Колонка главного редактора

Пользуясь случаем, позволю себе рассказать о другом проекте, который занимает значимую часть моей жизни и тоже посвящен тематике FLOSS. 1 февраля мы запустили в рамках публичной беты качественно новую версию сайта www.nixp.ru, в первую очередь известного как информационный ресурс по GNU/Linux и UNIX-подобным системам, а также Open Source в целом. Затянувшийся процесс разработки позволил нам сделать из ресурса прообраз социально ориентированного веб-портала, который со временем «обрастет» множеством дополнительных интересных возможностей для удобного и взаимно полезного общения. Хочется верить, что это будет способствовать развитию и сплочению русскоязычного Open Source-сообщества.

Вообще появление и развитие подобных ресурсов в Рунете явным образом указывает на факт растущего числа пользователей открытого и свободного ПО, что не может не радовать. Для дальнейшей популяризации FLOSS-продуктов осталось лишь обеспечить этим новичкам достойную поддержку сообщества, которое на ведущих русскоязычных ресурсах славится своей разношерстностью: зачастую можно вместо помощи встретить сильную неприязнь от местных гуру. Впрочем, ситуация в последнее время исправляется, так что достаточные условия уже сформированы.

Главный редактор
Дмитрий Шурупов
(osa@samag.ru)

«Open Source»
электронное приложение к журналу «Системный администратор»
№55, 11 февраля 2010 г.

РЕДАКЦИЯ
Исполнительный директор
Владимир Положевец
Главный редактор
Дмитрий Шурупов
Верстка и оформление
Владимир Лукин
Сайт электронного приложения:
http://osa.samag.ru

Новости мира Open Source

Oracle завершила поглощение Sun: первые последствия

27 января корпорация Oracle официально объявила о завершении поглощения компании Sun Microsystems. В этот день Oracle провела презентацию из главного штаба. Мероприятие транслировалось в прямом эфире для клиентов и партнеров Oracle, а также для прессы и аналитиков. Oracle пообещала с приобретением портфолио продуктов Sun расширить линейки своих продуктов и услуг, благодаря чему «клиенты выиграют в производительности, надежности и безопасности эксплуатируемых систем, а расходы на интеграцию этих систем и их обслуживание понизятся».

Первые кадровые перестановки, последовавшие за объединением компаний, вызвали весьма неоднозначную реакцию. С одной стороны, Кен Якобс (Ken Jacobs), один из ветеранов Oracle, покинул корпорацию и поделился своим оптимистичным взглядом на будущее СУБД с открытым кодом MySQL: «Я знаю, что в Open Source-сообществе полно людей, скептически относящихся к планам Oracle насчет MySQL. Они зря беспокоятся. Oracle продолжила развивать MySQL и помогать этому бизнесу расти. Oracle сделает MySQL лучше. И такой подход позволит Oracle выйти на новые рынки, а ресурсы и инвестиции корпорации означают хорошие перспективы как для покупателей, так и для сообщества».

Но с другой стороны, уже вскоре разработчики GNOME Accessibility, занимающиеся созданием технологий и приложений для людей с ограниченными возможностями, выразили свое недовольство политикой Oracle. Оно было вызвано тем, что корпорация уволила нескольких участников их проекта, которые раньше работали в Accessibility Program Office (APO) компании Sun Microsystems. Вскоре в Интернете появилось открытое письмо, автор которого просит корпорацию Oracle пересмотреть свой взгляд на те проекты, которыми занимались представители Sun Accessibility Program Office: «Решение Oracle грозит тем, что многие люди с ограниченными возможностями со всего мира лишатся современного окружения рабочего стола. Я нахожу это трагичным».

Тед Тсо, автор ext4, перешел из Linux Foundation в Google

Теодор Тсо (Theodore «Ted» Ts'o), который был техническим директором организации Linux Foundation чуть больше года, перешел в компанию Google для внедрения файловой системы ext4 в ИТ-инфраструктуру интернет-гиганта.

Тсо хорошо известен в Linux-сообществе как один из ключевых авторов файловых систем ext3 и ext4. В середине января стало известно о том, что в Google завершили тестирование нескольких файловых систем (JFS, XFS, ext4) для того, чтобы перевести на одну из них свою существующую Linux-инфраструктуру (на базе ext2). Результаты тестирования оставили Google выбор между ext4 и XFS, но ключевым аргументом в пользу ext4 стала простота перехода на эту файловую систему с используемой до сих пор ext2.

Теперь именно Теодор Тсо поможет Google с переходом на современную файловую систему ext4. Кто станет новым техническим директором Linux Foundation, пока не сообщается.

21% немцев используют офисный пакет OpenOffice.org

По данным ИТ-компании Webmasterpro.de, более 21% немецких пользователей персональных компьютеров работают с OpenOffice.org в качестве офисного пакета.

Такие статистические данные были получены в результате исследования более 1 миллиона немецких интернет-пользователей. Само исследование проводилось с помощью сервиса FlashCounter Statistics Service, который собирает данные более чем со 100 тысяч веб-страниц. Определение установленных в системе офисных пакетов осуществлялось на основе проверки представленных в системе шрифтов.

21% – это сумма голосов за сам пакет OpenOffice.org и другие офисные продукты, основанные на нем: StarOffice и IBM Lotus Symphony. Результаты опроса по использованию других офисных пакетов немцами таковы:

- ☑ Microsoft Office – 72%;
- ☑ Corel WordPerfect – 2,7%;
- ☑ Apple iWork – 1,4%;
- ☑ SoftMaker Office – 0,3%;
- ☑ KOffice – 0,03%.

Кроме того, у 17,1% пользователей вообще не установлены офисные пакеты на компьютерах. По заявлениям исследователей, погрешность в оценке числа пользователей Microsoft Office может составлять 10%, а всех других офисных продуктов – 0,5%.

В GIMP 2.8 появится однооконный интерфейс

В грядущем релизе популярнейшей Open Source-программы для редактирования растровой графики, GIMP 2.8, появится однооконный интерфейс, которого многие пользователи ждали уже не один год.

Исторически у GIMP сложился многооконный интерфейс, т.е. разные компоненты редактора (панель инструментов, само изображение, дополнительные инструменты вроде панели слоев) были доступны в разных окнах. Это вызывало ряд неудобств у пользователей, и вот разработчики графического редактора проанализировали проблему, провели мозговой штурм вместе с Open Source-сообществом и пришли к выводу, что реализация классического однооконного интерфейса (как, например, у Adobe Photoshop) необходима.

В нестабильном релизе GIMP 2.7.1 уже сейчас можно увидеть результаты работы над этой проблемой. Теперь все панели графического редактора располагаются в едином окне – их можно перемещать по усмотрению или «отцеплять» в отдельные окна, возвращая GIMP «привычный» для него интерфейс.

Первый стабильный релиз GIMP 2.8 запланирован на декабрь 2010 года.

У Paint.NET появился аналог на базе GTK# – Pinta

Джонатан Побст (Jonathan Pobst) из компании Novell представил первый релиз графического редактора Pinta, позиционируемого как аналог Paint.NET, основанный на библиотеке GTK#.

Проект начался с того, что разработчик увидел потребность в подобном приложении и решил «поиграть несколько часов» с Cairo и GTK# – подобного опыта у него еще не было. Создав первый холст для картины и кисточку, он решил развить успех и примерно через неделю получил очень простую программу для рисования. После этого Джонатан продолжал разработку в течение месяца и теперь представил публике первый релиз получившегося детища – проект нового графического редактора Pinta.

Возможности программы пока скромны,среди них отмечаются основные инструменты для рисования (кисточка, карандаш, готовые формы, ластик, различные виды выделения частей изображения), поддержка слоев, неограниченная история операций, простые эффекты (инвертирование, сепия и другие). Pinta распространяется под Open Source-лицензией MIT X11. Программа доступна в виде исходников, а также готовых пакетов для openSUSE и Ubuntu Linux на сайте pinta-project.com.

Symbian Foundation открыла исходный код ОС Symbian

Организация Symbian Foundation, как и обещала еще в 2008 году, открыла исходный код мобильной операционной системы Symbian.

Symbian – самая популярная в мире операционная система для смартфонов. Одним из ключевых производителей мобильных телефонов, который в свое время сделал на нее ставку, является Nokia. Именно финская компания купила разработчика этой системы Symbian Software Limited и создала новую организацию Symbian Foundation, объявив о том, что в дальнейшем Symbian станет Open Source-платформой.

