

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ
КАЗАХСТАН

Казахский национальный исследовательский технический
университет имени К. И. Сатпаева

Г. Е. Сейдахметова, Б. М. Мустафина

**СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ ДЛЯ ОС
APPLE**

Рекомендовано Научно-методическим советом университета в
качестве учебного пособия

Алматы 2016

УДК 004.45 (075.8)
ББК 32.973.26 – 018.2 я73
С 28

Рецензенты:

Ахметова М.А., канд.техн. наук, доц., АУЭС
Амиргалиев Е.Н., д-р техн. наук, проф.,
Университет им. С. Демиреля
Балгабаева Л.Ш. канд.техн.наук., доц.,
КазННТУ им. К.И.Сатпаева

Печатается по плану издания Министерства образования и науки
Республики Казахстан на 2016 г.

Сейдахметова Г. Е.

С 28 Системное программирование для ОС Apple: Учеб. пособие /
Г. Е. Сейдахметова, Б. М., Мустафина. – Алматы: КазННТУ
им. К. И. Сатпаева, 2016. – 128 с. – Ил. 5. Библиогр. – 5 назв.

ISBN 978–601–228–958–9

Учебное пособие содержит материал по разделам основного курса «Системное программирование MOSA», читаемого авторами для специальностей 5B060200 «Информатика», 5B070400 «Вычислительная техника и программное обеспечение» в КазННТУ имени К.И. Сатпаева.

Системное программирование охватывает огромное количество направлений разработки системного ПО. Пособие содержит описания компонентов и технологий операционных систем Apple, инструментов и сред разработки приложений для OS X и iOS.

Предназначено для студентов, обучающихся по специальностям 5B060200 «Информатика», 5B070400 «Вычислительная техника и программное обеспечение» КазННТУ имени К.И. Сатпаева, а также для всех, кто желает освоить курс самостоятельно.

УДК 004.45 (075.8)
ББК 32.973.26 – 018.2 я73

ISBN 978–601–228–958–9

© Сейдахметова Г. Е., Мустафина Б. М., 2016

© КазННТУ им. К. И. Сатпаева, 2016

ВВЕДЕНИЕ

Системное программирование рассматривает темы, связанные с компонентами и технологиями операционной системы, а также включает инструменты и среды программирования для реализации этих компонентов и технологий. Разработка программного обеспечения для операционных систем Apple стало модным и востребованным явлением в области современного программирования и разработки ПО. Системное программирование охватывает огромное количество направлений разработки системного ПО. Это компоненты операционной системы OS X (управление процессами, потоками, памятью, файловой системой, сетевыми возможностями, устройствами ввода-вывода, безопасностью системы и т.д.), разработка приложений для iOS (приложений для iPhone, iPod touch, iPad устройств) и т.д.

Xcode – основной инструмент разработки приложений для OS X и iOS, разработанный компанией Apple. И на сегодняшний день является единственным вариантом для написания программ под iOS с последующим размещением продукта в App Store.

Программирование для операционных систем iOS и OS X предполагает изучение и применение языка Objective-C, Swift.

Основные темы, связанные с программированием мобильных устройств iPhone и iPad с помощью языка Objective-C, Swift, – блоки, расширения класса, наследование, композиция, инициализация объектов, категории, протоколы, управление памятью и организация исходных файлов. Также рассматриваются механизм управления памятью ARC и механизм Grand Central Dispatch, подробно описывается использование библиотек UIKit и APKit.

Глава1

ТЕХНОЛОГИИ MAC

Операционная система OS X сочетает в себе стабильное ядро с передовыми технологиями. ОС OS X имеет многоуровневую архитектуру с ключевыми технологиями в каждом слое [3]. К ним относятся:

- Cocoa (Application)
- Media
- Core Services
- Core OS
- Kernel and Device Drivers

Слой Cocoa (Application) включает в себя технологии для создания пользовательского интерфейса приложения, для реагирования на события пользователя, и для управления поведением приложения.

Мультимедийный слой (Media) включает в себя специализированные технологии для воспроизведения, записи и редактирования информации аудиовизуальных средств и для рендеринга и анимации 2D и 3D графики.

Основной слой сервисов (Core Services) содержит много основных услуг и технологий, которые варьируются от автоматического подсчета ссылок и низкого уровня коммуникационной сети до обработки строк и форматирования данных.

Слой ядра ОС (Core OS) определяет интерфейсы программирования, которые относятся к оборудованию и сетям, в том числе интерфейсы для выполнения высокопроизводительных вычислений на процессорах CPU и GPU компьютера.

Слой ядра и драйверов устройств (Kernel and Device Drivers) состоит из окружения ядра Mach, драйверов устройств, (libSystem) функций библиотеки BSD, и других низкоуровневых компонентов. Слой включает поддержку файловых систем, сетевых технологий, безопасности, межпроцессного взаимодействия, языков программирования, драйверов устройств и расширений ядра.

1.1. Виды программного обеспечения для Mac

С помощью инструментов разработчика и системных фреймворков, можно разработать широкий ассортимент программного обеспечения для Mac, включая следующие:

Приложения (Apps). Приложения помогают пользователям решать задачи, которые варьируются от создания контента и управления данными до общения с другими людьми. ОС OS X обеспечивает изобилие системных технологий, которые можно использовать для расширения возможностей вашего приложения и повышения опыта пользователей.

Фреймворки и библиотеки (Frameworks and libraries). Библиотеки и фреймворки позволяют приложениям совместно использовать код. Frameworks - это особый тип пакета используется для распределения общих ресурсов, включая код библиотеки, файлы ресурсов, файлы заголовков и справочной документации. Apple использует фреймворки для распространения открытых интерфейсов OS X (и iOS), которые упакованы в наборы средств разработки программного обеспечения. Комплект разработки программного обеспечения (SDK) собирает фреймворки, файлы заголовков, инструменты и другие ресурсы, необходимые для разработки программного обеспечения, ориентированные на конкретную версию платформы.

Инструменты командной строки и демоны (Command-line tools and daemons). Инструменты командной строки позволяют работать с данными в среде командной строки приложения Терминал. Демоны запускаются и выступают в качестве сервера для обработки запросов клиента.

Приложения плагинов и загружаемых пакетов (App plug-ins and loadable bundles). Плагины расширяют возможности других приложений; пакеты содержат код и ресурсы, которые приложения могут динамически загружать во время выполнения.

Системные плагины (System plug-ins). Системные плагины, такие как элементы аудио, расширения ядра, комплекты ввода-вывода драйверов устройств, панели настроек, импортеры Spotlight, и экранные заставки, расширяют возможности системы.

1.2. Слой приложений Cocoa

Слой приложений Cocoa в первую очередь отвечает за внешний вид приложений и их реагирование на действия пользователя. Термин Aqua относится к общему внешнему виду и поведению OS X, который характеризуется последовательным, удобным поведением в сочетании с мастерским использованием макета, цвета и текстуры. Рекомендации, приведенные в *OS X Human Interface Guidelines* позволяют создавать красивые, привлекательные приложения.

