Акционерное общество «Иридиум»

Операционная система «Иридиум»

Описание жизненного цикла и поддержки программного обеспечения

Листов 42

Москва, 2023

АННОТАЦИЯ

Данный документ содержит:

- описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения (ПО);
- процессы устранения неисправностей, выявленных в ходе эксплуатации программного обеспечения;
- описание процессов совершенствования программного обеспечения;
- информацию о персонале, необходимом для обеспечения поддержки ПО.

Данный документ предназначен для администраторов ПО.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящий документ представляет собой описание жизненного цикла изделия Операционная система «Иридиум» (далее – ОС «Иридиум»), производящегося АО «Иридиум».

Цели

Настоящий документ предоставляет определение всех предпринимаемых процессов, реализующих жизненный цикл ПО.

1 ПРОЦЕССЫ ЖИЗНЕННОГО ЦИКЛА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1 Общие сведения

Жизненный цикл программных средств, входящих в состав ОС «ИРИДИУМ», обеспечивается в соответствии с требованиями ГОСТ Р ИСО/МЭК 12207-2010. Основные процессы жизненного цикла программных средств в соответствии с указанным ГОСТ описаны в данном разделе.

1.2 Процесс внедрения программного обеспечения

1.2.1 Основной процесс внедрения

В результате успешного осуществления основного процесса внедрения (в ГОСТ Р ИСО/МЭК 12207-2010 используется термин «реализации») программных средств:

- определяется стратегия внедрения;
- определяются ограничения по технологии реализации проекта;
- изготавливается программная составная часть;
- программная составная часть упаковывается и хранится на предприятии изготовителе.

1.2.2 Процесс анализа требований к программным средствам

В результате успешного осуществления процесса анализа требований к программным средствам:

- определяются требования к программным элементам системы и их интерфейсам;
- требования к программным средствам анализируются на корректность и тестируемость;
- осознается воздействие требований к программным средствам на среду функционирования;
- устанавливается совместимость и прослеживаемость между требованиями к программным средствам и требованиями к системе;
- определяются приоритеты реализации требований к программным средствам;
- требования к программным средствам принимаются и обновляются по мере необходимости;
- оцениваются изменения в требованиях к программным средствам по стоимости,

графикам работ и техническим воздействиям;

- требования к программным средствам воплощаются в виде базовых линий и доводятся до сведения заинтересованных сторон.

1.2.3 Процессы проектирования программных средств

В результате успешной реализации процесса проектирования архитектуры программных средств:

- разрабатывается проект архитектуры программных средств и устанавливается базовая линия, описывающая программные составные части, которые будут реализовывать требования к программным средствам;
- определяются внутренние и внешние интерфейсы каждой программной составной части;
- устанавливаются согласованность и прослеживаемость между требованиями к программным средствам и программным проектом.

В результате успешного осуществления процесса детального проектирования программных средств:

- разрабатывается детальный проект каждого программного компонента, описывающий создаваемые программные модули;
- определяются внешние интерфейсы каждого программного модуля и устанавливается совместимость и прослеживаемость между детальным проектированием, требованиями и проектированием архитектуры.

1.2.4 Процесс конструирования программных средств

В результате успешного осуществления процесса конструирования программных средств:

- Определяются критерии верификации для всех программных блоков относительно требований;
- изготавливаются программные блоки, определенные проектом;
- устанавливается совместимость и прослеживаемость между программными блоками, требованиями и проектом;
- завершается верификация программных блоков относительно требований и

проекта.

1.2.5 Процесс комплексирования программных средств

В результате успешного осуществления процесса комплексирования программных средств:

- разрабатывается стратегия комплексирования для программных блоков, согласованная с программным проектом и расположенными по приоритетам требованиями к программным средствам;
- разрабатываются критерии верификации для программных составных частей, которые гарантируют соответствие с требованиями к программным средствам, связанными с этими составными частями;
- программные составные части верифицируются с использованием определенных критериев;
- программные составные части, определенные стратегией комплексирования, изготавливаются;
- регистрируются результаты комплексного тестирования;
- устанавливаются согласованность и прослеживаемость между программным проектом и программными составными частями;
- разрабатывается и применяется стратегия регрессии для повторной верификации программных составных частей при возникновении изменений в программных блоках (в том числе в соответствующих требованиях, проекте и кодах).

1.2.6 Процесс квалификационного тестирования программных средств

В результате успешного осуществления процесса квалификационного тестирования программных средств:

- определяются критерии для комплектованных программных средств с целью демонстрации соответствия с требованиями к программным средствам;
- комплектованные программные средства верифицируются с использованием определенных критериев;
- записываются результаты тестирования;
- разрабатывается и применяется стратегия регрессии для повторного тестирования

комплектованного программного средства при проведении изменений в программных составных частях.

1.3 Процесс поддержки программного обеспечения

1.3.1 Процесс управления документацией программных средств

В результате успешного осуществления процесса управления документацией программных средств:

- разрабатывается стратегия идентификации документации, которая реализуется в течение жизненного цикла программного продукта или услуги;
- определяются стандарты, которые применяются при разработке программной документации;
- определяется документация, которая производится процессом или проектом;
- указываются, рассматриваются и утверждаются содержание и цели всей документации;
- документация разрабатывается и делается доступной в соответствии с определенными стандартами;
- документация сопровождается в соответствии с определенными критериями.

1.3.2 Процесс управления конфигурацией программных средств

В результате успешного осуществления процесса управления конфигурацией программных средств:

- разрабатывается стратегия управления конфигурацией программных средств;
- составные части, порождаемые процессом или проектом, идентифицируются, определяются и вводятся в базовую линию;
- контролируются модификации и выпуски этих составных частей;
- обеспечивается доступность модификаций и выпусков для заинтересованных сторон;
- регистрируется и сообщается статус составных частей и модификаций;
- гарантируются завершенность и согласованность составных частей;
- контролируются хранение, обработка и поставка составных частей.

