Современная инфраструктура для обеспечения совместимости Linux-платформ и приложений

В.В. Рубанов
vrub@ispras.ru

Аннотация. В статье описывается подход к построению инфраструктуры для эффективной разработки и использования спецификаций Linux-платформ. Подобные спецификации описывают программные интерфейсы (API) для обеспечения совместимости между различными реализациями таких платформ и различными приложениями для них. Задача рассматривается в условиях эволюционирующих версий спецификации платформы и наличия множественных платформенных реализаций и приложений, удовлетворяющих той или иной версии спецификации. Предлагаемый подход основан на использовании централизованной базы данных, содержащей структурированную информацию о различных версиях спецификации и различных реализациях платформ и приложений, а также средств автоматической верификации фактического соответствия реализаций платформ и приложений той или иной версии спецификации. Подход иллюстрируется на примере инфраструктуры для поддержки стандарта Linux Standard Base (LSB), основного промышленного стандарта на интерфейсы базовых библиотек операционной системы Linux.

1. Введение

Одной из четко выделенных современных тенденций в мире разработки новых системных платформ можно с уверенностью назвать появление множества решений на базе операционной системы Linux. Новые платформы на базе этой ОС активно появляются в совершенно разных сегментах - от корпоративных серверов и рабочих станций до мобильных и встраиваемых систем. К сожалению, в условиях такого разнообразия системных платформ возникает проблема переносимости приложений между их различными реализациями. И если переносимость между различными сегментами (например, перенос приложения для мобильного телефона на сервер) не всегда востребована, то обеспечение совместимости различных платформ и приложений в рамках одного сегмента существенно необходимо как с точки зрения производителей платформ, так и с точки зрения производителей приложений и, в конце концов, с точки зрения конечных пользователей.

Это обусловлено тем, что проблемы переносимости приложений между разными платформами ограничивают число доступных приложений для каждой конкретной платформы, тем самым оказывая негативное влияние на сферу их применимости и, в конечном счете, популярность. С точки зрения разработчика приложений, проблемы переносимости увеличивают стоимость разработки, так как зачастую необходимо поддерживать различные версии приложения для каждой конкретной целевой платформы, или приводят к тому, что распространение приложения ограничивается одной частной платформой. Наконец, возможность использовать полный набор необходимых приложений на конкретной платформе или возможность смены платформы для предотвращения зависимости от поставщика (vendor lock-in) являются значимыми факторами для конечных пользователей, но именно эти возможности ставятся под угрозу из-за проблем переносимости приложений.

Лучшее решение, которое было найдено на пути обеспечения переносимости приложений - это выработка *открытых спецификаций* (в том числе *стандартов*) программных интерфейсов, которые предоставляются платформой и на которые могут надежно опираться приложения. Такие спецификации описывают сигнатуры интерфейсов (имя функции, параметры, возвращаемые значения) и их поведение и являются своего рода публичным контрактом между производителями платформ и приложений. Корректные реализации программных интерфейсов платформы становятся совместимыми с приложениями, если последние соответственно ограничиваются использованием только специфицированных интерфейсов.

Однако наличие только неформального описания в виде текста спецификации платформенных интерфейсов является недостаточным для эффективного использования такого подхода на практике. Как создатели платформенных спецификаций, так и разработчики соответствующих реализаций платформ и различных приложений для них нуждаются в инструментальной и методологической поддержке для успешного решения своих задач в практической плоскости.

В данной статье рассматривается подход к организации такой поддержки на примере инфраструктуры, связанной с открытым стандартом на интерфейсы базовых библиотек операционной системы Linux - **Linux Standard Base (LSB)** [1]. Эта инфраструктура была построена в рамках совместной научно-исследовательской и производственной деятельности ИСП РАН и Linux Foundation. Аналогичные инструменты и методы вполне применимы и для поддержки других системных платформ, в первую очередь основанных на Linux (например, Moblin [2], LiMo [3], Android [4], Maemo [5]).