На уровне Cocoa реализованы **высоко-уровневые возможности**: Notification Center, Game Center, Sharing (пользовательский интерфейс для обмена контентом среди многих видов услуг), Resume (автоматическое возобновление приложения), Automatic and Sudden Termination Apps (Автоматическое завершение работы приложения), User Interface Preservation (Сохранение пользовательского интерфейса), Auto Save, Full-Screen Mode, Cocoa Auto Layout (используется для размещения элементов пользовательского интерфейса), Popover (вид, который отображает дополнительные данные, имеющие отношение к существующему контенту на экране), Software Configuration, Accessibility (OC OS X обеспечивает множество встроенных функций и технологий, которые помогают пользователям с особыми потребностями), AppleScript (основной язык для создания приложений с поддержкой сценариев), Spotlight (предоставляет расширенные возможности поиска приложений), Ink Services (обеспечивает распознавание рукописного ввода для приложений).

Frameworks. Слой Cocoa (Application) включает фреймворки: Cocoa Umbrella Framework, AppKit, Game Kit, Preference Panes, Screen Saver, Security Interface.

Cocoa umbrella framework (Cocoa.framework) импортирует основные фреймворки Objective-C для разработки приложений: AppKit, Foundation и Core Data.

AppKit (AppKit.framework). AppKit. AppKit является ключевым фреймворком для приложений Cocoa. Классы фреймворка AppKit реализуют пользовательский интерфейс

(UI) приложения, включая окна, диалоговые окна, элементы управления, меню, обработка событий. Они также обрабатывают большую часть поведения приложения, включая управление меню, окнами, документами, открытие и сохранение диалогов, и поведения буфера обмена (Clipboard).

Foundation (Foundation.framework). Классы Foundation фреймворка (который находится в слое Core Services), реализуют управление данными, доступ к файлам, уведомление процесса, сетевое взаимодействие и другие низкоуровневые функции. AppKit имеет прямую зависимость от Foundation, поскольку многие ее методы и функции принимают экземпляры классов в качестве параметров, или возвращает экземпляры в качестве значений.

Core Data (CoreData.framework). Классы фреймворка Core Data (находится в слое Core Services) управляют моделью данных приложений на основе шаблона проектирования Model-View-Controller (Модель-Представление-Контроллер).

Информацию о платформе AppKit см. *Application Kit Framework Reference, Mac App Programming Guide*

Game Kit. Game Kit framework (GameKit.framework) предоставляет API, которые позволяют вашему приложению участвовать в Game Center (см. *Game Kit Framework Reference*)

Preference Panes. Preference Panes framework (PreferencePanes.framework) позволяет создавать плагины, содержащие пользовательский интерфейс для настройки предпочтений приложения (см. *Preference Pane Programming Guide*).

Screen Saver. Screen Saver framework (ScreenSaver.framework) содержит классы для создания динамически подгружаемые пакеты, реализующие заставки.

Security Interface. Security Interface framework (SecurityInterface.framework) содержит классы, обеспечивающие элементы пользовательского интерфейса для программ, реализующих функции безопасности, такие как авторизация, доступ к цифровым сертификатам и доступ к элементам в keychains файлах (см. *Security Interface Framework Reference*).

1.3. Media Layer

Технологии Media layer позволяют включить 2D и 3D графику, анимацию, эффекты изображения, и профессиональную аудио и видео функциональность в ваше приложение. Media слой включает AV Foundation, Core Animation, Core Audio, Core Image, Core Text, OpenAL, OpenGL, Quartz.

Технологии Графики. Отличительным качеством любого приложения OS X является высокое качество графики пользовательского интерфейса. Самый эффективный способ обеспечения высокого качества графики в пользовательском приложении является использование стандартных представлений и руководств платформы AppKit (см. Media Layer Frameworks).

Графика и рисунок. ОС OS X предлагает несколько системных технологий для графики и рисования.

Сocoa Рисование. Платформа AppKit предоставляет объектно-ориентированный интерфейс для многих функций, найденных в Quartz 2D. Сocoa обеспечивает поддержку для рисования примитивных фигур, таких как линии, прямоугольники, овалы, дуги и кривые Безье, поддерживает рисование в стандартных и пользовательских цветовых пространствах и поддерживает манипуляции контента, используя графические преобразования.

Другие платформы для графики и рисования. В дополнение к AppKit есть несколько других важных структур для графики и рисования:

- Core Graphics (CoreGraphics.framework)
- Core Animation
- Scene Kit (SceneKit.framework)
- OpenGL (OpenGL.framework)
- GLKit (GLKit.framework)

Текст, Типографика и шрифты. ОС OS X обеспечивает расширенную поддержку для расширенной типографики для приложений Сocoa.

- Cocoa text system (см. *Cocoa Text Architecture Guide*)
- Core Text (CoreText.framework).

Images. AppKit и Quartz позволяет создавать объекты,

которые представляют образы (NSImage и CGImageRef) из различных источников, рисовать эти образы до соответствующего графического контекста. Помимо стандартных возможностей AppKit и Core Graphics, вы можете делать другие вещи с изображениями, используя следующие фреймворки:

- Image Capture Core (ImageCaptureCore.framework).
- Core Image.
- Image Kit.
- Image I/O (ImageIO.framework)

Color Management. ColorSync является системой управления цветом в OS X

Printing. ОС OS X реализует поддержку печати через набор API и сервисов системы, которые доступны для всех приложений среды.

Аудио Технологии. ОС OS X включает поддержку для качественной записи, синтеза, обработки и воспроизведения звука. AV Foundation framework предлагает интерфейсы, которые можно использовать:

- AV Foundation (AVFoundation.framework)
- OpenAL (OpenAL.framework).
- Core Audio (CoreAudio.framework).

Video Technologies. OS X обеспечивает несколько технологий для воспроизведения видео-контента:

- AV Foundation (AVFoundation.framework).
- Core Media (CoreMedia.framework).
- Core Video (CoreVideo.framework).

Media Layer Frameworks. ОС OS X включает множество технологий для рендеринга и анимации 2D и 3D-контента (включая текст) и для воспроизведения, записи и редактирования аудиовизуальных сред и содержит фреймворки:

- Application Services umbrella framework (ApplicationServices.framework)
- AV Foundation framework (AVFoundation.framework)
- ColorSync framework (ColorSync.framework)
- Open Audio Library (OpenAL.framework)
- OpenGL framework (OpenGL.framework)
- Quartz Core framework (QuartzCore.framework)

1.4. Core Services Layer

Слой основных сервисов включает Address Book, Core Data, Core Foundation, Foundation, Quick Look, Social, Security, WebKit

Интеграция Социальных Медиа:

- Accounts (Accounts.framework).
- Social (Social.framework).

iCloud Storage. Существует три типа хранилища iCloud:

- Document storage.
- Key-value storage.
- Core Data storage.