1.3.3 Процесс обеспечения гарантии качества программных средств

В результате успешного осуществления процесса гарантии качества программных средств:

- разрабатывается стратегия обеспечения гарантии качества;
- создается и поддерживается свидетельство гарантии качества;
- идентифицируются и регистрируются проблемы и (или) несоответствия с требованиями;
- верифицируется соблюдение продукцией, процессами и действиями соответствующих стандартов, процедур и требований.

1.3.4 Процесс верификации программных средств

В результате успешного осуществления процесса верификации программных средств:

- разрабатывается и осуществляется стратегия верификации;
- определяются критерии верификации всех необходимых программных рабочих продуктов;
- выполняются требуемые действия по верификации;
- определяются и регистрируются дефекты;
- результаты верификации становятся доступными заказчику и другим заинтересованным сторонам.

1.3.5 Процесс валидации программных средств

В результате успешного осуществления процесса валидации программных средств:

- разрабатывается и реализуется стратегия валидации;
- определяются критерии валидации для всей требуемой рабочей продукции;
- выполняются требуемые действия по валидации;
- идентифицируются и регистрируются проблемы;
- обеспечиваются свидетельства того, что созданные рабочие программные продукты пригодны для применения по назначению;
- результаты действий по валидации делаются доступными заказчику и другим заинтересованным сторонам.

1.3.6 Процесс ревизии программных средств

В результате успешного осуществления процесса ревизии программных средств:

- выполняются технические ревизии и ревизии менеджмента на основе потребностей проекта;
- оцениваются состояние и результаты действий процесса посредством ревизии деятельности;
- объявляются результаты ревизии всем участвующим сторонам;
- отслеживаются для закрытия позиции, по которым необходимо предпринимать активные действия, выявленные в результате ревизии;
- идентифицируются и регистрируются риски и проблемы.

1.3.7 Процесс аудита программных средств

В результате успешного осуществления процесса аудита программных средств:

- разрабатывается и осуществляется стратегия аудита;
- согласно стратегии аудита определяется соответствие отобранных рабочих программных продуктов и (или) услуг или процессов требованиям, планам и соглашениям;
- аудиты проводятся соответствующими независимыми сторонами;
- проблемы, выявленные в процессе аудита, идентифицируются, доводятся до сведения ответственных за корректирующие действия и затем решаются.

1.3.8 Процесс решения проблем в программных средствах

В результате успешной реализации процесса решения проблем в программных средствах:

- разрабатывается стратегия менеджмента проблем;
- проблемы регистрируются, идентифицируются и классифицируются;
- проблемы анализируются и оцениваются для определения приемлемого решения (решений);
- выполняется решение проблем;
- проблемы отслеживаются вплоть до их закрытия;

- известно текущее состояние всех зафиксированных проблем.

2 ПОРЯДОК ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Общие сведения

По условиям партнерского договора, все партнеры разработчика (производителя) ПО АО «ИРИДИУМ», поставляющие решения на базе данного ПО конечным пользователям, оказывают конечным пользователям услуги технической поддержки. При прямой продаже ПО АО «ИРИДИУМ» поддержка оказывается непосредственно разработчиком ПО. В данном разделе описываются минимальные требования к условиям технической поддержки. По условиям конкретного контракта партнер или разработчик могут предоставлять более высокие уровни технической поддержки.

2.2 Техническая поддержка первого уровня

Техническая поддержка первого уровня подразумевает регистрацию обращения и консультацию, оказываемую конечному пользователю партнером производителя ПО, проводившей работы по внедрению ПО. Она осуществляется по телефону и электронной почте в режиме 8x5 (восемь часов в день, пять рабочих дней в неделю).

2.3 Техническая поддержка второго уровня

Под технической поддержкой второго уровня понимается устранение возникших неполадок, осуществляемое техническими специалистами организации, проводившей работы по внедрению ПО, в режиме 8x5 (восемь часов в день, пять рабочих дней в неделю).

2.4 Техническая поддержка третьего уровня

Техническая поддержка третьего уровня оказывается непосредственно производителем ПО в ситуациях, когда партнер не может справиться с возникшей проблемой самостоятельно и нуждается в помощи технических специалистов производителя ПО.

В рамках технической поддержки третьего уровня оказываются следующие услуги:

- консультации технических специалистов по АО «ИРИДИУМ»;
- предоставление необходимых руководств по ПО ОС «ИРИДИУМ»;
- предоставление рекомендаций или готовых решений по устранению проблем, возникающих у пользователя в процессе установки или эксплуатации ПО ОС «ИРИДИУМ»;
- предоставление обновлений, повышающих функциональность или устраняющих ошибки в работе ОС «ИРИДИУМ»;
- выезд специалиста производителя ПО для проведения обследования и устранения проблемы.

Техническая поддержка оказывается производителем ПО только в случае:

- действия срока бесплатной технической поддержки или оплаты его продления;
- использования ПО ОС «ИРИДИУМ» с лицензионной продукцией;
- соблюдения всех условий применения ПО и лицензионного договора.

3 УСТРАНЕНИЕ НЕИСПРАВНОСТЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Перечень этапов процесса устранения неисправностей программного обеспечения (ПО) приведено в п. 1.3.8 «Процесс решения проблем в программных средствах». Общий порядок технической поддержки ПО приведен в п. 2.

Штатный порядок работы ПО определяется эксплуатационной документаций, предоставляемой производителем ПО. Поддерживаемый ПО набор функций определяется требованиями технического задания (ТЗ), утвержденного Заказчиком.

В случае обнаружения ошибок в работе ПО, которые являются нарушением требований ТЗ или противоречат порядку работы ПО, описанному в документации, администратор ПО должен направить заявку в службу технической поддержки (СТП) организации, проводившей работы по внедрению ПО. СТП организации, внедрившей ПО, проверяет, при необходимости уточняет полученную заявку и пытается выполнить ее, использую собственные ресурсы и знания.

В случае, если силами СТП организации, внедрившей ПО, выполнить заявку не удается, указанная организация обращается за помощью к производителю ПО. СТП производителя, проверяет наличие ошибки и рекомендаций по ее устранению в базе знаний технической поддержки.

