

Колонка главного редактора



Безусловно, одним из ключевых событий уходящего года стало известие о поглощении гигантом Oracle легендарной ИТ-компании Sun Microsystems. Процесс слияния не завершился, но это лишь подливает масла в огонь — дискуссии о дальнейшей судьбе продуктов Sun не утихают ни на минуту еще с весны. Сделка уже одобрена правительством США, но Еврокомиссия пока обдумывает перспективы этого поглощения. Тем временем, Open Source-сообщество взволновано судьбой популярной СУБД с открытым кодом, ныне принадлежащей Sun, — MySQL. Причины понятны: Oracle никогда не отличилась поддержкой FLOSS и строит бизнес вокруг проприетарных решений, главенствующая роль в которых отведена своей СУБД.

И вот буквально на днях Майкл Видениус, автор оригинальной версии MySQL и со-основатель MySQL AB, через новостное агентство CNews обратился к российскому ИТ-сообществу с просьбой помочь повлиять на решение Еврокомиссии по сделке между Oracle и Sun. По его мнению, поглощение Oracle грозит негативными последствиями для открытой СУБД. Сбор 100 тысяч подписей в поддержку MySQL должен существенно помочь ситуации. Что ж — поможем! Подробности смотрите на <http://monty-says.blogspot.com/2009/12/help-saving-mysql.html>.

Главный редактор
Дмитрий Шурупов
(osa@samag.ru)

«Open Source»

электронное приложение к журналу
«Системный администратор»
№54, 25 декабря 2009 г.

РЕДАКЦИЯ

Исполнительный директор

Владимир Положевец

Главный редактор

Дмитрий Шурупов

Верстка и оформление

Владимир Лукин

Сайт электронного приложения:

<http://osa.samag.ru>

За содержание статей ответственность несет автор. Все права на опубликованные материалы защищены.

Новости мира Open Source

Подведены итоги конкурса свободных разработок России

22 декабря были подведены итоги первого российского конкурса свободных разработок «Лучший свободный проект России», проводимого журналом Linux Format при поддержке спонсоров — Intel, PingWin Software, «ГНУ/Линуксцентр», Wimaxstore и официальной поддержке Министерства связи и массовых коммуникаций РФ.

Для участия в конкурсе было зарегистрировано более 100 свободных проектов, разрабатываемых программистами из России и стран СНГ. Победителями конкурса признаны:

В номинации «Лучший индивидуальный проект»:

✓ **1 место** — отказоустойчивый HTTP-сервер, HTTP-прокси-сервер, почтовый прокси-сервер для ресурсов с высоким трафиком nginx; приз — 100 тысяч рублей.

✓ **2 место** — созданный методом инженерного анализа Linux-драйвер madwimax для устройств доступа к сетям Mobile WiMAX (802.16e), выполненных на основе чипа Samsung CMC-730; приз — 80 тысяч рублей.

✓ **3 место** — открытые кросс-платформенные драйверы принтеров чеков и этикеток Open Ticket Printer Drivers/OTPD; приз — 70 тысяч рублей.

В номинации «Лучший групповой проект»:

✓ **1 место** — система оптического распознавания текстов российской компании Cognitive Technologies, CuneiForm; приз — 100 тысяч рублей.

✓ **2 место** — редактор векторной графики sK1, ориентированный на профессиональное использование в печатной индустрии и поддерживающий проприетарные форматы; приз — 80 тысяч рублей.

✓ **3 место** — консольный файловый менеджер Midnight Commander; приз — 70 тысяч рублей.

Благодаря спонсорам призовой фонд удалось увеличить до 600 тысяч рублей, что позволило наградить специальным призом «Выбор редакции Linux Format» еще два проекта:

✓ **Kalpa.Cloud** — средство разработки сетевых приложений с развитым графическим интерфейсом (танDEM библиотеки C++ — разработчика сервера приложений и универсального тонкого клиента, приз — 50 тысяч рублей.

✓ **Электронный деканат (Free Dean's Office)** — модуль для среды дистанционного обучения Moodle, который добавляет возможность управления процессом обучения, типичным для российских школ, колледжей и вузов, приз — 50 тысяч рублей.

Помимо официальной тройки победителей конкурса в каждой из номинаций, экспертное жюри отметило высокий уровень следующих свободных проектов:

✓ **Ananas** — учетная платформа для разработки бизнес-приложений;

✓ **«Кумир»** — учебная система программирования: предназначена для поддержки начальных курсов информатики и программирования в средней и высшей школе;

✓ **quTiM** — кросс-платформенный клиент сетей обмена мгновенными сообщениями;

✓ **Scan Tailor** — интерактивный инструмент для пост-обработки сканированных страниц;

✓ **Webacula** — веб-интерфейс для системы резервного копирования Bacula.

Подобный конкурс проводится на территории России и стран СНГ впервые. Основной целью конкурса является стимулирование российских разработчиков свободного ПО за счет поддержки наиболее успешных и востребованных проектов, а также составление современной карты свободных проектов, разрабатываемых на территории Российской Федерации и ее ближайших соседей.

Французская армия модифицировала Thunderbird для себя

По данным Reuters, французские военные внесли вклад в код Mozilla Thunderbird и сами активно используют модифицированную версию этого популярного Open Source-клиента электронной почты.

По словам военных Франции, выбор на Thunderbird пал из-за того, что проприетарные решения вроде Microsoft Outlook не позволяют создавать им необходимые для работы дополнения, что просто сделать в случае Thunderbird. В итоге, часть проведенных работ над Thunderbird была опубликована для всех желающих в рамках проекта Trustedbird.

На данный момент Thunderbird и Trustedbird используются на 80 тысячах компьютеров французских военных, и даже более того — этот клиент электрон-

ной почты получил распространение и в других министерствах Франции. Также стоит отметить, что французское правительство уже не первый год помимо Thunderbird использует и другие программные продукты с открытым кодом – в частности, операционную систему GNU/Linux вместо Windows и офисный пакет OpenOffice.org вместо Microsoft Office.

А незадолго до этого сообщения стало известно о выходе новой, третьей, версии Thunderbird. Среди новых возможностей: продвинутые возможности фильтрации результатов поиска (по отправителю, тегу, людям, почтовым рассылкам и т.п.); автоматическое дополнение данными из контакт-листа в поле глобального поиска; новый мастер настройки почтового аккаунта (с поддержкой базы данных популярных почтовых клиентов); обновленная панель инструментов почты (кнопки ответа, пересылки, удаления и другие стали частью каждого почтового сообщения); поддержка табов (вкладок) при просмотре сообщений; режим «умных каталогов» теперь включен по умолчанию; обновленный вид общей информации о письме; новая система архивации писем; менеджер активности (фиксирует взаимодействия Thunderbird с почтовым сервером); новый менеджер дополнений (позволяет просто искать и устанавливать расширения к клиенту); улучшенная интеграция с Gmail; интеграция с результатами поиска в Windows Vista; интеграция с Spotlight в Mac OS X, возможности импорта из Mail.app, а также чтения адресной книги Mac OS X и использования Growl для уведомлений о новой почте; функция автоматической загрузки сообщений по протоколу IMAP в фоновом режиме.

SFLC подала иск на 14 компаний по поводу нарушения GPL

Юридический центр свободы программного обеспечения (Software Freedom Law Center, SFLC) подал судебный иск на 14 компаний-производителей бытовой электроники, обвинив их в нарушении лицензии GNU GPL. Среди компаний, на которые был подан иск, такие гиганты индустрии, как Samsung, JVC и Best Buy. Согласно поданному SFLC иску все они используют в своих продуктах модифицированные версии BusyBox – популярного компактного набора базовых UNIX-утилит, распространяемого под лицензией GPL. Как известно, условия GPL обязывают публиковать исходный код программного обеспечения, созданного на основе кода, лицензированного под ней.

