АНО «Институт логики, когнитологии и развития личности»
ALT Linux

# Третья конференция разработчиков свободных программ на Протве

Обнинск, 24–26 июля 2006 года

Тезисы докладов

Москва,
Институт Логики,
2006

В книге собраны тезисы докладов, одобренных Программным комитетом Третьей конференции разработчиков свободных программ. Круг рассматриваемых тем весьма широк: от новейших системных и прикладных разработок до правовых проблем, вопросов организации работы в проектах и аналитики.

# Программа конференции

### 23 июля

19.00–22.00: Регистрация в холле гостиницы

### 24 июля

10.00–12.30: Регистрация в холле гостиницы

12.00–12.30: Кофе

**Дневное заседание**
**12.30–14.00**

12.30–12.40: Открытие конференции

12.40–13.20: Александр Давыдов

FOSS-центричные информационные системы как основное направление ИТ. Опыт бизнеса и технологий NAUMEN 9

13.20–14.00: Федор Зуев

Проект GPLv3 и российское законодательство — возможные конфликты ..... 11

14.00–14.45: Обед

### Вечернее заседание
### 14.45–19.10



## 25 июля

### Утреннее заседание
### 9.30–13.45

## Дневное заседание 14.40–16.40

16.05–16.40: Кирилл Колышкин, Кирилл Коротаев



16.40–17.00: Кофе-брейк

**Вечернее заседание**
**17.05–19.00**

17.05–19.00: Круглый стол, ведущий Дмитрий Левин

## 26 июля

**Утреннее заседание**
**9.30–11.35**

9.30–9.55: А. В. Балагута, Е. А. Чичкарев



9.55–10.20: Павел Виноградов



10.20–10.45: Павел Морозов



10.45–11.10: Денис Медведев



11.10–11.35: Е. А. Чичкарев, Н. В. Назаренко



11.35–11.55: Кофе-Брейк

**Семинар**
**Регулирование информационных технологий, используемых в государственном управлении**

12.00–12.15: Открытие семинара, вступительное слово от организаторов (Алексей Новодворский, ALT Linux, Москва)

12.15–12.35: Вступительное слово от Минэкономразвития РФ

## 23 июля

19.00–22.00: Регистрация в холле гостиницы * *

## 24 июля

10.00–12.30: Регистрация в холле гостиницы * *

12.00–12.30: Кофе * * * * *

# Дневное заседание
## 12.30–14.00

12.30–12.40: Открытие конференции * * *

12.40–13.20

Александр Давыдов Екатеринбург, «Наумен»

### FOSS-центричные информационные системы как основное направление ИТ. Опыт бизнеса и технологий NAUMEN

Аннотация

Бизнес компании NAUMEN — проприетарные решения CRM, Service Desk, BPM, e-Learning, Contact Center и заказные проекты, основанные на технологиях FOSS. Еще два решения имеют лицензию FOSS:

- управления поддержкой разработки софта (Software Lifecycle Management);
- управления открытыми конкурсами, заключением и исполнением контрактов в рамках целевых государственных программ.

В 2005 году выпущена FOSS-лицензия на платформу Naumen Kernel прикладной разработки на Java, на которой разработаны большинство продуктов компании. В докладе рассказывается стратегия компании и развитие стека технологий, делаются выводы и даются рекомендации для компаний и учебных заведений.

Предпосылки выбора FOSS для бизнеса.

1. Частью, софт становится более общим, универсальным по отражаемым законам и распространенности, подобен артефактам фундаментальной науки.
2. На универсальные законы природы невозможны имущественные права (патенты). Не продать закон Ньютона или уравнения Максвелла.

Из этих соображений вытекает:

- Проблемы с продажей лицензий будут нарастать по мере универсализации и улучшения софта.
- В центр распространенных ИС встанет универсальный бесплатный открытый софт.
- Частный софт займет место на периферии, отражая частности, многообразие мира.
- Универсальный частный софт неизбежно становится FOSS.
- Чем лучше становится софт, тем меньше денег за лицензии.

Выводы по результатам деятельности:

- Ресурсы и технологии FOSS стали основой проектного и мелкосерийного бизнеса.
- Компания получила финансовую и политическую независимость от поставщиков, современные технологии и экспертизу (а затем госконтракты на FOSS).
- Проекты FOSS в России могут развиваться только государством, оно заинтересовано в продвижении FOSS как основы самостоятельного бизнеса (но пока не осознает этого).
- Ведущая сила в продвижении FOSS — университеты, там выигрывается или проигрывается пространство ИТ-технологий.

## 13.20–14.00

Фёдор Зуев Иркутск, Институт Земной Коры СО РАН

### Проект GPLv3 и российское законодательство — возможные конфликты

Аннотация

Хочу сразу подчеркнуть, что все нижеперечисленные проблемы все же не настолько серьезны, чтобы сделать использование GPLv3 в России даже и в нынешнем виде невозможным или всерьез опасным. Русское гражданское право достаточно гибко и разумно, чтобы быть в состоянии простить — и фактически регулярно прощать — и куда более грубые огрехи. Однако это все же требует некоторой доброй воли со стороны судей и в некоторых специальных ситуациях лицензия может не сработать как ожидалось. Кроме того, части проблем, имеющихся уже в GPL2 также можно избежать.

Проект третьей версии GPL, опубликованный FSF в начале этого года, имеет, по-видимому, гораздо больше сложностей с российским законодательством и российскими обычаями, чем действующая вторая версия.

Проблемы можно разделить на следующие группы.

- Несоблюдение формы авторского договора.
- Чрезмерная подстройка под софтверно-патентный режим.
- Разница в правовой концепции Интернета в европейском и американском праве.
- Гослицензирование видов деятельности.

#### Форма договора

Самое важное и самое простое. Ст. 31 ЗоАП требует упоминания в авторском договоре ряда существенных условий. Отсутствие их может, теоретически, послужить основанием для признания авторского договора ничтожным. Уже в GPL2 эти условия были явно прописаны не полностью, кое-что приходилось домысливать по контексту.

В GPL3 они еще более покорежены. Так, ЗоАП требует указывать размер авторского вознаграждения, а ГК предписывает считать любой договор по умолчанию возмездным. Однако из GPL3 было выброшено упоминание о безвозмездном характере передачи авторских прав.

Более того, в пункте 9 GPL3 вообще заявляется, что GPL-де не является договором. Ввиду очевидной нелепости подобного утверждения в любой юрисдикции кроме некоторых (далеко не всех) штатов США, последствия его совершенно непредсказуемы.

### Софтверные патенты

С ними вопрос не столько юридический, сколько, так сказать, принципиальный, и выражается он в известной поговорке «Не буди лихо, пока оно тихо». Софтверные патенты в России пока запрещены законом. Но — софтверные патенты никогда и нигде законом и не разрешались явно. Их легализация везде происходила путем накопления все более нелепых интерпретаций, толкований, прецедентов и обычаев. Россия уже сделала ряд шагов по этому пути. Еще ни один софтверный патент не был подтвержден в суде (хотя в ряде случаев до этого оставалось совсем немного), но известен целый ряд случаев когда разного рода жулики благополучно *получали* их.

Язык GPL3, в ряде мест ссылающийся на софтверные патенты как на неоспоримую данность, может послужить нам плохую службу, способствуя их легализации в России. И не одних патентов. В качестве примера можно привести неоднократно поминаемое в GPL3 «permission to run program». Авторское право не знает такого permission, запуск не относится к числу исключительных авторских прав, и его появление в GPL3 связано исключительно с интеграцией туда патентной лицензии. Объяснять людям, что какую-то часть GPL можно и нужно игнорировать, поскольку она ссылается на не существующие у нас сущности, — будет весьма нетривиальным занятием.

### Интернет

В части, регламентирующей распространение лицензируемой программы через Интернет (§ 6d), GPL3 существеннейшим образом опирается на популярную в США (но и там, сколь мне известно, не единственную) юридическую теорию, представляющую передачу по цифровым сетям в качестве серии копирований, осуществляемых отправите-

лем, получателем, промежуточными провайдерами и т. п. Теория эта приятна сердцу адвокатов копирайт-картелей, но за пределами США не употребляется. В странах с более вменяемой юридической системой (в том числе и в РФ) передача по интернету рассматривается как род вещания, подобного эфирному и кабельному вещанию.

Чтобы еще более запутать ситуацию, напомню, что с сентября в российском ЗоАП появляется новое исключительное право «доведение до всеобщего сведения», по декларированному намерению законодателя означающее именно распространение через Интернет, но по юридической технике имеющее мало с ним общего. Предсказываю, что это новое право породит многолетнюю неразбериху.

Иными словами, распространение способами, включающими распространение по Интернету, следовало бы описывать в возможно более общей форме и не закладываться на их конкретную юридическую интерпретацию.

### Гослицензирование

§ 12 GPL3 требует, чтобы лицензиат отказался от распространения программы, если он по каким-либо причинам не в состоянии передать получателям копий все права, зафиксированные в лицензии. Возникает вопрос — относится ли это требование также и к ситуациям, связанным с государственным лицензированием определенных видов деятельности, в частности криптографии? Окончательного ответа у меня пока нет, но общем — очень похоже на то. И в таком случае GPL-лицензированные криптографические, медицинские и другие подобные пакты оказываются в РФ на птичьих правах.

Эта проблема до определенной степени присутствует уже в GPL2, но в GPL3 это требование расширено.

**14.00–14.45: Обед** * * * * *

# Вечернее заседание
14.45–19.10

## 14.45–15.20

Константин Осипов Москва, MySQL AB
Проект: MySQL http://dev.mysql.com, http://mysql.com

### Новые возможности MySQL 5.1

Аннотация

MySQL 5.1 — новая версии в серии баз данных MySQL AB, предоставляющая следующие новые возможности:

- partitioning;
- новый формат репликации, основанный на репликации данных;
- возможность смешанного хранения данных на диске и памяти в кластере;
- внутренний диспетчер задач (cron);
- новые таблицы для аудита: `mysql.slow_log` и `mysql.general_log`.

Доклад направлен на освещение новых возможностей версии 5.1, причин, по которым они были включены в релиз, общему видению разработчиков MySQL, в каком направлении следует развивать продукт.

В конце 2005 г. MySQL AB выпустила новую версию популярного сервера баз данных — 5.0. Эта версия добавила наибольшее количество новых возможностей за всю историю MySQL — поддержку хранимых процедур, представлений, триггеров, новые типы таблиц. Новые возможности позволили MySQL приобрести новых пользователей, у тех же, кто и до этого пользовался сервером, возможно, возник вопрос: не превращается ли эта некогда легкая и быстрая база данных в неповоротливого монстра? Доклад рассказывает о новых возможностях версии 5.1 в свете обозначенной проблемы, освещает на что направлены усилия разработчиков и чего следует ждать в его последующих версиях (5.2).

После выпуска версии 5.0 команда MySQL сконцентрировала свои основные усилия на качестве продукта — за 2005–2006 год были исправлены тысячи ошибок. Исправления некоторых ошибок требовали более серьезных изменений, и поэтому они были включены в новую версию.

Таким серьёзным «исправлением» стал новый формат репликации. Тем, кто знаком с MySQL давно, известно что репликация в нём реализована не самым традиционным способом: для экономии пространства на диске и увеличения производительности MySQL реплицирует не данные, а запросы. Так, когда пользователь выполняет `UPDATE t1 SET a=3` MySQL создаёт сообщение репликации, содержащее текст запроса, а не изменённые данные. Этот текст затем пересылается на replication slave и выполняется там.

Это позволяет сэкономить десятки килобайт пересылаемых данных на средний запрос, но и имеет свои недостатки.

Так, если запрос содержит данные, не реплицируемые в текстовом виде, к примеру: `UPDATE t1 SET a=TRUE_RAND()`, где `TRUE_RAND()` возвращает случайное число, запрос заведомо выполнится иначе на replication slave.

MySQL 5.1 добавляет возможность реплицировать такие запросы по-новому: в новом формате реплицируются не запросы, а изменённые данные, таким образом replication slave получает те же изменённые строки, что и таблицы на master.

Сервер автоматически выбирает нужный режим репликации, чтобы максимизировать скорость и сохранить надёжность.

Другой важной возможностью добавленной в MySQL 5.1 стал partitioning — административные средства управления большими объёмами данных. Идея partitioning состоит в эксплуатации принципа локальности данных для облегчения работы с большими таблицами. Пользователю предоставляется возможность указать, по какому принципу данные следует разбивать при хранении, и в дальнейшем оптимизированное хранение используется для ускорения выборки и управления данными.

Наиболее типичный пример использования partitioning — управление архивными данными за десятки лет. Так, таблица, хранящая записи о всех сделках, совершённых начиная с 1990 года, может быть разбита на разделы (partitions) по годам. Тогда запрос, который работает только с данными одного года истории, будет автоматически

перенаправлен сервером для работы с одним участком диска и сможет быть выполнен существенно быстрее.

Подробнее об этих и других возможностях новой версии будет рассказано на конференции.

## 15.20–15.55

Пётр Новодворский Москва, ВМК МГУ

### Исследования в области операционных систем: ждать ли рассвета?

В 80-ые и начале 90-ых исследование операционных систем переживало бурное развитие, однако множество идей, описанных в статьях этих лет, так и не прижились в широко используемых ОС. На сегодняшний день определились явные проблемы, которые существуют в старых операционных системах и начинается процесс поиска старых идей, разрешающих их, и адаптации их к существующим реалиям. В докладе будет описано, каких изменений, по мнению автора, следует ожидать в ближайшие годы в операционных системах, а также в моделях их разработки.

Стандартная архитектура ОС, которая лежит в основе всех современных ОС общего назначения, окончательно сформировалась в середине 80-ых и уходит корнями в ОС Multics[7]. В ней появилась технология защиты памяти и понятие процесса, которые существуют в почти неизмененном виде по сегодняшний день. Интерфейс к файловой системе был разработан в UNIX, наследнике Multics, в конце 70-ых и также существенно не изменялся. Последним сильным изменением в операционных системах оказалось появление сетевых технологий и стандартизация TCP/IP в качестве сетевого протокола в начале 80-ых.

С тех пор исследователями операционных систем были разработаны новые технологии, которые решали многие проблемы, возникающие перед разработчиками ОС, однако они не находили применения из-за высокого требования к ресурсам или сложности в реализации. Разработчики ОС предпочитали смиряться с существованием проблем или создавать обходные пути, не решающие (workarounds) сути проблем. В результате игнорирования этих проблем, операционная система сохра-

няла в целом старую архитектуру, однако обрастала заплатами, которые в эту архитектуру не совсем вписывались (такие куски кода обычно называют hack). Такие заплаты накапливались как снежный ком, и в результате структура операционной системы значительно усложнялась.

Рассмотрим одну из наиболее актуальных проблем — проблему изоляции контекстов, в которых исполняются потенциально опасные программы, и ограничение ущерба, наносимого ошибками в этих программмах. Эта проблема целиком не решена до сих пор, хотя было приведено много способов её решения. Из решений этой проблемы, предложенных исследователями ОС, можно привести следующие: микроядерная архитектура [2, 1], написание потенциально опасных программ на языках с безопасной типизацией [9], создание безопасных оболочек для драйверов [3]. Если посмотреть на изменения в архитектурах ОС общего назначения, мы заметим, что ни одно из этих изменений не было применено. Разработчики более аккуратно обходились с архитектурой и не решались изменять её в корне: в данном контексте можно рассмотреть такие разработки как `chroot` и `jail`: надо отметить, что в их основе лежат изменения в планировщике задач, сетевой и файловой подсистемах ядра, однако архитектура остается той же. В результате обе этих разработки не решают проблемы целиком.

Нежелание изменять существующую архитектуру ведет к латанию старых проблем и не ведет к их решению. Однако это является не единственной проблемой. Операционная система остается библиотекой обращения пользовательских программ к оборудованию. Архитектура оборудования изменяется, и поддержка этих изменений в ОС тоже оказывается зачастую заплатой к архитектуре ОС. Это можно хорошо видеть в том, как долго появлялась поддержка многопроцессорности в FreeBSD и Linux. Похожих проблем можно ожидать при появлении многоядерных процессоров с большим количеством ядер.

Введение новых функциональностей в ОС также затруднено их архитектурой, в частности, из-за их монолитности. Разработка Linux осложнена тем, что существует единственный источник всех поддерживаемых драйверов устройств, и включить поддержку устройства можно, лишь договорившись с Линусом или другим главным разработчиком. Исследователи показали [4], как можно реализовать четкие интерфейсы между компонентами ОС и их взаимозаменяемость. Первым реальным изменением в архитектуре ОС за последние годы стала поддержка виртуализации. Виртуализация сама по себе не является новой идеей, существуют ранние разработки по виртуализации [5], и

она использовалась ещё с 70-ых, однако эта технология не использовалась в ОС общего назначения на дешевых компьютерах. Однако в конце 90-ых и начале 2000-ых она начала использоваться для изоляции небольших контекстов [10] или рапространения ПО [8], решения проблем, о которых IBM, фактически главный популяризатор виртуализации до 2000-ых, не задумывалась. В нашем контексте виртуализация заслуживает внимания из-за того, что разработчики ОС согласились с радикальным изменением архитектуры своих ОС (Xen, vserver, openvirtuozzo) ради решения существующих проблем. С другой стороны, виртуализация позволяет внедрять совершенно новые технологии, не изменяя архитектуру существующих систем[6], позволяя найти компромисс между исследователями и разработчиками ОС.