В случае, если в базе знаний обнаружить описание ошибки не удается, СТП производителя пытается воспроизвести обнаруженную пользователем ошибку в тестовой среде. После подтверждения найденной ошибки СТП производителя передает разработчикам ПО задание на устранение обнаруженной ошибки.

После устранения неисправности разработчики ПО выпускают обновление к текущей версии ПО или включают исправление в следующую версию ПО. Информация о наличии обновления или новой версии ПО доводится до партнеров производителя ПО. В случае наличия у Заказчика контракта или договора на поддержку ПО, Заказчик имеет право на получение обновления ПО.

4 СОВЕРШЕНСТВОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Работа по совершенствованию ПО включает в себя два основных направления:

- повышение качества и надежности ПО;
- актуализация перечня функций, поддерживаемых ПО.

В ходе постоянно проводимой работы по совершенствованию ПО используются хорошо зарекомендовавшие себя методы повышения качества и надежности ПО:

- совершенствование процесса разработки ПО – повышение качества ПО за счет использования современных методик и инструментов разработки;
- совершенствование процесса тестирования ПО – обеспечение необходимой полноты покрытия.

Актуализация перечня функций, поддерживаемых ПО, включает в себя:

- добавление новых и изменение существующих функций в соответствии со стратегией развития ПО;
- добавление новых и изменение существующих функций по предложениям Заказчиков и партнеров производителя ПО;
- исключение устаревших функций.

5 ОТЕЧЕСТВЕННАЯ ПРОГРАММНАЯ ИНФРАСТРУКТУРА ПОДДЕРЖКИ, РАЗРАБОТКИ И ОБНОВЛЕНИЯ ПО

В качестве технологического комплекса для выпуска ОС «Иридиум» используется инфраструктура в основе, которой, лежат технологии отечественного репозитория свободного программного обеспечения Sisyphus. Репозиторий не только является хранилищем пакетов программ, но и сопровождается набором оригинальных технологий, которые поддерживают его целостность, обеспечивают возможность взаимодействия для разработчиков (в том числе и сторонних), и позволяют создавать на этой базе дистрибутивы Linux и другие решения различного назначения.

Кроме того, данная инфраструктура позволяет обеспечить интеграцию в окружение операционной системы и поддержку программных продуктов различных разработчиков.

Инфраструктура поддержки, разработки и обновления ПО включает в себя следующие компоненты:

1. **Банк исходных кодов программ**

В банке хранятся все исходные коды вместе с историей изменений. Обеспечивается ведение нескольких веток одного проекта как одним разработчиком, так и разными, в целях поддержки нескольких версий дистрибутива, а также нескольких дистрибутивов. Банк исходных кодов интегрирован с системой сборки пакетов.

2. **Система контроля зависимостей между пакетами**

При формировании пакетов программ, разработчик указывает разного рода зависимости с другими пакетами (по выполнению, по сборке, и др.). Автоматическая система контроля анализирует как исполняемые бинарные файлы, так и программы на скриптовых языках в целях обнаружения неразрешенных зависимостей или паразитных зависимостей. Такая проверка обеспечивает высокий уровень интеграции всех пакетов в единый репозиторий, на основании, которого можно создавать целостные решения, а также уменьшает объѐм ручного тестирования, необходимого в технологическом процессе.

3. **Система сборки пакетов в гарантированных сборочных средах**

При сборке пакета из исходных кодов автоматически формируется виртуальная файловая система, гарантирующая воспроизводимость сборки независимо от

конфигурации сборочной системы. Сборка пакетов в виртуальной среде позволяет фиксировать среду сборки, а также обеспечивает еѐ безопасность. Система сборки интегрирована с банком исходных кодов программ и позволяет отслеживать связь между собранными пакетами и исходным кодом, что облегчает тестирование и отладку решений.

4. Система распараллеливания сборки

Обеспечивает распределенную сборку пакетов на доступных компьютерных мощностях, что позволяет достигать разумного времени сборки пакетов и, при необходимости, пересборки всего репозитория.

5. Единая модульная система управления

Управление различными системными объектами обеспечивается средствами системы Alterator, которая предоставляет API для создания и подключения различных модулей управления, специфичных для данного дистрибутива, а также набор готовых модулей для управления типичными Unix-сервисами.

6. Единая система управления пакетами программ

Управление программным обеспечением осуществляется единой системой, представляющей ПО в виде так называемых пакетов и контролирующей зависимости между пакетами, включая версии пакетов. Наличие этой системы позволяет формировать дистрибутивы с контролированной замкнутостью по зависимостям между программными компонентами, а в дальнейшем – организовывать выборочную установку пакетов или их обновление, без нарушения целостности системы и без потери пользовательских настроек.

7. Единая система установки

Установка дистрибутива осуществляется единой настраиваемой системой, интегрированной с системой управления системными объектами и с системой управления пакетами.

8. Система генерации дистрибутивов

Данная система позволяет создавать инсталляционные образы дистрибутивов (наборы iso- образов CD или DVD, образа сетевой загрузки) по формализованному описанию (набору целевых пакетов) с учѐтом зависимостей между пакетами.

6 БАНК ИСХОДНЫХ КОДОВ ПРОГРАММ

Все исходные текста входящих в дистрибутив программ находятся в специально созданном для этого банке исходных кодов. Все изменения фиксируются в системе контроля версий. Именно из этого банка в дальнейшем и создаются бинарные пакеты. Таким образом, исходные программный код из банка исходных кодов проходит стадию сборки в гарантированной сборочной среде, в результате чего попадает в банк бинарных пакетов.

Исходный текст является свободно доступным, что позволяет производить его своевременный аудит и адаптацию. Структура хранения исходных кодов может быть произвольной. Для сборки пакетов из произвольно устроенного репозитория используется утилита gear.

