоматически разрывается по знакам и командам: `+ - = > < \*` $(\cdot)$ `\x` $(\times)$ с дублированием этих знаков на границах строк переноса.

Создают выделенные в отдельную строку ненумерованные формулы с их центрированием конструкцией:

```
$$ формулы $$     или     \[ формулы \].
```

а формул с их автоматической нумерацией — командами[7]:

```
       \eq[метка]{ формула }  или
\begin{equation}\label{метка} формула \end{equation}
```

Многострочные формулы с нумерацией каждой

```
1 \eqs{ формула1  \label{e:1}  \\    %(#.1)
2                 ...
3     формулаN  \label{e:N}      %(#.N)
4 }
```

К этим базовым конструкциям относят окружения: `split` — блок формул с общим номером и выравниванием по &; `gather` — блок формул с выравниванием по центру и другие.

Далее приводятся примеры математических формул и конструкций с различными символами (греческими буквами, математическими операциями, стрелками) и их расположением, а также команды, задающие систему уравнений, матрицы, выбора по условию. Заканчивается эта тема информацией о регулировании промежутков в формулах и тексте и оформлении пояснений элементов формул.

Следующая важная тема — **создание химических формул и уравнений реакций**. Химические формулы неорганических и простых органических соединений, изотопов и уравнений в TEX-документе предлагается задавать как аргумент команды `\ce`[8]. Вот несколько типичных примеров:

$[AgCl_2]^-$ ⟵ `\ce{ [AgCl2]- }`

$^{227}_{90}Th^+$ ⟵ `\ce{^{227}_{90} Th+}`

$\frac{1}{2}H_2O$ ⟵ `\ce{1/2H2O}`

[7] Пакет `nccmath` / Александр Роженко `http://www.ctan.org/pkg/nccmath`

[8] Пакет `mhchem`, версия $\geqslant 3$ / Martin Hensel `http://www.ctan.org/pkg/mhchem`.

$\mathrm{KCr(SO_4)_2 \cdot 12\,H_2O} \longleftarrow$ `\ce{KCr(SO4)2*12H2O}`

$\mathrm{A{-}B{=}C{\equiv}D} \longleftarrow$ `\ce{A\sbond B\dbond C\tbond D}`

```
\ce{        Zn^2+ <=>[\ce{+ 2OH-}][\ce{+ 2H+}]
  $\underset{\text{амфотерный гидроксид}}{\ce{Zn(OH)2 v}}$
    <=>C[+2OH-][{+ 2H+}]
  $\underset{\text{гидроксоцинкат}}{\cf{[Zn(OH)4]^2-}}$ }
```

$$\mathrm{Zn^{2+} \underset{+2\,H^+}{\overset{+2\,OH^-}{\rightleftharpoons}} \underset{\text{амфотерный гидроксид}}{Zn(OH)_2\downarrow} \underset{+2\,H^+}{\overset{+2\,OH^-}{\rightleftharpoons}} \underset{\text{гидроксоцинкат}}{[Zn(OH)_4]^{2-}}}$$

Для группы уравнений химических реакций предлагается использовать команду `\cee`:

Несколько уравнений реакций

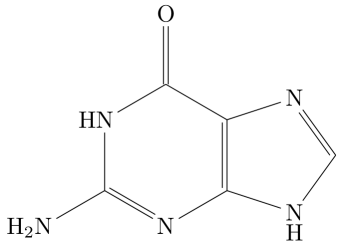
```
1 \begin{align*}
2     \cee{ RNO2 &<=>C[+e] RNO2^{-} \\
3         RNO2^{-} &<=>C[+e] RNO2^2-  }
4 \end{align*}
```

$$\begin{aligned}
\mathrm{RNO_2} &\overset{+e}{\rightleftharpoons} \mathrm{RNO_2^-} \\
\mathrm{RNO_2^-} &\overset{+e}{\rightleftharpoons} \mathrm{RNO_2^{2-}}
\end{aligned}$$

Как наиболее универсальный вариант создания самых сложных 2D/3D неорганических и органических структур для студентов рекомендуется использовать команду `\chemfig`[9]. Вот несколько интересных примеров:

Гуанин

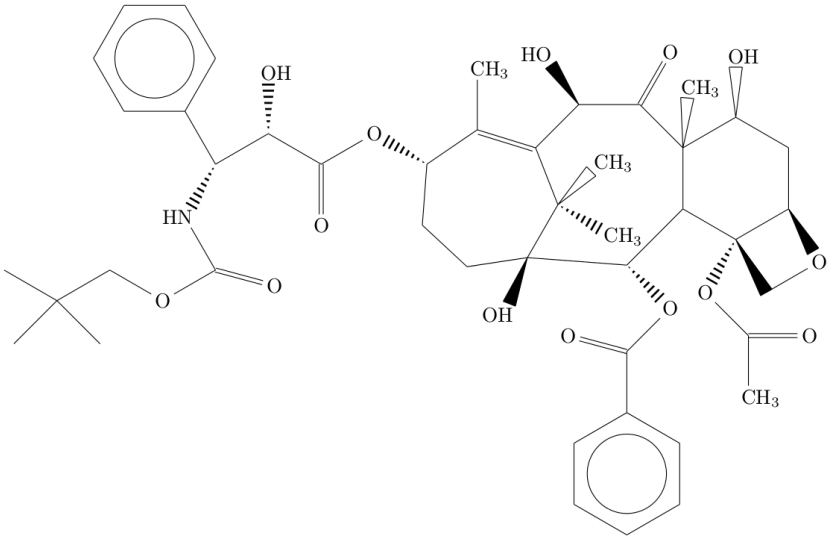
```
\chemfig{  *6((-H_2|N)=N-*5(-\chembelow{N}{H}-=N-)=-(=O)-HN-[,,2])  }
```



[9] Пакет `chemfig` / Christian Tellechea `http://www.ctan.org/pkg/chemfig`.

Таксотер

```
\chemfig{ -[::-30](-[5])(-[7])-[::+60]-[::-60]O-[::+60](=[::-45]O)-[::+90]
   HN>:[::-60](-[::+60]**6(------))-[::-30](<:[2]OH)-[::-60](=[6]O)
   -[::+60]O>:[::-60]*7(---?(<[::-120]OH)-(<|[1]CH_3)(<:[::-90]CH_3)
   -(-[1](<[::+80]HO)-[0](=[::+60]O)-[7](<|[::+130]CH_3)(-[::+75]
   (<|[2]OH)-[::-60]-[::-60](<[::+30]O-[::-90])-[::-60](<[::+90])
   (<:[::+30]O-[7](-[6]CH_3)=[0]O)-[::-60])-[6]-[5,1.3]?(<:[7]O-[5]
   (=[::-60]O)-[6]**6(------)))=(-[2]CH_3)-)   }
```



Создание небольших (плавающих) и протяжённых таблиц — следующая важная тема. Здесь предлагается универсальная команда `tablica` с 1-м опциональным (для протяжённых таблиц) и 5 обязательными параметрами:

Структура таблицы

```
\tablica  [Количество столбцов]
  {Метка таблицы}
  {Название таблицы}
  {Формат столбцов}
  {Шапка таблицы}
  {Содержимое таблицы}
```

Синтаксис параметров *Формат столбцов*, *Шапка таблицы* и *Содержимое таблицы* полностью совпадает с подобными в классических окружениях `tabular` и `longtable`[10]. Использование *Количество*

[10] Пакет `longtable` / `http://www.ctan.org/pkg/longtable`

столбцов позволяет автоматически разбивать материал таблицы на страницы, дублировать обозначения столбцов шапки и создавать надписи *Продолжение таблицы #.#* Здесь рассматриваются табличные команды `hline`, `cline`, `multicolumn`, `makebox`, `cbox` и `tc`[11].

Следующая важная тема — **включение графических материалов и рисунков**[12] в документ с помощью универсальной команды `foto`, у которой есть 1-й, 3-й и 5-й (последний) опциональные и 2-й, 4-й — обязательные параметры:

```
\foto[Размер]{Имя файла}[Пояснения]{Подпись к рисунку}[Строки обтекания]
```

Параметром *Размер* — указывают долю (0.1 . . . 1) от ширины печатной области страницы под рисунок. По умолчанию это 1.

Параметром *Строки обтекания* — указывают для небольших рисунков число строк текста, обтекающих рисунок слева. Если этот параметр указан (>0), то рисунок будет расположен *у правой стороны* страницы и он не будет «плавающим».