Сегодня можно ожидать, что разработчики ОС наконец начнут реализовывать изменения, предложенные исследователями. Это стало возможным как благодаря технологии виртуализации, так и из-за смещения акцента в разработке компьютеров новых поколений. Война гигагерцев на рабочих столах прошла, и на смену ей пришла война новшеств в архитектурах. В качестве примера можно привести технологию многоядерности, технологии поддержки виртуализации и грядущие изменения в архитектуре кэшей. Прирост гигарцев не требовал изменения архитектуры ОС, и разработчики могли просто поддерживать старый код, добавляя поддержку стороннего оборудования, но при появлении этих технологий им придется заняться чтением статей для поиска новых идей.

### Литература

[1] *Liedtke J.* Toward real microkernels // *Communications of the ACM*. — 1996. — Vol. 39, no. 9. — Pp. 70–77. `citeseer.ist.¬ psu.edu/liedtke96toward.html`.

[2] Mach: A new kernel foundation for unix development // USENIX Conference Proceedings. — USENIX, 1986. — Pp. 93–112.

[3] Nooks: an architecture for reliable device drivers // EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop: beyond the PC. — New York, NY, USA: ACM Press, 2002. — Pp. 102–107.

[4] The pebble component-based operating system // Proc. 1999 USENIX Technical Conference. — 1999. — Pp. 267–282. citeseer.ist.psu.edu/gabber99pebble.html.

[5] *Popek G. J., Goldberg R. P.* Formal requirements for virtualizable third generation architectures // *Commun. ACM*. — 1974. — Vol. 17, no. 7. — Pp. 412–421.

[6] Pre-virtualization: Uniting two worlds / J. LeVasseur, V. Uhlig, B. Leslie et al. // Poster session of the 20th ACM Symposium on Operating Systems Principles (SOSP-20). — Brighton, United Kingdom, 2005. — . http://l4ka.org/publications/.

[7] *Saltzer J. H.* Protection and the control of information sharing in multics // *Commun. ACM*. — 1974. — Vol. 17, no. 7. — Pp. 388–402.

[8] *Sapuntzakis C., Lam M. S.* Virtual appliances in the Collective: A road to hassle-free computing // Proceedings of the Ninth Workshop on Hot Topics in Operating System. — 2003. — May.

[9] SPIN - an extensible microkernel for application-specific operating system services // ACM SIGOPS European Workshop. — 1994. — Pp. 68–71. citeseer.ist.psu.edu/article/chambers94spin.¬ html.

[10] *Whitaker A., Shaw M., Gribble S.* Denali: Lightweight virtual machines for distributed and networked applications // Proceedings of the USENIX Annual Technical Conference. — 2002. citeseer.¬ ist.psu.edu/whitaker02denali.html.

15.55–16.30

Александр Боковой, Steven French Москва,
IBM Linux Technology Center

Проект: Samba Team http://linux-cifs.samba.org

## Опыт использования файловой системы CIFS для организации обмена данными в вычислительных кластерах

Современные вычислительные кластерные системы имеют достаточно сложную и зачастую гетерогенную архитектуру, интегрируя вычислительные ресурсы различных аппаратных архитектур и операционных систем. Эта интеграция происходит на разных уровнях, однако одной из наиболее насущных проблем является необходимость совместного доступа к исследуемым данным. Для этого используются сетевые и так называемые «кластерные» файловые системы, которые специально разрабатываются с учетом особенностей одновременного доступа к данным с разных сетевых систем. Для разграничения такого доступа и предотвращения возможности повреждения данных в файловых системах реализуют различные механизмы, как правило, основанные на использовании совместного блокирования доступа к файлам.

Одной из наиболее популярных и простых в эксплуатации сетевых файловых систем, применяемых при построении вычислительных кластеров, является NFS — Network File System, реализации которой присутствуют во всех современных UNIX-подобных операционных системах. При всей своей простоте NFS обеспечивает приемлемый уровень производительности для небольших кластерных систем, однако крайне сложно масштабируется при росте размера вычислительного кластера. К этому стоит добавить отсутствие адекватных средств контроля доступа к данным на уровне пользователей в наиболее распространенной версии протокола NFS — версии 3.

Специализированные кластерные файловые системы позволяют обойти эти и другие естественные ограничения, однако сложнее во внедрении и разработке. Неудивительно, что количество таких файловых систем невелико — масштаб разработки служит определенного рода ограничителем. Из существующих реализаций стоит выделить

Global Parallel File System (GPFS) компании IBM, CXFS компании SGI, Parallel Virtual File System (PVFS) разработки университета города Клемзона (Южная Каролина, США) и Аргоннской национальной лаборатории (США), Global File System (GFS) компании RedHat и Lustre компании Cluster File Systems, Inc.

Перечисленные кластерные файловые системы обладают определенными достоинствами и недостатками, позволяющими при выборе файловой системы для конкретного решения склониться в пользу того или иного подхода. Одним из факторов, учитываемых при таком выборе, является наличие или отсутствие реализации файловой системы для эксплуатируемых версий операционных систем и аппаратных архитектур. В качестве примера можно привести проект по реализации вычислительного кластера для НПО «Сатурн», который был реализован в 2005 году компаниями КРОК и IBM с применением оборудования и программного обеспечения компании IBM. В частности, кластер представляет собой гетерогенную среду из машин архитектур x86_64 и IA-64. Для последней из них недоступна реализация выбранной разработчиками в качестве базовой файловой системы GPFS на платформе GNU/Linux, поэтому было решено применить комбинированный подход и интегрировать доступ к GPFS средствами NFS.

Примененный подход типичен и позволил полученному кластеру занять четвертую позицию в списке крупнейших вычислительных систем СНГ (`www.supercomputers.ru`), однако недостатки использования NFS проявились в процессе запуска реальных исследований. Программное обеспечение НПО «Сатурн» создает неравномерную нагрузку на подсистему хранения и сетевую инфраструктуру, используемую для доступа к файлам, заставляя NFS генерировать множество операций записи, ограниченных размером буфера на клиенте (в данном случае — GNU/Linux на IA-64), приводя к фрагментированию результирующих пакетов и увеличению времени обработки на стороне сервера.

Потенциальным способом увеличения производительности было бы использование асинхронных операций записи, поддерживаемых NFSv3, однако в данном случае разработчиков кластера ждал неприятный сюрприз: использование асинхронной записи в NFSv3 иногда приводит к разрушению данных, хранимых на разделах GPFS, поскольку GPFS умеет динамически перестраивать свою внутреннюю структуру, к чему сервер NFS оказывается не готов. Таким образом, использование синхронной записи по NFS существенно ограничивает пропускную способность файловых операций.

Для решения этой проблемы предпринимались различные подходы. Одним из них стало применение альтернативной NFS сетевой файловой системы CIFS и ее свободной реализации в проекте Samba (`www.¬ samba.org`), а также новой версии клиента CIFS для ядра Linux, разрабатываемой в Центре технологий Linux компании IBM. Этот реализация клиентской части протокола (которую мы далее будем называть CIFS.ko, чтобы отличать от собственно протокола) включена в ядро Linux достаточно давно, однако в течение последних двух лет были выполнены работы, увеличившие стабильность и производительность, особенно для операций записи.

Необходимо отметить, что серверная часть CIFS, реализованная в проекте Samba, обеспечивает более полную интеграцию с кластерными файловыми системами, чем NFS, и позволяет эффективно использовать различные методы кэширования данных и асинхронного доступа, которые по-прежнему недоступны при использовании NFS для организации доступа к GPFS. К тому же, GPFS по-прежнему недоступна для операционных систем семейства Windows, где протокол CIFS является основным методом доступа к файлам по сети.

В результате экспериментов было установлено, что для тех задач, которые решаются в НПО «Сатурн», CIFS.ko подходит лучше NFS, достигая на операциях записи фантастических результатов с 5000-кратным увеличением производительности. Однако для реального внедрения необходимо было обеспечить плавную миграцию с NFS.

Одним из основным постулатов NFS является «клиент всегда прав» — в частности, операции по проверке прав доступа выполняются на клиентской стороне, разгружая тем самым сервер. В случае CIFS ситуация прямо противоположная — сервер выполняет проверку доступа, производя, в зависимости от конфигурации, сложные преобразования пользовательских идентификаторов и даже транслируя одни из них в другие перед проверкой. Такая схема более гибка и позволяет клиентским станциям иметь отличные от сервера идентификаторы пользователей, однако в данном конкретном случае она создает больше неудобств.

В результате была реализована NFS-подобная схема, когда CIFS.ko выполняет проверку прав доступа локально, используя права, установленные на сервере, и текущие права процесса, выполняющего доступ к ресурсам. Это позволило не модифицировать имеющуюся модель распределения прав между пользователями в организации, но в то же время добиться эффективного ее контроля.

Для увеличения общей производительности операций записи в CIFS.ko была реализована поддержка больших размеров передаваемых блоков данных. NFSv3 поддерживает размеры блоков до 32 Кб и использует размер блока в 8 Кб по умолчанию. CIFS.ko поддерживает размеры блоков до 1 Мб и при взаимодействии с сервером Samba версии 3.0 использует размер блока в 52 Кб, драматически увеличивая производительность. Для некоторых типов задач помогает увеличение размера блока до 128 Кб, особенно на архитектурах, поддерживающих большие размеры страниц памяти (largepages, например, POWER5).

Особо хотелось бы отметить специальный режим работы CIFS.ko, направленный на интеграцию с не-Windows системами под управлением Samba. В этом случае поддерживаются так называемые «CIFS POSIX Extensions», позволяющие в рамках CIFS передавать информацию, специфическую для POSIX-совместимых приложений. В частности, это касается обработки прав доступа, описываемых POSIX ACLs и реализованных в GPFS и практически всех локальных файловых системах в GNU/Linux и FreeBSD. В этом случае CIFS.ko запрашивает, а сервер Samba возвращает реальную информацию о POSIX-совместимой системе, которая могла бы быть неверно оттранслирована в термины CIFS, как, в частности, и происходило с преобразованием определенных POSIX ACLs в NT ACLs в связи с тем, что эти модели представления прав доступа не полностью совместимы между собой. Сохранение правильных прав доступа к данным играет существенную роль в реальных ситуациях, поскольку довольно часто вычислительные кластера используются для обсчета данных, носящих конфиденциальный и даже секретный характер.

Внесенные изменения позволили повысить общую производительность кластера НПО «Сатурн» при решении задач заказчика, увеличить общую надежность обмена данными системы и добиться повышения уровня сохранности конфиденциальных данных. Результаты модификаций кода CIFS.ko были включены в ядро Linux версии 2.6.16 и старше, а также были разработаны специальные версии CIFS.ko для более старых ядер, применяемых в коммерческих версиях дистрибутивов GNU/Linux от компаний RedHat (ядро версии 2.6.9) и Novell (SUSE LINUX, ядро версии 2.6.5). Все модификации и специальные версии доступны с сайта `http://linux-cifs.samba.org/`, а также в рамках ядра Linux, начиная с версии 2.6.16.

В дальнейшем предполагается продолжить работу по усовершенствованию CIFS.ko, в частности, интеграции с Kerberos и увеличение

производительности операций чтения путем их распараллеливания, а также интеграции результатов уже проведенных исследований по увеличению производительности при использовании соединений с высокой латентностью.

## 16.30–17.05

Вартан Хачатуров, Александр Шишкин Санкт-Петербург,
Siemens Corporate Technology, Embedded Linux Competence Center

Проект: SLIND `http://emdebian.org/slind.html`

# SLIND: опыт разработки embedded-дистрибутива на базе Debian

Аннотация

Доклад будет посвящён разработке сравнительно небольшого дистрибутива для встраиваемых систем, с использованием Debian package management. Будут изложены как общие соображения относительно полезности этой затеи, так и ряд технических проблем, возникших в процессе разработки, методы их решения и возможные перспективы.

### Введение

В течение продолжительного времени разработка встраиваемых систем предполагала использование специализированных ОС и прикладного ПО, что было оправдано небольшими мощностями платформ. За последние несколько лет процессорные мощности возросли, а энергопотребление и стоимость были сведены к уровню, который позволил применять сравнительно быстрое «железо» для встраиваемых систем. В свою очередь, на нём уже (после незначительных модификаций) стало возможным использование ядер ОС общего назначения, вместе с огромным количеством существующего прикладного ПО. Естественно, Linux не остался в стороне.

### Постановка задачи

Основная проблема, с которой нам пришлось столкнуться в процессе разработки Linux-платформ для различных устройств, состоя-

ла в том, что умы инженеров всё ещё остались в прошлом — в тех прекрасных временах, когда мужчины были настоящими мужчинами, женщины — настоящими женщинами, маленькие мохнатые создания с Альфа Центавра — настоящими маленькими мохнатыми созданиями с Альфа Центавра, а embedded-инженеры мыслили терминами «прошивок», «программаторов» и «прожига». Поэтому пресловутые «прошивки» Linux-устройств представляли из себя беспорядочную свалку из бинарных запускаемых файлов и библиотек, со всеми возможными типами скриптов загрузки, без какого бы то ни было контроля версий или чего-либо, хотя бы отдалённо напоминающего package management. Соответственно, процесс обновления прошивок был классическим — «запустите наш прошиватель с образом прошивки и молитесь, чтобы не было скачка питания». Нас, выросших в условиях дистрибутива, который можно аккуратно целиком обновить даже со скачком через релиз, в котором отдельные пакеты можно ставить и удалять вместе с зависимостями, подобная ситуация устроить не могла. Так родился SLIND.

## SLIND

SLIND состоит из двух основных частей: target-части, то есть пакетов, непосредственно устанавливаемых на устройство, и host-части, то есть набора кросс-компиляторов в поддерживаемые архитектуры. На сегодняшний момент SLIND состоит из приблизительно 40 пакетов в target-части, собранных для платформ PowerPC, x86, ARM и MIPS, доступных в вариантах uClibc и glibc, и использующих на выбор busybox или GNU coreutils/sysvinit. Компиляторы предназначены для установки на Debian «sarge»: gcc-3.4 и gcc-4.1. Кросс-сборка пакетов осуществляется с помощью инфраструктуры dpkg-cross, разработка ведётся при помощи subversion.

### Проблемы

Разумеется, без проблем не обошлось. Вот некоторые из них:

- Установка пакетов требует исполнения их сценариев времени инсталляции. Следовательно, установка пакетов для target-архитектуры невозможна на host-архитектуре, а следовательно, без участия самого устройства невозможно подготовить образ его файловой системы.

- Установка дистрибутива на target требует инсталлятора, а существующие инсталляторы всё же слишком велики для запуска на некоторых типах целевых платформ. Разработчики «большого Debian» не слишком заинтересованы в подобных применениях идей и пакетов Debian, и поэтому, например, часть сборочной системы gcc, реализующая сборку пакетов с кросс-компиляторами, практически не обновляется.
- Большинство пакетов в Debian не готовы для кросс-сборки «из коробки». Крайне трудно адаптировать инфраструктуру Debian для автоматической кросс-сборки пакетов для всех целевых платформ.

**Выводы**

Несмотря на отмеченные проблемы, проект уверенно развивается, хотя и страдает от недостаточного количества подготовленных разработчиков, авторов документации и переводчиков. На сегодняшний момент мы можем гордиться тем, что мы создали единственную сколь-нибудь полную реализацию масштабной идеи Embedded Debian, зародившейся ещё в середине девяностых годов.

17.05–17.25: Кофе-брейк * * * *

## 17.25–18.00

Станислав Иевлев Москва, ALT Linux

Проект: Alterator `http://wiki.sisyphus.ru/alterator`

### Alterator: интерфейс пользователя

Аннотация

Альтератор — это не платформа, это язык, на котором формулируются задачи. За последний год особенно продвинулась подсистема, связанная с описанием диалогового интерфейса пользователя. Язык получился простой, регулярный и легко расширяемый. Пользователь может

пользоваться как заранее подготовленными примитивами, так и легко создавать новые. Одинаково легко создаются как простые виджеты, так и сложные, состоящие из большого числа компонент.

Свобода языка компенсируется простой, но эффективной системой безопасности.

Alterator взаимодействует с пользователем по принципу «описание интерфейса — браузер». Браузер может представлять интерфейс как в текстовом режиме, так и в графическом, и даже, при желании, и в виде голосовых сообщений. Разные варианты представления имеют разные функциональные возможности и особенности, а описание интерфейса должно быть единым и, что немаловажно, простым для использования.

Ряд проблем приходится решать на стороне браузера, например, сведение различных моделей размещения элементов документа (layout) к одной. Особенно это актуально для http-интерфейса, как самого ограниченного в возможности создания пользовательских алгоритмов размещения (custom layouts). В alterator принято указывать размеры элементов в процентах относительно родительского. Возможны также два варианта упаковки виджетов — вертикальная и горизонтальная.