7 СИСТЕМА КОНТРОЛЯ ЗАВИСИМОСТЕЙ МЕЖДУ ПАКЕТАМИ

В современных системах на базе Linux огромное число общих ресурсов, которыми пользуются сразу несколько программ: разделяемых библиотек, содержащих стандартные функции, исполняемых файлов, сценариев и стандартных утилит и т. д. Удаление или изменение версии одного из составляющих систему компонентов может повлечь неработоспособность других, связанных с ним компонентов, или даже вывести из строя всю систему. В контексте системного администрирования проблемы такого рода называют нарушением целостности системы. Задача администратора – обеспечить наличие в системе согласованных версий всех необходимых программных компонентов (обеспечение целостности системы).

Для установки, удаления и обновления программ и поддержания целостности системы в Linux в первую очередь стали использоваться менеджеры пакетов. С точки зрения менеджера пакетов программное обеспечение представляет собой набор компонентов – программных пакетов. Такие компоненты содержат в себе набор исполняемых программ и вспомогательных файлов, необходимых для корректной работы программного обеспечения. Менеджеры пакетов облегчают установку программ: они позволяют проверить наличие необходимых для работы устанавливаемой программы компонент подходящей версии непосредственно в момент установки, а также производят необходимые процедуры для регистрации программы во всех операционных средах пользователя: сразу после установки программа может быть доступна пользователю из командной строки и – если это предусмотрено – появляется в меню всех графических оболочек.

Часто компоненты, используемые различными программами, выделяют в отдельные пакеты и помечают, что для работы ПО, предоставляемого пакетом A, необходимо установить пакет B. В таком случае говорят, что пакет A зависит от пакета B или что между пакетами A и B существует зависимость.

Отслеживание зависимостей между такими пакетами представляет собой серьѐзную задачу для любого дистрибутива – некоторые компоненты могут быть

взаимозаменяемыми: может обнаружиться несколько пакетов, предлагающих затребованный ресурс.

Задача контроля целостности и непротиворечивости установленного в системе ПО ещѐ сложнее. Представим, что некие программы A и B требуют наличия в системе компоненты C версии 1.0. Обновление версии пакета A, требующее обновления компоненты C до новой, использующей новый интерфейс доступа, версии (например, до версии 2.0), влечѐт за собой обязательное обновление и программы B.

Однако менеджеры пакетов оказались неспособны предотвратить все возможные коллизии при установке или удалении программ, а тем более эффективно устранить нарушения целостности системы. Особенно сильно этот недостаток сказывается при обновлении систем из централизованного репозитория пакетов, в котором последние могут непрерывно обновляться, дробиться на более мелкие и т. п. Этот недостаток и стимулировал создание систем управления программными пакетами и поддержания целостности системы.

Для автоматизации этого процесса и применяется **Усовершенствованная система управления программными пакетами APT** (от англ. Advanced Packaging Tool). Такая автоматизация достигается созданием одного или нескольких внешних репозиториев, в которых хранятся пакеты программ и относительно которых производится сверка пакетов, установленных в системе. Репозитории могут содержать как официальную версию дистрибутива, обновляемую его разработчиками по мере выхода новых версий программ, так и локальные наработки, например, пакеты, разработанные внутри компании.

Таким образом, в распоряжении APT находятся две базы данных: одна описывает установленные в системе пакеты, вторая – внешний репозиторий. APT отслеживает целостность установленной системы и, в случае обнаружения противоречий в зависимостях пакетов, руководствуется сведениями о внешнем репозитории для разрешения конфликтов и поиска корректного пути их устранения.

8 СИСТЕМА СБОРКИ ПАКЕТОВ В ГАРАНТИРОВАННЫХ СБОРОЧНЫХ СРЕДАХ

8.1 Сборочная среда hasher

hasher – инструмент для сборки пакетов в «чистой» и контролируемой среде. Это достигается с помощью создания в chroot минимальной сборочной среды, установки туда указанных в source-пакете сборочных зависимостей и сборке пакета в свежесозданной среде. Для сборки каждого пакета сборочная среда создаѐтся заново.

Такой принцип сборки имеет несколько следствий:

8.1.1 все необходимые для сборки зависимости должны быть указаны в пакете. Для облегчения поддержания сборочных зависимостей в актуальном состоянии в Sisyphus имеется инстру- мент под названием buildreq;

8.1.2 сборка не зависит от конфигурации компьютера пользователя, собирающего пакет, и может быть повторена на другом компьютере;

8.1.3 изолированность среды сборки позволяет с лѐгкостью собирать на одном компьютере паке- ты для разных дистрибутивов и веток репозитория – для этого достаточно лишь направить hasher на различные репозитории для каждого сборочного окружения.

Дополнительно к сборке пакетов hasher:

8.1.4 проверяет их с помощью утилиты `sisyphus_check`;

8.1.5 создаѐт локальный APT-репозиторий с результатами сборки, позволяя последовательно со-бирать пакеты, опираясь на уже собранные.

Система сборки пакетов в гарантированных сборочных средах предъявляет следующиетребования:

8.1.6 не снижать уровень безопасности хост-системы;

8.1.7 обеспечивать собственную безопасность от атак со стороны пакетов;

8.1.8 обеспечивать безопасность сборки пакетов от атак со стороны других пакетов;

8.1.9 гарантировать надѐжность (воспроизводимость) результатов сборки;

8.1.10 обеспечивать приемлемый уровень производительности.

8.2 Архитектура hasher

В основе архитектуры hasher лежит трѐх пользовательская модель: вызывающий непривилегированный пользователь (C) и два непривилегированных вспомогательных псевдопользователя; первый (R) играет роль root в порождаемой сборочной среде, второй (U) – обычного пользователя, собирающего программы.

Переключение между вызывающим и вспомогательными пользователями осуществляется с помощью специальной привилегированной программы (вызываемой посредством `sudo`), написанной с применением мер защиты от непривилегированных пользователей. Кроме того, с помощью этой программы удаляются процессы, запущенные вспомогательными псевдопользователями и не завершившиеся в срок, а также создаются устройства. Наконец, эта программа предоставляет возможность контролировать ресурсы, выделяемые процессам непривилегированных пользователей, для защиты от DOS-атак.