Automatic Reference Counting. Автоматический подсчет ссылок (ARC) - это компилятор уровня функций, который упрощает процесс управления жизнью Objective-C объектов. ARC освобождает от необходимости помнить когда нужно сохранить или освободить объект, потому что он оценивает потребности существования объектов и автоматически вставляет соответствующие вызовы в методы во время компиляции (см. *What's New in Xcode и Transitioning to ARC Release Notes*).

File Coordination. NSFileCoordinator class, NSFilePresenter protocol, NSFileManager method и др.

Bundles and Packages. (см. Bundle Programming Guide)

Internationalization and Localization (см. Data Formatting Guide for Core Foundation)

Block Objects. (см. Blocks Programming Topics)

Grand Central Dispatch (GCD). GCD обеспечивает простой и эффективный API для реализации параллельного выполнения кода в приложении. GCD предоставляет инфраструктуру для выполнения любой задачи в приложения асинхронно, через очередь отправки. Очереди отправки собирают свои задачи и работают с ядром, чтобы облегчить их выполнение в основном потоке. Одной очередью диспетчер может выполнять задачи последовательно или параллельно, и приложения могут иметь несколько очередей чтобы направить на выполнение задачи в параллельном режиме (см. Concurrency Programming Guide).

Источники отправки предоставляют возможность обрабатывать следующие типы ядра на уровне событий, что

является более эффективным, чем BSD альтернативы:

- уведомлений о таймере
- обработка сигнала
- события, связанные с операциями над файлами и сокетами
- важные события, связанные с процессом,
- события, связанные с Mach
- настраиваемые события, которые вы определяете и триггер

Bonjour является мощной системой для публикации и обнаружения сервисов по IP-сети.

Security Services. Безопасность OS X построена на нескольких открытых технологиях, в том числе BSD и Kerberos добавляет свои особенности к этим технологиям. Security framework (Security.framework) – эти высокоуровневые сервисы предоставляют удобные абстракции и делают его возможным для Apple и третьих сторон, для реализации новых функций безопасности, не нарушая ваш код.

ОС OS X обеспечивает высокоуровневые интерфейсы для следующих функций:

- User authentication
- Certificate, Key, and Trust Services
- Authorization Services
- Secure Transport
- Keychain Services

Поддержка Time Machine. Time Machine защищает пользовательские данные от случайной потери за счет автоматического резервного копирования данных на другой жесткий диск.

Keychain Services. Keychain Services обеспечивает более безопасный способ хранения паролей, ключей, сертификатов и другой конфиденциальной информации, связанной с пользователем. Пользователям часто приходится управлять несколькими Идентификаторами пользователей и паролями для доступа к различным логинам учетных записей, серверов, защищенных веб-сайтов, обменом мгновенными сообщениями услуг и так далее. Keychain – это зашифрованный контейнер,

который содержит пароли для нескольких приложений и безопасных услуг. Доступ к связке ключей осуществляется через один мастер-пароль (см. Keychain Services Programming Guide).

Core Service Frameworks. ОС OS X включает несколько основных услуг, которые упрощают разработку приложений. Эти технологии варьируются от утилиты для управления внутренними структурами данных высокого уровня, для распознавания речи и доступа к данным календаря.

Core Services Umbrella Framework включают следующие фреймворки:

- Launch Services (LaunchServices.framework) (см. Launch Services Programming Guide).
- Metadata (Metadata.framework) (см. Spotlight Importer Programming Guide и File Metadata Search Programming Guide).
- Search Kit (SearchKit.framework) (см. Search Kit Reference).
- Web Services Core (WebServicesCore.framework) (см. Web Services Core Programming Guide).
- Dictionary Services (DictionaryServices.framework) (см. Dictionary Services Programming Guide).

Accounts. Accounts framework (Accounts.framework) обеспечивает единый вход в модель для поддерживаемых типов учетных записей, таких как Twitter и Facebook.

Core Data. Core Data framework (CoreData.framework) управляет моделью данных приложения с точки зрения архитектуры Модель-Представление-Контроллер шаблона проектирования.

1.5. Core OS Layer

Технологии и платформы уровня ядра ОС обеспечивают низко-уровневые сервисы по оборудованию и сетям. Эти услуги основаны на возможностях уровня ядра и драйверов устройств. Уровень ядра ОС включает Accelerate, Directory Services, Disk Arbitration, OpenCL, System Configuration.

Уровень Core OS реализует **высоко-уровневые возможности**, связанные с безопасностью приложения.

Gatekeeper. Gatekeeper (сторож), позволяет пользователям

блокировать установку программного обеспечения, которое приходит не из Mac App Store и идентифицированных разработчиков. Если ваше приложение не подписано сертификатом ID разработчика выпущенным компанией Apple, оно не будет запускаться на системах, у которых выбрана эта опция безопасности.

App Sandbox. Приложение **Sandbox** обеспечивает последнюю линию обороны против украденных, поврежденных или удаленных данных пользователя, если вредоносный код использует ваше приложение. Приложение песочница также уменьшает ущерб от ошибок кодирования. Приложения Sandbox позволяет описать, как ваше приложение взаимодействует с системой. Затем система предоставляет вашему приложению доступ только к тем ресурсам, с которыми он должен работать, и не более. А также позволяет пользователю прозрачно предоставлять вашему приложению доступ к дополнительным возможностям с помощью открытия и сохранения файлов, перетаскивания и других привычных действий пользователей.

Code Signing (Подписывание Кода). ОС OS X использует технологии безопасности, известные как подписывание кода, чтобы можно было подтвердить, что ваше приложение действительно создано вами. После того, как приложение будет подписано кодом, система может обнаружить любое изменение в приложении — внесено ли изменение случайно или вредоносным кодом.

Core OS Frameworks. Следующие технологии и фреймворки относятся к уровню Core OS в OS X:

- Accelerate
- Disk Arbitration
- OpenCL
- Open Directory (Directory Services)
- System Configuration

Accelerate framework (Accelerate.framework) содержит API, которые могут ускорить сложные операции и повысить производительность.

Disk Arbitration framework (Disk Arbitration.framework) уведомит ваше приложение, когда локальные и удаленные тома монтируются и размонтируются. Он также предоставляет

другие обновления о статусе удаленного и локального мониторинга и возвращает информацию о подключенных томах.

OpenCL. Open Computing Language (OpenCL) предоставляет высокопроизводительные параллельные вычислительные мощности графических процессоров, делая их доступными для вычислений общего назначения. Язык OpenCL является универсальным компьютерным языком, а не специальным графическим языком, чтобы абстрагировать низко-уровневые детали, необходимые для выполнения параллельной обработки данных вычислительных задач на GPU и CPU процессорах (см. OpenCL Programming Guide for Mac).