Для описания разметки используются s-выражения, для программирования динамики — язык Scheme. Использование единого языка как для разметки интерфейса, так и для описания поведения позволяет легко описывать такие вещи, как генерация интерфейса на основе данных, принятых из внешнего источника (backend). В проектах, где используются различные языки, приходится прибегать к разным дополнительным средствам поддержки. Например, в XUL для этого предлагается язык шаблонов (templates).

Вот простой пример описания интерфейса в alterator:

```
(document:surround "/std/base") (label "Hello" pixmap
"hello.png")
(button "Quit" (when clicked (document:end)))
```

Интересным моментом является то, что в alterator используются единые правила для задания первоначального состояния виджета, изменения и опроса оного. Например команда `(my-button pixmap "hello.png")` — приведёт к изменению иконки у кнопки. А команда `(my-button pixmap)` — вернёт текущее значение атрибута `pixmap`. Идея проста: у каждого атрибута имеется минимальное количество параметров, необходимое для его корректного задания. Вызов виджета с полностью определённым атрибутом — запрос на модификацию.

Вызов с недоопределённым атрибутом — запрос текущего состояния. Если производится запрос состояния атрибута, являющийся функцией обратного вызова — значит, эту функцию надо исполнить. Например, «программный» щелчок по кнопке задаётся командой `(my-button clicked)`.

Важно, что простота синтаксиса не исключает гибкости. Язык описания — многоуровневый. В самой основе лежит язык описания атрибутов и контейнеров с атрибутами. Браузер визуализирует контейнеры исходя из значений их атрибутов и своих возможностей по отображению. Например, в целях экономии площади изображения список `(listbox)` может отобразится в свёрнутой форме `(combobox)`.

Пользователь волен задавать собственные атрибуты и собственные контейнеры. Сам язык, на котором описываются большинство интерфейсов в alterator, задан в стандартных файлах (обратите внимание на имя `/std/base` в предыдущем примере). Вот пример инструкции задания новых атрибутов:

```
(document:envelop with-attributes (my-text my-pixmap
my-width
my-height))
```

Совершенно тривиально создаются комплексные виджеты, например, виджет для изменения пароля пользователя. Что задание, что работа с подобным виджетом ничем не отличается от работы с базовыми элементами, такими как `label` и `button`. Это возможно, потому что каждое описание документа есть виджет, начиная с самого примитивного, состоящего из одной единственной инструкции задания типа. Например:

```
(document:surround "/std/attributes") type "label"
```

И наоборот, каждый виджет есть некий документ. Поэтому виджет создаётся простой инструкцией, наподобие:

```
(document:envelop with-container-presentations ( (label
'/std/label)))
```

Где `/std/label` есть примитивный документ, показанный в предыдущем примере.

Всё это, и кроме того неограниченные возможности самого базового языка Scheme по расширению, позволяют строить язык описания

интерфейса, наиболее подходящий для решения той или иной задачи, стоящей перед пользователями alterator.

### Ссылки

1. http://wiki.sisyphus.ru/Alterator/evolution
   Простейший язык описания интерфейсов.
2. http://wiki.sisyphus.ru/Alterator/start
   Низкий старт, или как сделать свой первый модуль для Alterator за пять минут.

## 18.00–18.35

Виктор Вагнер Москва, ООО Криптоком

Проект: OpenSSL
http://www.cryptocom.ru/OpenSource/OpenSSL_rus.html

## Проблемы встраивания российской криптографии в дистрибутивы свободных ОС

Аннотация

Обзор возможных применений российских криптоалгоритмов в Linux и FreeBSD и анализ опыта встравивания этих алгоритмов в приложения на базе OpenSSL.

В современных дистрибутивах свободых ОС (Linux, *BSD) применяются две основные группы криптографических протоколов:

1. Протоколы с распределенной сетью доверия (gnupg, ssh).
2. Протоколы с централизованной сетью доверия (с использованием ключевой инфраструктуры X509).

Российские нормативные акты, требующие применения национальных стандартов, регламентируют преимущественно решения с централизованной сетью доверия.

Таким образом, основной областью применения национальной криптографии являются защита электронной почты в формате S/MIME и

XML-документов в формате XMLSEC и защита каналов с использованием TLS или каких-либо VPN.

Имеет также смысл использование российских алгоритмов в ssh, но если для S/MIME и X509 PKI уже приняты RFC на использование российских алгоритмов, а для TLS и XMLSEC существуют черновые версии (drafts), то о существовании работ по расширению протокола ssh с целью использования российских алгоритмов нам пока неизвестно.

В типичной Linux или *BSD системе имеется по крайней мере три реализации TLS — OpenSSL, GnuTLS и NSS (библиотека, используемая Mozilla). Большая часть прикладных программ использует одну из этих библиотек либо позволяет при компиляции выбрать одну из них.

При построении своего решения мы выбрали OpenSSL, так как эта библиотека поддерживается наибольшим количеством серверного программного обеспечения.

Мы реализовали российские алгоритмы в виде отдельного подгружаемого модуля (engine), чтобы избежать излишнего разрастания кода ядра OpenSSL. Функциональность системы модулей OpenSSL версии 0.9.8 оказалась недостаточной для добавления новых алгоритмов с открытым ключом, поэтому пришлось разработать патч, реалзиующий необходимую функциональнсть. В настоящий момент ведется совместная работа с OpenSSL team по включению необходимой функциональности в следующий релиз OpenSSL.

Сертификационные требования к средствам защиты информации требуют побитового совпадения используемого криптографического модуля с представленным на сертификацию, поэтому сертифицировать как средство криптографической информации решение, распространяемое в исходных текстах, невозможно. Поэтому мы пошли по пути создания двух независимых совместимых реализаций — одной коммерческой, которая будет сертифицироваться, а второй — открытой, распространяемой по BSD-style лицензии.

Тестирование различных приложений на поддержку работы с этими реализациями привело к следующим результатам:

1. Большинство приложений требуют модификации для того, чтобы поддержать использование подгружаемых модулей (engines) в OpenSSL. Эта проблема касается не только взаимозаменяемых реализаций национальных алгоритмов, но и использования аппа-

ратных криптоакселераторов. Так, например поддержка директивы SSLCryptoDevice в Apache2 появилась только в версии 2.2.0.

2. Около четверти серверных приложений используют RSA-специфичный API для загрузки секретных ключей, т. е. не способны работать не только с российскими алгоритмами, но и с DSA (или требуют конфигурирования уровня компиляции для работы с DSA, как stunnel 3.x).

3. Чем более специализировано и гибко приложение, тем больше вероятность использования в нем недокументированных функций OpenSSL и прямого доступа к полям внутренних структур, которые нам пришлось модифицировать при добавлении поддержки российских алгоритмов.

4. В большинстве дистрибуитивов поставляются разнообразные скриптовые врапперы, упрощающие генерацию сертификатов или заявок. Как правило, все они требуют модификации для работы с алгоритмами, отличными от RSA.

Таким образом, работа по поддержке российских криптоалгоритмов не ограничивается реализацией этих алгоритмов в системных библиотеках, реализующих протоколы TLS и SMIME. Требуется проведение достаточно большой работы по тестированию всех приложений, использующих OpenSSL.

На данный момент нами исследованы следующие приложения:

Apache 1.3 и 2.2
stunnel 3.x и 4.x
lynx
tcltls
wget
mutt (как S/MIME-приложение и как TLS-клиент)
openldap
PostgreSQL 8.1.x
netkit telnet и netkit telnetd
Postfix
dovecot
Emacs/GNUS

В планах:

1. openvpn как средство реализации VPN на базе TLS и DTLS.

2. courier

3. jabberd

4. KDE
5. proftpd
6. libneon

Кроме того, в libxmlsec нами была реализована работа с российскими криптоалгоритмами на базе MS CryptoAPI. Планируется поддержать и работу с российскими криптоалгоритмами с использованием OpenSSL.

## 18.35–19.10

Андрей Михеев Москва, Консалтинговая группа РУНА
Проект: RUNA WFE http://sourceforge.net/projects/runawfe

### Разработка OpenSource workflow-системы

Аннотация

RUNA WFE — это open source решение по управлению бизнес-процессами, основанное на популярном workflow-ядре JBOSS-JBPM, ориентированное на конечного пользователя.

Характеристики:

- возможность интеграции существующих разнородных приложений предприятия;
- удобный веб-интерфейс пользователя;
- графический редактор бизнес-процессов;
- боты для выполнения автоматических заданий;
- гибкая система определения исполнителей на основе ролей;
- простая интеграция с существующими реляционными базами данных;
- система безопасности, позволяющая интеграцию с LDAP/MS Active Directory;
- локализация на английский, французский, немецкий, голландский, испанский, русский, украинский и китайский языки;
- поддержка ОС Windows, Linux, Solaris, FreeBSD.

### Workflow-системы

В настоящее время перспективным подходом к организации управления предприятием становится процессный подход, в соответствии с которым деятельность предприятия представляется в виде множества бизнес-процессов — наборов заданий, выполняемых как людьми, так и информационными системами предприятия.

Вследствие этого у предприятий возникает потребность в workflow-системах — гибких компьютерных системах, реализующих процессный подход к управлению. Важной характеристикой этих систем является возможность быстрой разработки и изменения бизнес-процессов предприятия без изменения кода системы. Workflow-систему можно также рассматривать как центральную часть современных систем масштаба предприятия. Если в КИС отсутствует workflow-компонента, то логика бизнес-процессов оказывается рассеянной по различным элементам системы.

Эксперты прогнозирует значительный рост доли workflow-систем на рынке информационных систем в ближайшие годы. В течение двух последних лет активно развиваются OpenSource workflow-проекты, они практически достигли состояния, когда их можно использовать на промышленных предприятиях. В ближайшей перспективе OpenSource workflow-системы могут составить реальную конкуренцию коммерческим проприетарным системам.

### Архитектура системы

Компоненты системы:

- Ядро системы (на основе JBOSS JBPM)
  - Содержит набор определений бизнес-процессов;
  - Содержит набор выполняющихся экземпляров бизнес-процессов.
- Клиент
  - Task list. (Набор графических форм, содержит очереди поступивших работ, сортировки и фильтры.)
  - Проигрыватель. форм. (Визуализирует формы, разработанные в редакторе процессов.)

  - Административный интерфейс (Показывает состояния процессов, позволяет фильтровать и останавливать процессы.)

- Графический редактор процессов.
- Боты. (Приложения специального вида, которые также как и обычные пользователи могут выполнять задания.)
- Подсистема управления правами доступа.

Система состоит из следующих слоев:

- delegate
- service
- logic
- dao

Слои delegate и service реализуют J2EE pattern service-delegate, слой dao реализует pattern dao, в слое logic реализована основная логика работы системы.

### Выбор OpenSource-компонентов, сравнение с другими проектами

С самого начала проекта систему предполагалось построить на основе уже существующих OpenSource-компонентов.

В качестве возможного ядра системы были рассмотрены следующие проекты:

- Jboss
- jBpm
- Bonita
- Shark
- WfmOpen
- Openwfe
- Yawl

Все проекты являются зрелыми системами. Jboss jBpm и Openwfe базируются на собственных языках описания бизнес-процессов, однако в Jboss jBpm недавно появилось расширение для языка BPEL. Проект WfmOpen и Shark используют язык XPDL.

Проекты Jboss jBpm и Openwfe включают OpenSource графические редакторы бизнес-процессов, для WfmOpen компания Danet GmbH разрабатывает коммерческий проприетарный редактор. Это ядро, так же как и Shark, позволяет использовать XPDL-совместимые редакторы, например jawe. Bonita не содержит встроенного языка описания бизнес-процессов, представляет собой единую среду для разработки и исполнения процессов. Проект Yawl базируется на одноименном языке определения бизнес-процессов YAWL. Этот проект интересен скорее для академических научных исследований в области workflow, чем для промышленного использования, так как слишком сложен для практического использования менеджерами предприятий.

В результате анализа был выбран проект Jboss jBpm как имеющий простую архитектуру и вместе с тем развитую функциональность. Однако проект Jboss jBpm ориентирован скорее на разработчиков систем, нам же требовалась система, ориентированная на конечного пользователя. Поэтому workflow-окружение решено было разработать самостоятельно. После консультаций с командой Jboss jBpm на sourceforge был заведен проект RUNA WFE, целью которого является разработка компонентов workflow-окружения для ядра Jboss jBpm.

В проекте использованы следующие технологии:

- EJB 2.0 (stateless session beans) — интерфейс взаимодействия с серверной частью и декларативная транзакционность JSP 2.0;
- Servlet 2.3;
- Struts 1.2 — web-интерфейс;
- Hibernate 2.1 — ORM;
- Eclipse — графический редактор;
- JAAS — аутентификация.

### Статус проекта

На портале sourceforge проект имеет статус Production/Stable, система RUNA WFE эксплуатируется в КГ РУНА с июля 2005 г. Также

система используется OpenSource сообществом в различных странах — произведено 8259 скачиваний системы.

### Краткая история

Проект начался в сентябре 2003 года, первый дистрибутив wfe (environment) был выложен на sourceforge в ноябре 2004 года, в июне 2005 г. появилось OnLine demo, в декабре 2005 года был готов первый дистрибутив графического редактора процессов. В 2005 г. проект стал дипломантом конкурса Java-технологий, проводившимся корпорацией Sun Microsystems при официальной поддержке Министерства информационных технологий и связи РФ. В 2006 г. проект получил Honorable Mentions статус на конкурсе JBoss Innovation Award в двух категориях: Управление бизнес-процессами и Хранение информации.

# 25 июля

## Утреннее заседание

9.30–13.45

9.30–9.55

Тарас Кошелев Великий Новгород, Новгородский ГУ им. Ярослава Мудрого

### FOSS как дальнейшая стадия развития глобального социума и культуры

Глобализация проявляется во многих сферах, в экономике, политике, культуре.

Пространственный барьер для коммуникации был фактически снят техникой, что послужило одной из причин создания информационной среды. К ней применимы принципы синергетики: развитие происходит нелинейно. Другими словами внутри могут возникать конструкции, сложность и качественные характеристики которых не могут быть получены простым сложением исходных элементов. Таким образом информационная среда обладает системными свойствами.

Информационные потоки за счет нелинейного взаимодействия создают на порядок более сложные и структурированные системы. Одной из таких систем стало движение FOSS.

Современное авторское право ведет к тому, что складывается рынок программного обеспечения (и вообще информации), более того, такой рынок, который неизбежно порождает монополии. Красноречивым примером может считаться рынок ПО для персональных компьютеров.

Сторонники FOSS настаивают, что программное обеспечение (а более широко — информация вообще) не может быть рыночным продуктом и следовательно не может быть товаром в привычном смысле. Дабы ограничить монополии и предупредить их появление в будущем,

сторонники FOSS ратуют за повсеместный переход к новой модели авторского права.

Феномен FOSS как альтернативный подход к авторскому праву возник в конце 70-х годов прошлого века и до какого-то момента был уделом немногих единиц и идеалистически настроенных лидеров-основателей. Все продолжалось бы и дальше столь неспешно, если бы в западный мир не ворвались компьютерные сети и — конечно — интернет. Для будущего сообщества[1] FOSS появление такого способа коммуникации стало даром небес. В то время количество сторонников было крайне мало, но благодаря налаживанию связей между ними, возник синергетический эффект – появилось сообщество. Создаются методы коллективной работы.

При этом сообществу для сплочения своих рядов требовался какой-то символ, который бы показывал значимость самого движения, эффективность механизма разработки и адекватность системы альтернативного авторского права. И такой символ появился. Им, что логично, стало ядро операционной системы — Linux. Наиболее красноречивым признаком жизнеспособности такого подхода является факт конкуренции систем на основе ядра Linux с коммерческими системами *NIX, а скоро и Windows. Где проиграла OS/2, системы GNU/Linux вполне нормально развиваются. В итоге, к началу нового тысячелетия системы GNU/Linux стали потенциально конкурентоспособны[2]. Что было замечено крупными корпорациями. То есть механизм разработки программного обеспечения на основе принципов свободы вполне адекватен и жизнеспособен.

Как и всякая система с хаотической составляющей, сообщество создает огромное количество «дочерних» флуктуаций, которые относительно быстро исчезают. Речь идет о тысячах незаконченных проектов. Перспективы увеличения эффективности подобной схемы разработки очень велики, очевидно, что дальнейшая коммерциализации FOSS неизбежна.

Очевидных основания самого феномена два. Первое — «этическое», состоит в том, что программные продукты, документация к ним должны быть свободными[3]. Символом данного основания в полной мере

[1]Речь идет именно о сообществе, где велика доля личной харизмы и влияния на основе реальных заслуг и проделанной работы.

[2]Вообще говоря, правильней было бы говорить не о конкуренции, а о замещении, но как-то прижилось именно понятие конкуренции.

[3]Именно свободными, а не просто открытыми.

можно считать Ричарда М. Столлмена (Richard M. Stallman). Основной упор делается на том, что современной авторское право устарело, и с появлением новой техники, должно измениться. Именно за это ратуют сторонники свободного ПО. [1]

Вторым основанием появления и развития FOSS считается «утилитарное». Оно состоит в том, что, поделившись своей работой с другими разработчиками, взамен можно получить их расположение, а также активное участие в выявлении ошибок, усовершенствовании и т. д. Более того, как показывает опыт, программное обеспечение с открытым кодом может продаваться.