Представители SFLC пытались урегулировать нарушение GPL мирным путем, однако компании проигнорировали их предупреждения. Теперь эту проблему будут решать в суде. Стоит напомнить, что у SFLC уже есть опыт успешных дел против компаний, нарушивших условия лицензии GNU GPL.

И буквально через несколько дней после этого сообщения OpenLogic, специализирующаяся на услугах по программному обеспечению с открытым исходным кодом, объявила о готовности помочь всем компаниям, получившим повестку в суд по причине нарушения лицензии GPL.

Для оказания юридической помощи по вопросам нарушения лицензии GPL в OpenLogic создали специальный «Центр исполнения Open Source» (Open Source Fulfillment Center). Среди его услуг – рекомендации по разработке и распространению ПО с соблюдением Open Source-лицензий, а также аудит приложений, анализ лицензий и соответствующих им требований на содержащийся в программах код. Кроме того, OpenLogic запускает веб-сайт, с которого ее клиенты смогут скачивать лицензированный под GPL исходный код, требующий публикации для выполнения условий лицензии.

Стив Грандчамп (Steve Grandchamp), исполнительный директор OpenLogic: «Наш Open Source Fulfillment Center позволя-

ет компаниям быть уверенными в том, что они выполняют требования GPL и других Open Source-лицензий. Это в свою очередь позволяет нашим клиентам сосредоточить усилия непосредственно на своем бизнесе».

Шаттлворт уходит с поста директора Canonical

Марк Шаттлворт (Mark Shuttleworth), инициировавший проект Ubuntu Linux, объявил о том, что уходит с поста исполнительного директора (CEO) компании Canonical – коммерческого «крыла» этого популярного Linux-дистрибутива.

Тем не менее Шаттлворт вовсе не намерен отстраняться от Ubuntu Linux и Canonical – просто он стремится посвятить больше времени дизайну продукта и расширению взаимодействия Canonical с вендорами и сообществом.

Новым исполнительным директором Canonical станет хорошо известная в сообществе Джейн Силбер (Jane Silber), которая сейчас занимает в Canonical позиции главного директора по операциям (COO) и директора подразделения Online Services. Ожидается, что кадровые перестановки в Canonical должны пройти весьма «гладко».

VideoLAN Project представил видеоредактор VLMC

Проект VideoLAN, известный своим популярным Open Source-медиаплеером VLC, объявил о разработке нового программного продукта – видеоредактора VLMC (VideoLAN Movie Creator).

Официальный анонс VLMC состоялся в рамках мероприятия VideoLAN Dev Days 2009, которое проходило с 18 по 20 декабря в Париже. VideoLAN Movie Creator – это свободный кроссплатформенный инструмент для редактирования видео, позиционируемый как «простое и дружелюбное средство для создания полупрофессиональных роликов».

Первый предварительный релиз VLMC для операционных систем GNU/Linux, Mac OS X и Windows ожидается в ближайшее время. А пока энтузиасты могут получить исходный код видеоредактора на GitHub – там же доступны инструкции для сборки этой программы. Код VLMC распространяется под лицензией GNU GPLv2.

Firefox 3.5 – самый популярный веб-браузер в мире

Согласно последней статистике StatCounter по мировой популярности различных версий веб-браузеров, Firefox 3.5 занял первую строчку.

По итогам прошлой (51-й в этом году) недели, самая большая доля на мировом рынке веб-браузеров – у последнего релиза Open Source-решения от Mozilla, Firefox 3.5. У него 21,93% мирового рынка, что позволило обойти лидировавшего до сих пор Microsoft Internet Explorer 7, у которого теперь 21,2%. Падение популярности IE7 отчасти связано с ростом доли использования нового релиза браузера Microsoft – IE8, «на счету» которого уже 20,33% рынка. Следующая позиция – у другого релиза MSIE, 6.0, с 13,89%.

Среди других учитывавшихся в исследовании StatCounter браузеров: Firefox 3.0 (9,01%), Safari 4.0 (3,03%), Firefox 2.0 (1,01%), Opera 9.6 (0,42%). У «других» браузеров – оставшиеся 9,19%. Очевидно, заметная доля из оставшихся процентов принадлежит Google Chrome, который почему-то остался без особого внимания в отчете.

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

Собираем свое легковесное окружение рабочего стола

В этой статье по сути нет ничего нового. К ее созданию меня побудила тема на форуме [archlinux.org.ru](http://archlinux.org.ru/forum/viewtopic.php?f=15&t=2561) (<http://archlinux.org.ru/forum/viewtopic.php?f=15&t=2561>), которая звучала так: «Подскажите, на базе чего можно собрать графическую среду». На своем примере я покажу, как можно собрать легковесное окружение рабочего стола из множества небольших компонентов.

И сразу же отвечу на несколько вопросов, которые могут возникнуть у читателей еще до знакомства с основным содержанием статьи. «Для чего это нужно?» — для понимания устройства рабочей среды. «Будет ли функционально?» — вполне. «А если я не хочу использовать..?» — да пожалуйста, ведь любой компонент несложно заменить на другой.

Основные компоненты

Для начала попытаюсь определить, что в моем понимании должно включать рабочая графическая среда. Этот список крайне субъективен, и каждый вправе самостоятельно решать, что понадобится лично ему. Вот что получилось у меня:

- ☑ **Менеджер сессий** — он будет управлять запуском приложений, входящих в окружение, и по возможности предоставлять сохранение рабочих сессий, выполнять завершение сессии.
- ☑ **Оконный менеджер** — для управления поведением окон.
- ☑ **Рабочий стол** — для иконок и/или фона рабочего стола.
- ☑ **Файловый менеджер** — для работы с файлами/директориями в графическом виде.
- ☑ **Панель** — для отображения меню приложений, окон запущенных программ, возможности переключения между рабочими столами, отображения иконок приложений в трее. Если готового решения нет, можно сделать несколькими утилитами или вообще обойтись без панели.
- ☑ **Дополнительные мелочи** — без которых жить можно, но неудобно.

Теперь — о том, что я выбрал в каждой из перечисленных «номинаций».

Менеджер сессий

Забегая вперед, сообщу, что в качестве менеджера окон я остановился на Openbox. Поэтому выбор менеджера сессий пал на Staybox (<http://staybox.sourceforge.net>). К тому же данная программа — разработка моего коллеги (w00zy) с уже упомянутого форума, а поддерживать отечественного производителя приятно, особенно когда на то есть веские причины. Итак, этот компонент запускает сам оконный менеджер (по умолчанию — Openbox), а также осуществляет автоматический запуск приложений в едином контексте в соответствии со стандартами freedesktop.org. Собственные средства позволяющие не только выполнять автозапуск, но и осуществлять слежение и перезапуск программ в случае их краха. (Кстати, более подробно о запуске приложений можно почитать в моем блоге: http://hatred.homelinux.net/wiki/zhurnal:2009-10-30_09.09_kak_zapustit.)

Оконный менеджер

Его основная задача — управлять поведением окон на экране. Заголовок окна, рамка, кнопки распахивания, сворачивания и т.п. — все это в его ведомстве. Кроме того, он занимается рас-

положением окон на экране и на рабочих столах. При помощи Openbox, к примеру, можно поставить терминал в качестве фона рабочего стола. При потребности в выборе легкого и управляемого варианта я, как уже упоминал, остановился на Openbox (<http://icculus.org/openbox>). В Интернете много документации о его настройке, а тонкости конфигурации — не моя задача. Справедливости ради лишь отмечу, что использование Openbox далеко не принципиально — с легкостью можно настроить, например, и IceWM с выключенной панелью (на мой взгляд, она не очень удобна, хотя, опять же, кому как).