В первом разделе статьи дается обзор одной из наиболее успешных спецификаций программных интерфейсов, а именно стандарта Linux Standard Base (LSB). Второй раздел содержит описание современных достижений в области построения инфраструктуры поддержки интерфейсных стандартов на примере LSB. Среди подразделов выделяется историческая справка о

разработке инфраструктуры, а также отдельные подразделы, описывающие базу данных LSB, инструменты работы с ней, веб-портал разработчиков, сертификационные автоматизированные тесты и систему онлайн сертификации. В заключении приводится сводка результатов, полученных в ходе разработки инфраструктуры LSB, и делаются выводы.

2. LSB – спецификация единой платформы Linux

Одним из наиболее ярких примеров спецификации Linux-платформ является стандарт Linux Standard Base (LSB) [1], основная идея которого заключается в описании общепринятого подмножества программных интерфейсов различных библиотек, составляющих «единую платформу Linux» с точки зрения производителей приложений. Это подмножество полностью представлено во всех основных дистрибутивах Linux в сегменте серверов и рабочих станций. Кроме собственно сигнатуры и поведения функций, спецификация интерфейсов в LSB включает также информацию бинарного уровня (в основном имена ELF-символов), необходимую для обеспечения переносимости приложений без перекомпиляции. Для разработки стандарта LSB в 2000 году был создан консорциум Free Standards Group (в настоящее время Linux Foundation [6]), который поддерживается ведущими ИТ-компаниями, такими как IBM, Intel, HP, Novell, Oracle и др. Первая версия стандарта была опубликована в июне 2001 года и описывала около 3000 интерфейсов. В последующие годы стандарт развивался и приобретал зрелость - в каждой версии появлялось все больше и больше интерфейсов (некоторые устаревшие исключались). Текущая версия LSB 3.2 включает более 30000 интерфейсов из более чем 40 библиотек. Большинство основных дистрибутивов Linux сертифицированы или фактически соответствуют этому стандарту (разные поколения дистрибутивов соответствуют разным версиям LSB).

Важной особенностью стандарта LSB является то, что его разработчики не пытаются диктовать что-то новое для производителей дистрибутивов Linux. В большинстве случаев LSB просто ссылается на устоявшиеся промышленные стандарты и документацию, которым основные дистрибутивы удовлетворяют де-факто. Среди спецификаций, на которые ссылается LSB, стоит отметить Single Unix Specification (POSIX), ISO C99, ISO C++ Language и различную документацию разработчиков отдельных компонентов (как правильно библиотек) Linux. Только в случае отсутствия устоявшейся документации на тот или иной интерфейс LSB описывает его независимым образом, как правило, на основе анализа реализации интерфейса в основных дистрибутивах Linux.

При принятии решений о стандартизации новых интерфейсов LSB использует критерии «наилучшей практики». Это означает, что интерфейс становится кандидатом для добавления в очередную версию LSB, если он присутствует в реализациях всех основных дистрибутивов Linux и этот интерфейс активно используется в приложениях. Другими словами, интерфейсы для

стандартизации должны быть достаточно популярны как среди разработчиков дистрибутивов, так и среди разработчиков приложений. Также, должны быть соблюден ряд технических требований, таких как наличие достаточно стабильных реализации, документации и тестов интерфейса. Важно отметить, что роль независящего от конкретного поставщика международного консорциума позволяет Linux Foundation принимать непредубежденные решения при разработке стандарта.

Современная версия LSB 3.2 включает 5 обязательных модулей:

- **LSB Core** – низкоуровневые системные интерфейсы C (библиотеки libc, libcrypt, libdl, libm, libpthread, librt, libutil, libpam, libz and libncurses).
- **LSB C++** - библиотека поддержки времени выполнения для языка C++ (libstdcxx).
- **LSB Desktop** – различные функции для работы с графическим интерфейсом пользователя и вспомогательные сервисы (в основном XML, X11, GTK и Qt).
- **LSB Interpreted Languages** – параметры окружения и стандартные модуля для языков Perl и Python.
- **LSB Printing** – в основном библиотека libcups.

Первые три модуля включают как архитектурно независимые описания, так и элементы, специфические для конкретных аппаратных архитектур, которые можно грубо представить в виде иерархии «модуль -> библиотека -> группа -> интерфейс». LSB 3.2 поддерживает 7 архитектур - IA32 (x86), AMD64 (x86_64), IA64 (Itanium), Power PC 32, Power PC 64, IBM S390 и IBM S390X. Модули Interpreted Languages и Printing включают только архитектурно независимые описания.