И вот стало известно о том, что Symbian Foundation начала свободное распространение исходного кода Symbian под Open Source-лицензией Eclipse Public License (EPL). Как сообщил в интервью BBC News представитель Symbian Foundation Ли Уильямс (Lee Williams), инициатива Symbian Foundation – «это крупнейшее открытие исходного кода за всю историю».

Раньше исходный код Symbian был доступен только участникам организации Symbian Foundation, куда помимо Nokia входят такие компании, как Motorola, Samsung, Sony Ericsson, AT&T и Vodafone. Разработкой операционной системы Symbian на данный момент преимущественно занимаются специалисты из Nokia, но в компании планируют, что к 2011 году их вклад будет составлять не более 50%.

Исходный код платформы Symbian можно будет скачать с www.symbian.org.

В Ubuntu Netbook Remix заменили OpenOffice.org на Google Docs

По данным о последнем релизе редакции Linux-дистрибутива Ubuntu для нетбуков (Ubuntu Netbook Remix, UNR), из него решили убрать офисный пакет OpenOffice.org, отдав предпочтение онлайн-сервису Google Docs.

Отказ от OpenOffice.org в качестве офисного приложения в UNR произойдет уже с ближайшим релизом, который ожидается в апреле, – Ubuntu Netbook Remix 10.04 «Lucid Lynx». Все документы отныне будут открываться через онлайн-сервис Google Docs. Решение разработчиков вполне понятно: всерьез работать с офисными документами на нетбуке вообще не очень удобно, а использование на устройствах такого класса столь «тяжелого» приложения, как OpenOffice.org, лишь усугубляет ситуацию. Поэтому логично, что пользователям, работающим с инсталляцией UNR по умолчанию, будет достаточно Google Docs в качестве офисного решения.

Помимо OpenOffice.org из UNR убрали еще и систему заметок Tomboy (которая написана на C#, из-за чего требует Mono и ряд других зависимостей), а также GNOME Pilot для синхронизации с устройствами под управлением PalmOS. Ранее также сообщалось, что из «главного» релиза Ubuntu Linux 10.04 уберут GIMP, так что его можно не ждать и в редакции UNR.

Зато в UNR добавили несколько других программ: клиент микроблоггинга Gwibber; приложение Cheese для веб-камер; игру gbrainy.

Дмитрий Шурупов,
по материалам www.nixp.ru
(osa@samag.ru)

Обзор дистрибутива Zenwalk Linux

Zenwalk (http://www.zenwalk.org) – дистрибутив, разработанный на базе Slackware Linux. Он появился на свет благодаря стараниям Жана-Филиппа Гильомена (Jean-Philippe Guillemin (http://en.wikipedia.org/wiki/Jean-Philippe_Guillemin)) в мае 2004 года и назывался тогда MiniSlack. Нынешнее название дистрибутив получил в октябре 2005 года. Zenwalk задумывался с целью популяризации Slackware и создания оптимизированной рациональной системы. Дистрибутив собран с оптимизацией для набора инструкций i686, но с возможностью запуска на машинах i486. А вот версия для 64-битных процессоров отсутствует. Выход релизов не привязан к какому-либо графику, но в среднем новая версия появляется каждые 2-3 месяца. Соответственно программное обеспечение всегда достаточно свежее.

Существует 4 редакции Zenwalk:

- ☑ Zenwalk Standard Edition (среда рабочего стола – Xfce);
- ☑ Zenwalk Core Edition (отсутствуют графические приложения, рекомендуется как отправная точка для построения настроенной под себя системы);
- ☑ Zenwalk Live Edition (та же функциональность, что и у Standard Edition + инструментарий для восстановления системы, разметки диска);
- ☑ Zenwalk Gnome Edition.

В данном обзоре я рассматриваю Zenwalk Standard Edition последней на текущий момент версии 6.2 (релиз вышел 6 сентября).

Установка

Инсталлятор имеет текстовый (ncurses) интерфейс. Впрочем, это никак не влияет на простоту установки: она проходит быстро и без лишних вопросов. Во время инсталляции в качестве загрузчика доступен только lilo (как и в случае со Slackware). Его установку можно пропустить, если у вас уже установлен grub. Для тех, кто нуждается в подробных инструкциях, рекомендую ознакомиться с пошаговым руководством, доступным по адресу http://manual.zenwalk.org/ru/Пошаговое руководство.html.

Первый запуск

После успешной установки при первом запуске вы сможете добавить обычного пользователя системы (по умолчанию единственным пользователем является root). Вход в систему происходит в графическом режиме. В качестве среды рабочего стола используется Xfce версии 4.6.1 (последняя стабильная версия на текущий момент). Конфигурация данной среды полностью управляется мышью (конфигурационные файлы скрыты от пользователя), что, несомненно, порадует новичков. Настройке подлежат внешний вид окон, раскладка клавиатуры, системное время, загружаемые при старте= сервисы, параметры сети, печати, а также можно указать точки монтирования для ваших разделов диска и многое другое. Выглядит рабочий стол примерно так (см. **рис. 1**). Для меня, как пользователя рабочей среды KDE, наличие двух панелей на рабочем столе поначалу казалось излишним (думаю, эта мысль может появиться и у пользователей, работавших до сих пор в Windows). Но со временем я увидел в этом некоторые преимущества. Для доступа к каталогам и приложениям, пиктограммы которых расположены на рабочем столе, необходимо сделать двойной щелчок, но в файловом менеджере для этого достаточно одного клика. Поначалу немного путаешься, но вскоре привыкаешь. Стоит отметить легкость перемещения окна с одного виртуального рабочего стола на другой. Достаточно просто перемещать окно вправо или влево, как говорится, «до упора». После освоения среды можно изучить список приложений, доступных сразу после установки (см. **рис. 2**).

Создатели дистрибутива придерживаются мысли, что для решения одной задачи достаточно одного приложения (один браузер, один аудиоплеер и т.д.). Для повседневных задач предусмотрены:

- ☑ работа с документами – OpenOffice 3.1;
- ☑ веб-серфинг – Iceweasel 3.5.2 (Mozilla Firefox без «бренда»);
- ☑ электронная почта – Icedove (аналогично Thunderbird);
- ☑ общение в Интернете – Pidgin;
- ☑ прослушивание музыки – Exaile;
- ☑ просмотр видео – Totem;
- ☑ запись дисков – Brasero;
- ☑ редактирование изображений – GIMP.

Но не ищите здесь редакторов аудио или видео, а также игр – даже самых простых вроде сапера или тетриса. Впрочем, не спешите расстраиваться, ведь у вас есть менеджер пакетов, именуемый здесь netpkg, о котором расскажу немного ниже. А также здесь есть и знакомый пользователям Slackware pkgtool – он поможет, если у вас нет выхода в Интернет, но есть необходимый пакет (netpkg, как можно догадаться по его названию, ориентирован на установку из сети) (см. **рис. 3**).

Есть текстовый и графический интерфейсы для менеджера пакетов netpkg (последний запускается командой xnetpkg или через пункт меню «Меню XFCE → Система → netpkg»). Установка в графическом режиме достаточно проста: необходимо выбрать источник пакетов из списка доступных зеркал, установить фильтр (например, «неустановленные»), выбрать пакет из отобразившегося списка и нажать на кнопку установки. Этот процесс проиллюстрирован ниже. Более подробно процесс установки (в том числе в текстовом режиме) описан в руководстве (http://manual.zenwalk.org/ru/Управление пакетами.html). Менеджер пакетов поддерживает пакеты как в формате tgz, так и в txz, появившемся в Slackware 13. Управление зависимостями – основное отличие netpkg от pkgtool. К сожалению, русификация интерфейса имеющихся программ оставляет желать лучшего.

Приятно отметить, что при использовании дистрибутива на машине с довольно скромной по нынешним меркам системной конфигурацией (процессор с частотой 1.5 ГГц, 256 Мб оперативной памяти, встроенные видео- и звуковая карты) не ощущалось какого-либо дискомфорта. Единственным заметным отличием было несколько более продолжительное время старта самой системы и тяжеловесных приложений вроде GIMP.