Рабочий стол и файловый менеджер

Такое объединение не случайно — это почти повсеместная практика, которая повелась еще со времен gmc. В качестве решения я выбрал rstanfm (<http://pcmanfm.sourceforge.net>). Он умеет демонизироваться (т.е. не запускать окно, а только предоставлять рабочий стол), управлять рабочим столом, предоставляя при этом выбор, разрешать ли оконному менеджеру рисовать свое меню. Помимо всего, позволяет автоматически монтировать флешки — причем он сразу и из коробки поддерживал возможность задания опций монтирования, из-за чего проблем с кодировками не было как класса. Ценное, на мой взгляд, свойство rstanfm — возможность быстрого запуска терминальной программы в текущей директории: для этого нужно просто нажать <F4> (для сравнения: Thunar, файловый менеджер из XFCE, такую возможность тоже предоставляет, но только из контекстного меню). Поддерживается и аргумент командной строки, который позволяет сразу запустить диалог поиска (команда `rastan -f`). И последнее удобство, которое нельзя не упомянуть, — возможность открытия директорий во вкладках.

Панель

Тут выбор широк: и tint2 (она не оснащена переключателем рабочих столов, хотя в остальном приятна), и fbpanel (для Fluxbox), и lxpanel (идеи взяты из fbpanel, но более или менее доведены до логического конца), и rupanel, и многие другие. Я остановил свой выбор на lxpanel (<http://wiki.lxde.org/en/LXPanel>): хоть она и является частью проекта LXDE, лишней зависимостей при этом не тянет. Поддерживается меню приложений (даже оно реализовано плагином) и множество расширений, среди которых: Launch для значков запуска приложений, переключатель рабочих столов, TaskBar, Tray, Сри, часы, использование батареи (актуально для ноутбуков). Все это управляется через щелчок правой кнопкой мыши далее «Настройка панели» (или правкой конфигурационного файла).

Мелочи

К этой категории я отношу базовый набор утилит для комфортной работы. Вот они:

- ☑ **Эмулятор терминала** — `termit` (<http://code.google.com/p/termit/wiki/Termit>). Удобный и простой терминал, который позволяет писать внутренние расширения возможностей на Lua. В качестве примера приведено сохранение сессий и переключение кодировок (полезно бывает при администрировании разных серверов).
- ☑ **Текстовый редактор** — `medit` (<http://moedit.sourceforge.net>). Не супер, но достаточно легкий и удобен. Функции можно

Электронное приложение «Open Source»

расширять своими плагинами на C/Python или дополнительными инструментами («Параметры → Инструменты») на shell/Python/Lua.

✓ **Управление архивами** – xarchiver (<http://xarchiver.xfce.org>). Просто и со вкусом. Хотя и разрабатывался в рамках проекта XFCE, зависимостей за собой не тянет.

✓ **Управление раскладками клавиатуры и их индикация** – axkb (<http://github.com/disels/axkb.git>). Простая и удобная утилита, написанная на Qt 4. Как альтернатива: fbxbk (у меня отказалась запускаться) или sbxbk (от автора Staybox) – последняя у меня странно себя «вела» (иногда отказывалась показывать раскладку, хотя переключалась нормально).

✓ **Просмотр картинок** – geeqie (<http://geeqie.sourceforge.net>). Продолжение развития замечательного просмотрщика GQview, обладает минимумом зависимостей, всеми необходимыми функциями и поддержкой просмотра RAW.

✓ **Управление сетевыми подключениями** – wicd (<http://wicd.sourceforge.net>). Особенно удобно для ноутбуков, когда приходится работать в разных сетях.

✓ **Персональный планировщик** – osmo (<http://sourceforge.net/projects/osmo-pim>). Маленький и удобный, предоставляет календарь, записную и адресную книги.

✓ **Электронная почта** – claws-mail (<http://www.claws-mail.org>).

✓ **Браузер** – тут все неоднозначно. Можно попробовать Midori (http://www.twotoasts.de/index.php?pages/midori_summary.html), Arora (<http://arora.googlecode.com>), Kazehakase (<http://kazehakase.sourceforge.jp>), но удобного для себя пока не нашел, хотя у всех есть свои перспективы. Поэтому использую Firefox (единственный, кто выпадает из общей картины легковесности, хотя я собираю с поддержкой рго и использованием распределителя памяти из OpenBSD).

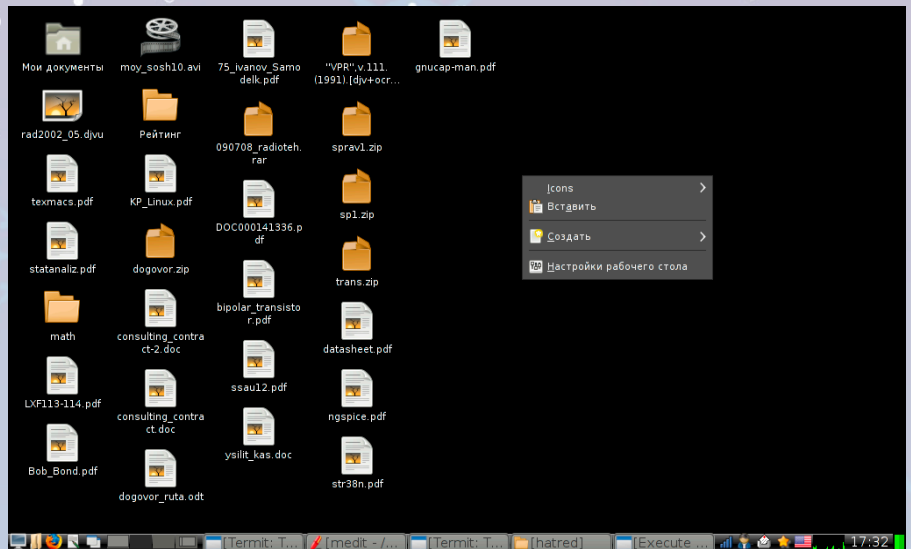
Собираем, склеиваем, настраиваем

Осталось собрать все эти приложения вместе. И сразу ремарка: я предполагаю, что графическая система запускается командой startx. Реализация всего описанного через xdm/gdm/kdm/slim и подобные менеджеры тоже возможна, но останется за рамками статьи.

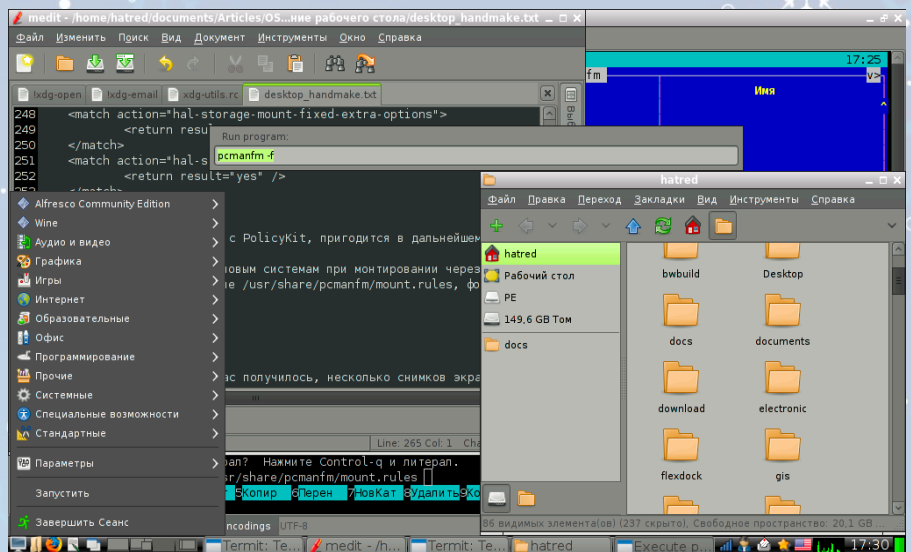
Заострять внимание на установке всех компонентов тоже не буду. Как установить, зависит от дистрибутива. Единственное, о чем стоит предупредить: staybox скорее всего не окажется в вашем дистрибутиве, и его придется собирать из исходных кодов. Пользователям ArchLinux будет проще – пакет есть в AUR (<http://aur.archlinux.org>).