Важно отметить, что LSB не нацелен на *все* дистрибутивы и приложения Linux – он лишь для наиболее распространенных решений общего назначения в сегменте серверов и рабочих станций. Специализированные дистрибутивы и некоторые системные приложения могут быть не вполне совместимы с LSB. Тем временем, даже если приложение использует некоторые интерфейсы за рамками LSB, то LSB все же имеет значение для таких приложений, так как позволяет сократить издержки на обеспечение переносимости в области интерфейсов, пересекающихся с LSB, на которые приложение может надежно опираться. Для обеспечения совместимости для интерфейсов вне LSB можно использовать отдельные методы, такие как статическая линковка необходимых библиотек или разработка специальных библиотек-переходников, которые могут скрывать различия между различными дистрибутивами для заранее заданного набора необходимых интерфейсов. LSB позволяет сократить число интерфейсов, для которых нужно применять такие специальные (относительно дорогостоящие) меры.

3. Техническая инфраструктура поддержки LSB

Важнейшим фактором для эффективной разработки и поддержки интерфейсных спецификаций (в том числе стандартов) для системных платформ является техническая инфраструктура, обеспечивающая автоматизацию ключевых процессов в работе над спецификацией и облегчающая ее использование разработчиками. На примере LSB в качестве основных компонентов инфраструктуры поддержки можно назвать генераторы текста стандарта на основе центральной базы данных, веб-портал для разработчиков, системы анализа и принятия решений по развитию стандарта, тесты реализаций платформ (дистрибутивов) и приложений, включая системы автоматизированного запуска тестов, анализа результатов и сертификации. В последующих разделах дается подробный обзор этих элементов инфраструктуры после небольшого исторического отступления, описывающего историю разработки инфраструктуры LSB.

3.1. Проектная программа LSB Infrastructure

Российские специалисты активно вовлечены в LSB сообщество. Первым крупным проектом в этом направлении был Open Linux VERification (OLVER) [7]. Проект выполнялся Центром верификации ОС Linux [8] при Институте системного программирования РАН [9] по заказу Федерального агентства по науке и инновациям (Роснаука). В проекте был проанализирован текст основной (core) части стандарта LSB для около 1500 системных функций Linux, были формализованы требования на поведение этих функций и построены тесты для автоматической проверки дистрибутивов Linux на соответствие этим требованиям [10]-[11].

Результаты проекта OLVER заинтересовали комитет по стандартизации LSB - в тот момент консорциум Free Standards Group (FSG), который предложил ИСП РАН долгосрочное сотрудничество в области построения новой инфраструктуры использования и развития стандарта LSB, а также разработки технологий автоматизации тестирования и создания собственно новых тестов для Linux, что являлось основным больным местом, сдерживающим дальнейшее развитие и продвижение этого стандарта. Впоследствии, в результате слияния FSG с OSDL был создан консорциум Linux Foundation [6] и сотрудничество с ИСП РАН было расширено.

Важно отметить, что все результаты работ Linux Foundation и ИСП РАН в рамках этого сотрудничества являются открытыми и свободными (open-source) для сообщества разработчиков Linux. Первые результаты по созданию новой инфраструктуры в рамках проектной программы LSB Infrastructure [12] были представлены в июне 2007 года на конференции Linux Foundation Collaboration Summit 2007 и с тех пор они постоянно развиваются с помощью системы выпуска регулярных версий. Далее рассматривается текущее (конец 2008 года) состояние инфраструктуры LSB.