Информационная поддержка дистрибутива

Документация по дистрибутиву создается и поддерживается

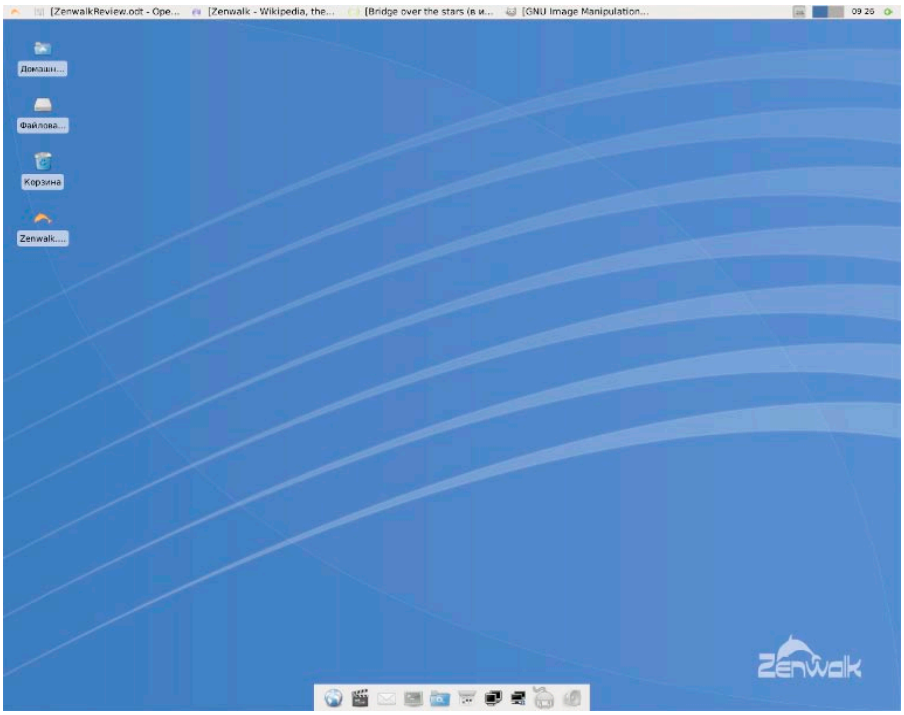

Рисунок 1. Рабочий стол Zenwalk

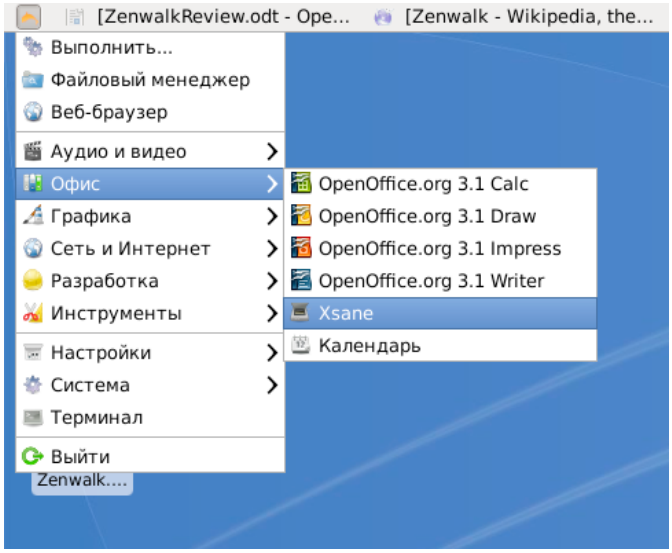

Рисунок 2. Список установленных приложений

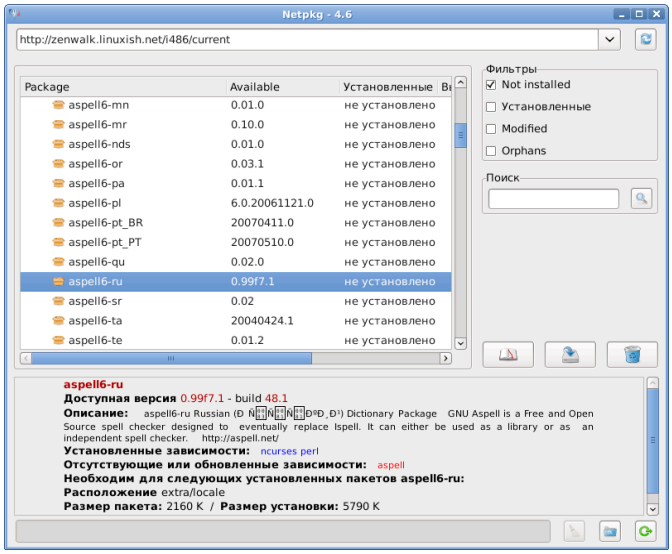

Рисунок 3. Менеджер пакетов

членами Zenwalk Documentation Team. Справочник пользователя Zenwalk на русском языке доступен по ссылке http://manual.zenwalk.org/ru (за основу этого руководства взята Slackbook (http://www.slackbook.org)). Если вам будет недостаточно имеющейся там информации, можно обратиться к Zenwalk Wiki (http://wiki.zenwalk.org) (правда, большинство информации здесь уже на английском). К сожалению, информация не всегда самая новая. Поэтому при необходимости обращайтесь на форум (http://support.zenwalk.org), тем более есть русская ветка (http://support.zenwalk.org/viewforum.php?f=44). Помимо официальной документации можно найти подборку русскоязычных материалов на linuxcenter.ru (http://www.linuxcenter.ru/lib/articles/distrib/slackware/zenwalk_ru.phtml) (она в основном за 2007 год, но еще не утратила полностью своей актуальности).

Заключение

Zenwalk Linux – достаточно легковесный (имеет скромные по нынешним меркам аппаратные требования), легкий в освоении и настройке дистрибутив. Он легко может быть дополнен необходимыми программами. Этот дистрибутив достаточно популярен (входит в первую двадцатку по рейтингу сайта Distrowatch.com, немного уступая своему предку).

На мой взгляд, он неплохо подойдет для тех, кто все еще не решается установить Slackware. Ведь Zenwalk как раз отличается более простой процедурой установки, наличием собственного пакетного менеджера, отслеживающего зависимости устанавливаемых пакетов, принципом «одна задача – одно приложение». Использование по умолчанию среды Xfce снижает системные требования. К тому же у этого дистрибутива довольно короткий, хоть и нестабильный цикл релизов. Перечисленные особенности способствуют успешному его использованию в качестве настольной системы. Явным недостатком является плохо русифицированный интерфейс.

Виталий Зборовский
(mismatch@yandex.ru)

Знакомство с Tcl/Tk. Часть 1: основы Tcl

Tcl (http://www.tcl.tk) можно смело назвать одним из самых необычных языков программирования. Человек, знающий как минимум один популярный язык, при изучении следующего видит уже знакомые элементы синтаксиса и концепции, заимствованные из других (в конечном счете обычно из C или Pascal). Tcl же кажется не похожим ни на что – его ближайшими «родственниками» являются языки UNIX Shell. Такая неординарность часто отпугивает новичков, несмотря на то что сам язык концептуально очень прост и по-своему элегантен.

В этой статье приведен очень краткий обзор Tcl, ориентированный на читателей, уже имеющих опыт программирования на других языках. В Сети имеется огромное количество материалов как для опытных программистов, так и для новичков (в том числе и на русском языке: http://ru.wikipedia.org/wiki/Tcl), откуда можно почерпнуть подробности.

Tcl относится к динамическим языкам высокого уровня. Его прямые конкуренты – Perl, Python, Ruby и Lua. Однако Tcl – единственный язык из этого класса, для которого существует своя собственная GUI-библиотека Tk. Как правило, тандем Tcl/Tk (часто называемый «тикль» или «так-тикль») рассматривается как неразрывное целое. Библиотека Tk была изначально написана для Tcl, тогда как для других языков доступны только «обертки» вокруг сторонних GUI-библиотек, зачастую чужеродных им по стилю использования (к слову, интерфейсы для Tk существуют для всех перечисленных выше языков).

Tcl появился в 1988 году в Университете Беркли. В 1990 году он вышел за пределы академической среды, а после скорого появления GUI-библиотеки Tk стал стремительно завоевывать популярность. В последние годы интерес к Tcl/Tk несколько уменьшился. Немаловажную роль в этом сыграл внешний вид компонентов GUI Tk версий 8.4 и ниже – по сравнению с современными аналогами они выглядели настолько архаично, что вызывали активное неприятие у пользователей (хотя и работали прекрасно). С выходом Tk 8.5 ситуация резко изменилась. Виджеты Tk теперь могут менять свой внешний вид с помощью тем и стилей, а в Windows и Mac OS они реализованы с помощью «родных» компонентов. Это по праву считается вторым рождением Tk. Появление хороших интерфейсов для других интерпретируемых языков, особенно для Python, также добавило Tk популярности.