Staybox

В домашней директории создаем файл .xinitrc и заполняем его примерно следующим содержанием:



Получившийся рабочий стол



Основные приложения и меню

```
# http://www.linux-archive.org/debian-kde/20524-fish-ssh-kioslave-
doesnt-work.html
export KDE_FORK_SLAVES=true
```

```
# xbindkeys - демон управления горячими клавишами,
# который не зависит от среды поэтому он у меня запускается
# для любых оконных менеджеров
xscreensaver -nosplash &
xbindkeys
```

```
# отработывает только при запуске отсюда
(sleep 5; axkb) &
```

```
source /etc/X11/xinit/xinitrc.d/*
exec ck-launch-session staybox-session
```

Теперь после запуска startx у нас стартует Openbox под управлением staybox.

Переопределить оконный менеджер можно в файле /etc/xdg/staybox/config:

```
[general]
manager=openbox-staybox
```

Например, прописать «icewm» вместо «openbox-staybox». Определим, что будет запускаться при старте сессии. Для этого исправляем файл ~/.config/staybox/autostart:

```
@lxpanel
@pcmanfm -d
dbus-launch wicd
```


«Собака» (@) перед командой означает, что Staybox будет «следить» за приложением (если оно упадет, произойдет его повторный запуск). Теперь вместе с оконным менеджером будут запускаться панель, рабочий стол с иконками, индикатор раскладок в трее и клиент к wicd.

Ещё можно установить программу staylaunch (<http://sourceforge.net/projects/staybox/files>), которая состоит из двух частей: служебная (собственно staylaunch), запускаемая при старте, и графическая (staylaunch_config), позволяющая настраивать, какие программы и в каком порядке запускаться. Staybox её «подцепляет» автоматически.

В общем, все довольно просто и лаконично. Разве что советую ещё почитать дополнительно про sc-launch-session и dbus-launch.

Openbox, Pcmnfm, Lxpanel

Но от «допиливания» нас никто не освобождал... Для базовой настройки Openbox советую поставить приложение ObConf (<http://icculus.org/openbox/index.php/ObConf>About>), чтобы упростить себе работу с его конфигурационным файлом (он хранится в XML).

Впрочем, без ручной правки все равно не обойдется – например, чтобы сделать вызов окна Run для быстрого запуска приложений, настроить переключение между рабочими столами и комбинацию для разворачивания окна на полный экран. Нужный файл – ~/.config/openbox/rc.xml, за основу которого можно взять /etc/xdg/openbox/rc.xml. Найдём секцию <keyboard> и добавим туда примерно следующее:

```
<keybind key="C-F1">
  <action name="Desktop">
    <desktop>1</desktop>
  </action>
</keybind>
<keybind key="C-F2">
  <action name="Desktop">
    <desktop>2</desktop>
  </action>
</keybind>
<keybind key="C-F3">
  <action name="Desktop">
    <desktop>3</desktop>
  </action>
</keybind>
<keybind key="C-F4">
  <action name="Desktop">
    <desktop>4</desktop>
  </action>
</keybind>

<keybind key="A-F5">
  <action name="ToggleMaximizeFull"/>
</keybind>

<keybind key="A-F2">
  <action name="Execute">
    <command>gmrn</command>
  </action>
</keybind>
```

Эти настройки добавляют возможность переключаться между рабочими столами по сочетанию клавиш <Ctrl> + <F1-F4>, распахивать окно на весь экран – по <Alt> + <F5>, вызывать gmrn – по <Alt> + <F2>.

Последнее (gmrn, <http://gmrn.sourceforge.net>) – это утилита с минималистским интерфейсом для быстрого запуска приложений, обладающая такими дополнительными возможностями, как вывод страниц руководства (для этого надо набрать, например, «man:printf») и открытие сайтов в браузере (достаточно ввести URL – например, <http://google.com/linux>).

Для прочих настроек читайте существующий конфигурационный файл и документацию с официального сайта или ищите конфигуризаторы. В pcmnfm и lxpanel нужно сделать ещё мень-

ше настроек. В первом достаточно зайти в «Правка → Параметры → Дополнительно» и задать терминал, который будет открываться при нажатии <F4> в любом окне файлового менеджера (в моём случае – termite). В lxpanel – зайти в настройки (правой кнопкой мыши по панели, «Настройки панели») на вкладку «Опытные» и задать:

- ☑ Менеджер файлов – pcmnfm %s.
- ☑ Эмулятор терминала – termite.
- ☑ Команда выхода из системы – staybox-logout.

Последняя настройка – самая важная, она предоставляет возможность завершать работу со средой и будет доступна через «Меню → Завершить сеанс». Для возможности выключения и перезагрузки системы отредактируйте файл /etc/PolicyKit/PolicyKit.conf, вписав туда примерно следующее (внутри секции <config>):

```
!-- Power management -->
<match action="org.freedesktop.hal.power-management.shutdown">
  <return result="yes"/>
</match>
<match action="org.freedesktop.hal.power-management.reboot">
  <return result="yes"/>
</match>
<match action="org.freedesktop.hal.power-management.suspend">
  <return result="yes"/>
</match>
<match action="org.freedesktop.hal.power-management.hibernate">
  <return result="yes"/>
</match>
<match action="org.freedesktop.hal.power-management.lcd-panel">
  <return result="yes"/>
</match>
```

Для автоматического монтирования носителей – в том же конфигурационном файле указываем:

```
!-- Mount -->
<match action="org.freedesktop.hal.storage.*">
  <return result="yes"/>
</match>
<match action="hal-storage-mount-fixed-extra-options">
  <return result="yes" />
</match>
<match action="hal-storage-mount-removable-extra-options">
  <return result="yes" />
</match>
```

И в любом случае полезно изучить работу с PolicyKit – это может пригодиться в дальнейшем. Параметры, передаваемые файловым системам при монтировании через pcmnfm/hal, задаются в файле /usr/share/pcmnmfm/mount.rules.

Заключение

Результатом работы, в общем-то, стало частичное повторение такой рабочей среды, как LXDE, но мы использовали свой менеджер сессий и, надеюсь, узнали что-то новое. Любой компонент можно заменить на тот, который вам больше всего подойдет или понравится (альтернативы некоторым приложениям я приводил в статье). В результате, получаем свою достаточно легковесную рабочую среду с приемлемыми для работы возможностями.

Держайте!

Александр Дроздов
(hatred@inbox.ru)

Программирование проверки правописания. Часть 1: Aspell

Исторический экскурс

Проверка правописания (далее для краткости – ПП) стала неотъемлемой частью многих программ, где пользователю приходится вводить много текста. Причем если с наличием ПП в браузерах можно согласиться, то зачем ПП в клиентах ICQ или Jabber, мне непонятно. Всё равно при обмене мгновенными сообщениями роль играет только естественная, «встроенная в человека», грамотность, а не подчеркивание красной линией.

Так сложилось, что в мире UNIX-подобных систем сейчас два мощных игрока в нише ПП: Aspell (<http://aspell.net>) и Hunspell (<http://hunspell.sourceforge.net>). Существует еще Enchant (<http://www.abisource.com/projects/enchant>), однако сам по себе этот продукт – прослойка между вашей программой и другими движками ПП (теми же Aspell, Hunspell и рядом других). Кратко проследим историю развития Aspell и Hunspell и попытаемся разобраться, зачем появился Hunspell, если существует такой капитальный движок, как Aspell.

Вначале был не Aspell, а Ispell. Его алгоритмы ПП были написаны еще в 1971 году на ассемблере для PDP-10 программистом по фамилии Горин. Затем программу переписали на Си и стали развивать, причем, вероятно, впервые в истории программирования появилось понятие API – надо было как-то состыковать Ispell с Emacs, для чего и придумали интерфейс между библиотекой и программой. Затем Кевин Аткинсон написал Pspell, который был совместим с Ispell в режиме pipe и по API. Но с 2001 года Pspell не развивается – Аткинсон двигает продолжателя Aspell. Вернее, код Pspell был слит воедино с кодом Aspell.