Open directory (directory services) -это архитектура служб каталогов, которая обеспечивает централизованный способ извлечения данных, хранящихся в локальных или сетевых базах данных. Службы каталогов, как правило, обеспечивают доступ к собираемой информации о пользователях, группах, компьютерах, принтерах и другой информации, которая существует в сетевой среде. Вы используете Open Directory для получения информации из этих локальных или сетевых баз данных.

System Configuration (SystemConfiguration.framework) - это платформа, которая помогает приложениям настраивать сети и определять, могут ли сети подключиться к ним.

1.6. Kernel and Device Drivers Layer (Слой ядра и драйверов устройств)

Самый нижний слой OS X включает ядро, драйверы и элементы системы BSD и в основном базируется на технологиях с открытым исходным кодом. Слой ядра и драйверов устройств включает BSD, File System, Mach, Networking.

1.6.1. Высокоуровневые возможности

XPC Interprocess Communication and Services. XPC является технологией межпроцессного взаимодействия в OS X, которая дополняет приложение Sandbox (песочница), позволяя разграничение полномочий. Разграничение полномочий - это стратегия развития, в которой вы делите приложение на части в

зависимости от доступа к системным ресурсам, которые требуются каждой части. Составные части, которые вы создаете, называются XPC сервисы. Каждая служба получает свою собственную песочницу, в частности, она получает свой собственный контейнер и собственный набор прав. Кроме того, XPC сервис, который вы включите в ваше приложение доступен только вашему приложению. Эти преимущества позволили XPC стать самой лучшей технологией для осуществления разделения полномочий в приложениях OS X.

XPC интегрирована с Grand Central Dispatch (GCD). Когда вы создаете соединение, вы связываете его с очередью диспетчера, на котором выполняется трафик сообщений.

Когда приложение запускается, система автоматически регистрирует каждую XPC услугу, которую находит в пространстве имен видимом для приложения. Приложение устанавливает соединение с одним из его XPC сервисов, и отправляет сообщения, содержащие события, которые затем сервис обрабатывает (см. *Creating XPC Services in Daemons and Services Programming Guide*, *App Sandbox Design Guide*).

Caching API. libcache API является низкоуровневым API очищаемого кэширования. Агрессивное кэширование - это важный метод повышения эффективности приложения. Однако, когда требования кэширования превышают объем доступной памяти, система должна освободить память по мере необходимости, чтобы справиться с новыми требованиями. Как правило, это означает подкачку кэшированных данных и от относительно медленных устройств хранения, иногда даже приводящие к снижению системной производительности. Ваше приложение должно избегать возможных накладных подкачек, активно управляя своими кэшируемыми данными, освобождая их, как только перестанет нуждаться в кэшированных данных.

In-Kernel Video Capture. I/O Video обеспечивает на уровне ядра интерфейс C++ программирования для написания драйверов устройств видео захвата.

1.6.2. Kernel (Ядро)

Основа системы OS X - POSIX-совместимая операционная система Darwin, являющаяся свободным программным обеспечением. Её ядром является XNU, в котором используется

микроядро Mach и стандартные сервисы BSD.

В OS X (как и в любой UNIX-системе) используется вытесняющая многозадачность и защита памяти, позволяющие запускать несколько изолированных друг от друга процессов, каждый из которых не может прервать или модифицировать остальные.

В следующих разделах описаны некоторые из ключевых особенностей ядра и части Darwin.

Mach. Mach находится в центре Darwin, т.к. обеспечивает самые важные функции операционной системы. Mach управляет ресурсами процессора, такими как использование процессора и памяти, обрабатывает планирования, осуществляет защиту памяти и реализует инфраструктуру, ориентированную на обмен сообщениями для нетипизированного межпроцессного взаимодействия. Mach предоставляет следующие важные преимущества Mac компьютеров:

Protected memory (Защита памяти). Mach гарантирует, что приложение не может писать в память другого приложения или в память операционной системы.

Preemptive multitasking (Вытесняющая многозадачность). Mach эффективно разделяет процессор между процессами. Mach следит за процессором компьютера, определяет приоритеты задач, задавая активность на максимуме, гарантирует, что каждая задача получает необходимые ему ресурсы.

Advanced virtual memory (Усовершенствованная виртуальной памяти). Система виртуальной памяти Mach выделяет каждому процессу собственное виртуальное адресное пространства. Для 64-разрядных приложений теоретический максимум составляет около 18 экзабайт или 18 биллион биллионов байт. Mach поддерживает карты адресов, которые управляют трансляцией виртуальных адресов задачи в физическую память. Как правило, только часть данных или кода, содержащаяся в задаче виртуального адресного пространства находится в физической памяти в любой момент времени. Как только страницы потребуются, они загружаются в физическую память из хранилища. Mach дополняет эти семантики абстракциями объектов памяти. Именованные объекты памяти дают возможность одной задаче (на достаточно

низком уровне) отобразить диапазон памяти, отменить его, и передать его другой задаче. Эта возможность имеет существенное значение для осуществления отдельных сред выполнения в той же системе.

Real-time support (Поддержка в режиме реального времени). Эта функция гарантирует низкие задержки доступа к ресурсам процессора для критичных по времени сред приложений. Mach также предоставляет кооперативную многозадачность, вытесняющую поточность и кооперативную поточность.

64-Bit Kernel. Как в v10.8, OS X требует чтобы Mac, использовал 64-битное ядро. 64-разрядное ядро обеспечивает несколько преимуществ:

- Ядро может поддерживать более эффективно конфигурации с большим объемом памяти.
- Максимальный размер буферного кэша увеличивается, что повышает производительность ввода/вывода.
- Повышение производительности при работе со специализированным сетевым оборудованием имитирует отображение памяти через провод или с несколькими видеокартами, содержащими более 2 ГБ видеопамати.

Потому что 64-битное ядро не поддерживает 32-битные драйверы и расширения ядра (KEXTs), эти элементы должны быть построены как 64-разрядные (см. 64-Bit Transition Guide).

Device-Driver Support (Поддержка Драйверов Устройств). Darwin предлагает объектно-ориентированный фреймворк для разработки драйверов устройств под названием I/O Kit framework (Платформа комплекта ввода-вывода). Эта структура облегчает разработку драйверов для OS X и обеспечивает большую часть инфраструктуры, в которой они нуждаются. Написано ограниченное подмножество на C++ и предназначено для поддержки ряда семейств устройств ввода-вывода, I/O Kit является модульным и расширяемым.

Драйверы устройств, созданных с помощью I/O Kit приобретают несколько важных особенностей:

- Истинное plug and play (подключи и играй)
- Динамическое управления устройством (“hot plugging” - “горячее подключение”)

- Управление питанием (для настольных компьютеров и ноутбуков)

Если ваше устройство соответствует стандартным спецификациям - например, мышки, клавиатуры, звуковые устройства ввода, современные MIDI-устройства, и так далее - оно должно работать, только когда вы подключите его. Если ваше устройство не соответствует опубликованному стандарту, можно использовать ресурсы I/O Kit для создания пользовательского драйвера, чтобы удовлетворить ваши потребности. Устройства, таких как карты AGP, PCI и PCIe карты, сканеры, и принтеры обычно требуют специальных драйверов или другого программного обеспечения, поддерживаемого для работы с OS X (см. IOKit Device Driver Design Guidelines).