Путь пакета через сборочную систему в общих чертах выглядит следующим образом:Пользователь **C** порождает среду (aptbox) для работы с apt.

Полностью удаляется сборочная среда, возможно оставшаяся от предыдущей сборки. Удаление происходит последовательно в chroot пользователем **U**, в chroot пользователем **R** и, наконец, пользователем **C**.

Пользователь **C** создаѐт каркас новой сборочной среды, состоящий из вспомогательных каталогов и вспомогательных статически слинкованных программ (ash, find и cpio). С помощью вспомогательной привилегированной программы создаѐтся фиксированный набор устройств, достаточный для нормального функционирования сборочной среды и при этом не несущий угрозы host-системе.

Порождается базовая установочная среда, представляющая собой набор средств, необходимых для штатной установки пакетов в эту среду. Пользователь **C** с помощью apt определяет набор пакетов, необходимых для порождения базовой установочной среды.

Пользователь **R** с помощью вспомогательных статически слинкованных программ распаковывает эти пакеты.

Порождается базовая сборочная среда, представляющая собой набор средств, необходимых для сборки любого пакета. Пользователь **C** с помощью apt определяет набор пакетов, пользователь **R** устанавливает их.

Порождается сборочная среда для данного пакета. Пользователь **U** извлекает сборочные зависимости пакета, пользователь **C** с помощью apt определяет набор пакетов для установки, и пользователь **R** устанавливает их.

Пользователь **U** осуществляет сборку пакета.

Такая схема призвана исключить атаки вида **U->R**, **U->C**, **R->C**, а также все виды атак на

- **root**.

Для повышения производительности, особенно важной при сборке большого числа пакетов,

применяется кэширование базовой сборочной среды. С помощью средств apt реализованавозможность использования собранных ранее пакетов для сборки последующих пакетов.

9 ЕДИНАЯ СИСТЕМА УПРАВЛЕНИЯ ПАКЕТАМИ ПРОГРАММ

Используемая усовершенствованная система управления программными пакетами APT, является системой управления пакетами с простым пользовательским интерфейсом, позволяющим производить установку, обновление и повседневные «хозяйственные» работы с установленными на машине программами без необходимости изучения тонкостей используемого в дистрибутиве менеджера программных пакетов.

Система APT состоит из нескольких утилит. Чаще всего используется утилита управления пакетами apt-get: она автоматически определяет зависимости между пакетами и строго следит за их соблюдением при выполнении любой из следующих операций: установка, удаление или обновлениепакетов.

9.1Репозитории

Репозитории, с которыми работает APT, отличаются от обычного набора пакетов наличием метаинформации – индексов пакетов, содержащихся в репозитории, и сведений о них. Поэтому, чтобы получить всю информацию о репозитории, APT достаточно получить его индексы.

APT может работать с любым количеством репозиториев одновременно, формируя единую информационную базу обо всех содержащихся в них пакетах. При установке пакетов APT обращает внимание только на название пакета, его версию и зависимости, а расположение в том или ином репозитории не имеет значения. Если потребуется, APT в рамках одной операции установки группы пакетов может пользоваться несколькими репозиториями.

П р и м е ч а н и е . Подключая одновременно несколько репозиториев, нужно следить за тем, чтобы они были совместимы друг с другом по пакетной базе, т. е. отражали один определѐнный этап разработки. Например, совместимыми являются основной репозиторий дистрибутива и репозиторий обновлений по безопасности к данному дистрибутиву. В то же время смешение среди источников APT репозиториев, относящихся к разным дистрибутивам, или смешение

стабильного репозитория с нестабильной веткой разработки (Sisyphus) чревато различными неожиданными трудностями при обновлении пакетов.

APT позволяет взаимодействовать с репозиторием с помощью различных протоколов доступа. Наиболее популярные – HTTP и FTP, однако существуют и некоторые дополнительные методы.

Для того чтобы APT мог использовать тот или иной репозиторий, информацию о нем необходимо поместить в файл `/etc/apt/sources.list`, либо в любой файл `.list` (например, `mysources.list`) в каталоге `/etc/apt/sources.list.d`. Описания репозиториев заносятся в эти файлы в следующем виде:

```
rpm [подпись] метод: путь база название
rpm-src [подпись] метод: путь база название
```

Здесь:

- rpm или rpm-src – тип репозитория (скомпилированные программы или исходные тексты);
- [подпись] – необязательная строка-указатель на электронную подпись разработчиков. Нали- чие этого поля подразумевает, что каждый пакет из данного репозитория должен быть под- писан соответствующей электронной подписью. Подписи описываются в файле

/etc/apt/vendor.list;

- метод – способ доступа к репозиторию: ftp, http, file, cdrom, copy;
- путь – путь к репозиторию в терминах выбранного метода;
- база – относительный путь к базе данных репозитория;
- название – название репозитория.

Непосредственно после установки ОС «Иридиум» в файлах `/etc/apt/sources.list.d/*.list` обычно указывается интернет-репозиторий, совместимыйс установленным дистрибутивом.

После того как отредактирован список репозиториев в `sources.list`, необходимо обновить локальную базу данных APT о доступных пакетах. Это делается командой `apt-get update`.

Если в `sources.list` присутствует репозиторий, содержимое которого может изменяться (как происходит с любым постоянно разрабатываемым репозиторием, в частности, обновлений по безопасности), то прежде чем работать с APT, необходимо синхронизировать локальную базу данных с удалѐнным сервером командой `apt-get update`. Локальная база данных создаѐтся заново каждый раз, когда в репозитории происходит изменение: добавление, удаление или переименование пакета. Для репозиториев, находящихся на компакт-дисках и подключѐнных командой `apt-cdrom add`, синхронизация производится единожды в момент подключения.