И долгое время Aspell оставался (да и сейчас остается) стандартом де-факто. Под него существует множество словарей, Aspell поддерживается уймой программ и так далее. Однако есть одно «но»: внутренне Aspell работает с 8-битными символами. Конечно, ему можно передать на проверку документ в UTF-8, да и в любой кодировке, которую поддерживает iconv, но затем Aspell переведет текст в 8-битную кодировку. Такой подход – безболезненный для языков, где букв не так уж много. Например, русский язык хорошо уживается и в 8-битных рамках. Но ведь есть языки, которым только юникод подавай. И здесь Aspell начинает буксовать. А Кевин Аткинсон хочет и дальше держать внутреннее представление символов в 8 битах, поскольку юникод ему не подходит по причине потери производительности, расхода дискового пространства и так далее – он даже целую страницу документации посвятил возражениям против юникода в пользу 8 бит.

Так возникла необходимость в новом универсальном движке ПП, который бы внутренне был юникодовым. Он возник не сразу и не на пустом месте. Кевин Хендрикс и известный уже Кевин Аткинсон написали для OpenOffice.org движок ПП, который получил имя MySpell. MySpell тоже был «восьмибитным», как и Aspell. И уже на основе MySpell был создан Hunspell, где символы представлены как 16-битный юникод. Однако, многие словари под Hunspell остаются в других кодировках – например, русский в KOI8-R. При этом, используя API, вам надо сначала перевести передаваемое на проверку слово в нужную кодировку, а потом уже «сормить» её Hunspell.

В двух частях этой статьи я рассмотрю работу с Aspell и Hunspell только через API – без pipe, консольного интерфейса и тому подобных вещей.

Работа с Aspell

А API в Aspell – непростое. Но сначала о том, как подключать библиотеку к программе. У Aspell нет пакетного файла для pkgconfig, поэтому правила сборки с Aspell надо прописывать вручную. Для системы сборки autotools надо прописать в файле configure.in примерно такое:

```
AC_CHECK_HEADER(aspell.h,
  LIBS="$LIBS -laspell"
  AC_DEFINE(HAVE_LIBASPELL, 1, [use aspell]),
  echo "no aspell.h - aspell support is disabled"
)
```

Так мы проверяем, есть ли в системе заголовочный файл aspell.h, и если да, то добавляем его в командную строку линковки («\$LIBS -laspell»), а также устанавливаем в истину переменную HAVE_LIBASPELL, которую потом можно проверить из исходника, скажем, на Си или C++:

```
#ifdef HAVE_LIBASPELL
код работы с функциями проверки правописания
#endif
```

Отмечу, что команда AC_CHECK_HEADER проверяет наличие заголовочного файла, однако не прерывает настройку исходника, если условия не удовлетворены. То есть в примере выше Aspell будет опциональной зависимостью программы. Для системы сборки qmake код проверки наличия ASPELL будет таким:

```
exists("/usr/include/aspell.h") {
  message ("Aspell enabled")
  LIBS += -laspell
  DEFINES += ASPELL_ENABLE
}
```

Подключение в исходнике (вне зависимости от системы сборки) выглядит так:

```
#ifdef ASPELL_ENABLE
#include "aspell.h"
#endif
```

Перейдем теперь к API. Алгоритм начала работы с Aspell разобьем на четыре этапа:

1. Создать особый класс для настройки Aspell.
2. Задать с его помощью параметры настройки.
3. Создать экземпляр класса движка ПП, используя заданные выше параметры.
4. Пользоваться.

Этап первый на Си/C++ можно написать так:

```
AspellConfig *spell_config = new _aspell_config();
```

Этап второй – попробуем задать русский язык и кодировку UTF-8:

```
aspell_config_replace(spell_config, "lang", "ru");
aspell_config_replace(spell_config, "encoding", "UTF-8");
```

Итак, функция aspell_config_replace служит для изменения значений параметров, присущих движку, таких как язык, кодировка и тому подобное.

Выбранная кодировка определяет:

- ✓ в какой кодировке движку будет передаваться текст на проверку;
- ✓ в какой кодировке будет сохраняться пользовательский словарь.

При смене языка кодировка сбрасывается к значению по умолчанию, поэтому после:

```
aspell_config_replace (spell_config, "lang", такой-то язык);
```

Всегда вызывайте:

```
aspell_config_replace (spell_config, "encoding", "
такая-то кодировка);
```

Файлы словарей у Aspell могут иметь разные кодировки. Сам Aspell – 8-битный внутри, но вас это не должно беспокоить, поскольку главное – правильно указать кодировку, в которой вы желаете передавать движку слова. Остальное – дело движка. Пользовательские словари Aspell сохраняются в корневом каталоге пользователя. На каждый язык – свой словарный файл. Например, для русского имя файла будет `aspell.ru.pws`.

В приведенном примере использования `aspell_config_replace` язык прописан жестко – «ru», а в реальной программе лучше использовать локаль пользователя, прочитав её из переменной среды окружения `LANG`. В Qt вам поможет статическая функция `QStringList QProcess::systemEnvironment()` – ищите значение `LANG` в возвращенном ею списке, а в GTK+ можно сделать так:

```
gchar *lang = g_getenv ("LANG");
```

Если `LANG` не определена – ставим локаль «C» (стандартная локаль для языка Си). Пример для GTK+:

```
gchar *lang = g_getenv ("LANG");
if (lang)
    aspell_config_replace (config, "lang", lang);
else
    aspell_config_replace (config, "lang", "C");
aspell_config_replace (config, "encoding", "UTF-8");
```

В Qt строки обычно хранятся в `QString`, поэтому, если вы будете передавать какой-то текст в API Aspell, надо переводить `QString` в обычный массив `char` примерно таким образом:

```
mystring.toUtf8().data()
```

Еще насущный вопрос: как узнать, какие словари установлены? Aspell может ответить на него целым рядом функций. Для вызова первой, `get_aspell_dict_info_list`, нам понадобится экземпляр `AspellConfig`, можно даже только что созданный, без каких-либо настроек. Вот код:

```
// список словарей:
AspellDictInfoList *dlist;
// список структур со сведениями о словарях:
AspellDictInfoEnumeration *dels;
// указатель на текущий элемент со сведениями:
const AspellDictInfo *entry;
// получаем список словарей
dlist = get_aspell_dict_info_list (spell_config);
// список сведений о них
dels = aspell_dict_info_list_elements (dlist);

// проходим по всему циклу
while ((entry = aspell_dict_info_enumeration_next (dels)) != 0)
    if (entry)
        cout << entry->name << endl;

// чистим память
delete_aspell_dict_info_enumeration (dels);
```

В этом примере в цикле мы выводим на консоль имена словарей – например, `ru`, `en`, `ru-uo` (русский с «ё») и так далее. Именно эти имена надо использовать при выборе языка для движка:

```
aspell_config_replace (spell_config, "lang", "ru");
```

Перейдем к третьему этапу – созданию экземпляра класса движка ПП. В случае Aspell для этого надо сделать определенный трюк. Вот две функции:

```
struct AspellCanHaveError* new_aspell_speller (
    struct AspellConfig *config);
struct AspellSpeller* to_aspell_speller (
    struct AspellCanHaveError *obj);
```

Прежде чем говорить о них, замечу, что хотя интерфейс у Aspell – «сишный», сама библиотека написана на C++. Внутренней структуре `spellCanHaveError` соответствует класс `CanHaveError`, а `AspellSpeller` – класс `Speller`. И вот что происходит вне «сишного» интерфейса: функция `new_aspell_speller` пытается создать экземпляр класса `Speller`, однако в случае ошибки `new_aspell_speller` возвращает экземпляр `CanHaveError`. Если же всё в порядке, то возвращается `Speller` (на самом деле), которого функцией `to_aspell_speller` надо перевести из `CanHaveError` в `Speller`.