3.2. База данных LSB

Основой инфраструктуры поддержки стандарта LSB является централизованная база данных (используется СУБД MySQL), которая содержит интегрированную информацию о структуре стандартизованных элементов (от модулей до конкретных интерфейсов, их параметров, типов данных, виртуальных таблиц, и т.п.). Также эта база данных включает информацию о составе существующих дистрибутивов Linux и внешних зависимостях различных популярных приложений. Кроме того, в базе данных отражается различная оперативная информация (например, статус сертификации различных продуктов, в том числе промежуточный). Текущая база в общей сложности содержит 92 таблицы с более чем 65 миллионами записей, которые можно условно разделить на три части:

1. *Часть поддержки стандартизации* – включает информацию о стандартизованных элементах LSB и об элементах-кандидатах.
2. *Экосистемная част*ь – содержит информацию о составе (предоставляемые библиотеки и интерфейсы) основных дистрибутивов и зависимостях (требуемые от дистрибутива библиотеки и интерфейсы) популярных приложений.
3. *Сертификационная часть* – содержит оперативную информацию о статусе сертификации различных продуктов (дистрибутивов и приложений), об операциях аудита, о платежах за процедуру официальной сертификации и т.п.

Часть поддержки стандартизации описывает следующие элементы:

- группирующие элементы:
  - модули (наборы библиотек и команд);
  - библиотеки (наборы интерфейсных групп и заголовочных файлов);
  - интерфейсные группы (наборы классов и интерфейсов);
  - заголовочные файлы (наборы интерфейсов, типов данных и констант);
- основные элементы:
  - команды (например, ls, cat и т.п.);
  - классы C++;
  - интерфейсы (включая глобальные переменные);
  - константы и макросы;
  - типы данных.

Все эти элементы связаны в граф зависимостей различных видов так, что имеющейся информации достаточно для полностью автоматической генерации заголовочных файлов, которые содержат декларации всех необходимых элементов, входящих в стандарт.

Идея *экосистемной части* заключается в объединении в одном месте информации о составе (предоставляемые библиотеки и интерфейсы) основных дистрибутивов и зависимостях (требуемые от дистрибутива библиотеки и интерфейсы) популярных приложений. Эта информация постоянно

обновляется и позволяет анализировать текущее положение с точки зрения принятия решений о развитии стандарта. Например, вот основные вопросы, на которые становится возможным ответить: какими дистрибутивами предоставляется (или наоборот в каких отсутствует) тот или иной интерфейс или насколько этот интерфейс популярен среди производителей приложений?

Сертификационная часть поддерживает процесс сертификации, в том числе содержит официальный реестр сертифицированных продуктов, удовлетворяющих той или иной версии LSB. Также данные в этой части используются для оперативной работы онлайн системы сертификации LSB Certification System.

3.3. Инструменты работы с БД и веб-портал разработчиков

База данных LSB используется различными инструментами, среди которых стоит отметить более 40 скриптов, которые генерируют различные части «технического пакета поддержки LSB», такие как части текста самой спецификации стандарта, заголовочные файлы, части реализации инструментов тестирования и разработки (Software Development Kit - SDK).

Для эффективного использования данных из БД человеком был создан веб-портал *LSB Navigator*, который предоставляет графический интерфейс пользователя и позволяет эффективно работать с данными в базе данных, включая поиск, рубрикатор, кросс-ссылки, возможности обратной связи и т.п. Этот портал находится в публичном доступе (http://linux-foundation.org/navigator/) и предназначен как для рабочей группы по стандартизации LSB, так и для разработчиков различных дистрибутивов и приложений Linux. Основные возможности LSB Navigator включают:

- Навигацию по стандартизованным элементам LSB от модулей до конечных элементов в виде интерфейсов, типов данных, констант и т.п.
- Глобальные фильтры по версии LSB и по аппаратной архитектуре.
- Индивидуальные «домашние странички» для более миллиона интерфейсов Linux (доступных всего в 2 перехода с главной страницы с помощью специализированного поиска), которые в свою очередь содержат следующую информацию:
    - LSB статус интерфейса («включен в LSB», «никогда не был в LSB», «планируется для включения», «был исключен», «будет скоро исключен» - deprecated);
    - контекст интерфейса (модуль, библиотека, заголовочный файл, к которым относится интерфейс и т.п.);
    - сигнатура интерфейса (имя, параметры и возвращаемое значение);
    - прямая ссылка на описание (в первую очередь поведения) интерфейса;
    - список дистрибутивов, которые предоставляют этот интерфейс;

    - список приложений, которые используют этот интерфейс;
    - список открытых тестов, которые проверяют этот интерфейс;
    - обсуждения сообщества, связанные с этим интерфейсом.
- Информацию о дистрибутивах (предоставляемые библиотеки и интерфейсы).
- Информацию о приложениях (внешние зависимости по библиотекам и интерфейсам).
- Статистику по LSB элементам (общее число интерфейсов, команд, классов и т.д. в каждой версии LSB).
- Статистику по использованию интерфейсов приложениями:
    - Какие интерфейсы и библиотеки наиболее часто используются приложениями;
    - «LSB рейтинг» приложений – список приложений с указанием числа LSB и не-LSB библиотек и интерфейсов, используемых каждым приложением.