Следующая тема — **буквально воспроизведение листингов программ и результатов расчёта** командой `lstinputlisting` или окружениями `lstlisting`[13] и `Verbatim`[14]. Указывается, что окружением `comment`[15] можно обозначить блок текста, который будет полностью игнорироваться, т.е. *закоментирован*.

Вопросы **физической разметки текста**, меняющей внешний вид текста, обсуждаются в рамках классических команд влияющих на размер шрифта и команд `textit` (курсивный), `textsl` (наклонный), `emph` (выделенный), `texttt` (машинописный) `textbf` (полужирный текст), а для не шрифтового выделения текста — команды[16] `so` (разрядкой), `ul` (подчеркиванием), `st` (перечеркиванием) и `caps` (мелкописное начертание).

Заключительные темы — **альбомная ориентация** страниц с помощью окружения `landscape`[17] и **разбиение большого файла** документа на части командой `input`.

---

[11] Пакет `nccboxes` / Александр Роженко `http://www.ctan.org/pkg/nccboxes`
[12] Нужны векторный **PDF** или растровые **JPG, PNG**.
[13] Пакет `listings` / Jobst Hoffmann etc. `http://www.ctan.org/pkg/listings`
[14] Пакет `fancyvrb` / `http://www.ctan.org/pkg/fancyvrb`
[15] Пакет `verbatim` / Rainer Schöpf `http://www.ctan.org/pkg/verbatim`
[16] Пакет `soul` / Melchior Franz `http://www.ctan.org/pkg/soul`
[17] Пакет `lscape` / David Carlisle `http://www.ctan.org/pkg/lscape`

Практические занятия проходят в форме компьютерного практикума, где студенты вначале устанавливают и настраивают систему LaTeX, а затем верстают pdf-файл в соответствии с заданием методических указаний [4].

Литература

[1] *Роженко, А. И.* Искусство верстки в LaTeX'е / А. И. Роженко; Под ред. А. С. Алексеев. — Новосибирск: Изд-во ИВМиГК СО РАН, 2005. — 398 с.

[2] *Котельников, И.* LaTeX по-русски / И. Котельников, П. Чеботаев. — Новосибирск: Сибирский хронограф, 2004. — 489 с.

[3] *Львовский, С. М.* Набор и вёрстка в системе LaTeX / С. М. Львовский. — 3-е, испр. и доп. изд. — М.: Космосинформ, 2003. — 448 с.

[4] Методические указания к лабораторному практикуму по дисциплине «Компьютеризация управления производством». Цикл работ: TeX–LaTeX 2_ε–BibTeX. Для специальности 8.091606/7.091606 / Сост. А. Н. Гороховский. — Донецк: ДонНТУ, 2008. — 84 с.

Сергей Мартишин, Марина Храпченко
Москва, Институт системного программирования РАН

Проект: Название проекта(ов) URL проекта

Распределенные данные в модели облачных вычислений SaaS

Аннотация

Анализируется использование NoSQL («не только SQL») систем управления базами данных (СУБД) свободного и свободно распространяемого программного обеспечения (СПО) для манипулирования «Big Data» (большими данными) в модели облачных вычислений. Рассматриваются вопросы информационной безопасности.

Появление термина «Big Data» в 2008 году зафиксировало понимание проблемы роста объёмов и многообразия обрабатываемых данных. Эти данные возникают в самых различных сферах: научных исследованиях, с устройств аудио и видеорегистрации, социальных сетях и т.п. При этом необходимость обработки больших объемов как

структурированной, так и неструктурированной информации за разумное время стала еще более востребованной. Более того, часто операции необходимо производить в реальном времени и в многопользовательской среде. Для такого рода работы с данными необходимо иметь вычислительные системы соответствующей производительности. В последние годы для хранения «Big Data» используются NoSQL СПО СУБД, а для манипуляции с данными облачные сервисы.

Облачные вычисления (cloud computing) — технология, основанная на распределенной обработке данных, которая была предназначена для решения трудоемких вычислительных задач при помощи системы нескольких параллельно функционирующих компьютеров. Облачные сервисы основаны на клиент-серверной архитектуре, при этом ресурсы облака, такие как: процессорное время, оперативная память, сетевые каналы, программное обеспечение и пр. предоставляются пользователю как некоторый сервис. Доступ к сервисам пользователь получает через Интернет, используя клиентское приложение, например, браузер.