При выборе пакетов для установки, APT руководствуется всеми доступными репозиториями вне зависимости от способа доступа к ним. Так, если в репозитории, доступном по сети Интернет, обнаружена более новая версия программы, чем на компакт-диске, то APT начнѐт загружать данный пакет из Интернет. Поэтому если подключение к Интернет отсутствует или ограничено низкой пропускной способностью канала или высокой стоимостью, то следует закомментировать те строчки в `/etc/apt/sources.list`, в которых говорится о ресурсах, доступных по Интернет.

9.1.1 Утилита apt-repo для работы с репозиториями

Для редактирования репозиториев можно воспользоваться скриптом `apt-repo`:

- просмотреть список активных репозиториев:

```
apt-repo
```

- добавить репозиторий в список активных репозиториев:

```
apt-repo add репозиторий
```

- удалить или выключить репозиторий:

```
apt-repo rm репозиторий
```

- обновить информацию о репозиториях:

```
apt-repo update
```

- справка о команде apt-repo:

```
man apt-repo
```

или

```
apt-repo --help
```

Примечание. Для выполнения большинства команд необходимы права администратора.

Типичный пример использования: удалить все источники и добавить стандартный репозиторий P10 (архитектура выбирается автоматически):

```
# apt-repo rm
all # apt-
repo add p10
```

Или то же самое одной командой:

```
# apt-repo set p10
```

9.1.2 Добавление репозитория на CD/DVD-носителе

Для добавления в `sources.list` репозитория на компакт-диске в APT предусмотрена специальная утилита – `apt-cdrom`. Чтобы добавить запись о репозитории на компакт-диске, достаточно вставить диск в привод и выполнить команду `apt-cdrom add`. После этого в `sources.list` появится запись о подключѐнном диске.

Примечание. Если при выполнении команды `apt-cdrom add`, вы получаете ошибку:

```
Не найдена точка монтирования /media/ALTLinux/ диска
```

Необходимо:

- в файл `/etc/fstab` добавить строку:

```
/dev/sr0 /media/ALTLinux udf,iso9660
ro,noauto,user,utf8,nofail,comment=x-gvfs-
show  0 0
```

- создать каталог для монтирования:

```
# mkdir /media/ALTLinux
```

- затем использовать команду добавления носителя:

```
# apt-cdrom add
```

9.1.3 Добавление репозиториев вручную

Для изменения списка репозиториев можно отредактировать в любом текстовом

редакторефайлы из каталога /etc/apt/sources.list.d/.

Примечание. Для изменения этих файлов необходимы права администратора.

В файле alt.list может содержаться такая информация:

```
# ftp.altlinux.org (ALT Linux, Moscow)

# ALT Platform 10
#rpm [p10]
ftp://ftp.altlinux.org/pub/distributions/ALTLinux
p10/branch/x86_64classic
#rpm [p10]
ftp://ftp.altlinux.org/pub/distributions/ALTL
inuxp10/branch/x86_64-i586 classic
#rpm [p10]
ftp://ftp.altlinux.org/pub/distributions/ALTLinux
p10/branch/noarchclassic

rpm [p10]
http://ftp.altlinux.org/pub/distributions/ALTLinux
p10/branch/x86_64classic
rpm [p10]
http://ftp.altlinux.org/pub/distributions/ALT
Linuxp10/branch/x86_64-i586 classic
rpm [p10]
http://ftp.altlinux.org/pub/distributions/ALTLinux
p10/branch/noarchclassic


#rpm [p10] rsync://ftp.altlinux.org/ALTLinux
p10/branch/x86_64 classic
#rpm [p10] rsync://ftp.altlinux.org/ALTLinux
p10/branch/x86_64-i586 classic#rpm [p10]
```

```
rsync://ftp.altlinux.org/ALTLinux p10/branch/noarch
classic
```

По сути, каждая строчка соответствует некому репозиторию. Не активные репозитории – строки, начинающиеся со знака #. Для добавления нового репозитория, достаточно дописать его в этот или другой файл.

После обновления списка репозиториев следует обновить информацию о них (выполнить команду `apt-get update` или `apt-repo update`).

9.2 Поиск пакетов

Если точное название пакета неизвестно, то для его поиска можно воспользоваться утилитой

`apt-cache`, которая позволяет искать не только по имени пакета, но и по его описанию.

Команда `apt-cache search` позволяет найти все пакеты, в именах или описании которых присутствует указанная подстрока. Например:

```
$ apt-cache search ^gimp
gimp - The GNU Image Manipulation
Programlibgimp - GIMP libraries
libgimp-devel - GIMP plugin and extension
development kitgimp-help-en - English help
files for the GIMP
gimp-help-ru - Russian help files for the GIMP
gimp-plugin-separateplus - Improved version of the CMYK
Separationplug-in [...]
gimp-script-ISONoiseReduction - Gimp script for reducing
sensor noise[...]
gimp-plugin-gutenprint - GIMP plug-in for gutenprint
gimp-plugin-ufraw - GIMP plugin for opening and converting
RAW files[...]
```

Обратите внимание, что в данном примере в поисковом выражении используется символ ^, указывающий на то, что необходимо найти совпадения только в начале

строки (в данном случае – в начале имени пакета).

Для того чтобы подробнее узнать о каждом из найденных пакетов и прочитать его описание, можно воспользоваться командой `apt-cache show`, которая покажет информацию о пакете из репозитория:

```
$ apt-cache show gimp-
help-ruPackage: gimp-
help-ru
Section: Graphics
Installed Size:
37095561
Maintainer: Alexey Tourbin
<at@altlinux.ru>Version: 2.6.1-alt2
Pre-Depends: rpmlib(PayloadIsLzma)
Provides: gimp-help-ru (= 2.6.1-
alt2) Obsoletes: gimp-help-common
(< 2.6.1-alt2)Architecture: noarch
Size: 28561160
MD5Sum:
0802d8f5ec1f78af6a4a19005af4e37d
Filename: gimp-help-ru-2.6.1-
alt2.noarch.rpmDescription: Russian
help files for the GIMPRussian help
files for the GIMP.
```

Утилита `apt-cache` позволяет осуществлять поиск и по русскому слову, однако в этом случае будут найдены только те пакеты, у которых есть описание на русском языке. К сожалению, русское описание на настоящий момент есть не у всех пакетов, хотя описания наиболее актуальных для пользователя пакетов переведены.