Network Kernel Extensions (Сетевые Расширения Ядра).

Darwin позволяет разработчикам ядра добавить сетевые возможности операционной системы путем создания сетевых расширений ядра (NKEs). Возможности NKE позволяют создавать сетевые модули и даже целые стеки протоколов, которые могут быть динамически загружены в ядро и выгружены из него. NKEs также позволяют автоматически настроить стеки протоколов.

NKE модули имеют встроенные возможности для мониторинга и изменения сетевого трафика. На уровнях канала передачи данных и сетевого, они также могут получать уведомления об асинхронных событиях от драйверов устройств, такие как изменение в статусе сетевого интерфейса (см. Network Kernel Extensions Programming Guide).

1.6.3. BSD

Интегрированный с Darwin - это настраиваемая версия дистрибутива Berkeley Software Distribution (BSD) операционной системы. Реализация BSD в Darwin включает в себя многие из POSIX API, которые высокоуровневые приложения также могут использовать для реализации основных функций приложения. BSD служит основой для файловых систем и сетевых возможностей OS X. Кроме того, он

предоставляет несколько программных интерфейсов и сервисов, в том числе:

- Модель процесса (IDs - идентификаторы процессов, сигналы и т. д.)
- Основные политики безопасности, такие как разрешения для файлов и идентификаторы пользователей и групп
- Поддержка потоков (POSIX threads)
- Поддержка сети (BSD сокет)

Информацию о демонах, используемых для реализации основных BSD услуг см. *Daemons and Services Programming Guide*. Рассмотрим ключевые возможности BSD слоя в OS X.

IPC and Notification Mechanisms (IPC и Механизмы уведомлений). ОС OS X поддерживает следующие технологии для межпроцессного взаимодействия (IPC) и для доставки уведомлений через систему:

- *File System Events* (События ФС). События файловой системы (FSEvents) - это механизм для уведомления приложения об изменениях в файловой системе, таких как создание, изменение или удаление файлов и каталогов. API FSEvents дают возможность контролировать многие каталоги сразу и выявлять общие изменения в иерархии файлов (см. *File System Events Programming Guide*).

- *Kernel queues and kernel events* (Очереди ядра и события ядра). Эти механизмы позволяют перехватывать события на уровне ядра, чтобы получать уведомления об изменениях в сокетах, процессах, файловой системе, и других аспектах системы. Очереди и события ядра являются частью уровня FreeBSD операционной системы и описана в разделах `target="_self" kqueue/a` и `target="_self" kevent/a` руководства `man`.

- *BSD notifications*. BSD уведомления имеют несколько преимуществ по сравнению с сервисом уведомлений, которые предлагаются в Core Foundation и Foundation. Например, ваша программа может получать BSD уведомления через механизмы, которые включают порты Mach, сигналы, и дескрипторы файлов. Кроме того, эта технология является легкой, эффективной, и способна к объединению уведомлений. Можно добавить поддержку BSD уведомлений в любом типе программ, включая приложения Cocoa (см. *Mac Notification Overview* или

target="_self" notify/a man page. Информацию о Cocoa и интерфейсах Core Foundation для межпроцессных уведомлений см. Distributed Notifications).

- *Sockets and ports.* Сокеты и порты являются портативным механизмом межпроцессного взаимодействия. Сокет представляет собой один конец канала связи между двумя процессами локально или по сети. Порт - это канал между процессами или потоками на локальном компьютере. Core Foundation и Foundation обеспечивают высокоуровневые абстракции для портов и сокетов, что облегчает их реализацию и предлагает дополнительные услуги. Например, вы можете использовать CFSocket с CFRunLoop для мультиплексирования данных, полученных из сокета с данными, полученными из других источников (для получения дополнительной информации см. a target="_self" CFSocket Reference/a и a target="_self" CFRunLoop Reference/a).

- *Streams.* Поток - это механизм для передачи данных между процессами, когда вы общаетесь с помощью установленного механизма транспорта, таких, как Bonjour или HTTP. Высокоуровневые интерфейсы Core Foundation и Foundation (которые работают с CFNetwork) обеспечивает способ на основе потоковой передачи для чтения и записи сетевых данных и может использоваться с циклами запуска, чтобы эффективно функционировать в сетевой среде.

- *Pipes.* Это коммуникационные каналы, которые обычно создаются между родительским и дочерним процессами, когда дочерний процесс разветвляется. Данные, записанные в pipe, буферизуются и читаются в порядке FIFO (first-in, first-out) первым пришел - первым вышел. Можно создать именованные каналы (pipe function) для связи между родственными процессами или именованные каналы для связи между любыми двумя процессами. Именованный канал должен быть создан с уникальным именем, известным как процессу отправителю, так и процессу получателю.

- *Shared memory.* Разделяемая память - это область памяти, которая была выделена процессом, специально для того, чтобы быть читаемой и, возможно, записываемой среди нескольких процессов. Вы можете создать области разделяемой памяти, используя несколько BSD подходов, в том числе функцию

shm_open и процедуры shm_unlink, и через процедуру mmap. Доступ к разделяемой памяти осуществляется через семафоры POSIX, которые реализуют виды блокирующих механизмов. Хотя разделяемая память позволяет любому процессу с соответствующими разрешениями читать или писать в область разделяемой памяти напрямую, она очень уязвима, -- что ведет к опасности повреждения данных и нарушения безопасности -- ее следует использовать с осторожностью. Ее лучше использовать только в качестве хранилища исходных данных (например, элементов изображения или аудио), с контролирующими структурами данных, доступными с помощью более традиционного межпроцессного взаимодействия. Для получения информации о функциях и процедурах shm_open, shm_unlink, и mmap, см. a target="_self" shm_open/a, a target="_self" shm_unlink/a, и a target="_self" mmap/a man pages.

- *Apple event*. События Apple - это высокоуровневое семантическое событие, которое приложение может отправлять себе, другим приложениям на том же компьютере или приложениям на удаленном компьютере. Приложения могут использовать события Apple, чтобы запросить услуги и информация из других приложений. Для предоставления услуг необходимо определить объекты в вашем приложении, которые могут быть доступны с помощью событий Apple, а затем предоставить обработчики событий Apple, чтобы ответить на запросы этих объектов. Несмотря на то, что события Apple не являются технологией BSD, они являются низкоуровневой альтернативой для межпроцессного взаимодействия (см. Apple Events Programming Guide).

Network Support (Поддержка Сети).

ОС OS X-это одна из ведущих платформ для вычислений во взаимосвязанном мире. Он поддерживает доминирующие типы носителей, протоколов и служб в индустрии, а также дифференцированные и инновационные услуги от компании Apple.