Tcl/Tk работает абсолютно одинаково на разных платформах, в том числе в Linux, Windows и Mac OS, что обеспечивает прекрасную переносимость программ. Интерпретатор Tcl компили-

рует программу во внутренний байт-код, который, однако, нельзя сохранить на диск. Программы выполняются достаточно быстро, но, как и у всех подобных языков, производительность гораздо ниже, чем у компилируемых языков. Производительность Tcl в математических вычислениях невелика, т.к. все значения изначально интерпретируются как строки, поэтому использовать этот язык для интенсивных расчетов не стоит. Напротив, обработка текстов происходит достаточно быстро. В последних версиях Tcl эта особенность несколько сглаживается за счет разных внутренних типов данных для строк и чисел.

Tcl и Tk распространяются по особой открытой лицензии Tcl/Tk license, похожей на лицензию BSD. В частности, язык можно бесплатно использовать в коммерческих разработках.

Установка

Tcl и Tk доступны во всех основных дистрибутивах Linux. Для Windows и Mac OS лучше всего установить пакет «все в одном» ActiveTcl (http://www.activestate.com/activetcl). В Ubuntu достаточно ввести:

```
sudo apt-get install tcl8.5 tk8.5 tcllib tkcon
```

Tkcon – это интерактивная консоль для Tcl и Tk (она сама написана на Tcl/Tk) с подсветкой синтаксиса, историей, автодополнением и другими полезными функциями. Очень рекомендую ее установить, т. к. возможности интерактивной работы из командной строки «штатного» интерпретатора tclsh крайне ограничены. На мониторах с высоким разрешением шрифты по умолчанию в Tkcon оказываются слишком мелкими. Чтобы увеличить их, нужно создать в домашней директории пользователя файл .tkconrc, например, с такой строкой:

```
tkcon font Courier 12
```

Естественно, тип и размер шрифта можно установить любые. Еще можно настроить цвета и многие другие параметры – подробнее смотрите в документации (http://docs.activestate.com/activetcl/8.4/tkcon/ref/docs/tkconrc.5.html).

Основы языка

Программа на Tcl, по сути, состоит исключительно из команд типа «имя_команды аргумент аргумент...». Причем семантическое значение отдельных «слов» может меняться в зависимости от контекста, т. е. у языка нет формальной грамматики (!). Управляющие конструкции и функции также ничем не выделены и реализованы как команды. В языке нет зарезервированных слов – можно назвать переменную или функцию if или for, несмотря на наличие таких управляющих конструкций (но делать этого не стоит, поскольку управляющая конструкция может стать недоступной). Tcl использует исключительно польскую (префиксную) нотацию. Имя команды всегда идет первым, затем имена подкоманд и только затем аргументы.

Понятия оператора в Tcl также нет. Переменные вводятся командой «set имя_переменной начальное_значение».

Для доступа к значению переменной (разыменования) нужно написать «$имя_переменной».

Комментарии начинаются с символа «#» – он должен быть первым в строке (не считая пробелов). Внутри строки символ «#» не несет никакого особого смысла. Чтобы поместить комментарий после некоего кода в той же строке нужно написать «код ;# комментарии».

Каждая команда обычно занимает отдельную строку, но если отделять команды точкой с запятой, их можно размещать по несколько на одной строке. Длинную команду можно переносить на другую строку, ставя в конце строки символ «\».

Строки и списки

Несколько упрощая ситуацию, можно сказать, что любые данные в Tcl интерпретируются как списки строк. Список в Tcl – это индексированный набор элементов (в других языках это назвали бы массивом, но в Tcl понятие массива имеет другой смысл). Элементы нумеруются всегда с нуля. Основная команда работы с отдельными строками – string. Для работы со списками есть набор команд, начинающихся с буквы «l» с вполне интуитивными названиями: lindex, lappend, lsearch, lset, llength и т.п.

В зависимости от контекста одно и то же значение может интерпретироваться либо как строка, либо как список строк. Например:

```
% set var "Вышел зайчик погулять"
Вышел зайчик погулять
% string length $var ;# интерпретируем как единую строку
из 21 символа
21
% llength $var ;# интерпретируем как список из трех строк
3
% string length [lindex $var 1] ;# получаем число символов
во втором слове
6
```

Строки в двойных кавычках позволяют вставлять в них значения переменных, тогда как в строках, заключенных в фигурные скобки, разыменования не происходит:

```
% set a "Some value"
Some value
% puts "Value of a is: $a"
Value of a is: Some value
% puts {Value of a is: $a}
Value of a is: $a
```

Команда puts выводит свой аргумент на экран. Все, что ей передано, преобразуется в строку. Списки могут быть вложены друг в друга:

```
% lappend a 1
1
% lappend a {2 3}
1 {2 3}
% lindex $a 0
1
# фигурные скобки отделяют вложенные списки, но они не являются
# элементами списка:
% lindex $a 1
2 3
% string length $a  ;# однако в представлении строки они учитываются
7
```

Подпрограммы и управляющие конструкции

Следующий пример иллюстрирует некоторые управляющие конструкции языка и подпрограммы:

```
proc add_me {arg1 arg2} {
      return "The sum of $arg1 and $arg2 is [expr $arg1+$arg2]"
}

for {set i 1} {$i<20} {incr i} {
      if {$i>10} {
            puts [add_me $i [expr $i+1]]
      }
}
```

Результат:

```
The sum of 11 and 12 is 23
The sum of 12 and 13 is 25
...
The sum of 19 and 20 is 39
```

Цикл for очень похож на свой аналог в Си. Как легко догадаться, команда incr увеличивает переменную на единицу. Команда, заключенная в квадратные скобки, заменяется на результат ее выполнения. В Tcl часто можно встретить «матрешки» вроде этой:

```
set result [lindex [lsort [lappend some_list ↵
      [expr [lindex $var1 $i] + [lindex $var2 $j]]]] 5]
```

Нужно признать, что такой код читается очень плохо, поэтому перед подобными выражениями желательно помещать комментарии или разбивать их на несколько частей. Команда expr служит для выполнения математических операций в привычной нотации. Просто так написать в коде «set a 2+3» в надежде получить «5» нельзя, т.к. Tcl не имеет представления о числах и арифметических операторах вне команды expr и получится просто строка «2+3». Однако запись:

```
for {set i 1} {$i<20+10} {incr i} {...
```

корректна: в условиях остановки цикла и некоторых других управляющих конструкциях математические выражения допустимы.

Внутри команды expr можно использовать переменные и любые команды в квадратных скобках. Из-за того что все управляющие конструкции – это просто команды, пробелы перед «{» обязательны: «if{» без пробела будет восприниматься как цельное слово и вызовет ошибку.

Все переменные, введенные в подпрограммах, локальны и исчезают, как только выходят из области видимости. Глобальные переменные внутри подпрограммы также не видны, пока их не объявляют доступными явно командой global:

```
set a "Some text"
proc some_proc {} {
      global a
      puts $a
}
```

Массивы

Кроме списков, в Tcl есть массивы (arrays), которые эквивалентны хешам или ассоциативным массивам в других языках. В массивах хранятся пары «ключ-значение». Вот пример использования массива:

```
% set arr(x) 11
% set arr(y) "some text"
% set arr(z) [list a b c]
% array names arr ;# получаем список всех ключей
x y z

% array get arr ;# весь массив в виде списка «ключ значение ключ значение...»
x 11 y {some text} z {a b c}

# Итерация по массиву
% foreach key [array names arr] { puts "$key --> $arr($key)" }
x --> 11
y --> some text
z --> a b c
```

Одна из неприятных особенностей массивов в Tcl состоит в том, что их нельзя напрямую (по значению) передать в подпрограмму, а передачи по ссылке как таковой в Tcl нет. Передачу по ссылке можно имитировать с помощью команды upvar, но мы не будем подробно касаться этой темы.

Пространства имен

В Tcl имеются пространства имен, позволяющие предотвратить конфликты переменных:

```
set a Text
% namespace eval N {
      variable a 10
      variable b 20
      variable c 30
      proc some_proc {} {
            variable a
            puts $a
      }
}
% puts $::a
Text

% puts {Value of a is: $a}
10
```

Переменные и подпрограммы из глобального пространства имен доступны из любого места программы с префиксом «::» (глобальное пространство имен). Обращение к переменным пространства любой вложенности формируются с добавлением префикса «::пространство». Если в пространство имен входит подпрограмма, то переменные того же пространства нужно явно объявлять доступными внутри нее, как это показано в примере выше.