Таким образом мы видим, что само движение за открытый/свободный исходный программный код не является монолитным, а внутри сообществ идут дискуссии. Особенно это заметно в связи с обсуждением новой лицензии. Данное противостояние не какая-то досадная неприятность или упущение, это фундаментальное свойство системы. В данном споре можно видеть соединение меркантильных интересов и этических идеалов, что в каком-то смысле — очередной спор материалистов и идеалистов.

Если бы сообщество придерживалось и действовало только исходя из одного основания, то оно было бы обречено на поражение в конфликте с «миром» закрытых программ. Однако в сообществе имеется некоторый дуализм. Поэтому можно утверждать, что движение FOSS — это именно следующая, а не альтернативная стадия развития общественных отношений, начало которой было положено с появлением глобальных средств коммуникаций, в частности интернациональных компьютерных сетей.

### Литература

[1] `http://www.gnu.org/philosophy/reevaluating-copyright.¬ html` — Reevaluating Copyright: The Public Must Prevail (Перевод: `http://www.gnu.org/philosophy/reevaluating-copyright.ru¬ .html` — Пересмотр системы авторских прав: общество должно преобладать)

9.55–10.20

Николай Шмырёв Москва, НИИСИ РАН

## Привлечение разработчиков в проект GNOME

Интересно оценить, сколько активных разработчиков из России участвует в OSS проектах. Начать можно с проекта KDE, довольно представительный список разработчиков которого можно найти на сайте блогов [4]. Для карты GNOME Россия — чёрные пятна [5]. Карта Debian тоже говорит о многом [6]. Из полутора тысяч учётных записей GNOME российских разработчиков гораздо меньше процента, если вообще наберётся человек 10. Точных данных, конечно, нет, но приблизительные оценки уже говорят о многом.

Развитие процесса разработки очень важно. Конечно, локализация — это тоже серьёзный процесс при адаптации свободного ПО в России, но одной локализацией дело не обойдётся. Без поддержки разработчиков установка больших объёмов ПО в российских условиях просто невозможна. Кроме того, участие в серьёзном OSS-проекте — бесценный опыт для начинающих.

Конечно, считать только активных разработчиков некорректно, гораздо интереснее было бы рассматривать варианты применения. В целом интересны не только локализаторы, но и тестировщики, писатели технической документации и т. д.

Какие же препятствия мешают нам набрать уровень, сравнимый с Европой:

- плохое знание английского;
- недостаточно качественный доступ в Internet, или его отсутствие;
- отсутствие культуры разработки;
- многие готовы помогать GNOME, но не знают как, или имеют недостаточные навыки;
- проект имеет тяжёлую структуру.

Конечно, больше не значит лучше, ещё Брукс [3] об этом наглядно писал. Но нужно учитывать, что сейчас модель разработки ПО

очень сильно изменилась, всё больше ПО ориентируется на постоянные обновления, постоянную отдачу пользователя. Люди вовлекаются в процесс разработки повсюду — от прикладных приложений до аппаратуры (есть проекты, например, мобильных телефонов) позволяющие сообщать об ошибках в них.

Другая особенность настоящего времени — тогда всё-таки речь шла о сравнительно маленьких по сегодняшним временам проектах, сейчас есть место, где разработчики просто нужны. Около десятка ключевых проектов GNOME не имеют главных разработчиков, если вообще разрабатываются. В целом происходит только исправление ошибок, развитие многих компонентов просто остановлено. Сейчас GNOME находится на перепутье — сил и способностей существующих разработчиков недостаточно (опять цитата из Брукса «программист всю жизнь делает одну и ту же программу»). Нужно как-то развиваться дальше, именно новые разработчики сделают GNOME 3.0.

В проекте GNOME можно принимать участие, и нам действительно нужны люди. Пример gimp-help показывает, даже непрофессионалы могут участвовать в разработке. В проекте GNOME понимают это хорошо, так же как и в компаниях, связанных с GNOME — Google, NOKIA. Исследуются проблемы развития ПО, вовлечения большего количества участников в проект.

Что нам нужно делать и чем мы занимаемся:

- проект GNOME Love [2] и русские Love Days;
- локализация;
- интерактивная помощь в канале IRC;
- новый русский сайт [1];
- новости в средствах массовой информации;
- личные встречи;
- сертификация.

Мы достигли значительных успехов, вот уже два выпуска GNOME выходят со статусом перевода 100 процентов, сейчас идёт активная работа над документацией. Участники начинают работать с кодом.

Хорошо было бы:

- Принимать более активное участие в международных конференциях, приглашать международных гостей. Мы надеемся, что когда-нибудь и в Москве пройдёт GUADEC и GNOME Foundation нас может в этом вопросе поддержать.
- Получить более активную поддержку GNOME со стороны российских дистрибутивов.
- Проводить активный маркетинг русского GNOME.
- Сделать акцент на архитектурные и программистские аспекты рабочей среды GNOME, важно улучшить среду разработки и документацию разработчика.
- Выпускать бумажную литературу.

Распространение GNOME заключается в информировании людей, сборе их мнений и в ответных действиях. Это звучит легко, нужно только взяться за дело.

### Литература

[1] Русский сайт о проекте GNOME. `http://gnome.org.ru`

[2] Проект GNOME Love. `live.gnome.org/GnomeLove`

[3] *Брукс Ф.* Мифический человеко-месяц или как создатся программные системы. `http://www.lib.ru/CTOTOR/BRUKS/¬ mithsoftware.txt`

[4] Блоги проекта KDE. `http://planet.kde.org`

[5] Карта разработчиков GNOME. `http://live.gnome.org/¬ GnomeWorldWide`

[6] Карта разработчиков Debian. `http://www.debian.org/devel/¬ developers.loc`

## 10.20–10.45

Александр Сигачёв Москва, МГТУ им. Баумана

# Википедия: достижения и проблемы

Аннотация

В докладе рассказывается о текущем состоянии Википедии, её развитии, а также о проблемах свободной энциклопедии и путях их решения.

> Лучший способ в чём-то разобраться до конца — это попробовать научить этому компьютер.
>
> Дональд Кнут

Свободная энциклопедия «Википедия» не нуждается в особом представлении, за последний год она не была обделена вниманием прессы, однако перед тем как переходить к проблемам развития проекта, мне кажется необходимым дать краткую справку о текущем его состоянии.

### Техническая сторона

- По данным Alexa Википедия входит в 20 наиболее популярных сайтов Интернета. Примерно каждый 30-ый пользователь Интернета за последнюю неделю посетил Википедию.
- 12 000 HTTP-запросов в секунду. Всего используется 240 серверов, из них 12 серверов БД.
- Размер базы данных — 15 Гб (медиаданные не включаются).
- ПО: MySQL + Apache + PHP + Lucene + Squid + Memcached. Вики-движок MediaWiki.
- Фонд Викимедиа. Состоит из совета попечителей, комитетов, региональных отделений. Не вмешивается непосредственно в работу над статьями.

- Сотрудники фонда: 2 на административной работе, 1 системный администратор, 2 программиста. Разработчики и системные администраторы: около 30.

- Бюджет за 4-ый квартал 2005 года — 350 000 долларов (60 % на закупку оборудования, 10 % на зарплату). Ежегодная международная конференция, исследования Википедии.

**Содержательная сторона**

- 11 языковых разделов Википедии включают более 100 000 статей, 150 — более 100 статей, всего есть разделы на 229 языках. Английская версия привлекает 60 % посетителей. На русском языке около 95 000 «статей». Каждый день появляется около 180 новых (в английской — 1 300 000 и 2100).

- Рост сообщества. В русскоязычном разделе Википедии за апрель 2006 года 704 участника сделали более 5 правок, 139 — более 100 правок, годом ранее эти цифры били соответственно 202 и 38.

- Выработаны правила, которые приняты общим голосованием. Например, Правила именования статей, Об оригинальных исследованиях, Критерии значимости персоналий. Правила в разных языковых разделах могут отличаться.

- Голосования. Решения принимаются консенсусом (обычно необходимо больше двух третей голосов за), свой голос необходимо подкреплять аргументами, все участники равны.

- Арбитражный комитет для решения конфликтных ситуаций. 5 человек избираются на 6 месяцев.

- Культурный обмен. Например. Во всех русских источниках указано, что капитуляция нацистской Германии была подписана 9 мая в Берлине, в западных — 8 мая в Реймсе. В Википедии рассказывается о двух этих актах.

- Тематические порталы. Представляют информацию по определённой теме, организуют участников на работу по ней.

- Другие проекты. Кроме Википедии есть ряд родственных проектов, например хранилище свободных медиаданных, исторических

документов, многоязычный словарь (существенно технически перерабатывается в настоящее время).

Перечислим проблемы, с которыми сталкивается Википедия и которые указывают её критики.

**Концептуальные проблемы**

- Вандализм — явно вредительское добавление, удаление или изменение содержания, совершённое умышленно. Спам. *Как ни странно, эта проблема не является острой.*

- Неполнота. Множество «заготовок» статей, автоматическая загрузка ботом (статьи о фильмах, галактиках и т. д.). *Проблема должна решиться со временем.*

- Невозможность ссылаться на Википедию. Статьи постоянно меняются, неофициальность проекта. *Википедия (как и любая энциклопедия) не есть первичный источник, её авторитетность заключается в количестве и подробности ссылок на первичные источники.*

- Сложность фактической проверки данных. Использование неавторитетных источников. *Решается привлечением к редактированию статьи большего количества участников, обсуждением, выражением нескольких точек зрения.*

- Склонность к перекосам в охвате тем. Например, про огромных человекообразных роботов (меха) написано больше чем про лауреата премии имени Нобеля по экономике Леонида Канторовича или конфликт в Дарфуре, унёсший сотни тысяч жизней. Средний участник Википедии — это мужчина в возрасте 20–40 лет, из Северной Америки или Западной Европы, с высшем образованием, высокими навыками работы с компьютером. Википедия отражает в первую очередь его интересы. *Межкультурный обмен, привлечение новых участников, специальные тематические проекты.*

- Статьи нельзя подписывать. *Википедия — это энциклопедия, нужно понимать особенности «жанра».*

- Нейтральная точка зрения. Возможно ли вообще достичь нейтральности в исторических статьях, политических? *Нейтральность не исключает приведения полного спектра мнений по данному вопросу.*

- Цензура. Например удаление примеров порнографических фотографий из статьи «Порнография». *Официально в Википедии нет цензуры, но есть требования писать статьи литературном языком в научном стиле, достижения консенсуса по содержанию статьи заинтересованными сторонами.*

- Разрастание статей. Количество не переходит в качество. *Сегментирование больших статей на более узкие темы, периодическое полное переписывание статей.*

- Лицензия GNU Free Documentation License. *Авторские права остаются за участниками (возможно множественное лицензирование). В Википедии нет неизменяемых разделов.*

- Анонимность и приватность. Некоторые считают, что анонимные правки недопустимы. Если вы раскрываете своё настоящее имя, то остальные могут наблюдать ваши интересы, когда вы тратили своё время на Википедию и т. д.

### Проблемы сообщества

- Войны правок. В спорах рождается истина, но иногда это отнимает очень много сил.

- Личные конфликты, флейм вместо обсуждения. Администраторы могут действовать только согласно правилам, который достаточно либеральны, заблокировать кого-то можно только за явные оскорбления, длительность блокировки обычно 1—7 дней.

- Авторитаризм «администрации». Сообщество очень неоднородно, администраторы тоже, иногда правила действительно нарушаются, в том числе администраторами, но бывают и случаи личной обиды, даже когда всё было сделано согласно правилам, обвинения в «закостенелости», «подавлении свободы слова», «синдроме совкового вахтёра» и другие разнообразные.

- Фанатики и специфические интересы. Например, последователи Бахаи описывают бахаизм как мировую религию в общих статьях, появляются темы о гомосексуальности в различных сферах жизни общества и истории.

- Политическая ангажированность. Обсуждения политических статей всегда очень сложно (Нагорный Карабах, Катынский расстрел), кроме того, возникают более общие удаления молдавского (кириллического) раздела Википедии, использования «неофициальной» орфографии в белорусском разделе, что считать языком, а что диалектом, содержание китайского раздела.

- Ответственность. За статьи в Википедии никто не несёт ответственности, веские научные аргументы, критические замечания в адрес какого-либо материала могут оставить без внимания, потому что никто не следит за этой статьёй, это неинтересно. *Нужно создавать группу, занимающуюся взаимодействием с теми, кто не хочет осваивать вики-технологию, участвовать во внутренних обсуждениях.*

## 10.45–11.10

Михаил Якшин Москва, ALT Linux, БЕН РАН

### Организация хранилища слабоструктурированных данных на основе свободных Wiki

В последнее время всё большую актуальность приобретают задачи создания всевозможных масштабных хранилищ данных, и одно из наиболее сложных и неоднозначных направлений в этом вопросе — хранение слабоформализованных данных с произвольной, меняющейся в зависимости от задач, структурой.

Когда идёт работа со строго формализованным данными, обычно используются стандартные решения на реляционных базах данных, но зачастую возникает задача по обработке некоего произвольного множества данных, как-то связанных между собой. Некоторые из этих данных имеют форму произвольных полнотекстовых документов, некоторые — документов с четко выраженной структурой (разделами, под-

разделами), а некоторые — наборы структурированных данных, с разбиением по полям вплоть до атомарных значений.

Сама по себе задача сейчас не имеет четко обоснованного и теоретически красивого решения. Все универсальные подходы обладают теми или иными достоинствами и недостатками. Исторически, один из самых первых и самых простых способов создания связанной системы документов — это массив гипертекстовых файлов (с появлением стандарта HTML/XHTML — это файлы в этом формате) с расставленными ссылками друг на друга. Создание хранилища данных на этой основе возможно, но у этого решения существует несколько серьезных проблем:

1. Высокий порог вхождения. Для обычного пользователя создание даже одного HTML-документа — серьезная задача, которая требует неких специальных знаний и, обычно, привлечение WYSIWYG-средств (дополнительного программного обеспечения) разной степени сложности. Создание же системы связанных между собой документов вручную, даже для компетентного специалиста, становится уже сложной задачей, для решения которой, как правило, привлекаются специальные средства, называемые Content Management System — но они, в свою очередь, урезают универсальность подхода до решения какой-то конкретной задачи — например, ведения веб-сайта с новостными лентами, или ведения просто архива статей по датам.

2. Смешение оформления и содержания. Изначально HTML на SGML (а позднее — XHTML на XML) — язык произвольной разметки произвольных данных, включал в себя как семантически нагруженные элементы — например, `strong` для «усиления» высказываний (на печати обычно выделяется полужирным шрифтом) или `em` (emphasis) для акцентирования текста (на печати выделяется курсивом или разрядкой), так и элементы, изначально нацеленные только на оформление, такие как `font` (шрифт), `b` (полужирный), `i` (курсив), `u` (подчеркивание) и т. п. Особенно при использовании средств WYSIWYG пользователю в первую очередь предоставляются именно такие тэги — и зачастую происходит интенсивная работа над оформлением текста вместо создания собственно логически размеченного содержимого.

Таким образом, для создания системы связанных документов слабой формализации одних только механизмов, предоставляемых XML и XHTML, недостаточно. Как правило, используются некие внешние средства автоматизации деятельности по созданию такой информационной коллекции, некие CMS, которые следят за всем массивом документов и хранят их в качестве какого-то единого репозитария.

В последнее время получила распространение технология Wiki — технология, разработанная в конце 90-х годов на первой волне популяризации Internet. Изначально wiki-системы представляли собой просто сайты, страницы которых мог редактировать кто угодно, прямо из web. В современном варианте — это система для сбора и структурирования информации, характеризующаяся следующими признаками:

1. Многопользовательский режим работы — все редактирование осуществляется через web-интерфейс, есть центральный сервер (или кластер), хранящий весь массив данных.

2. Возможность многократно править текст посредством самой wiki-среды (вебсайта), без применения особых приспособлений на стороне пользователя.

3. Проявление изменений сразу после их внесения.

4. Разделение информации на однозначно идентифицируемые документы.

5. Несложный человеко-читаемый язык разметки, позволяющий легко отделить содержимое от оформления.

6. Учёт изменений (учёт версий) текста и возможность отката к ранней версии.

Преимущества Wiki:

1. Именование и идентификация. Wiki представляет собой коллекцию произвольных документов, единственный способ доступа к которым — идентификатор. Идентификатор — это обычная текстовая строка — «название документа» в его полном виде. В случае, когда у нас в коллекции есть два документа (две сущности) с одинаковыми названиями, есть механизм disambiguation, который заключается в том, что сами документы прозрачно именуются

с уточняющим суффиксом (например, «Иванов, Иван Иванович (физика)» и «Иванов, Иван Иванович (математика)»). В самом документе ссылка на другой документ создается либо полностью автоматически, либо полуавтоматически (обрамлением простым тэгом).

2. Шаблоны — предоставляют возможность хранения и представления структурированных данных, подробнее будут рассмотрены ниже.