LSB Navigator фактически представляет собой интерактивную версию стандарта LSB и базу знаний с различной дополнительной информацией (аналогично Microsoft MSDN). Другие системы могут легко ссылаться на странички внутри LSB Navigator, позволяя интегрировано предоставлять пользователям контекстную информацию.

3.4. Сертификационные тесты дистрибутивов

Ian Murdock, бывший руководитель рабочей группы LSB и технический директор Free Standard Group, а в настоящее время директор по операционным системам в Sun Microsystems, говорил: “Интерфейсный стандарт хорош настолько, насколько хороши автоматизированные тесты, проверяющие соответствие стандарту». Тогда он имел в виду тесты, проверяющие корректность реализации интерфейса в платформе (дистрибутивах в случае Linux), однако стоит отметить, что важны также и тесты приложений, которые проверяют, что приложения используют только стандартизованные интерфейсы и делают это корректным образом. Тем не менее, далее речь будет идти в основном о тестах дистрибутивов.

В конце 2006 года покрытие LSB интерфейсов тестами составляло около 15%, что означало, что 85% стандартизованных тестов вообще не имели тестов. Именно поэтому разработка новых автоматизированных тестов соответствия дистрибутивов требованиям стандарта является важнейшим приоритетом программы LSB Infrastructure. Однако проблема заключалась в том, что стандарт включает огромное количество интерфейсов и создать за разумное время необходимое количество по-настоящему мощных тестов не представлялось возможным. В то время как было неприемлемым и то, что некоторые интерфейсы вообще не имели каких-либо тестов. Для решения этих проблем в условиях ограничений на ресурсы специалистами ИСП РАН была

предложена стратегия, основанная на выделении трех уровней качества тестов.

1. *Глубокие (Deep)* – на этом уровне целевой интерфейс вызывается много раз (в среднем около 100) в различных ситуациях (с различными параметрами и в различных внутренних состояниях).
2. *Средние (Normal)* – это наиболее распространенный уровень качества тестирования, принятый в индустрии. Целевой интерфейс проверяется в нескольких (в среднем 5) основных ситуациях.
3. *Поверхностные (Shallow)* – простейшие тесты с единственной гарантированной целью – вызвать целевой интерфейс хотя бы один раз в какой-нибудь типичной ситуации и проверить, что не было аварийного сбоя (crash). Этот уровень близок к понятию тестов существования или санитарных тестов (“existence”, “smoke”, “sanity”).

Согласно предложенной стратегии, все интерфейсы, не имеющие тестов, были проанализированы и разбиты на три группы по степени важности интерфейса. Соответственно для особо важных интерфейсов было предложено разрабатывать глубокие тесты, для основной массы библиотек - тесты среднего уровня, а для многочисленных, но не важных, интерфейсов разработать поверхностные тесты.

Для автоматизации разработки тестов различных уровней было предложено соответственно три технологии (включая соответствующие инструменты поддержки).

1. **UniTESK** (см. [13]) – для разработки глубоких тестов.
2. **T2C** (см. [14]) – для разработки тестов среднего уровня.
3. **AZOV** (см. [15]) – для разработки поверхностных тестов.

В таблице 1 приведены данные о разработанных сотрудниками ИСП РАН новых тестах различного уровня. Трудоемкость разработки дана интегрально для полного цикла от изучения целевой системы до разработки и отладки теста на многочисленных системах и аппаратных архитектурах с последующим анализом/исправлением находимых проблем.

Уровень тестов	Кол-во тестируемых интерфейсов	Кол-во тестовых испытаний	Среднее кол-во испытаний на 1 интерфейс	Трудоемкость разработки (чел.-дней/интерфейс)
Глубокие	1 500	~150 000	100	5
Средние	3 700	17 200	5	1
Поверхностные	21 800	21 800	1	0,02

Таблица 1. Статистика по разработанным в ИСП РАН новым тестам для LSB.