Итак, что мы делаем? Сначала вызываем `new_aspell_speller`:

```
AspellCanHaveError *possible_err = new_aspell_speller (
    spell_config);
```

На деле она либо создала экземпляр `Speller`, либо не создала, но в любом случае вернула указатель на структуру `AspellCanHaveError`. В самом деле красноречивое название. То есть сейчас мы не знаем, что же именно вернула функция: экземпляр `AspellCanHaveError` или `AspellSpeller`. Надо проверить:

```
AspellSpeller *spell_checker;
if (aspell_error_number (possible_err) != 0)
    cout << aspell_error_message (possible_err) << endl;
else
    spell_checker = to_aspell_speller (possible_err);
```

Если проверка удалась на славу и вернулся `AspellSpeller`, то с помощью функции `to_aspell_speller` приводим `possible_err` к типу `AspellSpeller`, иначе же сообщаем в консоль об ошибке или делаем еще что-нибудь подобное.

Теперь важное замечание про очистку памяти. Если вернулся `AspellSpeller`, надо его освободить с помощью функции `delete_aspell_speller`. Если вернулся `AspellCanHaveError` – его освобождаем функцией `delete_aspell_can_have_error`. Однако не вызывайте обе функции, потому что произойдет `segfault`. Освобождайте память именно исходя из типа создавшегося экземпляра. Проще всего сделать так: сразу после «неудачного» вызова функции `aspell_error_number` чистим память посредством `delete_aspell_can_have_error` и завершаем код проверки правописания.

Итак, когда у нас будет рабочий экземпляр `AspellSpeller`, можно приступать к проверке правописания. Для этого предназначена функция `aspell_speller_check`:

```
int aspell_speller_check (struct AspellSpeller *speller,
    const char *word, int word_size);
```

Ей передается указатель на экземпляр `AspellSpeller`, слово и размер слова (в байтах). Если строка завершается символом `null`, то размер можно указать как `-1`. Функция возвращает: `0` – слово есть в словаре; `1` – слова нет в словаре; `-1` – ошибка движка. В коде для простоты, если не надо реагировать на ошибку движка, можно написать примерно так:

```
if (! aspell_speller_check (speller, word, size))
// некий код – скажем, подчеркивание слова с ошибкой
```

В принципе, вот и всё. Для получения списка предположительно верных вариантов написания слова действуем так:

```
// получаем список предположений:
AspellWordList *suggestions = aspell_speller_suggest (
    speller, error_word, -1);

// проверяем, есть ли они вообще?
if (! suggestions)
    ; // например, выходим по return

// иначе на основе suggestions получаем список elements:
AspellStringEnumeration *elements =
    aspell_word_list_elements (suggestions);
const char *word;

// перебираем elements по одному с выводом на консоль:
while (word = aspell_string_enumeration_next (elements))
    cout << word << endl;

// удаляем elements:
delete aspell_string_enumeration (elements);
```

А вот AspellWordList, возвращенный aspell_speller_suggest, удалять из памяти не надо – это указатель на const. Если посмотреть на код выше, да и вообще на примеры использования

Aspell, то возникает навязчивая мысль, что API можно было сократить вдвое...

И напоследок рассмотрим еще один важный аспект работы с Aspell – добавление пользователем нового слова в словарь. Собственно для добавления в текущую сессию работы с aspell служит функция:

```
aspell_speller_add_to_personal (speller, word, strlen (word));
```

Чтобы сохранить словарь в файл, вызовите:

```
aspell_speller_save_all_word_lists (speller);
```

Замечу, что всю работу по именам файлов словарей, и также по загрузке пользовательского словаря берет на себя движок Aspell. Пользовательский словарь загружается автоматически, если существует.

Вторую часть статьи, посвященную Hunspell, читайте в следующем выпуске «Open Source». – **Прим. ред.**

Петр Семилетов
(tea@list.ru)

Отслеживание ошибок с помощью Bugzilla. Часть 2

Первую часть обзора Bugzilla читайте в предыдущем выпуске «Open Source» (№053 от 10.12.2009). – **Прим. ред.**

Учетные записи пользователей

После того как вы настроите политики системы, можно переходить к работе с системой и посмотреть, какие возможности она предлагает. Для добавления пользователя необходимо перейти в секцию Administration → Users. Здесь вы можете производить поиск уже имеющихся пользователей и добавлять новых. Данная операция используется нечасто, поскольку обычно в системе разрешена регистрация новых пользователей, которая не требует участия администратора. Но секция бывает полезна в случаях, если пользователю требуются дополнительные разрешения, не определенные по умолчанию.

Итак, для добавления пользователя следует перейти по ссылке add a new user. От вас потребуются ввести логин (адрес электронной почты), действительное имя пользователя и пароль, определить, будет ли пользователь получать почту, задать текст,

который отображается в случае, если учетная запись отключена (если текст в этом поле указан, то учетная запись отключена) (см. **рис. 1**).

Поиск пользователя осуществляется по имени входа, идентификатору или реальному имени с учетом регистра или без его учета. Для демонстрации работы Bugzilla пройдем регистрацию как новый пользователь. Для этого на главной странице системы переходим по ссылке Open new Bugzilla account. При этом появится одна строка запросом адреса электронной почты нового пользователя и предупреждение, что пока письмо не будет получено, новый пользователь не сможет войти в систему. В письме содержится ссылка для продолжения регистрации пользователя. При переходе по ней вы попадете в раздел регистрации, где потребуются ввести реальное имя пользователя и желаемый пароль. После ввода всех необходимых данных учетная запись готова к использованию (см. **рис. 2**).

Продукты

Теперь, когда в системе есть пользователи, необходимо определить продукты и назначить этим продуктам ответственных за них лиц. Добавление продуктов осуществляется в секции Administration → Products. Там же можно посмотреть существующие продукты и удалить любой из них (при этом будут удалены и сведения, связанные с этим продуктом, такие как компоненты). Компоненты – это составные части продукта, которые не могут существовать сами по себе; они создаются и удаляются вместе с самим продуктом.

Для демонстрации работы добавим новый продукт «Банкомат» – программное обеспечение для некоторого абстрактного банкомата. Для этого заполним поля:

- ☒ **Product** – название добавляемого продукта («Банкомат»). Желательно использовать уникальные значения. Кроме того, название должно быть коротким и понятным.

Рисунок 1. Добавление нового пользователя

- ✓ **Description** – описание продукта. Здесь необходимо кратко описать назначение продукта и его основную функциональность. Например, «Программа банкомата» в нашем случае.
- ✓ **Closed for bug entry** – наличие флажка в этом поле означает, что к продукту не могут быть добавлены записи об ошибках. Его имеет смысл устанавливать для продуктов, поддержка которых уже прекратилась.

Следующие три поля имеет смысл устанавливать только в том случае, если вы используете функцию голосования за ошибки. Если функция отключена, то значения полей игнорируются. В нашем примере они не используются, но назначение полей я рассмотрю. Отмечу, что значение 0, установленное в любое поле, говорит о том, что данное поле не используется. Такое решение позволяет не использовать голосование для некоторых продуктов, если функция голосования включена в системе. Итак: Maximum votes per person – максимальное число голосов, которое может отдать один пользователь. Значение этого поля вы определяете в соответствии с вашими представлениями о при-

нятии решения, но чаще всего один человек может голосовать один раз. Следующее поле определяет, сколько голосов может отдать пользователь за одну ошибку. А Confirmation Threshold указывает, сколько голосов должна набрать ошибка, чтобы выйти из состояния «не подтверждена». По умолчанию все новые ошибки являются неподтвержденными (см. рис. 3).

- ✓ **Edit components** – позволяет задать компоненты, из которых состоит продукт. В нашем примере мы добавим компонент «Модуль авторизации», который отвечает за идентификацию клиента.
- ✓ **Edit versions** – позволяет задать версию продукта. В системе поддерживается наличие нескольких версий продукта. Версия продукта – это просто строка.
- ✓ **Edit Group Access Controls** – позволяет определить группу прав доступа для данного продукта. Благодаря этому можно иметь несколько правил для разных групп и разделять продукты для разных групп.
- ✓ **Bugs** – автоматический счетчик числа ошибок, зарегистрированных для данного продукта.