9.3 Установка или обновление пакета

Установка пакета с помощью APT выполняется командой:

```
# apt-get install имя_пакета
```

Примечание. Для установки пакетов требуются привилегии администратора.

Утилита `apt-get` позволяет устанавливать в систему пакеты, требующие для работы другие, пока ещѐ не установленные. В этом случае он определяет, какие пакеты необходимо установить, и устанавливает их, пользуясь всеми доступными репозиториями.

Установка пакета gimp командой `apt-get install gimp` приведѐт к следующему диалогу с APT:

```
# apt-get install gimp
Чтение списков пакетов... Завершено
Построение дерева зависимостей...
Завершено
Следующие дополнительные пакеты будут установлены:
icc-profiles libbabl libgegl libgimp libjavascriptcoregtk2
libopenrawlibspiro libwebkitgtk2 libwmf
Следующие НОВЫЕ пакеты будут установлены:
gimp  icc-profiles libbabl libgegl libgimp
      libjavascriptcoregtk2libopenraw libspiro libweb-
kitgtk2 libwmf
0 будет обновлено, 10 новых установлено, 0 пакетов будет
удалено и 0 небудет обновлено.
Необходимо получить 0B/24,6MB архивов.
После  распаковки потребуется дополнительно 105MB
       дисковогопространства.
Продолжить? [Y/n] y
. . .
Получено 24,6MB за 0s
(44,1MB/s).Совершаем
изменения...
```

```
Preparing... ########################################## [100%]
1: libbabl ############################################ [10%]
2: libwmf ############################################# [20%]
3: libjavascriptcoregtk2
################################### [30%]
4: libwebkitgtk2 ###################################################
[40%]
5: icc-profiles ####################################################### [50%]
6: libspiro ########################################### [60%]
7: libopenraw ######################################### [70%]
8: libgegl ############################################ [80%]
9: libgimp ############################################ [90%]
10: gimp ############################################## [100%]
Running /usr/lib/rpm/posttrans-
filetriggersЗавершено.
```

Команда `apt-get install` имя_пакета используется и для обновления уже установленного пакета или группы пакетов. В этом случае `apt-get` дополнительно проверяет, не обновилась ли версия пакета в репозитории по сравнению с установленным в системе.

При помощи APT можно установить и отдельный бинарный rpm-пакет, не входящий ни в один из репозиториев (например, полученный из Интернет). Для этого достаточно выполнить команду `apt-get install путь_к_файлу.rpm`. При этом APT проведѐт стандартную процедуру проверки зависимостей и конфликтов с уже установленными пакетами.

Иногда, в результате операций с пакетами без использования APT, целостность системы нарушается, и `apt-get` отказывается выполнять операции установки, удаления или обновления. В этом случае необходимо повторить операцию, задав опцию `-f`, заставляющую `apt-get` исправить нарушенные зависимости, удалить или заменить конфликтующие пакеты. В этом случае необходимо внимательно следить за сообщениями, выдаваемыми `apt-get`. Любые действия в этом режиме

обязательно требуют подтверждения со стороны пользователя.

Удаление установленного пакета

Для удаления пакета используется команда `apt-get remove имя_пакета`. Для того чтобы не нарушать целостность системы, будут удалены и все пакеты, зависящие от удаляемого: если отсутствует необходимый для работы приложения компонент (например, библиотека), то само приложение становится бесполезным. В случае удаления пакета, который относится к базовым компонентам системы, `apt-get` потребует дополнительного подтверждения производимой операции с целью предотвратить возможную случайную ошибку.

Примечание. Для удаления пакетов требуются привилегии администратора.

При попытке с помощью apt-get удалить базовый компонент системы, вы увидите следующий запрос на подтверждение операции:

```
# apt-get remove filesystem
Обработка файловых зависимостей...
ЗавершеноЧтение списков пакетов...
Завершено Построение дерева
зависимостей... Завершено Следующие
пакеты будут УДАЛЕНЫ:
basesystem filesystem ppp sudo
Внимание: следующие базовые пакеты будут удалены:
В обычных условиях этого не должно было произойти,
надеемся, вы точнопредставляете, чего требуете!
basesystem filesystem (по причине basesystem)
0 пакетов будет обновлено, 0 будет добавлено
новых, 4 будетудалено(заменено) и 0 не будет
обновлено.
Необходимо получить 0B архивов. После распаковки
588kБ будетосвобождено.
Вы делаете нечто потенциально опасное!
```

```
Введите фразу 'Yes, do as I say!' чтобы продолжить.
```

Каждую ситуацию, в которой APT выдаѐт такое сообщение, необходимо рассматривать отдельно. Однако, вероятность того, что после выполнения этой команды система окажется неработоспособной, очень велика.

9.4 Обновление всех установленных пакетов

Для обновления всех установленных пакетов используется команда `apt-get upgrade`. Она позволяет обновить те, и только те установленные пакеты, для которых в репозиториях, перечисленных в `/etc/apt/sources.list`, имеются новые версии; при этом из системы не будут удалены никакие другие пакеты. Этот способ полезен при работе со стабильными пакетами приложений, относительно которых известно, что они при смене версии изменяются несущественно.

Иногда, однако, происходит изменение в именовании пакетов или изменение их зависимостей. Такие ситуации не обрабатываются командой `apt-get upgrade`, в результате чего происходит нарушение целостности системы: появляются неудовлетворѐнные зависимости. Например, переименование пакета MySQL-shared, содержащего динамически загружаемые библиотеки для работы с системой управления базами данных MySQL, в libmysqlclient (отражающая общую тенденцию к наименованию библиотек в дистрибутиве) не приводит к тому, что установка обновлѐнной версии libmysqlclient требует удаления старой версии MySQL-shared. Для разрешения этой проблемы существует режим обновления в масштабе дистрибутива – `apt- get dist-upgrade`.