Стеки сетевых протоколов OS X основаны на BSD. Расширяемая архитектура обеспечивается сетевыми расширениями ядра, обобщена в Network Kernel Extensions, облегчает создание модулей, реализующих новые или

существующие протоколы, которые могут быть добавлены в этот стек.

File-System Support. Файловая система компонента Darwin основана на расширениях BSD и улучшенной модели Virtual File System (VFS). Компонент ФС включает в себя следующие функции:

- Permissions on removable media. Разрешения на съемный носитель. Эта функция основана на уникальном идентификаторе (globally unique ID), зарегистрированном для каждого подключенного съемного устройства (включая USB- и FireWire-устройства) в системе.

- Access control lists (Списки контроля доступа), которые поддерживают высококачественный доступ к объектам файловой системы.

- URL-based volume mounts. Монтирование тома на основе URL-адреса, которые позволяют пользователям (через команды Finder) монтировать такие объекты, как AppleShare и веб-сервера

- Unified buffer cache. Унифицированный буферный кэш, который консолидирует буферный кэш с кэшем виртуальной памяти

- Long filenames. Длинные имена файлов (до 255 символов или 755 байт, основанные на UTF-8)

- Support for hiding filename extensions on a per-file basis. Поддержка для скрытия расширения имени файла на основе per-file

- Journaling of all file-system types. Ведение журнала всех типов файловых систем, чтобы помочь в восстановлении данных после сбоя.

Из-за многочисленных сред приложений и различных видов устройств, которые ФС поддерживает, OS X обрабатывает файл данных во многих стандартных форматах тома.

Security. Корни OS X в операционной системе Unix обеспечивают надежную и безопасную компьютерную среду, чей послужной список насчитывает многие десятилетия. Сервисы безопасности OS X построены на BSD со стандартом открытым исходным кодом. BSD-это форма операционной системы Unix, которая обеспечивает базовый уровень безопасности для основных услуг, таких как файл и доступ к сети.

Библиотека CommonCrypto, который является частью libSystem, обеспечивает криптографические алгоритмы.

OS X включает в себя следующие функции безопасности:

- *Adoption of Mandatory Access Control.* Принятие общеобязательного контроля доступа, которая обеспечивает высококачественную безопасность для контроля выполнения процессов на уровне ядра. Эта функция предлагает “песочницы” для приложений, которые позволяют ограничить доступ данного приложения только к тем функциям, которые вы назначите.

- *Support for code signing and installer package signing.* Поддержка цифровой подписи и подписание пакета установщика. Эта функция позволяет системе проверять приложения, используя цифровую подпись и предупредить пользователя, если приложение подделано.

- Поддержка компилятора для укрепления исходного кода с учетом потенциальных угроз безопасности. Эта поддержка включает в себя варианты, чтобы запретить выполнение кода в стеке или других частях памяти, содержащих данные.

- Поддержка для сдачи подозрительных файлов в карантин. Это особенно полезно для разработчиков веб-браузеров или других сетевых приложений, которые получают файлы из неизвестных источников. Система предотвращает доступ к файлам в карантине, если пользователь явно подтверждает, что доступ разрешен (см. Security Overview).

Глава2

ОБЗОР АРХИТЕКТУРЫ ЯДРА

OS X обеспечивает множество преимуществ для пользователей Macintosh и разработчиков. Эти преимущества включают:

- улучшенную надежность и производительность;
- расширенные сетевые функции;
- объектно-ориентированный программный интерфейс системы;
- усиление поддержки отраслевых стандартов.

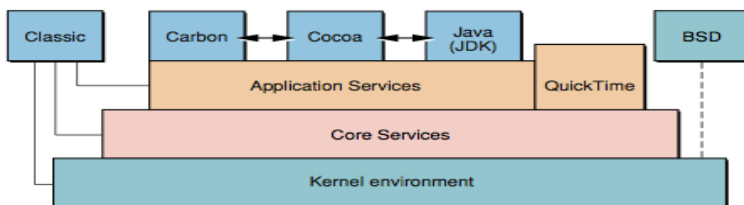


Рис. 2.1. Архитектура OS X

Classic – среда выполнения приложений в операционных системах (Mac OS 8 и Mac OS 9). Современная система Mac OS X не поддерживает приложения Classic в отличие от версий системы Mac OS X, предшествовавших версии Snow Leopard.

Carbon – технология компании Apple, позволяющая разработчикам создавать приложения, которые могут выполняться под управлением Mac OS 9 и Mac OS X под видом “родных” приложений для обеих операционных систем. С учетом того, что производство операционной системы Mac OS 9 прекращено еще 10 лет назад, лишь несколько разработчиков продолжают создавать приложения по технологии Carbon.

Cocoa – приложения, специально разработанные для системы Mac OS X. Они не совместимы с более старыми операционными системами для компьютеров Macintosh, такими как Mac OS 9. Приложения Cocoa обладают всеми современными возможностями, предоставляемыми системой Mac OS X, такими как расширенная память, вытесняющая многозадачность и интерфейс Aqua.

Java (JDK) – язык программирования, разработанный компанией Sun Microsystems. Язык Java является платформенно-независимым, что позволяет приложениям функционировать на любой платформе, на которой установлена Java-машина.

BSD (Berkeley Software Distribution) – существует много разных версий или стилей операционной системы Unix, включая дистрибутивы Linux, Solaris и AIX. “BSD” – это название стиля системы Unix, разработанного в Калифорнийском университете в Беркли (University of California at Berkeley). Система Mac OS X основана именно на этой версии системы Unix.

QuickTime – лицензионный кроссплатформенный механизм авторизации и распространения компании Apple. Термин “QuickTime” относится и к формату файла, и к набору приложений. В системе Mac OS X Snow Leopard и более поздних версиях соответствующий механизм называется *QuickTime X*.

При создании OS X Apple полностью модернизировала суть операционной системы Mac OS, лежащей в основе ядра OS X. Рис. 2.1 иллюстрирует архитектуру OS X.

Ядро предоставляет множество улучшений для OS X: preemption – вытесняющее планирование, защиту памяти, повышение производительности, улучшение сетевых средств, поддержку как для Macintosh (расширенный и стандартный), так и для не Macintosh (UFS, ISO 9660 и т. д.) файловых систем, объектно-ориентированных API и мн. др. Две из этих возможностей, вытесняющее планирование и защита памяти, приводят к более надежной среде.

В Mac OS 9 приложения сотрудничают в сфере разделения процессорного времени. Аналогичным образом все приложения разделяют память компьютера между собой. Mac OS 9 является кооперативной многозадачной средой. Взаимодействие всех процессов находится под угрозой, если хотя бы одно приложение не будет сотрудничать. С другой стороны, приложения реального времени, такие как мультимедийные, должны быть уверены в предсказуемости, критичности во времени, поведении системы.