Компоненты

Любой продукт, как известно, состоит из множества компонентов. Поэтому необходимо определить список компонентов, к которым уже, как правило, и привязываются конкретные ошибки (см. рис. 4). Доступные поля:

- ✓ **Component** – короткое название компонента, используемое для его идентификации.
- ✓ **Component Description** – описание компонента, где можно указать его назначение компонента и основные функции.
- ✓ **Default Assignee** – ответственный за компонент (по умолчанию). Пользователь, которому поручена поддержка данного компонента.
- ✓ **Default CC List** – список пользователей, которым рассылается информация об ошибке.

Рисунок 2. Добавление новой учетной записи

Рисунок 3. Изменение продукта

- ✓ **Bugs** – автоматическое поле, содержащее ошибки, зарегистрированные для данного компонента.

Группы

Для удобства разграничения полномочий и распределения ответственности по проектам между разработчиками предусмотрено понятие группы. В системе нет возможности ограничить права конкретного пользователя с помощью специальных полей, но для этого могут использоваться группы. По умолчанию в системе определено множество системных групп, которые позволяют разграничить права пользователей. Поэтому обычно (в случае небольшой команды) нет необходимости добавлять новые группы. Системные группы не могут быть удалены. Вот самые важные из них:

- ✓ **Администраторы (admin)** – этой группе разрешены любые действия в системе, они отвечают за ее настройку и поддержание в рабочем состоянии.
- ✓ **Имеющие право подтверждать (canconfirm)** – могут подтверждать ошибки (то есть изменять их состояние) или создавать новые копии существующих ошибок.
- ✓ **Имеющие возможность редактировать поля ошибки (editbugs)** – имеют право редактировать любые поля ошибки. По умолчанию все новые пользователи попадают в эту группу (в том числе и анонимные). Таким образом, любой представитель этой группы может задать ошибке приоритет и серьезность, что, на мой взгляд, является недостатком Bugzilla по сравнению с другими подобными системами.
- ✓ **Имеющие возможность редактировать компоненты (editcomponents)** – могут изменять, добавлять и удалять компоненты продуктов. Операция удаления по умолчанию отключена.

Управление группами осуществляется из секции Administration → Groups.

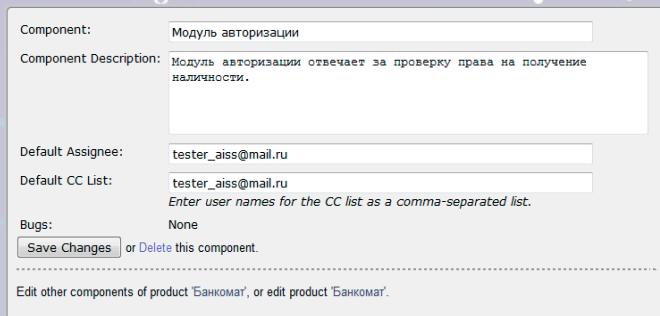


Рисунок 4. Добавление компонента

Процесс обработки ошибки

Так как основная цель подобных систем – отслеживание состояния ошибки, обязательно должна иметься возможность определения последовательности состояний, которые должна пройти запись об ошибке в процессе исправления. Для редактирования процесса обработки ошибки в Bugzilla предусмотрена секция Administration → Bug Status Workflow (см. рис. 5).

Здесь можно простым способом определить, через какие состояния проходит ошибка, устанавливая или снимая галочки в таблице. Слева подписаны начальные состояния, сверху – состояния, в которые можно перейти. Красным показаны состояния, переход в которые из текущего состояния невозможен. Последовательность по умолчанию подходит для большинства публичных проектов, но если вам, например, не нужно подтверждать ошибки, поскольку вносящий их источник является надежным, можете исключить это состояние. Кроме того, можно определить, какие переходы между состояниями требуют обязательного комментария.

Определение значений полей

Определение значений полей – важная функция в настройке системы под конкретные варианты использования. В Bugzilla существует шесть базовых полей, которые используются при описании продуктов, приоритетов и состояний ошибок:

- ☒ **OS** – операционная система. Значения этого поля определяют возможные значения операционной системы для продукта, когда вносится новая запись об ошибке. Это удобно, если ваш программный продукт является межплатформенным.
- ☒ **Hardware** – аппаратная платформа. Значения этого поля определяют возможные значения аппаратных платформ, т.е. архитектуру процессора, для которого предназначено программное обеспечение.

- ☒ **Priority** – приоритет ошибки. По умолчанию значения называются P1–P5, их можно легко изменить.
- ☒ **Severity** – серьезность ошибки. Варианты по умолчанию подходят для большинства продуктов. Они начинаются с минимального уровня – запрос на изменение и заканчиваются определением блокирующей ошибки.
- ☒ **Status** – варианты состояния ошибки. Если вы добавите новые состояния, то не забудьте внести изменения в процесс обработки ошибки.
- ☒ **Resolution** – резолюция. Это поле определяется ответственным за исправление ошибки лицом и определяет, какой «вердикт» вынесен ошибке. Например, может случиться так, что ошибка не подтвердится, и тогда ей будет поставлен вердикт – INVALID (ошибочный), а разрешенная ошибка будет отмечена как FIXED (исправлено).

К сожалению, в Bugzilla не предусмотрена возможность разграничения прав пользователей на перевод ошибки из одного состояния в другое, т. е. любой член группы canconfirm сможет изменить состояние ошибки.

Другие возможности

Кроме описанных возможностей в Bugzilla представлен еще ряд функций, подробно останавливаться на которых я не буду:

Добавление пользовательских полей позволяет ввести собственные информационные поля, которые будут отображаться пользователям при внесении записей об ошибках или запросов на изменения.

Классификация продуктов удобна, если в вашей системе имеется большое число продуктов. Она ускорит процесс поиска продуктов при добавлении новых записей и запросов.

Проверка правильности данных в БД позволяет найти и исправить ошибочные связи в базе данных ошибок и запросов на изменение.

Ключевые слова позволяют задать набор специальных тегов, которыми можно пометить записи об ошибках, что ускоряет процесс их поиска в системе.

Работа с Bugzilla

Мы рассмотрели возможности системы с точки зрения администрирования и определения правил, по которым система существует и работает. Теперь познакомимся с системой со стороны пользователя.

Для этого создадим запись об ошибке и проследим ее жизненный цикл. Войдя в систему с правами обычного пользовате-

This page allows you to define which status transitions are valid in your workflow. For compatibility with older versions of Bugzilla, reopening a bug will only display either UNCONFIRMED or REOPENED (if allowed by your workflow) but not both. The decision depends on whether the bug has ever been confirmed or not. So it is a good idea to allow both transitions and let Bugzilla select the correct one.

From	To						
	UNCONFIRMED	NEW	ASSIGNED	REOPENED	RESOLVED	VERIFIED	CLOSED
{Start}	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
UNCONFIRMED		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
NEW	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ASSIGNED	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
REOPENED	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RESOLVED	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
VERIFIED	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
CLOSED	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

When a bug is marked as a duplicate of another one or is moved to another installation, the bug status is automatically set to RESOLVED. All transitions to this bug status must then be valid (this is the reason why you cannot edit them above).

Note: you can change this setting by visiting the [Parameters](#) page and editing the `duplicate_or_move_bug_status` parameter.

[Commit Changes](#) - [Cancel Changes](#) - [View Comments Required on Status Transitions](#)

Рисунок 5. Настройка процесса обработки ошибки

ля (которого мы добавили ранее), увидим главную страницу системы (см. **рис. 6**).

Она незначительно отличается от той, что видит администратор: на ней нет ссылки на секцию администрирования и не отображается дополнительная информация о доступных обновлениях. Представлены следующие возможности:

- ☒ поиск существующей записи об ошибке;
- ☒ добавление новой записи об ошибке;
- ☒ получение сводных отчетов и графиков;
- ☒ изменение своих настроек и пароля.