Важно отметить, что поверхностные тесты отличаются от глубоких и средних не только малым количеством тестовых испытаний, но и почти полным отсутствием проверок корректности функциональности. Глубокие и средние тесты содержат систематически выделенные согласно разработанной методологии проверки каждого атомарного требования из текста стандарта. При этом, когда разработанные таким образом тесты обнаруживают ошибку, то выводится не только обычная диагностика о параметрах конкретного несоответствия (например, «ожидалось XX, но функция вернула YY»), но и сообщается ссылка на конкретное место в тексте стандарта, в котором описано нарушенное общее требование. Такой уровень диагностики существенно облегчает процесс реального тестирования и повышает комфорт пользователей с точки зрения тестирования соответствия стандарту.

Согласно принятой стратегии полное покрытие всех LSB интерфейсов тестами ожидается закончить к концу 2009 года.

3.5. Средства автоматизации LSB-сертификации

Для того чтобы тесты соответствия можно было эффективно использовать на практике, нужны не только сами тесты (каждый из которых грубо представляет собой отдельную программу), но и средства их автоматизированного запуска и анализа результатов. Кроме того, необходима информационная система, которая поддерживает рабочие процедуры (workflow) формальной сертификации на основе результатов тестирования.

В рамках программы LSB Infrastructure в начале 2008 года была разработана онлайн система поддержки LSB сертификации (**LSB Certification System**), которая направляет пользователей через необходимые шаги для выполнения тестирования их продуктов, разрешения проблем и в конечном итоге получения формальной сертификации. Эта система включает 3 основные части:

- *Управление сертификацией (Certification Workflow Management)* – предоставляет пошаговые инструкции с хранением текущего индивидуального статуса (шаги могут быть произвольно разнесены по времени) для проведения сертификации дистрибутивов и приложений на соответствие LSB.
- *Реестр сертифицированных продуктов (LSB Product Directory)* – публичная часть сертификационной системы, которая хранит список сертифицированных LSB продуктов, обеспечивая различные интерактивные группировки и фильтры при просмотре.
- *Подсистема управления проблемами (Problem Reporting)* – предназначена для онлайн поддержки в разрешении различных проблем, которые возникают в процессе сертификации, например, пользователи могут сталкиваться с ошибками в тестах, и необходимо исследовать такие случаи и игнорировать такие ошибки при рассмотрении результатов тестирования. Фактически эта подсистема представляет систему

управления публичной базой знаний, одновременно предоставляя возможность организованного индивидуального общения между пользователями и сотрудниками рабочей группы LSB.

Соответственно, в системе присутствуют два режима – для пользователей и для сотрудников рабочей группы LSB (администраторов).

Наконец, в основе технической части процесса сертификации лежит запуск автоматизированных тестов соответствия и анализ их результатов. Для этих целей были разработаны инструменты **Distribution Testkit Manager** и **Linux Application Checker** для тестирования соответственно дистрибутивов и приложений. Эти средства представляют собой программные комплексы автоматизации и управления тестированием, которые обеспечивают следующие основные возможности:

- интегрированный интерфейс пользователя (графический и командной строки на выбор), одинаковый для всех тестовых наборов LSB (созданных по различным технологиям);
- выбор, какие именно тесты запускать (все, заранее определенные поднаборы или вручную выбранное подмножество);
- сохранение всех настроек конфигурации для обеспечения непрерывности процесса тестирования в различных сеансах работы;
- автоматическая загрузка недостающих на локальном компьютере тестов с FTP-сайта Linux Foundation;
- запуск сертификационных тестов «одной кнопкой»;
- унифицированные интерактивные отчеты о результатах тестирования, интегрированные с базой знаний известных ошибок и дополнительных рекомендаций, а также с контекстными ссылками на странички в LSB Navigator. Это позволяет пользователям тестов эффективно анализировать результаты тестирования;
- реестр истории запусков с возможностью регрессионного сравнения.