Облако [3] предоставляет одну из моделей обслуживания клиентов: IaaS — инфраструктура как услуга, например, использование дата-центра с собственным администрированием, PaaS — платформа как услуга, клиент размещает прикладное программное обеспечение (ПО) на платформе, администрируемой провайдером (платформа включает ОС, СУБД, ПО и т.п.), SaaS — программное обеспечение как услуга, в которой клиент использует ПО облачного провайдера, в том числе прикладное. Заметим, что на практике многие пользователи предпочитают использовать модель SaaS, поскольку для доступа к ПО могут использоваться тонкие клиенты и терминальный доступ, а за ПО, развернутое на стороне сервера, его обслуживание, поддержку, обновление, надежность функционирования и безопасность конфиденциальных данных отвечает провайдер.

Совершенно очевидно, что в такого рода распределенной системе хранение данных имеет некоторые особенности.

Традиционные реляционные (СУБД) с вертикальной масштабируемостью не предназначены для распределенного хранения неструктурированной информации. Горизонтальное масштабирование, основанное на нескольких независимых серверах, потребовало организации иной структуры данных. Наиболее популярными в настоящее время являются СПО СУБД NoSQL типа «ключ-значение» и документные (например, Redis, MongoDB, CouchDB и т.п.). За счет отсутствия

жесткой структуры и использования механизмов репликации (синхронизация содержимого нескольких копий объекта, например, базы данных) и шардинга (распределение данных между разными физическими серверами) они лучше масштабируются и обеспечивают более быстрый доступ к данным, что является необходимым в условиях работы в многопользовательском режиме [2].

В случае распределенного хранения наиважнейшим становится вопрос информационной безопасности, в соответствии с [5] в данное понятие входят три основных составляющих:

— конфиденциальность — доступ к информации могут получить только ограниченная группа легальных пользователей, имеющих соответствующие полномочия;

— целостность — защищаемая информация может быть изменена только санкционированными пользователями, она достоверна, то есть адекватно отображает состояние предметной области и не искажена;

— доступность — гарантия беспрепятственного доступа к защищаемой информации для санкционированных пользователей.

Поскольку СУБД NoSQL не гарантируют выполнения требований ACID (Atomicity, Consistency, Isolation, Durability), а также в соответствии с теоремой CAP (Consistency, Availability, Partition tolerance) в любой реализации распределенных вычислений можно обеспечить не более двух из трёх следующих свойств: согласованность, доступность и устойчивость, наиболее актуальным вопросом становится защита конфиденциальных данных.

При использовании модели SaaS пользователи автоматически получают возможность использовать специальные средства защиты данных от их несанкционированного использования: аутентификация и контроль прав доступа пользователей, использование брандмауэра, VPN, шифрования (например, хеширование паролей или криптопротоколы для данных внутри СУБД), доверенных вычислений на базе криптосерверов, например, Trusted Platform Module (TPM) [4].

Заметим, что несмотря на стандартные способы защиты данных, возможна их утечка: защищенные модули уязвимы для хакерских атак, а в некоторых случаях сама информация о том, что клиент C вычислил значение функции f может быть конфиденциальной. Также и само облако может являться активным или пассивным противником [1].

Именно поэтому теоретически наиболее подходящим криптографическим протоколом для организации облачных вычислений над конфиденциальными данными представляются пороговые криптосистемы полностью гомоморфного шифрования (FHE). Однако в настоящий момент возможности их практического применения требуют дополнительных исследований.

Литература

[1] Варновский Н. П., Мартишин С. А., Храпченко М. В., Шокуров А. В., *Методы пороговой криптографии для защиты облачных вычислений*, Труды института системного программирования РАН, том 26, выпуск 2, 2014, с. 269–274.

[2] Фаулер М., Садаладж П. Дж. *NoSQL: новая методология разработки нереляционных баз данных* — NoSQL Distilled. — М.: «Вильямс», 2013. — 192 с. — ISBN 978-5-8459-1829-1

[3] Peter Mell, Timothy Grance, *The NIST Definition of Cloud Computing* NIST Special Publication 800, 2011.

[4] Мартишин С. А., Храпченко М. В. *Анализ способов защиты информации в базах данных.* Девятая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов/ Переяславль 25–26 января 2014 года. Альт Линукс с. 130–132.

[5] Ronald R. Krutz, Russell Dean Vines. *The CISSP Prep Guide—Mastering the Ten Domains of Computer Security.* — John Wiley and Sons, Inc., 2001.