3. Автоматическое создание всевозможных списков и классификаций. Один из примеров реализации таких механизмов — это механизм категорий или тэгов, применяющийся часто в разных wiki. Он заключается в том, что документ (либо специальным тэгом внутри текста документа, либо внешними атрибутами) помечается, как принадлежащий какой-то категории или как помеченный неким ключевым словом — тэгом. После этого при обращении к (пока чисто виртуальному) документу «Категория:Заданное имя» (в пространстве имен «Категория») будет выведен список документов, помеченных заданным именем. При всем этом «виртуальный» документ остается все равно документом, который можно заменить реальным, при этом список будет появляться так же автоматически. В сложных wiki engines, место появления списка и его внешний вид можно настраивать.

4. Ссылочная целостность. Как правило, wiki предоставляет возможность отследить как все ссылки с текущей страницы куда-либо, так и все ссылка откуда-то на текущую страницу. При этом реальный документ может не существовать, но список ссылок все равно будет проиндексирован в базе.

Во многих wiki существует механизм создания шаблонов. Шаблон — это отдельный документ, физически по способу хранения ничем не отличающийся от статей, но находящийся в отдельном пространстве имен и доступный для вставки с помощью специальных тэгов. Как правило, вызов шаблона из основного документа имеет вид вызова некоей процедуры, которой передается набор параметров — его и можно рассматривать как набор сильноструктурированных данных внутри произвольного документа.

Шаблон играет двоякую роль: с одной стороны — его вызов хранится в теле документа и таким образом формирует некую структуру машинно-обрабатываемых данных, которые в дальнейшем могут индексироваться, их можно анализировать, преобразовать при необходимости в отдельную реляционную форму, строить по ним статистики и т. п.

Существует и другой, альтернативный подход к созданию такого хранилища слабоструктурированных данных, который предполагает использование двух раздельных XML, привязанных к документу для хранения слабоструктурированной информации. Первый такой фрагмент XML — это собственно сами данные, а второй — стандартное XSLT-преобразование, которое будет применяться к этому фрагменту для показа его конечному пользователю. Этот способ — более гибкий, но он сложнее, так как требует создания трех отдельных документов.

Автором была создана серия шаблонов, который в качестве аргументов получает некое подмножество наиболее распространённых полей библиографического описания (например, монографического издания — `Шаблон:Книга`), а на выходе — для пользователя — выдает простейшее библиографическое описание по ГОСТ 7.1–2003.

Эти шаблоны доступны и применяются в русской части проекта Wikipedia, составляя вместе с англоязычными шаблонами вида `Template:Cite_book`, полный инструментарий для представления библиографических описаний во всевозможных стилях.

При этом же, для возможного последующего машинного анализа, по сути, сохраняется всё разбиение данных по полям: не теряется информация.

Шаблон имеет следующие аргументы:

```
{{Книга |автор = |заглавие = |параллельное_заглавие =
|ответственность
    = |место = |издательство = |год = |isbn = |ссылка =
    |дата_посещения_ссылки =}}
```

Стоит заметить, что почти все аргументы могут быть как просто текстовыми элементами, так и ссылками.

В качестве недостатков предложенного метода можно отметить чрезмерную простоту создаваемых отношений: создаваемые отношения хотя и классифицируются как «много-ко-многим», но не имеют возможности задания каких-либо атрибутов у самой связи. Частично

это реализуется с помощью создания связей из разных полей шаблона, что позволяет задать тип отношения между сущностями (например, «автор», «издательство» и т. п.), но не позволяет задать, например, временных рамок существования отношения или каких-либо более сложных атрибутов. Однако, стоит заметить, что эта простота точно так же является и огромным преимуществом для конечного пользователя — когда схема данных будет описана программистом таким образом в виде шаблонов, задача введения данных, поддержки их в актуальном состоянии и выборок значительно упрощается.

Таким образом, предложенная схема хранения данных может использоваться в системах электронных библиотек, где требуется организовать работу со слабоструктурированными данными. Разработанные шаблоны используются в русской части Wikipedia для представления библиографических данных в формализованном виде, аналогично шаблонам, которые делают это в других стилях цитирования (Гарвардском, Оксфордском и т. п.)

## 11.10–11.35

Кирилл Маслинский Санкт-Петербург, ALT Linux, СПбГУ

### Сравнительный анализ форматов wiki-разметки

Аннотация

Вопрос о сходстве и различиях различных форматов wiki-разметки особеннно актуализировался в последнее время в связи с попытками наладить способы передачи размеченных текстов из одних wiki-систем в другие. Этим докладом я хотел бы предложить свою каплю в это обсуждение: провести сравнительный анализ разных диалектов wiki-разметки, применив некоторые методы анализа, сложившиеся в рамках лингвистической типологии.

## 11.35–11.55: Кофе-брейк * * * *

## 11.55–12.20

Алексей Федосеев Москва, IBM Linux Technology Center

Проект: Samba
http://wiki.samba.org/index.php/Clustered_Samba

# Кластерная реализация Samba

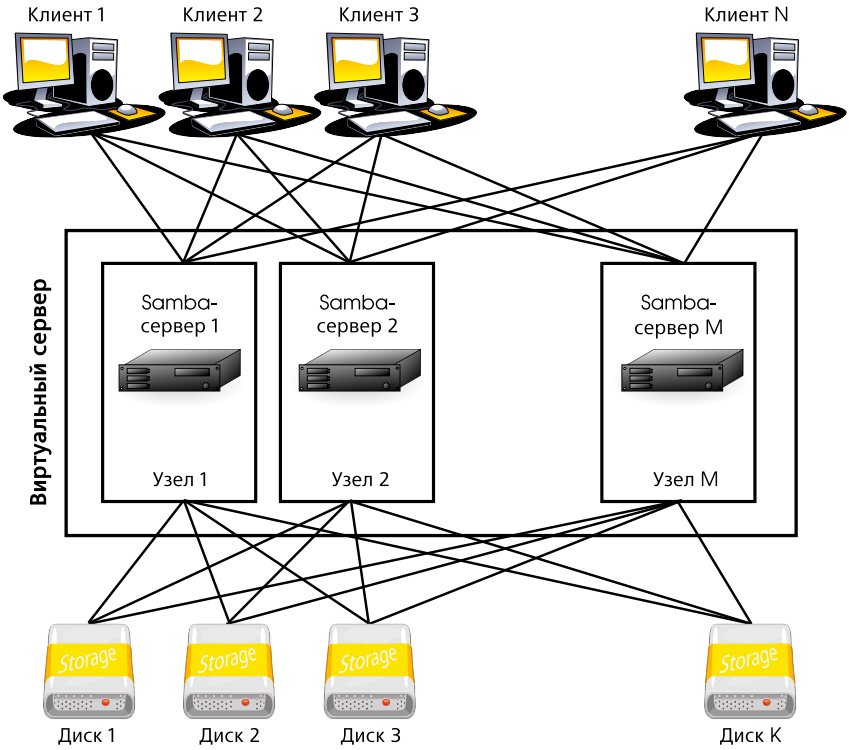
Аннотация

В докладе рассказывается о создании кластерной реализации файлового сервера Samba, о проблемах построения файловых серверов для кластеров и использовании распределённых файловых систем.

В современной компьютерной индустрии резко возросла потребность в высокопроизводительных системах и системах хранения большого объёма. В качестве примера можно взять область Digital Media, к которой относятся популярные сейчас компьютерная анимация и расчёт спецэффектов в киноиндустрии. Основным инструментом при решении этих задач являются кластерные системы с огромными распределёнными хранилищами данных. В докладе рассказывается о проблемах построения файлового сервера для кластерных систем на примере Samba и о текущей разработке кластерной версии Samba.

Кластерный файловый сервер в идеальном варианте должен обладать следующими свойствами:

- любой из клиентов может присоединиться к любому из серверов (узлов кластера);
- в случае отключения сервера клиент прозрачно переподключается к другому серверу;
- все серверы работают с общим набором файлов;
- все изменения в файлах становятся немедленно доступны всем серверам (распределённые файловые системы);
- легкая масштабируемость простым добавлением узлов или дисков;
- представляет собой единую большую систему.




В случае Samba, на каждом узле кластера располагается собственный CIFS-сервер. Координация состояний серверов становится освновной проблемой при построении такой распределённой системы. Вся система должна строиться на базе распределённой файловой системы, чтобы файловое хранилище было единым для всех узлов системы.

При разработке кластерного файлового сервера, не привязываясь к специфике используемого протокола доступа к файлам, можно выделить ряд проблем. Все они так или иначе связаны с сохранением состояния файла в распределенной среде при множественном доступе. Состояние файла можно разделить на:

- уникальный идентификатор файла — пара (dev, inode);
- режим одновременного доступа к файлу;

- блокировки (на уровне протокола — например, CIFS — и на уровне операционной системы).

Если доступ к файлам осуществляется на только через протокол CIFS (например, файлы могут изменяться локальными процессами), необходимо учитывать блокировки на уровне операционной системы (в частности, opportunistic locks в Samba должны учитывать текущее состояние файла). Кроме того, большинство файловых систем реализует семантику POSIX, которая может отличаться от семантики протокола доступа к файлам (так и есть в случае CIFS).

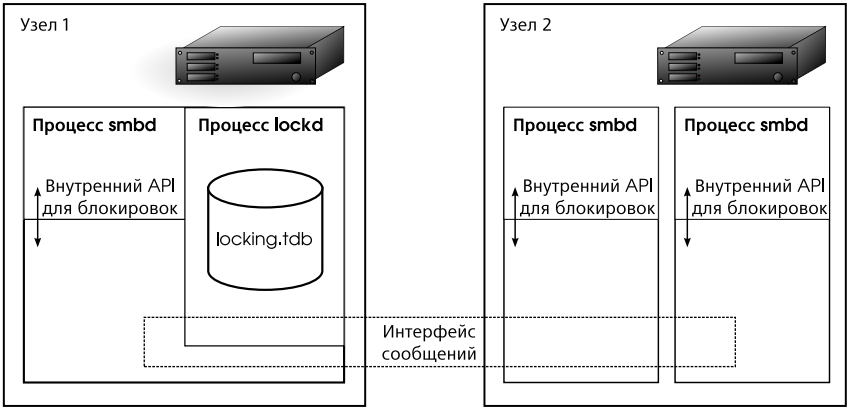
Эти проблемы усиливаются в случае распределённой файловой системы. В настоящее время существует ряд свободных (AFS, OpenGFS) и коммерческих (CXFS, GPFS) распределённых файловых систем. Предоставляя доступ к файлам в пределах всего кластера, они добавляют следующие сложности:

- невозможно установить «истинное» положение файла, который часто бывает распределён по всему кластеру;
- идентификатор файла (dev, inode) может изменяться в ходе работы системы;
- для координации состояний узлов распределённой ФС используются сообщения, а это может занимать значительно большее время, чем в локальных ФС.

Таким образом, на первый план выходит проблема реализации взаимных блокировок для файлов и координации отдельных Samba-серверов для избежания коллизий при обращении к одним и тем же файлам. Текущая версия сервера Samba использует локальную базу данных для хранения блокировок — этот механизм должен быть расширен до уровня кластера. Такая координация невозможна без организации механизма быстрых коммуникаций между Samba-серверами, расположенными на различных узлах кластера.

Было предложено следующее решение по созданию кластерной версии Samba-сервера:

- 1 шаг — расширение существующего механизма обмена сообщениями между процессами Samba, чтобы позволить взаимодействие в рамках всего кластера (для начала на основе TCP, затем MPI или любое другое средство коммуникаций);




- 2 шаг — реализация блокировок файлов через разработанную систему глобальных сообщений, для хранения блокировок используется специальный «демон блокировок» — общий для всего кластера;

- 3 шаг — тестирование на реальных задачах и оптимизация.

Сейчас первые два этапа работы завершены. Код кластерной версии Samba доступен любому в исследовательской ветке vl-messaging Subversion-репозитария проекта Samba.

## 12.20–12.45

Максим Лапань Рыбинск, ОАО НПО «Сатурн»

Проект: NoMachine NX `http://www.nomachine.com/developers.php`

### Сжатие протокола X11 с помощью NoMachine NX

При проведении инженерных и научных расчетов с использованием высокпроизводительных систем не последнее место занимает этап визуализации результатов. Чаще всего, объем данных делает нецелесообразным или невозможным передачу этих данных на клиентскую

машину для обработки. В этих случаях обычно используют терминальный доступ, при котором приложение работает на удаленном сервере, а визуализация выполняется на пользовательской машине.

Однако графические приложения, предназначенные для визуализации больших объемов данных, обычно используют библиотеки OpenGL, что накладывает определенные ограничения на схему работы пользователей, что в первую очередь касается скорости сетевого соединения.

В докладе рассматривается схема организации удаленного доступа к кластерному вычислительному комплексу НПО «Сатурн» (город Рыбинск) для инженерного центра в городе Перми. Приводятся результаты сравнительного тестирования производительности графических приложений при использовании несжатого протокола X11 и сжатии с помощью NX, свободной разработки фирмы NoMachine.

## 12.45–13.20

Алексей Гладков, Константин Лепихов Москва, ALTLinux, ИТМиВТ им. Лебедева

Проект: XULRunner http://developer.mozilla.org/en/docs/XULRunner

### XulRunner: платформа разработки

#### Почему XUL?

XUL или XML User Interface Language, это, как видно из его названия, язык, который позволяет создавать достаточно богатые по функциональности пользовательские интерфейсы, которые можно как запускать («отрисовывать») как стандартные приложения, так и загружать их из сети Internet. При этом приложения на XUL можно легко настраивать, менять в них текст или графические объекты, переводить на различные языки и т. д. Для программирования на XUL не требуется специальных навыков, любой web-разработчик, знакомый с Dynamic HTML (DHTML), сможет быстро выучить XUL и начать создавать свои приложения. Отличительные особенности XUL:

– Мощный язык разметки с поддержкой пользовательских элементов (widgets) (всех, которые поддерживаются платформой

mozilla). В отличие от DHTML, с помощью которого можно создавать web-страницы, с помощью XUL можно создавать переносимые приложения, содержащие окна, кнопки и ссылки.

- Основан на существующих стандартах. XUL — это язык XML, основанный на стандарте W3C XML 1.0. Приложения, написанные на XUL, используют дополнительные стандарты W3C, такие как HTML 4.0; Cascading Style Sheets (CSS) 1 and 2; Document Object Model (DOM) Levels 1 2; JavaScript 1.5, включающий ECMA-262 Edition 3 (ECMAscript); XML 1.0.
- Межплатформенная переносимость. XUL может быть использован на любой платформе, поддерживаемой XULRunner.
- Отделенная логика для отображения и формирования интерфейса.
- Легкость локализации, модификации или настройки.

Таким образом, XUL является легким в освоении, гибким и переносимым языком.

### XULRunner

С точки зрения браузера или сервера, «понимающего» XUL, отдельное приложение на этом языке ничем не отличается от разметки HTML/JS — это тот же код, который сначала надо скачать с сервера (либо передать его отдельному серверу на обработку), потом преобразовать (отрендерить) и показать пользователю — т. е. можно запускать пользовательские приложения, расположенные в сети, а не на локальном диске клиента, и XULRunner — это интерпретатор XUL-приложений. Эти приложения распространяются в виде обычного `.jar`-архива с xul-контентом и сопутствующими библиотеками (например, для поддержки ssl, ldap или sqlite).

Существует множество расширений (extensions) для стандартных продуктов mozilla.org. Они добавляют необходимую для пользователей функциональность. Крайняя форма таких расширений — это глобальные большие приложения, которые можно выделить в самостоятельные продукты. Но есть одна проблема — пользователь вынужден ставить себе браузер (или другой продукт) для запуска таких приложений, что затрудняет распространение ПО на базе XUL из-за большого

размера сопутствующего ПО и разработку из-за отсутствия отдельного приложения-интерпретатора, на котором можно производить процесс отладки. XULRunner и XRE (XUL Runtime Evironment) — попытка избавиться от этой зависимости, позволяет запускать XUL приложения обычным образом: через программу-загрузчик (XULRunner), или посредством встраиваемых библиотек для рендеринга (XRE или libxul). Кроме того, применение XULRunner позволяет устанавливать, обновлять или удалять приложения таким же способом, как и обычные программы mozilla.org (Firefox или Thunderbird).

### libxul

`libxul` является основным элементом в концепции развития Mozilla 2.0. Это будет библиотека, предоставляющая стабильный (или, как называют его разработчики, «замороженный») API для XUL-приложений и встроенных приложений на базе Gecko.

На сегодняшний день обычная сборка mozilla-based приложения предоставляет много различных разделяемых библиотек, которые являются как частью отдельных компонентов, так и отдельными библиотеками с интерфейсами для различных сторонних библиотек или протоколов (например ldap), но все они являются частью Gecko (т. е. движка для рендеринга). `libxul` призвана заменить это многообразие на одну статическую библиотеку, которая будет предоставлять Gecko, и куда будут входить все его составные компоненты (поддержка сетевых служб, DOM, рендеринг и т. д.). Все это позволит использовать прогрессивные технологии mozilla не только в продуктах mozilla.org, но и в любой сторонней разработке, делая ее переносимой между различными платформами, имеющей удобный и легкий GUI и поддержку сетевых служб.