Онлайн система LSB сертификации и инструменты автоматизации запуска тестов интегрированы друг с другом, обеспечивая прозрачные переходы между локальной и серверной функциональностью для облегчения процесса сертификации в целом от организационных аспектов до технических.

4. Заключение

Проблемы переносимости приложений между различными реализациями системных платформ создают трудности, как разработчикам самих платформ, так и разработчикам приложений и конечным пользователям. Именно эти проблемы являются огромным негативным фактором для успеха той или иной платформы. В мире Linux, где число различных реализаций платформ исчисляется сотнями, вопросы совместимости приобретают критическое значение.

Основной инициативой по обеспечению переносимости приложений между различными Linux-платформами в сегменте серверов и рабочих станций является стандарт Linux Standard Base. Опираясь в частности на один из старейших интерфейсных стандартов POSIX, LSB является наиболее зрелым примером спецификации системной платформы и набора технических и организационных решений, составляющих инфраструктуру поддержки ее эффективной эволюционной разработки и использования. В данной статье были рассмотрены основные компоненты этой инфраструктуры, такие как:

- централизованная база данных, содержащая структурированную информацию о стандартизируемых элементах и окружающей их экосистеме;
- инструменты автоматической генерации отдельных элементов текста спецификации и реализаций инструментов и тестов для разработчиков приложений на основе базы данных;
- веб-портал для эффективного представления информации в базе данных (интерактивная версия спецификации и база знаний);
- тесты для проверки реализаций платформ на соответствие спецификации;
- тесты для проверки приложений на предмет корректного использования специфицированных интерфейсов платформы;
- инструменты автоматизации запуска тестов и анализа результатов тестирования;
- интегрированная с вышеуказанными инструментами онлайн система сертификации продуктов (реализаций платформ и приложений).

Эта инфраструктура была построена коллективом Центра верификации ОС Linux при Институте системного программирования РАН в рамках сотрудничества с международным консорциумом Linux Foundation. Разработанные средства успешно применяются на практике и получили массу положительных отзывов, как от разработчиков самого стандарта, так и от разработчиков различных дистрибутивов и от разработчиков Linux-приложений.

Есть все основания, что построение аналогичной инфраструктуры будет чрезвычайно полезно и для других спецификаций системных платформ, в первую очередь основанных на Linux (например, Moblin, LiMo, Android или Maemo), особенно там, где существуют множественные реализации платформы, а сама спецификация платформы постоянно эволюционирует.

Литература

[1] *Linux Standard Base Homepage*. http://www.linuxfoundation.org/en/LSB/.
[2] Moblin Homepage. http://moblin.org/
[3] LiMo Homepage. http://www.limofoundation.org/
[4] Android Homepage. http://code.google.com/intl/ru/android/
[5] Maemo Homepage. http://maemo.org/
[6] *Linux Foundation*. http://linuxfoundation.org/

[7] Проект Open Linux VERification (OLVER). http://linuxtesting.org/project/olver
[8] *Linux Verification Center*. http://linuxtesting.org
[9] *Institute for System Programming of the Russian Academy of Sciences*. http://ispras.ru/
[10] V. Kuliamin, A. Petrenko, V. Rubanov, A. Khoroshilov. *Formalization of Interface Standards and Automatic Construction of Conformance Tests*. Proceedings of SECR 2006 conference, Moscow.
[11] А. И. Гриневич, В. В. Кулямин, Д. А. Марковцев, А. К. Петренко, В. В. Рубанов, А. В. Хорошилов
Использование формальных методов для обеспечения соблюдения программных стандартов. Труды ИСП РАН, том.10, , 2006. С.51-68.
[12] *LSB Infrastructure Program*. http://ispras.linuxfoundation.org
[13] http://www.unitesk.com
[14] В.В.Рубанов, А.В.Хорошилов, Е.А.Шатохин. T2C: технология автоматизированной разработки тестов базовой функциональности программных интерфейсов. Труды ИСП РАН: Том 14, часть 2, 2008. С.65-82.
[15] Р. С. Зыбин, В. В. Кулямин, А. В. Пономаренко, В. В. Рубанов, Е. С. Чернов. Технология Azov автоматизации массового создания тестов работоспособности. Труды ИСП РАН, том. 14, часть 2:83-108, 2008.