В противоположность этому, OS X является вытесняющей многозадачной средой. В OS X, ядро обеспечивает сотрудничество, планирование процессов для разделения

времени (преимущественного). Это поддерживается в реальном времени поведением в приложениях, которые требуют этого.

В OS X процессы обычно не разделяют память. Вместо этого ядро присваивает каждому процессу собственное адресное пространство, контролируя доступ к этим адресным пространствам. Такой контроль гарантирует, что ни одно приложение не может случайно получить доступ или изменить память другого приложения (защита). Размер не является проблемой; с виртуальной памятью включенное в OS X каждое приложение имеет доступ к собственному 4 ГБ адресному пространству.

Все приложения работают в пространстве пользователя, но это не означает, что они разделяют память. Пользовательское пространство просто термин для комбинированных адресных пространств для всех приложений пользовательского уровня. Само ядро имеет свое собственное адресное пространство, называемое *пространством ядра*. В OS X ни одно приложение не может напрямую изменять память системного программного обеспечения (ядра).

Хотя пользовательские процессы не разделяют памяти по умолчанию, как в Mac OS 9, коммуникации (и даже совместное использование памяти) между приложениями все же возможны. Например, ядро предлагает богатый набор примитивов, чтобы позволить некоторый обмен информацией между процессами. Эти примитивы включают разделяемые библиотеки, структуры и разделяемую память POSIX. Обмен сообщениями Mach обеспечивает другой подход, передавая память от одного процесса к другому. В отличие от Mac OS 9, однако, совместное использование памяти не может происходить без явных действий со стороны программиста.

2.1. Darwin

Ядро OS X является проектом с открытым исходным кодом. Ядро вместе с другими частями ядра OS X использует общий термин “*Darwin*”. Darwin является полной операционной системой на основе многих технологий, которые лежат в основе OS X. Тем не менее, Darwin не включает собственную графику

или слои приложений Apple, такие как Quartz, QuickTime, Cocoa, Carbon или OpenGL.

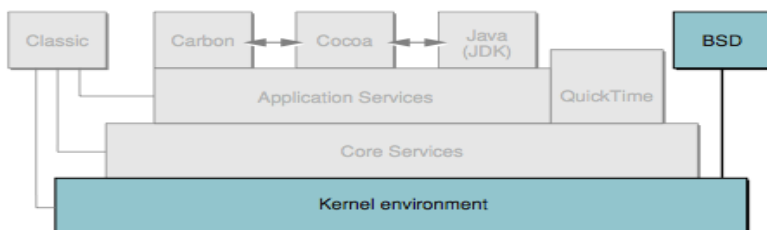


Рис.2.2. Darwin и OS X

На рис. 2.2 показана взаимосвязь между Darwin и OS X. Оба построены на одном и том же ядре, но OS X добавляет CoreServices, ApplicationServices и QuickTime, а также среды разработки приложений, такие как Classic, Carbon, Cocoa и Java (JDK). И Darwin, и OS X включает интерфейс командной строки BSD, однако в OS X использования этой окружающей среды не требуется, и, следовательно, она скрыта от пользователя, если они не выбирают доступ к нему.

Дарвин-технология основана на BSD-, Mach 3.0- и Apple-технологиях. Лучше всего технология “Дарвин” является технологией Open Source, что означает, что разработчики имеют полный доступ к исходному коду. По сути, сторонние разработчики OS X могут быть частью основной команды разработки программного обеспечения системы ядра Дарвина. Разработчики также могут увидеть, как Apple, делает вещи в ядре операционной системы и принимает (или адаптирует) код, чтобы использовать его в своих продуктах. Обратитесь к Apple Public Source License (APSL) для деталей.

Так как, то же программное обеспечение является основой как OS X и Дарвин, разработчики могут создавать низкоуровневое программное обеспечения, которое работает на OS X, как и Дарвин с немногими, если таковы изменения. Единственное различие, вероятно, будет в том, как программное обеспечение взаимодействует со средой разработки.

Дарвин-технология основана на проверенной технологии из многих источников. Большая часть этой технологии происходит

от FreeBSD, версия 4.4BSD, которая предлагает передовые сети, производительность, безопасность и совместимость функций.

Другая часть системного программного обеспечения, таких как Mach, основана на технологии ранее использовавшейся в MkLinux – проекте Apple, в OS X Server и в технологии, приобретенной у фирмы NeXT. Большая часть кода не зависит от платформы. Все коды ядра операционной системы доступны в виде исходного.

Основные технологии были выбраны по нескольким причинам. Mach обеспечивает чистый набор абстракций для работы с управлением памятью, межпроцессные взаимодействия (IPC) и другие низкоуровневые функции операционной системы. В сегодняшней быстро меняющейся аппаратной среде это обеспечивает полезный слой изоляции между операционной системой и базовым оборудованием.

BSD является тщательно спроектированной зрелой операционной системой с большим количеством возможностей. В действительности большинство коммерческих UNIX сегодняшнего дня и UNIX – подобные операционные системы содержат большую часть BSD-кода. BSD также предоставляет набор стандартных API.

2.2. Архитектура

Фундаментальный слой Дарвина и OS X состоит из нескольких архитектурных компонентов (рис. 2.3). Эти компоненты, взятые вместе, формируют среду ядра.

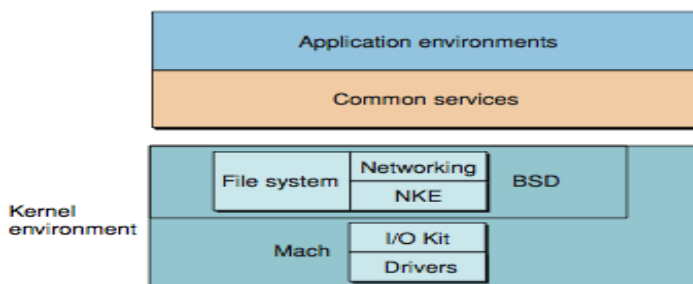


Рис.2.3. Архитектура ядраOSX

“Ядро в традиционной терминологии операционной системы – это небольшая центральная часть программного обеспечения, что обеспечивает только минимальные средства, необходимые для реализации дополнительных служб операционной системы” – от проектирования и реализации 4.4 BSD операционной системы, McKusick, Bostic, Karels и Quarterman, 1996.

Точно так же в традиционных операционных системах на Mach-основе ядро относится к микроядру Mach и игнорирует дополнительный код низкого уровня, без которого Mach очень маленький.

В OS X, однако, окружающая среда ядра содержит гораздо больше, чем само ядро Mach. Среда ядра OS X включает в себя ядро Mach, BSD, в комплект I/O, файловые системы и сетевые компоненты.

Так как OS X содержит три основных компонента (Mach, BSD и комплекте I/O), часто встречаются также целых три API для некоторых ключевых операций.