13.20–13.45

Алексей Турбин Рязань, ALT Linux Team
Проект: Sisyphus http://sisyphus.ru

## Анализ бинарной совместимости репозитория rpm-пакетов

Разработан метод анализа бинарной совместимости репозитория rpm-пакетов, основанный на извлечении информации о динамических символах из исполняемых файлов и разделяемых библиотек формата ELF. Создана реляционная модель данных, в которой каждый динамический символ, ассоциированный с ELF-файлом и rpm-пакетом, является либо предоставляемым, либо требуемым. Предметом анализа является ссылочная целостность (разрешимость) символов, которая формализуется с помощью оператора соединения (`JOIN`). Символы, которые не удается таким образом разрешить, образуют некоторый класс потенциальных ошибок (бинарной несовместимости).

Семантически символы соответствуют функциям и глобальным переменным, используемым в приложении. Как правило, требуемые символы исполняемого файла должны быть сопоставлены с предоставляемым символам разделяемых библиотек, с которыми этот файл связан (разделяемые библиотеки в свою очередь могут быть связаны с другими разделяемыми библиотеками); невозможность сопоставить (разрешить) какой-либо требуемый символ приводит к аварийному останову приложения. Известно, что на стадии компиляции (компоновки) исполняемого файла предусмотрены некоторые проверки разрешимости символов. Однако в ежедневно обновляемом репозитории rpm-пакетов этих проверок оказывается недостаточно, поскольку среда выполнения приложения может отличаться от среды, в которой собран соответствующий rpm-пакет.

В первоначально разработанной упрощенной модели производится проверка разрешимости на полном множестве символов, без учета связей с разделяемыми библиотеками. В рамках этой модели можно обнаружить лишь некоторое подмножество потенциальных ошибок в разрешимости символов, поскольку считается, что любой требуемый символ может разрешиться в любой соответствующий ему предоставляемый символ. Тем не менее, при выполнении некоторых ограничений

(в частности, при ограничениях на дублирование бинарного кода в репозитории) в рамках модели удается получить практически значимые результаты.

Дублирование бинарного кода само по себе является достаточно актуальной проблемой. Из соображений переносимости некоторые приложения включают в себя внутренние копии библиотек, доступных в репозитории в виде отдельных пакетов. Переносимость в таком случае противопоставляется модульной структуре и целостности репозитория: исправления в системной библиотеке оказываются недоступными приложениям.

В рамках рассмотренной модели разработан метод обнаружения дублирования бинарного кода, основанный на подсчете совпадающих символов, предоставляемых ELF-файлами. Большое число совпадающих символов говорит о потенциальном дублировании реализации какого-либо бинарного интерфейса. Этот метод также не является достаточным, поскольку формат ELF допускает сокрытие символов. Однако с помощью данного метода удалось, например, обнаружить в репозитории несколько внутренних копий библиотеки `zlib`.

Дальнейшее развитие модели предполагает извлечение дополнительной информации из ELF-файлов, в частности, связи с разделяемыми библиотеками (`DT_NEEDED`) и названия библиотек (`DT_SONAME`). В рамках новой модели удается обнаружить надмножество неразрешимых символов, поскольку разрешимость в этой модели не является транзитивной, тогда как динамический редактор связей допускает транзитивное сопоставление символов. Кроме того, в рамках этой модели можно ответить на следующие вопросы: 1) для данного ELF-файла, существуют ли символы, которые разрешаются в несколько разделяемых библиотек, с которыми этот файл связан? множество таких символов образует ещё один класс потенциальных ошибок; 2) среди библиотек, с которыми связан ELF файл, существуют ли такие, которые он не использует? последнее обнаруживает, в частности, излишние и чрезмерно жесткие зависимости между пакетами. Показано, что в среднем почти половина всех библиотек, с которыми связаны ELF файлы, образуют излишние связи. Впоследствии эта проблема была в основном решена с помощью включения по умолчанию специального режима компоновки `ld --as-needed`.

Наконец, разработан метод анализа бинарной совместимости между двумя «срезами» репозитория, сделанными в разное время. Предметом анализа является возможность частичного обновления системы,

построенной на основе «старого» репозитория, пакетами из «нового» репозитория, без нарушения совместимости между бинарными интерфейсами. В этой задаче символы, предоставляемые разделяемыми библиотеками, образуют отдельное отношение; производится сравнение разрешимости символов на множестве библиотек «старого» и «нового» репозитория. Если разрешимость невозможна в одном из случаев, но не в другом, это говорит о недопустимом изменении бинарного интерфейса соответствующей разделяемой библиотеки.

Рассмотрена реализация предложенных моделей, в которой отношения представлены в виде текстовых файлов, а операции реляционной алгебры реализованы с помощью стандартных утилит UNIX. Утверждается, что ни один доступный SQL-сервер не способен обеспечить приемлемую производительность для решения рассмотренных задач.

### Литература

[1] Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2. `http://www.x86.org/ftp/manuals/¬ tools/elf.pdf`

[2] Ulrich Drepper. *How To Write Shared Libraries.* `http://people.redhat.com/drepper/dsohowto.pdf`

[3] E. F. Codd. *A Relational Model of Data for Large Shared Data Banks.* Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377–387. `http://www.acm.org/classics/nov95/`

[4] Evan Schaffer and Mike Wolf. *The UNIX Shell as a Fourth Generation Language.* `http://www.rdb.com/lib/4gl.pdf`

13.45–14.40: Обед * * * * *

# Дневное заседание
## 14.40–16.40

### 14.40–15.05

Фёдор Зуев Иркутск, Институт Земной Коры СО РАН
Проект: libfracdim http://libracdim.narod.ru

### Фрактальная математика для обработки геофизических данных

**Задача.** У широкой компьютерной публики фракталы прочно ассоциируются с красивыми цветными картинками, компьютерной генерацией ландшафтов и тому подобными современными аналогами калейдоскопа. Построение математических фракталов привлекательно тем, что применение сравнительно простого правила позволяет получить нам структурно сложную, визуально богатую картину.

Однако в естественных науках куда более актуальна обратная задача: по имеющейся выборке данных — экспериментальных или наблюдаемых точек, фотографии, схеме — оценить характеристики той предположительно фрактальной (или говоря по другому, самоподобной) природной структуры, которую эта выборка представляет: размерность Хаусдорфа D, показатель Херста H, мультифрактальные меры.

**Предмет.** Особенно богата самоподобными объектами геофизика, с которой собственно и пошло развитие теории фракталов. Поле сейсмических толчков (см рис. 1), сетка тектонических разломов, речная сеть, даже просто рельеф являются структурами очевидно самоподобными, фрактальными. Поэтому неудивительно, что попытки применения к ним математического аппарата теории фракталов делались с самого момента возникновения этой теории.

Однако с весьма скромными успехами. Важнейший результат, имеющий отношение к самоподобию сейсмичности — закон повторяемости землетрясений — был получен задолго до возникновения теории фракталов. В значительной степени это объясняется крайне невысокой точностью получаемых численных оценок. В большинстве случаев мы не

можем быть вполне уверены даже в первой цифре после запятой. Отчасти это вызвано объективным недостатком данных, к количеству которых очень чувствительны все фрактальные методы. Так для вычисления корреляционной размерности $C$ с точностью в 5% (т.е. до первого знака после запятой) требуется не менее $40^m$ экспериментальных точек, где $m$ — размерность вмещающего пространства. При этом во многих случаях мы не можем даже оценить величину этой ошибки. Однако существуют и недоиспользуемые резервы сугубо вычислительного характера.

**Жанр.** Доклад выполнен в жанре руководства по методу вычислений, популярном в 70–80-е годы, но почти забытом в наши дни. То есть стоит на полпути между математикой и технологией.

Оценка фрактальной размерности по сути своей является таким же численным методом, как численное дифференцирование, нелинейная оптимизация или преобразование Фурье. Однако если последние методы подробно и систематически описаны в литературе, то информация о техниках вычисления фрактальных характеристик разбросана по множеству неочевидных, иной раз довольно-таки странных мест.

Поэтому весьма важным представляется собрать и пересказать по-русски имеющуюся информацию на эту тему.

**Black Art.** Некоторые из способов, позволяющих добиться повышения точности вычислений — от солидных математических теорем, до сугубо технических трюков, которые, однако, не следует забывать.

- Использование, по возможности, фрактальной размерности более высокого ранга — информационной или корелляционной, вместо клеточной.
- Минимизация (хотя бы очень грубая), числа ячеек (суммы, в случае информационной|корелляционной размерности) для каждого характеристического размера.
- Выбор начального и конечного масштаба вычислений (см рис. 2)
- Выбор вмещающей размерности и границ вмещающей области.
- Отсечение ненадежных точек.

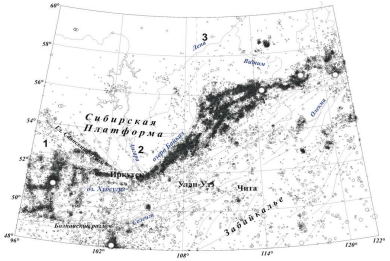



Рис. 1: Эпицентры землетрясений Байкальской рифтовой зоны

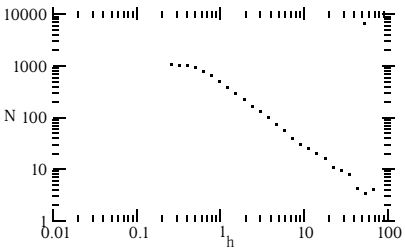



Рис. 2: Число покрывающих ячеек выборки, равномерно распределенной на триадной кривой Коха

- Внесение искусственной ошибки в данные, полученные с низкой разрядностью.

**Хозяйственное.** Предварительная версия библиотеки libfracdim, реализующая обсуждаемые алгоритмы и представляющая собой дополнительный пакет к библиотеке GNU Scientific library, выложена (на момент написания — будет выложена) по адресу libfracdim.narod.ru.

Исследование по изучению самоподобия сейсмичности БРЗ поддержано грантом РФФИ 06–05–64120.

15.05–15.30

Никита Гущин Москва, ВМК МГУ
Проект: Nigma.ru http://www.nigma.ru

## Разработка прототипа поисковой системы 3-го поколения на базе Nigma.ru

Аннотация

Рассматривается возможность создания поисковой системы, выдающей в качестве результатов информацию (а не набор ссылок, как это делают существующие сегодня поисковые системы). Для решения поставленной задачи изучаются и создаются алгоритмы кластеризации и ранжирования результатов поиска, позволяющих формировать блоки, содержащие информацию о запросе.

15.30–16.05

Пётр Савельев Санкт-Петербург, Моторола ЗАО
Проект: RAD GNU/Linux http://www.radlinux.org/

## RAD/GNU Linux: решения на базе дистрибутива

### Введение

RAD GNU/Linux задумывался как комплекс надстроек над стандартными утилитами работы с сетью, позволяющий «из коробки» развёртывать решения для шлюзов, систем учёта и фильтрации трафика. Однако архитектура системы позволяет применять его гораздо шире. Предусмотрено максимальное упрощение процесса загрузки и хранения файлов конфигурации, виртуальные контексты обеспечивают модульную организацию и безопасность, а система сборки достаточно проста, чтобы создавать свои варианты решения.

### Загрузка и корневая ФС

Самый простой процесс инициализации предусматривает загрузку только ядра системы. Минимум необходимой функциональности, теоретически, можно перенести в kernelspace в виде дополнительных драйверов. В таком случае потребуется загрузка только одного файла – монолитного ядра. Однако очевидно, что для универсального решения наличия только kernelspace недостаточно, к тому же, монолитное ядро создаёт некоторые проблемы, связанные с поддержкой разнообразного оборудования.

Варианты загрузки же в несколько этапов (например, stage 1 – initrd, stage 2 – rootfs) были отвергнуты как неоправданно громоздкие. В итоге, RAD GNU/Linux работает в initrd, не выходя из него. Это позволяет свести инициализацию системы к загрузке только двух файлов: ядра и образа initrd. Такой подход делает ненужными дополнительные стадии поиска и загрузки либо монтирования rootfs. На данный момент все основные загрузчики GNU/Linux имеют поддержку initrd, и это позволяет одинаково легко загружать RAD GNU/Linux как с дисковых носителей, так и по сети, не привлекая лишнего кода.

Работа в initrd означает также и то, что изменения, например, конфигурации системы не сохранятся в процессе перезагрузки, если не принять дополнительных мер. И хотя инсталляция как таковая не требуется для работы RAD GNU/Linux, дистрибутив имеет поддержку локальных дисковых накопителей для сохранения архивов конфигурации и для размещения на rootfs виртуальных контекстов.

### Виртуальные контексты

VServer был выбран как основа системы управления процессами. Такое решение продиктовано несколькими соображениями. Во-первых, изоляция процессов в виртуальных контекстах значительно повышет безопасность системы. Во-вторых, VServer позволяет вести строгий контроль ресурсов, доступных каждому контексту. С равным успехом контролируются память, процессорный ресурс и сетевые адреса. При должном подходе это позволяет избежать DoS-атак на сервисы.

Ещё одно применение VServer состоит в запуске в рамках виртуальных контекстов отдельных систем GNU/Linux со своими rootfs. Это может использоваться с равным успехом как для модульного расширения функциональности решения, так и для хостинга. В первом случае,

виртуальный контекст служит для запуска сервиса, отсутствующего в стандартном наборе RAD GNU/Linux, например, Squid. Во втором, контекст выделяется для установки клиентского ПО. Изоляция, предоставляемая VServer, позволяет не беспокоиться по поводу безопасности всех прочих контекстов.

### Решения на базе RAD GNU/Linux

Система сборки решения использует apt(8) и rpm(8) для инсталляции необходимых пакетов в будущую rootfs (она же initrd). Использование репозитариев позволяет устанавливать уже собранные пакеты, оставляя разработчику свободу в формировании решения. В случае, если стандартный набор RAD GNU/Linux по тем или иным причинам не устраивает, достаточно изменить список пакетов и стартовые скрипты системы. В качестве вариантов решений на базе дистрибутива можно привести X-терминал, не требующий nfs-root, а также описанную выше систему хостинга. Наконец, простота загрузки системы и небольшой размер образа позволяет применить RAD GNU/Linux, например, для запуска инсталляции других дистрибутивов.

## 16.05–16.40

Кирилл Колышкин, Кирилл Коротаев СВсофт-МФТИ, Москва

## Виртуализация в Linux

Аннотация

В докладе даётся обзор основных типов виртуализации (эмуляция, паравиртуализация, виртуализация на уровне ОС), даются примеры реализации всех трёх технологий, их сравнение и области применения. Подробно рассматривается виртуализация на уровне ОС на примере OpenVZ. Описываются основные компоненты ядра (изоляция и виртуализация, управление ресурсами, чекпойнтинг и миграция) и пользовательских утилит. Кратко демонстрируются основные возможности OpenVZ (на реальной системе).

### Виртуализация. Типы виртуализации

В контексте данного доклада *виртуализация* — это система или методология разделения ресурсов компьютера на множество независимых сред. Возможно выделить четыре типа виртуализации: эмуляция, паравиртуализация, виртуализация на уровне операционной системы и многосерверная (кластерная) виртуализация (последняя в данном докладе не рассматривается).

Каждый из этих типов имеет свои достоинства и недостатки, обуславливающие сферы применения решений, на нём основанных.

*Эмуляция* позволяет запускать произвольную операционную систему без необходимости её изменения, но обладает сравнительно низкой производительностью, масштабируемостью и плотностью размещения. Примеры реализации: продукты VMware, QEmu, Bochs, Parallels.

*Паравиртуализация* даёт лучшую производительность, чем эмуляция, но требует изменения «гостевой» операционной системы. Примеры реализации: Xen, UML.

*Виртуализация на уровне операционной системы* обладает наилучшей возможной производительностью и плотностью, а также наиболее полно реализует динамическое управление ресурсами. В то же время эта технология не позволяет одновременно запускать несколько ядер разных ОС на одном сервере. Примеры реализации: FreeBSD Jail, Solaris Zones/Containers, Linux-VServer, OpenVZ.

### Понятие VE

*Виртуальная среда* (*VE*, также используются термины *VPS*, *VDS*, *контейнер* (container), *раздел* (partition) и т. п.) — это отдельная изолированная среда выполнения программ, которая (с точки зрения её владельца и запущенных в VE программ) выглядит как обычный выделенный сервер.

VE имеет свои процессы (начиная с `init`), файловую систему, пользователей (включая `root`), сетевое устройство с IP-адресами, таблицы маршрутизации, правила сетевого фильтра (`netfilter/iptables`) и так далее.

Множество VE сосуществует в рамках одного физического сервера. В разных VE могут работать разные дистрибутивы Linux, однако все VE работают под одним и тем же ядром.

### Ядро OpenVZ

Ядро OpenVZ — это модифицированное ядро Linux, которое добавляет следующую функциональность: виртуализация и изоляция отдельных подсистем ядра, управление ресурсами, а также чекпойнтинг.

*Виртуализация и изоляция* позволяют в рамках единого ядра ОС создавать множество виртуальных сред, каждая из которых имеет свои собственные процессы, файлы, сетевые и другие устройства и т. п. *Подсистема управления ресурсами* позволяет ограничивать и делить между VE такие ресурсы, как процессорное время, память и дисковое пространство. *Чекпойнтинг* — это процесс «замораживания» VE с сохранением полного образа её состояния в файл и возможностью последующего полного восстановления рабочего состояния VE.