Bugzilla обладает еще одной полезной функцией для пользователей Mozilla Firefox – боковая панель для быстрого доступа к ее различным возможностям. Кроме того, можно установить поисковый сервис для Firefox и Internet Explorer 7. Итак, добавим новую запись об ошибке. Bugzilla заинтересуется, для какого продукта мы добавляем запись – выберем банкомат (см. **рис. 7**).

Обратите внимание, что синие поля заполняются автоматически в соответствии со значениями полей, способ изменения которых был рассмотрен выше. В качестве компонента выбираем модуль авторизации, задаем уровень серьезности ошибки critical и приоритет P1. Кроме того, нам позволено выбрать начальное состояние ошибки, так что укажем самое первое – UNCONFIRMED (не подтверждена). В поле Summary кратко опишем ошибку: «Ошибка с повторным вводом PIN-кода», а в поле Description – последовательность действий, приводящих к ошибке: «Осуществлен ввод неверного значения PIN-кода; Повторен ввод кода, код правильный; Все операции, совершенные в банкомате, теперь будут провалены». В случае такой необходимости можно было бы еще приложить файл или снимок экрана. Отправляем ошибку в базу, нажав на Commit.

После этого мы попадем в форму с информацией о зарегистрированной ошибке, где будет указано название и состояние ошибки. На адреса электронной почты ответственных лиц отправлены уведомления о новой ошибке. В этой форме мы, как ответственное лицо, можем написать свой комментарий, подтвердить или отклонить ошибку. Подтвердим ее и оставим комментарий: «Действия стабильно приводят к ошибке – недоработка модуля», установим статус ASSIGNED (назначена) – теперь мы ответственны за ее исправление. Представим, что ошибку исправили, и теперь устанавливаем ей статус RESOLVED (решена). От нас потребуют установить дополнительное состояние, которое говорит о варианте разрешения ошибки – выбираем FIXED (исправлена). Теперь специалист по тестированию произведет тестирование ошибки. Если он успешно протестировал модуль и не обнаружил больше ошибки, сможет отметить ее статусом VERIFIED FIXED.

Не забывайте писать комментарии к каждому изменению состояния записи – это облегчит дальнейшую работу всем участникам цикла разработки, тестирования и внедрения, а также пользователи будут знать, решены ли их проблемы, в какой версии программного обеспечения и в чем заключалась проблема.

По истечении большого периода времени наша ошибка не возникала вновь, и теперь мы можем ее закрыть, установив статус CLOSED FIXED.

Поиск записей

Поиск записей об ошибках организован интересно. Для быстрого поиска используются порядковые номера ошибок в базе данных. Кроме этого, есть расширенный поиск, позволяющий искать ошибку по значениям полей, если вы не знаете номер ошибки. В результате поиска по номеру будет выведена ошибка, зарегистрированная с таким номером. Если вы просматриваете ошибку как неавторизованный пользователь – не сможете ничего редактировать. Однако, при вхождении в систему станет доступным внесение в запись изменения, в том числе повторное открытие ошибки, если вы вдруг обнаружите, что она не решена полностью.

Отчеты

Система была бы неполной, если бы не позволяла создавать отчеты об ошибках в продуктах. Ведь статистика – важная часть процесса разработки, которая позволяет иметь количественные показатели. Она также позволяет лучше учитывать риски, связанные с ошибками в будущих проектах. В Bugzilla имеется возможность создания отчетов в графической и табличной формах. Графические отчеты представляют собой круговые и линейчатые диаграммы, при этом вам позволено выбрать, какие значения будут откладываться по обеим осям. Табличные отчеты создаются по аналогии с графическими – в них вы так же определяете, что будет записано в горизонтальные и вертикальные ячейки. Чтобы создать отчет, необходимо перейти по ссылке Reports, заполнить поля формы и запустить процесс генерации. Заполнять поля несложно, поскольку вам будут предложены существующие

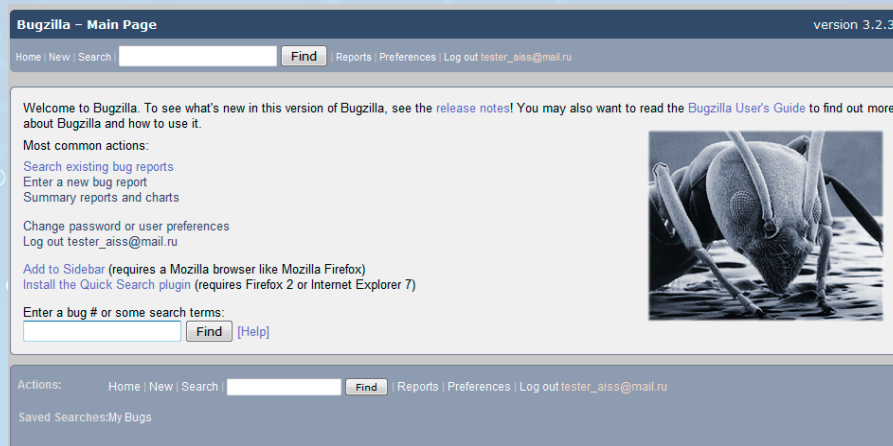


Рисунок 6. Главная страница для пользователя

Before reporting a bug, please read the [bug writing guidelines](#), please look at the list of [most frequently reported bugs](#), and please [search](#) for the bug.

Product: Банкомат	Reporter: tester_aiss@mail.ru
Component: Модуль авторизации	Component Description: Модуль авторизации отвечает за проверку права на получение наличности.
Version: 1.0	Severity: enhancement
	Platform: PC
	OS: Windows
	Priority: P5
Initial State: NEW	
Assign To: tester_aiss@mail.ru	
CC:	
Default CC: tester_aiss@mail.ru	

Рисунок 7. Форма ввода ошибки

Электронное приложение «Open Source»

варианты значений, а также отображены все продукты и входящие в их состав компоненты (см. рис. 8).

Выводы

Bugzilla предоставляет весь необходимый набор функций для успешного отслеживания ошибок и запросов на изменения. Система существует уже долгое время и зарекомендовала себя в среде разработчиков как надежная и достаточно простая в использовании.

Однако отметим и минусы системы:

- ☑ **Сравнительно сложный процесс установки**, связанный с установкой дополнительных модулей в Perl.
- ☑ **Сложный механизм формирования отчетов и довольно простая их форма**; невозможно экспортировать результаты в другие форматы (кроме HTML).
- ☑ **Неполные возможности разграничить права пользователей** на изменение состояния ошибок.

Банкомат		
Product		
	Банкомат	Total
Severity	critical	1
	Total	1

Bar | Line | Table | CSV

Edit this report

Рисунок 8. Отчет в табличной форме

- Тем не менее Bugzilla является замечательной альтернативой коммерческим системам отслеживания ошибок и запросов на изменение.

Александр Шайхразеев
(alexander.shaykhrazeev@gmail.com)

«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, основанной 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.

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 ра-
бочим 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-Пресссервис»
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 националь-
ным каталогам или по списку номенкла-
туры «АРЗИ»:

- ✓ **Азербайджан** – по объединенному
каталогу российских изданий через
предприятие по распространению пе-
чати «Гасид» (370102, г. Баку, ул. Джа-
вадхана, 21)

- ✓ **Казахстан** – по каталогу «Россий-
ская 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 номенклату-
ры «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 «По-
шта Молдовей» (МД-2012, г. Кишинев,
бул. Штефан чел Маре, 134)
по списку через ГУП «Почта Придне-
стровья» (МД-3300, г. Тирасполь, ул.
Ленина, 17)
по прайс-листу через ООО агентство
«Editil Periodice» (МД-2012, г. Киши-
нев, бул. Штефан чел Маре, 134)
- ✓ Подписка для **Украины**:
Киевский главпочтамт
Подписное агентство «KSS»
Телефон/факс (044)464-0220