Регулярные выражения

В Tcl имеется встроенная поддержка регулярных выражений, реализованная в командах regexp (поиск по регулярным выражениям) и regsub (поиск с заменой). Используется стандартный POSIX-синтаксис регулярных выражений с небольшими дополнениями (http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm). Описывать детали использования этих команд в короткой статье не имеет смысла, т. к. существует подробная документация (http://www.tcl.tk/man/tcl8.5/TclCmd/regsub.htm, http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm). Небольшой практический пример будет приведен ниже.

Чтение и запись файлов

Для открытия файлов используется команда «open имя_файла режим». Режимы r и w используются для чтения и записи соответственно. Есть также режимы для работы с бинарными файлами (rb, wb), дописывания в конец файла (a), и одновременного чтения-записи (r+, w+). Для чтения строки из текстового файла используется команда gets. Стандартными средствами считывать из файла по одному слову нельзя – нужно считать всю строку, а затем разбить ее по разделителям командой split. К со-

жалению, команда split не умеет интерпретировать несколько разделителей как один (что, как правило, и нужно). Поэтому типичный пример чтения двух колонок данных из файла выглядит неоправданно громоздко:

```
set f [open filename r]
while {[gets $f line] >= 0} {  ;# Читаем строки, пока не достигнут
конец файла
    # Заменяем все последовательности нескольких
    # разделителей на одинарные пробелы:
    regsub -all {\s+} $line " " line
    set L [split $line]   ;#  разбиваем строку по пробелам
    # извлекаем данные первой и второй колонок
    set col1 [lindex $L 0]
    set col2 [lindex $L 1]
}
close $f  ;# закрываем файл
```

«Многословность» этой тривиальной операции – одна из самых неприятных особенностей языка Tcl, которой приходится платить за другие преимущества.

Пакеты

Для расширения базового языка в Tcl используется понятие пакетов (packages). Вся стандартная библиотека и Tk реализованы как пакеты. Основная команда для работы с пакетами – «package require пакет». Она ищет пакет среди имеющихся в данной системе и загружает его в интерпретатор. Как дополнительный аргумент можно указать нужную версию пакета. Достаточно легко писать новые пакеты как с помощью чистого Tcl, так и в виде динамических библиотек на C.

Стандартная библиотека

Количество и качество пакетов в стандартной библиотеке tcllib впечатляют. В частности, среди них можно найти:

- ☑ Объектно-ориентированные расширения языка. Их несколько – incr tcl, Snit, XOTcl, что затрудняет выбор.
- ☑ Разнообразные пакеты для работы с файловой системой.
- ☑ Математика (константы, интерполяция, линейная алгебра, комплексные числа, геометрия, Фурье-анализ, решение дифференциальных уравнений и т. д.).
- ☑ Разнообразные структуры данных (стеки, графы, деревья).
- ☑ Текстовые утилиты и конвертеры.
- ☑ Работа с изображениями в форматах jpg, tiff, png.
- ☑ Криптография.
- ☑ Утилиты для построения документации.
- ☑ Работа с сетью (протоколы FTP, LDAP, POP3, SMTP; работа с Proxy, DNS и т.д.).

Интерфейс с языком Си

Интерпретатор Tcl написан на Си, и его можно расширять с помощью новых команд, реализованных на Си. С помощью команды load интерпретатор позволяет загружать динамические библиотеки с расширениями. Таким образом можно выносить в компилируемые библиотеки критические части кода, которые должны выполняться с максимальной скоростью, или писать «обертки» для сторонних библиотек. Интерфейс с Си, по мнению многих разработчиков, проще аналогичных интерфейсов других языков (Python, Perl). Встраивание интерпретатора Tcl в программы на C и C++ также не вызывает сложностей.

Выводы

Tcl может как притягивать, так и отпугивать своей неординарностью, однако, поняв его идеологию (которая на самом деле достаточно проста), пользователь приобретает мощный инструмент для разработки кроссплатформенных программ. В полной мере возможности Tcl раскрываются в связке с Tk, о котором речь пойдет в следующей части статьи (в выпуске «Open Source» 056. – **Прим. ред.**).

Семен Есилевский
(yesint4@yahoo.com)

Реализация отправки и приёма SMS с помощью Gnokii

В современном мире человек не может представить своё существование без целого вороха разных высокотехнологичных устройств: начиная от электронных термометров и заканчивая сложными сетями связи. Для многих главное место среди всей электроники занимает мобильный телефон. Действительно, сложно представить себе современного человека, который не носит с собой постоянно телефон. Очень удобно: мы привыкли к тому, что нам банки присылают счета, интернет-провайдеры напоминают о необходимости оплаты, да и мелкие мастерские уже зачастую уведомляют о завершенном ремонте электроплиты или даже обуви. И всё это нам приходит с помощью SMS. Почему бы системному администратору не воспользоваться этим, казалось бы, привычным методом общения в своих целях?

Конечно же, в Сети существует много сервисов, позволяющих это сделать, чего только стоят скрипты, отсылающие SMS через сайты сотовых провайдеров. В этой статье я расскажу, как это сделать, используя Debian GNU/Linux, Gnokii и обычный мобильный телефон, подключенный к серверу через USB, Bluetooth или последовательный порт. Кроме Gnokii, нам понадобится сервер MySQL (он может находиться и на другой машине), Perl, Perl-модуль DBI для работы с MySQL в скрипте отправки, а также базовые навыки работы с MySQL и GNU/Linux. Отличительная особенность данного решения – полная автономность и независимость от интернет-соединения, что позволяет строить более комплексные системы (например, аварийную сигнализацию в случае сбоя оборудования).

Установка, настройка и проверка Gnokii

Процесс установки необходимого ПО мы опустим – это слишком долго, да и читающие эту статью наверняка смогут выполнить все необходимые действия самостоятельно. Скажу лишь, что в Debian необходимые пакеты – это gnokii-smsd-mysql и libdbi-perl вместе с зависимостями.

Теперь немного о Gnokii. Проект изначально предназначался для синхронизации телефонов Nokia с компьютером, но впоследствии был расширен для работы с другими устройствами, а также получил много новых применений. Более подробно с проектом можно ознакомиться на официальном сайте http://www.gnokii.org.

После установки необходимо настроить Gnokii путём редактирования файла /etc/gnokiirc. Процесс настройки очень хорошо описан в http://wiki.gnokii.org, но самые главные параметры –

это port (для USB, скорее всего, /dev/ttyACM0; COM-порты – /dev/ttyS0 и /dev/ttyS1 – для COM1 и COM2 соответственно) и model («symbian» – для Symbian-устройств Nokia; «6110», «7110», «6510», «3110», «2110», «6160» – для соответствующих моделей; «AT» – для всех остальных AT-совместимых устройств).

У меня, например, Motorola ROKR Z6, подключенная к USB, настроена так:

```
port = /dev/ttyACM0
model = AT
```

Все остальные параметры оставлены по умолчанию. Замечу, что в некоторых телефонах нужно выбрать режим соединения «модем» либо «передача данных», так как с телефоном мы будем работать как с GSM-модемом, то есть посредством AT-команд. Теперь проверим работу, выполнив простую команду отправки SMS:

```
$ echo "Hello from Gnokii!" | gnokii --sendsms номер_телефона

GNOKII Version 0.6.26
Send succeeded!
```

Телефон, на который было отправлено тестовое сообщение, должен сразу же его получить. К слову, я упустил параметр -r, который включает получение отчета о доставке. Проверим работу в обратную сторону:

```
$ gnokii --smsreader

GNOKII Version 0.6.26
Entered sms reader mode...
```

Отправляем со своего телефона в ответ на тестовое сообщение, которое должен принять и обработать компьютер. Текст сообщения пока что значения не имеет – нам важно лишь проверить получение SMS. Видим, что сообщение получено успешно. Замечу, что на данном этапе сообщение может остаться в памяти телефона – это зависит от каждого конкретного аппарата. При использовании smsd оно будет удаляться из памяти телефона и останется исключительно в базе: этого можно не бояться – даже моя Motorola, часто уличаемая в некорректном поведении в качестве модема, нормально работала в такой связке.