*Система управления ресурсами* в OpenVZ состоит из трёх компонентов:

1. *Двухуровневая дисковая квота*. Администратор сервера OpenVZ может установить дисковые квоты на VE в терминах дискового пространства и количества айнодов (i-nodes). Это первый уровень дисковой квоты.

   В дополнение к этому администратор VE (root) может использовать обычные утилиты внутри своей VE для настроек стандартных дисковых квот UNIX для пользователей и групп.

2. *«Честный» планировщик процессов*. Планировщик процессора в OpenVZ также двухуровневый. На первом уровне планировщик решает, какой VE дать квант процессорного времени, базируясь на значении параметра `cpuunits` для VE. На втором уровне стандартный планировщик Linux решает, какому процессу в выбранном VE дать квант процессорного времени, базируясь на стандартных приоритетах процесса в Linux и т. п.

3. *User Beancounters*. Это набор счётчиков, ограничений и гарантий для различных ресурсов на каждое VE. Контролируется примерно 20 параметров, которые тщательно выбраны для того, чтобы никакая VE не могла злоупотребить каким-либо ресурсом, который ограничен для всего сервера, и, таким образом, помешать другим VE.

Ресурсы, которые считаются и контролируются — это в основном оперативная память и различные объекты в ядре, например, разделяемые сегменты памяти IPC, сетевые буферы и т. п.

*Чекпойнтинг* позволяет производить «живую» миграцию VE на другой физический сервер. VE «замораживается» и её полное состояние сохраняется в файл на диске. Далее этот файл можно перенести на другую машину и там «разморозить» (восстановить рабочее состояние) VE. Задержка этого процесса (время, когда VE в «замороженном» состоянии) составляет примерно несколько секунд. Важно подчеркнуть, что это задержка в обслуживании, но не отказ в обслуживании, так как разрыва сетевых соединений не происходит.

### Утилиты OpenVZ

В состав OpenVZ входит набор утилит `vzctl`, предоставляющий довольно высокоуровневый интерфейс командной строки для создания и управления виртуальными средами. Так, например, для создания и запуска новой VE необходимо всего две команды — `vzctl create` и `vzctl start`.

Для изменения различных параметров VE служит команда `vzctl set`. Все ресурсы (например, размер виртуальной памяти) могут быть изменены во время работы VE. В других технологиях виртуализации (напр. эмуляции или паравиртуализации) это или достаточно сложно, либо невозможно.

Существует также набор утилит `vzpkg`, реализующий функциональность создания и обновления образов VE для различных дистрибутивов. В текущей реализации `vzpkg` базируется на утилите `yum`, однако в будущем планируется добавить поддержку `apt`.

### Области применения

Средства виртуализации на уровне ОС дают возможность воспользоваться всеми преимуществами виртуализации, не платя при этом большую цену в виде падения производительности.

- *Консолидация серверов* позволяет сэкономить на стоимости оборудования, площади серверных площадок и затратах на обслуживание.

- Весьма привлекательно использование технологий виртуализации на уровне ОС для *хостинга* ввиду низкой себестоимости VE, возможностей массового управления, быстрого создания VE. Однако следует заметить, что в этом сценарии надёжная изоляция и управление ресурсами приобретают критическую важность.
- *Безопасность* может быть существенно улучшена при разнесении разных сетевых сервисов (демонов) в разные VE. В качестве дополнительных преимуществ пользователь получает динамическое управление ресурсами и возможность «живой» миграции.
- Использование VE в *процессе разработки и тестирования программного обеспечения* выглядит привлекательно ввиду хорошей производительности, возможностей иметь большое количество VE, различные дистрибутивы, версии библиотек, производить клонирование VE и т. п.
- *В образовательных целях* VE можно использовать, чтобы дать каждому студенту права root, не требуя для этого большого количества серверов. Возможность пробовать разные дистрибутивы, а также быстро создавать новые VE сводит к минимуму необходимость в частой переинсталляции.

**16.40–17.00: Кофе-брейк** * * * *

## Вечернее заседание
### 17.05–19.00

**17.05–19.00: Круглый стол, ведущий Дмитрий Левин** *

# 26 июля

## Утреннее заседание
9.30–11.35

9.30–9.55

А. В. Балагута, Е. А. Чичкарев Мариуполь, Украина,
Приазовский государственный технический университет

### Моделирование динамических систем на python

Данная работа посвящена опыту разработки пакета для моделирования детерминированных динамических систем. Данное направление — моделирование динамических систем — достаточно широко представлено как коммерческими (в первую очередь SimuLink), так и свободными средствами (например, `http://sourceforge.net/projects¬/pydstool`).

В теории динамических систем используется понятие математической динамической системы. Она представляет собой математическую модель, при помощи которой можно смоделировать на удивление много физических явлений. В самой простой формулировке динамическую систему можно определить как состоящую из двух частей: вектора состояний, который описывает некоторое конкретное состояние гипотетической или реальной системы, и функции (или правила), которое говорит нам, как изменится состояние системы в следующий момент времени. Данное правило часто называют правилом эволюции динамической системы. Это правило является либо детерминированным (когда текущему состоянию системы соответствует лишь одно будущее состояние), либо недетерминированным (каждое последующее состояние может наступить лишь с некоторой вероятностью — системы такого рода называют цепями Маркова и описывают одноименным математическим

аппаратом). По характеру изменения вектора состояний выделяют системы с непрерывным временем и дискретные системы. Первые описываются аппаратом дифференциальных уравнений, вторые — аппаратом разностных уравнений.

В данной работе рассмотрены лишь детерминированные динамические системы с непрерывным временем, которые описываются обыкновенными дифференциальными уравнениями (ОДУ).

Зачастую при исследовании моделей динамических систем возникает проблема жесткости. В общем случае, жесткой называется такая система, которая не может быть решена с приемлемым шагом явными методами, поскольку разностное уравнение, аппроксимирующее исходное дифференциальное, имеет некоторую паразитную осциллирующую составляющую. Характерным признаком жестких систем является наличие двух частей — быстро протекающей переходной и чрезвычайно пологой постоянной части. Ярким примером таких систем являются линейные системы с большими отрицательными коэффициентами. Для решения жестких систем применяются специализированные неявные или псевдонеявные (такие, как метод прогноза и коррекции) методы.

Кроме дифференциальной части в динамических системах часто присутствует и нелинейная алгебраическая составляющая (дифференциально-алгебраические уравнения — ДАУ, описывающие системы со стационарной и нестационарной частями). Как правило, такие системы являются жесткими, даже если исходные дифференциальные уравнения, описывающие нестационарную часть, изначально нежесткие.

В последние несколько лет в научной среде для расчетов интенсивно используется язык программирования Python и среда (или библиотека) SciPy. Данная связка является отличной бесплатной заменой «тяжеловесным» пакетам наподобие MATLAB, а развитый, но в то же время простой объектно-ориентированный язык делает ее гибкой и удобной. Поэтому рассматриваемый пакет изначально задумывался в качестве небольшой «надстройки» SciPy для решения динамических систем, и, как следствие, интенсивно использует ее внутренние информационные структуры и функции.

Пакет состоит из основного модуля, вспомогательного модуля-решателя дифференциально-алгебраических систем (на базе модуля решения ОДУ в SciPy), а также вспомогательного модуля с процедурами общего характера. Кроме того, разработан дополнительный модуль с вычислительным ядром на C, предназначенный для интегрирования больших дифференциально-алгебраических систем. Для визуализации

результатов использована библиотека MatPlotLib (достаточно полный аналог графической компоненты пакета MatLab).

В докладе представлены примеры моделирования как тестовых систем (аттрактор Лоренца, уравнение Ван-дер-Поля и т. п.), так и реальных технических систем. В результате тестированя установлено, что по быстродействию и возможности решения систем ОДУ+ДАУ данная работа практически не уступает Simulink и MatLab. Пока лишь задуман и начал разрабатываться модуль для преобразования описания моделей динамических систем в виде ориентированного графа в модуль на Python. В настоящее время пакет моделирования динамических систем на Python используется для курсового (ТАУ, системный анализ) и дипломного проектирования, а также решения научных задач.

## 9.55–10.20

Павел Виноградов Ижевск, Elewise

### Построение единого центра авторизации на базе LDAP-сервера

Построение единых центров авторизации (ЦА) становится важным элементом инфраструктуры не только больших предприятий, но и небольших фирм и контор численностью до 50 человек. Это связано в первую очередь с возрастанием количества сервисов и ресурсов, необходимых обычному сотруднику для выполнения его обязанностей. Сейчас в интернете можно найти множество примеров и рекомендаций по настройке различного ПО для авторизации через LDAP, но все эти примеры имеют ряд существенных недостатков: рассчитаны на конкретные версии конкретных программ; в большинстве своем предоставляют только хранение информации, но не интеграцию; не предоставляют единого интерфейса управления ЦА. Для преодоления этих недостатков ЦА должен иметь гибкую модульную структуру, позволяющую заменять отдельные модули, не нарушая общей целостности системы. Также необходим гибкий интерфейс управления записями в LDAP, позволяющий максимально отделить логику работы с сервером директорий от интерфейса управления.

Большинство серверного ПО уже имеет в своем составе модули для взаимодействия с LDAP-серверами, разработанные файлы схем атри-

бутов и объектов, рекомендации по настройке. Поэтому проблема настройки самого ПО уже не стоит так остро, как стояла раньше. Основная проблема сейчас заключается в интеграции всех этих атрибутов для исключения дублирования и конфликтов и обеспечения оптимальной структуры результирующего дерева. Поэтому основными целями было:

- разработать общие рекомендации по переводу ПО на авторизацию через LDAP-сервер;
- обеспечить интеграцию авторизационной информации хранящейся в LDAP-сервере;
- разработать интерфейс управления пользователями, а не описание настройки выбранного нами набора ПО.

Есть два основных пути интеграции ПО на базе LDAP:

- на уровне ПО;
- на уровне интерфейса управления.

Интеграция на уровне ПО позволяет перенести всю логику взаимодействия приложений в код самого ПО и тем самым значительно упростить интерфейс управления. Но в этом случае требуется значительная переработка ПО и согласование этих изменений с основными разработчиками. Интеграция на уровне интерфейса управления позволяет в большинстве случаев оставить неизменным код ПО (или внести в него незначительные изменения), а необходимый уровень интеграции, синхронизацию и проверку целостности возложить на интерфейс управления. Для реализации этих задач интерфейс управления должен иметь модульную структуру, позволять отделять описание интерфейса от реализации логики и предоставлять возможность замены и изменения разработанных модулей для адаптации к конкретным условиям применения. В качестве экспериментальной платформы для разработки такого интерфейса управления был выбран Alterator, заявленные возможности которого полностью удовлетворяют требованиям к платформе разработки интерфейса управления.

В качестве ПО для практической реализации проекта были выбраны:

- Fedora Directory Server 1.0.x (`http://directory.fedora.redhat.com/`)

- Alterator 2.9
  (`http://wiki.sisyphus.ru/Alterator`).

## 10.20–10.45

Павел Морозов Москва, РАМИ РИА «Новости»

### Система управления резервным копированием backup2

Поддерживаемые платформы:

- Linux
- FreeBSD
- Windows (Cygwin)

Плюсы:

- открытый исходный код;
- модульность;
- многопоточность;
- поддержка стандартных сетевых протоколов;
- удаленное управление хранилищами архивов;
- обновление системы из репозитария;
- пре- и пост-скрипты выполнения;
- гибкое ведение многоуровневых архивов;
- возможность криптования архивов;
- различные конфигурации для создания архивов;
- лёгкость добавления собственных модулей.

Минусы:

- невысокая надежность;
- некорректный вывод некоторых программ (curl,smbclient);
- невозможность нормального логирования ошибок.

### Общая информация о системе

Система полностью написана на bash и работает через именованные потоки. На сегодняшний момент, система работает на 20 Linux-серверах и нескольких FreeBSD, выполняющих различные задачи.

Распространяется по лицензии GPL.

Возможность добавлять свои модули. Этот метод предоставляет достаточную гибкость и широту адаптирования. Здесь можно использовать различные языки программирования и фантазию. На сегодняшний момент, написаны модули для хранения архивов на локальной файловой системе и удаленных системах, с работай по протоколам `ssh`, `smb`, `ftp`. Модули для создания архивов — `fs(tar)`, `mysqlhot`, `mysqldump`, `oracle9hot` (без архив-логов).

Система не требует временного создания файлов на локальной машине. Вот здесь все не так тривиально, как хотелось, т. к. многие программы не умеют работать с потоками. Большинство проблем можно устранить путем создания именованного потока, но некоторые программы и это игнорируют и пересоздают файл при открытии. В худшем случае, вам все таки придется создать временный файл.

Поскольку управление всеми архивами из единого места — дело небезопасное (многие бэкапы запускаются от привилегированного пользователя), а серьезно на эту тему я пока не думал, реализована только одна схема управления архивами — с компьютера, на котором установлен `backup2`. Думается, что для машин, работающих под unix, можно реализовать систему, которая будет с запускаться с единого сервера бэкапов и по `ssh` делать архивы на локальную машину, но это так, дело будущего.

Существует возможность сжатия и шифрование архивов «на лету», а также добавление прохождения программ, поддерживающих потоки. Здесь все также не очень гладко, т. к. многие архиваторы не умеют нормально работать с потоками, к примеру: `rar`, `7zip` и т. д., потому рекомендуется пользоваться `gzip` и `bzip2`.

Система позволяет создавать архивы независимо от частоты выполнения (раз в 1сек., раз в 1мин. и т. д.)

Одна из проблем при большом количестве серверов — это обновление программного кода. Здесь все решается простым созданием репозитария, где могут храниться все выпущенные версии пакетов для обновления с возможностью установки любой из них. Каждый пакет обновления подписывается gpg-ключом, который проверяется при об-

новлении кода и выдает предупреждение, если подписи не совпадают. При этом также создается локальная копия старого кода.

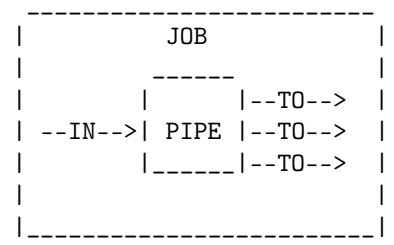
Запуск автоматического режима бэкапа производится посредством штатного планировщика системы, которую вы используете. Здесь для записи выходных сообщений в файл и отсылки по почте можно использовать программу `tail`. Пример использования таков:

```
./backup2.sh backup |tail back.log
```

### Устройство системы

Система состоит из 3 частей, которые вместе называются JOB:

1. входной модуль (создает архив и записывает в поток) — называется `IN`;
2. ядро (записывает входной поток в выходные потоки) — называется `PIPE`;
3. выходной модуль (читает поток и записывает архив на удаленную систему) — называется `TO`.




Таким образом, создаются сразу несколько туннелей до места хранения архивов. При «падении» одного из туннелей система продолжит выполнение в нормальном режиме и закончит процесс сохранения архивов в оставшиеся места доставки `TO`, если таковые имеются.

10.45–11.10

Денис Медведев Москва

## Серверные киоски на основе свободного программного обеспечения

Аннотация

Серверные киоски — это закрытые программные комплексы, предназначенные для выполнения специфичных для внутренней сети функций. Они представляют собой дистрибутив Linux с установленной и настроенной конфигурацией прикладной программы — CRM, сервера сортировки спама, почтовой программы, а также системы управления таким сервером. Так как все компоненты предустановлены и сконфигурированы, такая система будет требовать минимальной настройки при установке на месте использования.

«Киоск» — это программно-аппаратное устройство, предоставляющее некоторую определенную услугу и не конфигурируемое пользователем. «Серверным» киоском можно назвать подобное устройство, предоставляющее сервис для сети компании.

### Структурная схема и способ создания

Серверный киоск представляет собой преконфигурированный и предустановленный образ системы, загружаемый по сети, с CD/DVD, с USB или жесткого диска. После загрузки он работает либо полностью в ОЗУ, либо использует собственные или смонтированные по сети диски.

Образы подготавливаются при помощи системы подготовки образов — такой как разработанная в компании AltLinux система Separator/spt. Это позволяет относительно быстро и в рабочих условиях создавать измененные образы в случае изменения конфигурации или обновления программного обеспечения.

Предполагается использовать систему Alterator для удаленного управления такими киосками.

### Достоинства

*Безопасность*:

- Никакие системные действия не изменяют образ.
- В системе нет внутренних пользователей,
- Использование внешнего хранилища данных (NAS или iSCSI SAN) позволяет контролировать целостность данных полностью: образ неизменяем, а данные на хранилище контролируются внешним ПО. Это позволяет защититься от угрозы rootkit.

*Отказоустойчивость*:

- Снижает требования к надежности аппаратным ресурсов исполняющего компьютера, при аппаратном сбое достаточно сменить компьютер и перезапустить систему.

*Простота в обслуживании*:

- В промышленных условиях это системы «включил — заработало». Нет этапа установки системы — Возможность централизованного управления.

### Недостатки

- *Жесткость*: Серверный киоск сложно переконфигурировать — требуется перегенерация образа.
- *Требовательность к сетевым ресурсам*. В случае загрузки по сети и использования внешних хранилищ данных требуется надежность и высокая пропускная способность соответствующих сетей.