Для обновления всех установленных пакетов необходимо выполнить команды:

```
# apt-get update
# apt-get dist-upgrade
```

Первая команда (`apt-get update`) обновит индексы пакетов. Вторая команда (`apt-get dist-upgrade`) позволяет обновить только те установленные пакеты, для которых в репозиториях, перечисленных в `/etc/apt/sources.list`, имеются новые версии.

В случае обновления всего дистрибутива APT проведѐт сравнение системы с

репозиторием и удалит устаревшие пакеты, установит новые версии присутствующих в системе пакетов, а также отследит ситуации с переименованиями пакетов или изменения зависимостей между старыми и новыми версиями программ. Всѐ, что потребуется поставить (или удалить) дополнительно к уже имеющемуся в системе, будет указано в отчѐте `apt-get`, которым APT предварит само обновление.

П р и м е ч а н и е . Команда `apt-get dist-upgrade` обновит систему, но ядро ОС не будетобновлено.

9.5 Обновление ядра

Для обновления ядра ОС необходимо выполнить команду:

```
# update-kernel
```

П р и м е ч а н и е . Если индексы сегодня еще не обновлялись перед выполнением команды

`update-kernel` необходимо выполнить команду `apt-get update`.

Команда `update-kernel` обновляет и модули ядра, если в репозитории обновилось что-тоиз модулей без обновления ядра.

Новое ядро загрузится только после перезагрузки системы.

Если с новым ядром что-то пойдѐт не так, вы сможете вернуться к предыдущему варианту, выбрав его в начальном меню загрузчика.

После успешной загрузки на обновленном ядре можно удалить старое, выполнив команду:

```
# remove-old-kernels
```

10 СИСТЕМА РАСПАРАЛЛЕЛИВАНИЯ СБОРКИ

Обеспечивает распределенную сборку пакетов на доступных компьютерных мощностях, что позволяет достигать разумного времени сборки пакетов и, при необходимости, пересборки всего репозитория.

11 ЕДИНАЯ МОДУЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ

На платформе Alterator построены инсталлятор системы и штатный еѐ конфигуратор. В качестве языка описаний интерфейсов используется встроенный интерпретатор Scheme. Alterator позволяет, например, построить на своей основе удобный интерфейс для выполнения наиболее востребованных административных задач: добавление и удаление пользователей, настройка сетевых подключений, просмотр информации о состоянии системы, и т.п.

Важной особенностью является возможность сетевого доступа к такого рода интерфейсу, что позволяет осуществлять администрирование, в том числе и удаленно.

12 ЕДИНАЯ СИСТЕМА УСТАНОВКИ

Единая система установки состоит из модулей, написанных на платформе Alterator. Установка может проходить как из режима Live CD, так и сразу после загрузки с установочного диска или образа.

13 СИСТЕМА ГЕНЕРАЦИИ ДИСТРИБУТИВОВ

Система генерации дистрибутивов использует все преимущества банка пакетов и позволяет получить установочные образы. Для сборки используется утилита mkimage, которая использует для сборки «профиль», представляющий из себя набор файлов Makefile. В результате из пакетов репозитория создается установочный диск CD/DVD. Целостность репозитория и его непротиворечивость позволяют с легкостью генерировать новые образы при необходимости.

13.1 Образы бездисковых станций

Помимо создания установочных образов предусмотрена система для создания образов ОСдля бездисковых станций. Для этого используется утилита `mknfsroot`. Утилита принимает единственный параметр – местоположение профиля. Профиль – это каталог, содержащий следующие файлы:

packages – список пакетов для установки;

- modules – список модулей ядра для сетевых адаптеров;
- pxelinux.cfg – конфигурационный файл для pxelinux;
- autoinstall.scm – сценарий для инсталлятора.

Последний файл содержит инструкции для настройки системы, например:

- настройка системной локали;
- настройка раскладки клавиатуры;
- настройка часового пояса;
- задание пароля администратору.Запуск утилиты:

```
# mknfsroot /etc/mknfsroot/profiles/sample
```

В результате появляется файл `/var/lib/mknfsroot/mknfsroot.tar`, содержащий:

- настроенную систему;
- ядро, initrd, образ загрузчика pxelinux и конфигурационный файл для него.

13.1.1 Установка и удаление образа

Развѐртывание образа осуществляется при помощи утилиты `setupnfsroot`. Данная утили- та принимает два параметра: путь к tar-архиву и целевой каталог.

```
# setupnfsroot /var/lib/mknfsroot/nfsroot.tar
/var/lib/tftpboot
```

Утилита развѐртывает образ и настраивает необходимые точки монтирования. Если к этому моменту в системе уже настроены tftp и dhcp сервера, то можно уже попробовать загрузить бездис- ковую станцию.

Обратная операция осуществляется утилитой removenfsroot.

```
# removenfsroot /var/lib/tftpboot
```

В результате каталог очищается, и точки монтирования удаляются.

13.1.2 Особенности созданного образа

- имя машины (hostname) выставляется по результатам резолвинга еѐ ip-адреса;
- поскольку один и тот же образ используется для загрузки большого количества бездисковых узлов, то для каждого узла создаѐтся персональный каталог `/var`. Это перестраховка, поэтому для конкретного случая созданный автоматом образ желательно подправить;
- при остановке машины сеть не останавливается, так как корневая файловая система – сетевая.

13.1.3 Нестандартное расширение профиля

Работает `mknfsroot` следующим образом:

13.1.3.1 Утилита переключается на псевдопользователя, настроенного так, чтобы работал `hasher`.Две части профиля - общая для всех образов (`/etc/mknfsroot/template`) испецифичная (`/etc/mknfsroot/profiles/*`) – объединяются в один профиль `mkimage`.

13.1.3.2 Запускается `mkimage`.

Меняя содержимое `/etc/mknfsroot/template` можно неограниченно изменять поведе-ние mkfsroot вплоть до того что утилита начнѐт делать iso образы вместо tar-файлов.

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
	измененных	замененных	новых	аннулированных					