Самый главный зверь – smsd

Следующий этап – настройка и запуск главной части нашей системы, а именно – демона smsd, который будет принимать, отправлять и хранить в базе все сообщения. Создаём на MySQL-сервере базу данных и пользователя для демона. Я привык делать это через phpMyAdmin, хотя можно воспользоваться любым удобным вам способом. В базу нужно загрузить структуру таблиц – SQL-файл находится в системе (/usr/share/doc/gnokii-smsd-mysql/sms.tables.mysql.sql). Обратите внимание, что в этом дампе есть и создание базы, и создание учетной записи пользователя базы, поэтому будет логичным оставить из него лишь нужную часть, включающую в себя только создание таблиц:

```
CREATE TABLE inbox (
  id int(10) unsigned NOT NULL auto_increment,
  number varchar(20) NOT NULL default '',
  smsdate datetime NOT NULL default '0000-00-00 00:00:00',
  insertdate timestamp NOT NULL,
  text text,
  phone tinyint(4),
  processed tinyint(4) NOT NULL default '0',
  PRIMARY KEY (id)
);
```

```
CREATE TABLE outbox (
  id int(10) unsigned NOT NULL auto_increment,
  number varchar(20) NOT NULL default '',
  processed_date timestamp NOT NULL,
  insertdate timestamp NOT NULL,
  text varchar(160) default NULL,
  phone tinyint(4),
  processed tinyint(4) NOT NULL default '0',
  error tinyint(4) NOT NULL default '-1',
  dreport tinyint(4) NOT NULL default '0',
  not_before time NOT NULL default '00:00:00',
  not_after time NOT NULL default '23:59:59',
  PRIMARY KEY (id)
);

CREATE TABLE multipartinbox (
  id int(10) unsigned NOT NULL auto_increment,
  number varchar(20) NOT NULL default '',
  smsdate datetime NOT NULL default '0000-00-00 00:00:00',
  insertdate timestamp NOT NULL,
  text text,
  phone tinyint(4),
  processed tinyint(4) NOT NULL default '0',
  refnum int(8) default NULL,
  maxnum int(8) default NULL,
  curnum int(8) default NULL,
  PRIMARY KEY(id)
);
```

У демона нет файла конфигурации – все настройки для работы с базой задаются в командной строке. Работу с телефоном мы уже настроили и проверили – настройки читаются из /etc/gnokiirc. Попробуем запустить:

```
sudo /usr/sbin/smsd -u db_username -d db_name ↵
      -p db_password -c db_hostname -m mysql -b SM ↵
      -f /var/log/smsdaemon
```

Используемые здесь параметры db_username, db_name, db_password, db_hostname – это соответственно имя пользователя, имя базы, пароль и адрес сервера СУБД (у меня это был localhost). SM – это тип памяти, в которой хранятся SMS (в данном случае – Sim Memory); его значение подбирается скорее экспериментальным путём: в каждой конкретной модели телефона они могут быть перепутаны местами либо названы по-другому (полный список возможных значений можно найти на http://wiki.gnokii.org/index.php/Memory_type_codes). Здесь стоит заметить, что smsd работает по принципу, немного отличающему его от Gnokii: Gnokii просит телефон передавать ему все входящие SMS (из-за этого и неразбериха: некоторые телефоны сохраняют после этого входящие сообщения, некоторые – нет), а smsd проверяет наличие новых сообщений в памяти телефона, после чего принятые сообщения читаются и удаляются. Такой принцип работы отличается надежностью: например, телефон можно перезагрузить либо временно отключить – все принятые за это время SMS всё равно обработаются сервером.

После запуска мы должны увидеть в консоли сообщения об успешном подключении телефона, получении/приёме SMS и т.д. Если все работает, можно завершать процесс нажатием на <Ctrl> + <C>.

Для запуска сервера воспользуемся init-скриптом, приложенным к статье и написанным мною (точнее, отредактированным из примера). Важно не забыть сменить значение DAEMON_ARGS на свои параметры. Двоеточие в конце файла – это часть init-скрипта, сбрасывающая значение кода возврата.

Почти готово. Осталось записать всё это в /etc/init.d/smsd, сделать файл исполняемым, добавить его в загрузку и запустить. Последнее в Debian делается так:

```
update-rc.d smsd defaults
invoke-rc.d smsd start
```

Теперь smsd настроен, связь работает – можно начинать пожинать плоды своего труда.

Когда самое сложное позади...

Принцип действия smsd прост: всё, что приходит, записываем в таблицу inbox (**важно:** значение processed устанавливается равным нулю), а всё, что есть в таблице outbox с processed равное нулю (т.е. не обработано), отсылаем и ставим processed = 1. Не путайте таблицы inbox и outbox – у них обеих есть это поле. Интересно заметить, что в inbox это поле совсем не используется демоном – оно оставлено специально для внешних обработчиков. Мы его используем в обработке входящих SMS для обозначения уже обработанных сообщений. Для отправки SMS мною написан небольшой модуль на Perl, trim($) добавлен сюда же просто ради удобства – мы его всё равно потом будем использовать. Модуль следует обозвать Sms.pm, а в Debian Lenny записать в /usr/local/lib/perl/5.10.0 (в других дистрибутивах – по вкусу). Важно заметить, что здесь используется фильтр номеров для Украины (для России понадобится небольшая доработка).

```
#!/usr/bin/perl
# Sendsms for Gnokii
use DBI;
# Конфигурация для MySQL:
$mysqlHost = 'db_hostname';
$mysqlDb = 'db_name';
$mysqlUser = 'db_username';
$mysqlPassword = 'db_password';
$mysqlConnect = "dbi:mysql:$mysqlDb;$mysqlHost";

sub sendsms {
  my $number = @_[0];
  my $text = @_[1];
  if ($number =~ m/^\+?380\d{9}$/)
  {
    #print "Number is OK, sending SMS...\n";
  }
  else
  {
    #print "Number doesn't seem to be in international format.\n";
    if ($number =~ m/^80\d{9}$/)
    {
      #print "OK, number is in national format, restoring it...\n";
      $number = "+3" . $number;
      #print "The number is now " . $number . ↵
              " and it seems to be OK.\n";
    }
    else
    {
      print "ERROR: Could not understand the number ↵
          $number. Check the number and try again.\n";
      return 1;
    }
  }
  $text = trim($text);
  if ($text eq '')
  {
    print "ERROR: The string is empty!\n";
  }
  else
  {
    my $dbh = DBI->connect($mysqlConnect, $mysqlUser, ↵
      $mysqlPassword);
    my $query = "INSERT INTO `outbox` (`number`,`text`) ↵
      VALUES ('$number', '$text');";
    my $sth = $dbh->prepare($query);
    $sth->execute();
    $sth->finish();
    $dbh->disconnect;
    #print "OK, message sent. See you later!\n";
  }
}
# Функция trim для устранения пробелов в начале и конце строки
sub trim($)
{
  my $string = shift;
  $string =~ s/^\s+//;
  $string =~ s/\s+$//;
  return $string;
}
```

Пример использования – простая отправка SMS:

```
#!/usr/bin/perl
# Sendsms for Gnokii
use Sms;
$numArgs = $#ARGV + 1;
if ($numArgs != 2)
{
  print "Usage:\n\tsendsms destination text\n";
  if ($numArgs > 2)
  {
    print "\nError: too many arguments! Looks like you forgot ↵
      about whitespaces\n";
  }
  exit 1;
}
else
{
  $number = $ARGV[0];
  $text = $ARGV[1];
  &sendsms ($number,$text);
}
```

Теперь заставим сервер отвечать на SMS:

```
#!/usr/bin/perl
use DBI;
use Sys::Hostname;
use Sms;
# Конфигурация для MySQL:
$mysqlHost = 'db_hostname';
$mysqlDb = 'db_name';
$mysqlUser = 'db_username';
$mysqlPassword = 'db_password';
$mysqlConnect = "dbi:mysql:$mysqlDb;$mysqlHost";
$dbh = DBI->connect($mysqlConnect, $mysqlUser, $mysqlPassword);
$query = "SELECT `id`, `number`, `text` FROM `inbox` WHERE ↵
      `text` REGEXP '^".hostname."*' AND `processed` = 0;";
while (true) {
  $sth = $dbh->prepare($query);
  $results = $dbh->selectall_hashref($query, 'id');
  foreach my $id (keys %$results) {
    print "Value of ID $id is $results->{$id}->{text}\n";
    $hostname = hostname;
    $text = $results->{$id}->{text};
    if ($text =~ m/^$hostname/)
    {
      if ($text =~ m/^$hostname hello/)
      {
        sendsms($results->{$id}->{number}, ↵
                "Hello to you from $hostname!");
      }
      if ($text =~ m/^$hostname uptime/)
      {
        $message = "$hostname uptime:".qx ('uptime');
        sendsms($results->{$id}->{number},$message);
      }
      $dbh->do("UPDATE `inbox` SET `processed` = '1' WHERE ↵
                `id` = $id LIMIT 1 ;");
    }
  }
  sleep(1);
}
$dbh->disconnect();
```

Запускаем. Если отправить серверу SMS с текстом «имя_сервера uptime», в ответ придет время работы сервера. На запрос же «имя_сервера hello» детище поприветствует своего хозяина. Простора для творчества здесь много. Единственное условие – желательно, чтобы первое слово всё равно было именем сервера, ведь обработчик может быть запущен на нескольких машинах, а значит, SMS можно адресовать на любой из них.