### Рекомендуемая область применения

Такие киоски имеет смысл применять при автоматизации рутинных операций, выполняемых изо дня в день по определенной схеме.

Возможные применения серверных киосков:

- Сервер хранения информации о клиентах (на основе CRM-CTT, Hipergate).

- Сервер интернет-телефонии — Asterisk.
- BPEL-процессор — на основе BIE-GPL (Java, Tomcat).
- Сервер бизнес-аналитики на основе libferris (маркировка потоков информации, агрегация, индексирование).
- Сервер защиты от спама.
- Видеосервер — на основе VLC.
- Сервер контроля безопасности.
- Сервер управления промышленным оборудованием.

**Заключение**

Создание серверных киосков способно добавить в арсенал системного администратора предприятия простые в обслуживании и безопасные полезные устройства. Открытое программное обеспечение позволяет легко конфигурировать образы для таких устройств, а также обеспечивать удобное управление серверными киосками.

## 11.10–11.35

Е. А. Чичкарев, Н. В. Назаренко Мариуполь, Украина, ПГТУ, МГГУ

## Анализ возможностей и практическое использование свободных программных средств для моделирования методом конечных элементов и решения гидродинамических задач

Данная работа посвящена исследованию возможностей открытых программных средств для разработки моделей технических систем в первую очередь методом конечных элементов. Такого рода задачи широко используются для математического моделирования и проектирования многих технических устройств.

В качестве стандартного средства автоматизированного проектирования, вероятно, трудно найти замену лицензионному, сертифицированному и весьма мощному пакету ANSYS. Однако для многих задач научного характера, настройки систем автоматизированного управления и т. п., более простые пакеты могут оказаться куда удобнее. Кроме того, использование open-source ПО в качестве основного средства решения технических задач может существенно снизить суммарную стоимость разработки.

Поучительным для авторов примером стала серия публикаций в украинских периодических изданиях, посвященных черной металлургии, работ, посвященных использованию пакета ANSYS для решения задач моделирования затвердевания непрерывнолитых заготовок, а также многочисленным работам в мировой печати, посвященных использованию пакета Fluent для решения гидродинамических задач, связанных с агрегатами черной металлургии.

Однако, как показал опыт решения задач математического моделирования и настройки автоматизированных систем управления в черной металлургии (той же модели затвердевания непрерывнолитых заготовок и моделитрования системы управления вторичным охлаждением), в зависимости от сложности решаемых задач в рамках open-source ПО во многих случаях можно выбрать приемлемое средства для ее решения.

В частности, для круга задач умеренной сложности, которые можно решить средствами MatLab (надстройка PDETools) или FemLab, в Интернете представлены достаточно мощные открытые аналоги — FiPy, разработанный Metallurgy Division and Center for Theoretical and Computational Materials Science (CTCMS), реализующий метод конечных объемов, или Elliptd2D (`http://elliptd2d.sourceforge.net/install.html`), реализующий метод конечных элементов для двумерной области.

Существенными достоинствами Python как языка для решения научно-технических задач являются простота освоения, полностью объектно-ориентированная направленность, широкие возможности расширения и подключения внешних модулей на C, Fortran и т. п.

Для решения динамических задач, связанных с решением систем нелинейных параболических уравнений, авторами был дополнен пакет Elliptd2D, а полученный результат использован для создания математической модели формирования химической неоднородности непрерывнолитой заготовки при нестационарных режимах разливки. Эта же мо-

дель, а также ряд тестовых задач были реализованы параллельно и на MatLab. Установлено, что на одном и том же компьютере при сборке оптимизированного пакета может быть достигнут выигрыш в скорости счета на 20–30 % (тестирование программ на MatLab выполнялось в Windows, программ на Python — в Debian GNU/Linux). В настоящее время данная модель дополнена и модулем расчета напряжений в сечении заготовки. Достоинством программ на Python является достаточно хорошая переносимость в среду Windows с целью демонстрации хода расчетов и результатов потребителям результатов моделирования — технологам, проектировщикам и т. п.

Ряд задач идентификации математических моделей агрегатов черной металлургии был опробован и в рамках курсового и дипломного проектирования (FreeFem, mgghat и др.). Их сложно использовать для исследования поля напряжений в ответственных конструкциях, но для решения исследовательских задач и предпроектных расчетов возможности пакетов, входящих в репозитарий Debian, более чем достаточны. По существу, широкое использование FreeFem или FreeFem3D для решения учебно-научных задач сдерживается отсутствием русскоязычной документации по особенностям реализации и входному языку системы.

Кроме того, при разработке комплексных математических моделей технологических агрегатов черной металлургии существенный интерес представляет исследование скоростей потоков жидких фаз (обычно металла). Для потоков несжимаемой жидкости в рамках open-source ПО достаточно широкими возможностями обладает пакет GERRIS. В частности, на базе этого пакета авторам удалось построить математическую модель процессов перемешивания в сталеразливочном ковше, по информативности не уступающую результатам, представленным зарубежными авторами на базе пакета FLUENT. Однако использование open-source ПО студентами и аспирантами осложняется одним важным аспектом — малым объемом пользовательской документации и доступностью литературы. Так, tutorial по пакету gerris укзывает, что это руководство конечного пользователя, не описывающее технических деталей и технологии вычислений, использованных в пакете.

Таким образом, open-source ПО независимо от вида лицензии (public domain, BSD, GPL и т. п.) может использоваться для решения широкого круга технических задач методами конечных элементов или конечных объемов и решения гидродинамических задач умеренной сложности, при этом позволяя существенно снизить затраты на коммерческое ПО. Доступность исходных текстов обеспечивает полный

контроль за ходом решения, но обычно необходимо затратить существенные усилия на освоение и настройку соответствующего пакета.

**11.35–11.55: Кофе-Брейк** * * * *

Семинар

# Регулирование информационных технологий, используемых в государственном управлении

Семинар проводится при поддержке Министерства экономического развития и торговли РФ.

Основные материалы к семинару опубликованы в отдельной брошюре.

12.00–12.15: Открытие семинара, вступительное слово от организаторов (Алексей Новодворский, ALT Linux, Москва) *

12.15–12.35: Вступительное слово от Минэкономразвития РФ

12.35–13.00

Михаил Брауде-Золотарев Москва, Экспертно-аналитический центр АНХ при Правительстве РФ

## Почему нужно регулировать применение информационных технологий в государственном секторе

13.00–14.00

Роман Ермаков Москва, КТЕ

**Проблемы целеполагания и определения области регулирования технологий в государственных информационных системах**

14.00–15.00: Обед * * * * *

15.00–15.30

Виктор Серебряков Москва, ЗАО «Ланит»

**Реализация концепции регулирования использования ИКТ в гос. управлении. Текущие результаты и планы на будущее**

15.30–16.00

Татьяна Никифорова Санкт-Петербург, «Байтен Буркхардт»

**Правовые аспекты регулирования информационных технологий в публичном секторе: опыт европейских стран и предложения для России**

16. 00–16.30

Александр Давыдов Екатеринбург, «Наумен»

**Тенденция перехода бизнеса ПО на FOSS-центрированные ИС**

16.30–17.00

Егор Гребнев Москва, ALT Linux

## Программы, созданные по государственному заказу: кто хозяин? Мировые тенденции и российские инициативы

17.00–17.20: Кофе-брейк * * * *

17.20–18.00

Анатолий Якушин, Михаил Якушин, Алиса Цветкова
Москва, Госпиталь Ветеранов Войн № 3, МАИ, МГКИТ

## Сравнительная оценка современных офисных форматов файлов

Основной тенденцией последних лет в развитии офисных приложений является постепенных отказ от использования бинарных форматов файлов в пользу файлов, имеющих разметку на основе XML. Также все чаще форматы подобных файлов становятся доступны под открытыми лицензиями, что позволяет унифицировать хранение данных при разработке приложений.

Авторы провели сравнительное исследование двух наиболее популярных форматов файлов, базирующихся на XML — OpenDocument Format (ODF) и Microsoft Office Open XML. Несмотря на весьма высокую актуальность данной темы при анализе литературы удалось обнаружить только две работы, посвященные тематике доклада [1, 2], причем обе работы касаются отдельных аспектов реализации, а не собственно форматов в целом.

По мнению авторов, целесообразность выбора формата хранения данных при разработке офисных приложений определяется не только и не столько прогрессивностью идей, положенных в его основу, сколько

простотой реализации данного формата прикладным программистом. Поэтому рассматриваемые в докладе форматы файлов анализировались с различных точек зрения.

Учитывалась доступность описаний форматов и полнота предлагаемых разработчику документов; наличие дополнительной документации и библиотек для реализации под свободными лицензиями. Анализировался доступный код приложений, в которых данные форматы использовались.

Кроме того, рассматривались аспекты лицензионной политики разработчиков форматов, степень их стандартизации. В результате проведенных исследований можно сделать вывод о том, что оба рассматриваемых формата в принципе пригодны для создания офисных приложений.

Однако OpenDocument Format на сегодняшний день несомненно является более приемлимым для свободного разработчика ввиду его зрелости, более доступного синтаксиса, использования внешних стандартов, наличия значительного количества библиотек для обработки, а также практической реализации в нескольких многоплатформенных офисных приложениях.

### Литература

[1] *Alex Hudson, J. David Eisenberg, Bruce D'Arcus, Daniel Carrera* Format comparison between ODF and MS XML.
`http://opendocumentfellowship.org/Articles/¬ FormatODFVsMSXML`

[2] *George Ou* Performance analysis of OpenOffice and MS Office
`http://blogs.zdnet.com/Ou/?p=120`

18.00–18.30

Александр Прокудин Москва, linuxgraphics.ru

## Стандартизация в свободном ПО для работы с графикой

Говоря о стандартах на форматы графических данных, предоставляемых конечным пользователям, было бы нежелательно обойти вопрос о стандартах, используемых при создании и обработке этих данных.

В настоящее время существует немало свободных приложений, в которых реализована поддержка тех и или иных стандартов, не зависящих от одного разработчика.

### CREATE, Shared Resources

Проект CREATE объединяет под своей крышей разработчиков свободного ПО для работы с графикой. Это общее информационное пространство, в котором обсуждаются и разрабатываются различные стандарты, позволяющие единообразно реализовывать те или иные функции.

Среди участников представители команд разработчиков GIMP, Scribus, Inkscape, Cinepaint, digiKam, LProf и других программных продуктов. Основная идея проекта — выработка унифицированных открытых решений, на основе которых можно создавать новые продукты и улучшать существующие, сохраняя независимость пользователей ПО от его производителей.

По исторически сложившейся традиции почти каждый новый графический редактор заимствует палитры, кисти, градиенты или текстуры GIMP, а то и всё сразу, и копирует их в свои каталоги. Регламентируемое расположение таких ресурсов в системе позволяет использовать одни и те же ресурсы в разных программах. К примеру, если вы при работе над проектом изменили наиболее часто используемую палитру цветов в GIMP, при создании нового векторного рисунка в Inkscape будет использоваться та же самая, но уже обновлённая палитра, а не её копия, находящаяся в своём каталоге Inkscape. Это упрощает работу пользователя и позволяет избежать повторного копирования одних и тех же ресурсов при установке нового ПО.

Согласно спецификации ресурсы могут храниться в двух каталогах: `/usr/share/create/` и `~/.create/`. Ввиду кроссплатформенности многих приложений в спецификации определены системные каталоги ресурсов для Windows и Mac OS X. В настоящее время спецификация поддерживается в Scribus, Krita и Inkscape.

Подобные решения в ПО с закрытым кодом не практикуются.

Сайт проекта: `http://create.freedesktop.org`

### OpenRaster

Если в ПО для обработки звука и видео есть несколько распространённых стандартов обмена данными, то в ПО для работы с графикой таких стандартов по сути нет. Необходим расширенный, по возможности, стандартизованный формат с открытой спецификацией, в котором можно было бы сохранять коррекционные слои, повторно используемые изображения с сохранением исходных метаданных и качественных характеристик, текстовые слои, базовую векторную графику.

Наиболее перспективным является формат с кодовым названием OpenRaster, который создаётся как дополнение к семейству форматов OpenDocument в рамках проекта CREATE. Существует первый черновик спецификации. Краткий перечень возможностей приведён ниже.

*Модель документа*: наследуется от OpenDocument, минимальное сжатие файлов. *Метаданные*: хранятся как тэги XMP, возможность назначать/сохранять тэги для каждого слоя отдельно, исходные данные EXIF хранятся внутри каждого повторно используемого в виде слоя изображения, все текстовые данные — в UTF-8 или UTF-16. *Слои*: хранение нескольких слоёв в одном документе, режимы наложения слоёв, коррекционные слои, текстовые слои, группировка слоёв, хранение ICC-профилей. *Прочее*: контуры, текст по контуру, обтравочные контуры, маски.

Форматов, аналогичных OpenRaster, по сути дела, не существует. В какой-то степени аналогом можно считать формат Adobe Photoshop — PSD. Однако, последняя свободно доступная спецификация на PSD описывается версию 6.0, в то время как текущая версия формата — 9.0. Формат TIFF 6.0, позволяющий хранить в одном файле несколько слоёв с разными режимами наложения и обтравочными контурами, не соответствует всем современным требованиям к формату переноса данных проектов.

Рабочая страница проекта: http://create.freedesktop.org/¬
wiki/index.php/General_multilayered_bitmap_exchange_format

### W3C SVG

SVG — язык описания двухмерной графики на основе XML. Рекомендацией веб-консорциума является SVG 1.1. Версия 1.2 находится в процессе разработки. Существуют два «мобильных» профиля SVG: SVG Basic и SVG Tiny. Они предназначены для устройств с ограниченными вычислительными ресурсами и являются частью платформы 3GPP третьего поколения мобильных телефонов. Также существует спецификация SVG Print, описывающая печать документов SVG, работа над которой временно приостановлена до успешного завершения работы над SVG Tiny 1.2. В рабочую группу входят представители таких компаний как Adobe, Canon, ERICSSON, France Telecom, Nokia, Sharp Corporation, Sun Microsystems и т. д.

SVG удобен тем, что помимо вовлечённости упомянутых выше серьёзных вендоров и непатентованности, что гарантирует поддержку рынком и независимость от одного производителя, имеет массу независимых реализаций: от встроенных средств отображения в веб-приложениях (браузеры Mozilla Firefox, Konqueror, Opera) до средств создания статической (Inkscape) и анимированной графики (Beatware Mobile Designer).

В текущей реализации SVG отсутствуют некоторые возможности, необходимые для подготовки печатаемых иллюстраций. Часть этих недостатков будет устранена в редакции 1.2 стандарта. В этом случае SVG станет единственным стандартом, одинаково пригодным для печатного и экранного представления векторной графики.

Страница стандарта: http://www.w3.org/Graphics/SVG/

### OpenIcc

Под управлением цветом понимается возможность передать оригинальные цвета изображения через всю технологическую цепочку, от создания изображения до его печати или представления на экране. Управление цветом особенно критично в полиграфии. Для передачи исходных цветов используются ICC-профили — файлы, содержащие таблицу преобразований из одного цветового пространства в другое.

Управление цветом может быть реализовано как на уровне графической подсистемы, так и на уровне отдельных приложений. К примеру, X-сервер может обеспечивать загрузку корректного профиля для монитора, в то время как отдельное приложение, скажем, digiKam будет выполнять преобразования из исходного цветового пространства в рабочее цветовое пространство и обеспечивать просмотр цветопробы — эмуляции на экране бумажного отпечатка фотографии.

Проект OpenIcc появился по инициативе разработчиков Scribus — программы для вёрстки макетов. Цель — объединить усилия разработчиков свободного ПО в работе над общими принципами реализации управления цветом.

Это, пожалуй, самый успешный стандартизационный проект из рассматриваемых. В рабочую группу входят представители команд ArgyllCMS, Cinepaint, CUPS, GraphicsMagick, GIMP, Gutenprint, ImageMagick, Inkscape, Krita, Karbon, LittleCMS, LProf, Scribus и Oyranos. Кроме того, в работе проекта принимают участие представители Lexmark, Hewlett-Packard и Adobe, а также отдельные специалисты по управлению цветом с мировым именем, например, Крис Мэрфи.

В рамках проекта достигнуто соглашение определить системные каталоги Linux для хранения ICC-профилей: `/usr/share/color/icc`, `/usr/local/share/color/icc` и `~/.color/icc`. В соответствие с соглашением приведён ряд приложений. В ноябре компания Adobe выпустила стандартный пакет ICC-профилей под более либеральной лицензией; профили из RPM-пакета устанавливаются в каталоги, определённые рабочей группой OpenIcc.

Кроме того, начата работа над спецификацией поддержки управления цветом на уровне X-сервера.

В Windows и Mac OS X системные каталоги для хранения ICC- и ICM-профилей давно определены, но схожих по букве и духу стандартизационных проектов не существует. Сравнительно недавно консорциум ICC открыл свой проект с открытым кодом, SampleICC, но далеко не все открытые и разработанные программы работают под ОС, отличными от Windows.

Страница проекта: `http://www.freedesktop.org/wiki/OpenIcc`

**18.30–19.30: Дискуссия** * * * *