Вместо заключения

Внимательные читатели могли заметить, что нам никто и ничто не мешает запустить несколько обработчиков входящих сообщений на нескольких серверах (и соответственно слать SMS тоже с нескольких серверов), ведь мы работаем уже не с телефоном (с ним работает smsd), а с базой. Также предложенную мной схему работы можно немного улучшить – например, посредством udev запускать smsd при подключении телефона или останавливать после отключения (работать, правда, это будет только с USB). К примеру, я делаю рассылки с напоминаниями своим клиентам, а мой zabbix-сервер оповещает меня о проблемах. Всё в ваших руках!

Вадим Абрамчук
(abramm@gmail.com)

Программирование проверки правописания. Часть 2: Hunspell

Первая часть статьи была посвящена Aspell и опубликована в «Open Source» 054 (25.12.2009). Теперь речь пойдет о Hunspell (http://hunspell.sourceforge.net). Скажу сразу: если вам достаточно восьмибитных кодировок, поддерживаемых Aspell, от Hunspell особого проку не будет. Но тем программистам, которые приняли юникод как данность времени, ознакомиться с Hunspell – одно удовольствие. Ныне средства проверки правописания, предоставляемые Hunspell, задействованы в таких программах, как OpenOffice.org, Firefox, Opera, Google Chrome, The Bat! и TEA.

API Hunspell я буду показывать на примере интерфейса под C++, хотя он есть и для C. Разница, во-первых, в заголовочных файлах (hunspell.hxx для C++ и hunspell.h для C) и, конечно же, в коде. Для Си даются обычные функции, а для C++ – класс Hunspell. Всё описанное в этой статье, что не касается кода, относится к UNIX-подобным системам. Под Windows, а в частности Cygwin, мне совладать с Hunspell не удалось, даже несмотря на действия «по бумажке» (по руководству).

Чтобы при сборке вашей программы она «увидела» Hunspell, можно воспользоваться pkg-config. Для autotools это выглядит примерно так (прописываем в configure.in):

```
echo -n "checking for hunspell... "
if pkg-config --exists hunspell  ; then
      LIBS="$LIBS `pkg-config --libs hunspell `"
      CFLAGS="$CFLAGS `pkg-config --cflags hunspell `"
      AC_DEFINE(HUNSPELL_SUPPORTED, 1, [HUNSPELL_SUPPORTED])
      echo "yes"
else
      echo "no"
fi
```

То есть, если библиотека найдена, добавляем все нужные для сборки параметры, а также определяем HUNSPELL_SUPPORTED, чтобы потом в коде можно было написать:

```
#ifdef  HUNSPELL_SUPPORTED
//код с поддержкой Hunspell
#endif
```

Для сборочной системы qmake делаем так:

```
CONFIG += link_pkgconfig

exists("/usr/include/hunspell/hunspell.hxx") {
 message ("hunspell enabled")
 PKGCONFIG += hunspell
 DEFINES += HUNSPELL_SUPPORTED
}
```

Экземпляр класса Hunspell создается следующим конструктором:

```
Hunspell (const char *affpath, const char *dpath, ↵
      const char *key = NULL);
```

В общем случае нас интересуют здесь только два первых параметра:

- ☑ **dpath** – полный путь к файлу словаря;
- ☑ **affpath** – полный путь к так называемому аффиксному файлу. В нем содержатся различные указания, правила проверки правописания, то есть не просто грубый список слов (как обычный словарный файл dpath), а набор команд, внешне напоминающих язык ассемблера. У Hunspell есть целый макроязык для заданий таких правил правописания. В аффиксном файле также содержится и название кодировки словаря.

Раз и навсегда проясню положение с кодировками в Hunspell. Словари могут иметь любую кодировку. Внутренне Hunspell работает на основе UTF-8 и UTF-16 – в разных алгоритмах поразному (например, при анализе – в UTF-8, а при «предположении» – в UTF-16). Пользовательский словарь должен иметь ту же кодировку, что и основной словарь для текущего языка. Работая с Hunspell, вы должны предоставлять его движку строки в кодировке словаря. Например, если словарь в KOI8-R, то вы должны передавать строки именно в KOI8-R. Кодировка задается в файле аффиксов, парном основному словарю. Пользовательский словарь не имеет такого парного файла. Узнать кодировку словаря можно с помощью API-функции char* get_dic_encoding() (она существует как отдельная функция для «сишного» интерфейса и как функция-член класса Hunspell). Память для возвращаемого ею значения освобождать не нужно.

Теперь вернемся к параметрам конструктора. Итак, файлы аффиксов и словаря. В том же Aspell мы не сталкивались с этим напрямую. Там были «общесистемные» словари, устанавливаемые через менеджер пакетов. Aspell заведовал словарями, предоставлял нам через API список доступных языков проверки правописания. В Hunspell такого нет. Словари доступны для скачивания с http://wiki.services.openoffice.org/wiki/Dictionaries. Еще можно использовать Hunspell-словари, установленные для других программ (скажем, OpenOffice.org или Firefox), только при этом у словарей будут разные имена файлов. Например, у словарей с официального сайта Hunspell имена подчинены некоему шаблону: en_US.aff, en_US.dic, ru_RU.aff, ru_RU.dic. Не составит труда написать код, который бы из списка таких файлов вычленял названия локалей, ставил им в соответствие название языка и предоставлял бы такой удобочитаемый список для графического интерфейса, чтобы пользователь мог выбрать нужный ему язык.

Для Qt можно написать примерно такую функцию, возвращающую список словарных модулей без расширений файла (в параметре dict_dir передается каталог, где лежат словари):

```
QStringList CHunspellChecker::get_speller_modules_list ( ↵
      const QString &dict_dir)
{
  QDir dir (dict_dir);

  QStringList filters;
  QStringList sl;

  filters << "*.dic";

  dir.setSorting (QDir::Name);
  QFileInfoList fil = dir.entryInfoList (filters);

  for (int i = 0; i < fil.size(); i++)
   sl.append (fil[i].baseName());

  return sl;
}
```

Кроме «официальных» словарей есть еще словари для OpenOffice.org 3. Они либо как пакеты, либо в виде дополнений – в последнем случае берутся по ссылке http://extensions.services.openoffice.org/dictionary, каждый из них заключен в контейнерный файл OXT. OXT-файл – это обычный переименованный ZIP-архив, внутри которого содержатся .dic- и .aff-файлы, а также некоторые служебные:

- ☑ из dictionaries.xcu можно узнать название локали и поставленные ей в соответствие словарные файлы;
- ☑ в файле description.xml мы получаем развернутое текстовое описание словаря, причем в локализованном виде.

Для работы с ZIP-архивами под Qt советую попробовать библиотеку http://quazip.sourceforge.net/, а просто для C/C++ обратите внимание на Minizip (http://www.winimage.com/zLibDll/minizip.html). Как видите, чтобы работать с Hunspell, вам придется решить своими силами вопрос взаимодействия со словарями. Лично я в редакторе TEA поступил просто: дал пользователю возможность указать каталог, в котором лежат распакованные словари, а TEA выводит их в виде списка локалей. Пользователь выбирает локаль, и всё работает. Это не очень изящный, зато простой и действенный способ.

Также на ваших плечах, вернее, на коде вашей программы, лежит забота о пользовательском словаре – том, куда можно добавлять новые слова. Словарь пользователя – это файл со списком слов, первая строка которого содержит общее количество слов в файле. К слову, это вопреки документации: там сказано, что пользовательский словарь – просто список слов, без всяких «но». Однако без числа в первой строке Hunspell отказывается загружать словарный файл. Файл должен быть в той же кодировке, что и основной словарь.

Для загрузки пользовательского словаря надо вызвать функцию:

```
int add_dic (const char * dpath, const char * key = NULL);
```

Первый параметр – путь к словарю, второй параметр в общем случае опускаем. У Hunspell есть парные функции:

```
int add (const char * word);
int remove (const char * word);
```

Как следует из имен, первая добавляет слово, а вторая удаляет. Но куда и из чего? Вот Hunspell загрузил основной словарь, а затем пользовательский (предположим, он уже создан). Функция add добавляет слово в текущую сессию, а remove – удаляет. Всё это происходит на уровне оперативной памяти, то есть add и remove не меняют словарь пользователя. Вам нужно самому параллельно вести список слов – словарь пользователя – и добавлять/удалять слова одновременно с вызовами Hunspell add/remove.

После завершения работы сохраняйте (не забывая про нужную кодировку) свой список слов в файл, добавив в начало списка строку с числом, равным количеству слов в списке. Под «словом» я подразумеваю строку в кодировке словаря. Например, в Qt вам придется перекодировать из юникодовой QString в кодировку словаря приблизительно так:

```
Hunspell *speller;
QString word;

[...]

QTextCodec *codec = QTextCodec::codecForName (encoding);
QByteArray es = codec->fromUnicode (word);
speller->add (es.data());
```

Подобным образом надо перекодировать все строки, передаваемые в функции Hunspell. В GTK для перекодировки используйте функцию g_convert. Проверка слова на правильность в Hunspell осуществляется вызовом функции spell:

```
int spell (const char * word, int * info = NULL, ↵
      char ** root = NULL);
```

В общем случае достаточно только первого параметра – word – опять же в кодировке словаря. Возвращается ноль, если слово «плохое», и не ноль, если «хорошее»:

```
if (speller->spell ("foobar"))
   ;//
else
   ;//
```

В необязательный параметр int *info функция может вывести дополнительные сведения о проверяемом слове, а в char **root – корень слова. Получить список «предположений» для ошибочного слова можно функцией suggest:

```
int suggest (char ***slst, const char *word);
```

Здесь word – ошибочное слово, а slst – буфер для списка строк (Hunspell сам выделит им память). Возвращаемое значение – число элементов в списке slst. Проиллюстрирую вызов функции с привязкой к Qt. В примере мы наполняем «предположениями» список QStringList sl, чтобы потом использовать его еще где-то, например, построив на его основе контекстное меню:

```
QStringList sl;

QTextCodec *codec = QTextCodec::codecForName (encoding);
QByteArray es = codec->fromUnicode (word);

char **slst;

int size = speller-> suggest (&slst, es.data());

for (int i = 0; i < size; i++)
 sl.append (codec->toUnicode (slst[i]));

speller->free_list (&slst, size);
```

Как видите, в конце мы вызываем функцию free_list, чтобы очистить память, выделенную Hunspell под список. Количество «предположений» жестко прописано в заголовочном файле Hunspell:

```
#define MAXSUGGESTION 15
```

Кстати, подобным образом ограничено и число пользовательских словарей – их может быть загружено не более двадцати.

Заключение

В статье я говорил только об одной стороне Hunspell – средствах проверки правописания. Но ведь Hunspell – это еще и набор функций по работе с языком: по морфологическому анализу, словообразованию и т. п. Например, можно получить для слова множественное число из единственного числа. Но всё это работает не по умолчанию со стандартными словарями, а посредством специально написанных для подобных целей аффиксных файлов. Подробности смотрите в файле hunspell.4 man-руководства Hunspell.

Наше путешествие в проверку правописания продолжится в следующей части статьи, где я расскажу, как средствами GTK и Qt воплотить такую проверку в жизнь. Об этом читайте в следующем выпуске «Open Source» (056).

Петр Семилетов
(tea@list.ru)

Подписные индексы:

20780*
+ диск с архивом статей 2008 года

81655**
без диска

по каталогу агентства «Роспечать»

88099*
+ диск с архивом статей 2008 года

87836**
без диска

по каталогу агентства «Пресса России»

* **Годовой**
** **Полугодовой**
*** **Диск вкладывается в февральский номер журнала, распространяется только на территории России**

Подписка на журнал «Системный администратор»

Российская Федерация

- ☑ Подписной индекс: годовой – **20780**, полугодовой – **81655**
  Каталог агентства «**Роспечать**»
- ☑ Подписной индекс: годовой – **88099**, полугодовой – **87836**
  Объединенный каталог «**Пресса России**»
  Адресный каталог «**Подписка за рабочим столом**»
  Адресный каталог «**Библиотечный каталог**»
- ☑ Альтернативные подписные агентства:
  агентство «**Интер-Почта**»
  (495) 500-00-60, курьерская доставка по Москве
  агентство «**Вся Пресса**»
  (495) 787-34-47
  агентство «**Курьер-Прессервис**»
  агентство «**ООО Урал-Пресс**»
  (343) 375-62-74
- ☑ Подписка On-line
  http://www.arzi.ru
  http://www.gazety.ru
  http://www.presscafe.ru

СНГ

В странах СНГ подписка принимается в почтовых отделениях по национальным каталогам или по списку номенклатуры «**АРЗИ**»:

- ☑ **Азербайджан** – по объединенному каталогу российских изданий через предприятие по распространению печати «**Гасид**» (370102, г. Баку, ул. Джавадхана, 21)
- ☑ **Казахстан** – по каталогу «**Российская пресса**» через ОАО «**Казпочта**» и ЗАО «**Евразия пресс**»
- ☑ **Беларусь** – по каталогу изданий стран СНГ через РГО «**Белпочта**» (220050, г. Минск, пр-т Ф. Скорины, 10)
- ☑ **Узбекистан** – по каталогу «**Davriy nashrlar**», российские издания через агентство по распространению печати «**Davriy nashrlar**» (7000029, г. Ташкент, пл. Мустакиллик, 5/3, офис 33)
- ☑ **Армения** – по списку номенклатуры «**АРЗИ**» через ГЗАО «**Армпечать**» (375005, г. Ереван, пл. Сасунци Давида, д. 2) и ЗАО «**Контакт-Мамул**» (375002, г. Ереван, ул. Сарьяна, 22)
- ☑ **Грузия** – по списку номенклатуры «**АРЗИ**» через АО «**Сакпресса**» (380019, г. Тбилиси, ул. Хошараульская, 29) и АО «**Мацне**» (380060, г. Тбилиси, пр-т Гамсахурдия, 42)
- ☑ **Молдавия** – по каталогу через ГП «**Почта Молдовей**» (МД-2012, г. Кишинев, бул. Штефан чел Маре, 134)

по списку через ГУП «**Почта Приднестровья**» (MD-3300, г. Тирасполь, ул. Ленина, 17)

по прайс-листу через ООО агентство «**Editil Periodice**» (МД-2012, г. Кишинев, бул. Штефан чел Маре, 134)

- ☑ Подписка для **Украины**:

Киевский главпочтамт
Подписное агентство «**KSS**»
Телефон/факс (044)464-0220

«Open Source» приглашает к сотрудничеству!

Электронное приложение «Open Source» всегда открыто для сотрудничества с новыми авторами, с читателями и их конструктивными предложениями по улучшению издания, обоснованной критикой и любыми отзывами, с компаниями, занимающимися разработкой и продвижением программного обеспечения с открытым кодом. Приветствуются все энтузиасты, желающие опубликовать у нас свои статьи. Тематика нужных материалов очевидна из предназначения приложения, то есть FOSS (Free and Open Source Software): теория и практическое применение; исторические сведения, анализ сегодняшнего положения, прогнозы на будущее и другие аспекты, связанные с открытым ПО.

Среди наиболее интересных на данный момент общих тем можно выделить:

- ☑ общие обзоры новых и/или интересных проектов Open Source и конкретных приложений, свежих версий дистрибутивов Linux, *BSD и других систем;
- ☑ советы и рекомендации новичкам в GNU;
- ☑ истории успеха применения/распространения ПО с открытым кодом;
- ☑ философия и идеология Free Software;
- ☑ разработка приложений с применением средств Open Source.

Желательный объем статей: 6000 или 12000 символов (с пробелами). Примеры актуальных сейчас тем для статей публикуются на http://osa.samag.ru/todo. Но не стоит строго ограничиваться приведенными выше рамками!

Публичное обсуждение «Open Source» проводится на форуме сайта журнала «Системный администратор» по адресу: http://osa.samag.ru/forum. Связаться с редакцией можно по электронной почте osa@samag.ru.

P.S. **За статьи мы платим.**