Mach. Mach управляет ресурсами процессора, такими как использование процессора и памяти, отвечающее за планировку, обеспечивает защиту памяти, и обеспечивает инфраструктуру обмена сообщениями к остальной части операционной системы слоев. Компонент Mach обеспечивает:

- нетипизированные связи между процессами (IPC);
- вызовы удаленных процедур (RPC);
- поддержку планировщика для симметричной многопроцессорной обработки (SMP);
- поддержку услуг в режиме реального времени;
- поддержку виртуальной памяти;
- поддержку пейджеров;
- модульную архитектуру.

BSD. Над слоем Mach, BSD слой обеспечивает API и услуги своей части ОС. Слой BSD основан на ядре BSD, в первую очередь – FreeBSD. Компонент BSD предоставляет:

- файловые системы;
- сети (кроме уровня аппаратного устройства);
- модель безопасности UNIX;

- поддержку системного вызова;
- модель процесса BSD, включая идентификаторы процессов и сигналы;
- API ядра FreeBSD;
- многие из POSIX API;
- поддержку на уровне ядра для Pthreads (POSIXThreads).

Сетевые возможности. OS X сетей использует передовые сетевые возможности BSD, чтобы обеспечить поддержку современных функций, таких как Network Address Translation (NAT) и брандмауэры. Сетевой компонент обеспечивает:

- стек TCP / IP и API сокетов 4.4BSD;
- поддержка как IP и DDP (AppleTalk транспорт);
- множественную адресацию;
- маршрутизацию;
- многоадресную поддержку;
- настройку сервера;
- фильтрацию пакетов;
- поддержку Mac OS Classic (через фильтры).

Файловые системы. OS X обеспечивает поддержку многочисленных типов файловых систем, в том числе HFS, HFS+, UFS, NFS, ISO 9660 и др. Тип файловой системы по умолчанию является HFS+; корни OS X из HFS+, UFS, ISO, NFS и UDF. Расширенные возможности файловых систем OS X включают улучшенную конструкцию виртуальной файловой системы (VFS). VFS обеспечивает многоуровневую архитектуру (файловые системы наращиваемы). Компонент файловая система предоставляет:

- поддержку UTF-8 (Unicode);
- увеличение производительности по сравнению с предыдущими версиями MacOS.

I/O Kit. Комплект ввода/вывода обеспечивает основу для упрощения разработки драйвера, поддерживая многие категории устройств. Объектно-ориентированная архитектура ввода/вывода возможностей комплекта ввода/вывода реализована в ограниченном подмножестве C++. Наборы

комплeкта ввода/вывода являются модульными и расширяемыми. Компонент комплекта ввода/вывода обеспечивает:

- действительное plug and play;
- динамическое управление устройствами;
- динамическую (“по требованию”) загрузку драйверов;
- управление питанием для настольных систем также

хорошо, как и для портативных;

- многопроцессорные возможности.

2.3. Kernel Extensions

OS X обеспечивает механизм расширения ядра как средство, позволяющее динамическую загрузку частей кода в пространство ядра, без необходимости повторной компиляции. Эти части кода известны в общем как плагины или в среде ядра OS X как расширения ядра или KEXTs (kernel extensions).

KEXTs обеспечивают и модульность, и динамическую загружаемость, поэтому являются естественным выбором для любого относительно автономного обслуживания, что требует доступа к интерфейсам, которые не экспортируются в пространство пользователя. Многие из компонентов окружающей среды ядра поддерживают этот механизм расширения при том, что делают это по-разному. Например, некоторые из новых сетевых возможностей предполагают использование расширений сети ядра (NKEs).

Возможность динамически добавлять новую реализацию файловой системы основана на KEXTs виртуальной файловой системы VFS. Драйверы устройств и семейства устройств в комплекте I/O реализованы с использованием KEXTs. KEXTs делают разработку гораздо проще для разработчиков, пишущих драйверы или тех, пишущих коды для поддержки нового формата тома или сетевого протокола.

Глава 3

ОБЗОР MACH

Фундаментальные услуги и примитивы ядра OS X основаны на Mach 3.0. Apple изменила и расширила Mach, чтобы лучше соответствовать функциональным и исполнительным целям OS X.

Mach 3.0 изначально задумывался как простое расширяемое коммуникационное микроядро. Он способен работать как автономное ядро с другими традиционными сервисами операционной системы, такими как ввод/вывод, файловые системы и сетевые стеки, работающие как серверы пользовательского режима. Тем не менее, в OS X Mach связана с другими компонентами ядра в одном адресном пространстве ядра. Это в первую очередь для выполнения, что намного быстрее сделать прямой вызов между связанными компонентами, чем отправка сообщений или сделать удаленный вызов процедур (RPC) между отдельными задачами. Такая модульная структура приводит к более надежной и расширяемой системе, чем монолитное ядро позволило бы без потери производительности для чистого микроядра.

Таким образом, в OS X Mach в первую очередь не коммуникационный узел между клиентами и серверами. Вместо этого его достоинства состоят из его абстракций, его расширяемости и его гибкости. В частности, Mach представляет объектно-ориентированный API с каналами связи (например, порты) как ссылки на объекты высокопараллельного выполнения, в том числе превентивно запланированных потоков и поддержку SMP (symmetrical multiprocessing), гибкую структуру планирования с поддержкой использования режима реального времени, полный набор примитивов IPC, в том числе сообщения, RPC, синхронизацию и уведомления в поддержку большого виртуального адресного пространства, общих областей памяти и объектов памяти при поддержке постоянного хранилища, проверенных расширяемостью и портативностью, например, через архитектуры наборов инструкций и в распределенных средах безопасность и управление ресурсами в качестве основополагающего принципа проектирования. Все ресурсы являются виртуальными.

3.1. Абстракции ядра Mach

Mach представляет небольшой набор абстракций, которые были разработаны, чтобы быть простыми и мощными. Основные абстракции ядра:

Задачи (Task). Элементы ресурса владения; каждая задача состоит из виртуального адресного пространства, пространства имен порта прав (port right namespace) и одного или нескольких потоков (threads). (По аналогии с процессом.)

Потоки (Threads). Элементы, исполняемые CPU внутри задачи.

Адресное пространство. В сочетании с менеджерами памяти Mach реализует понятие виртуального адресного пространства и общей памяти.

Объекты памяти. Внутренние блоки управления памятью. Объекты памяти включают именованные записи и регионы, являющиеся представлениями потенциально постоянных данных, которые могут быть отображены в адресное пространство.

Порты. Безопасные, простые каналы связи, доступны только через возможности отправлять и получать (известный как права порта).

IPC. Очереди сообщений, вызовов удаленных процедур, уведомления, семафоры и наборы блокировок.

Время. Часы, таймеры и ожидание.

На уровне trap (ловушка) интерфейс к большинству Mach-абстракций состоит из сообщений, которые посылаются в и из портов ядра, представляющих эти объекты. Интерфейсы trap уровня (такие как mach_msg_overwrite_trap) и форматы сообщений сами абстрагируются при нормальном использовании генератором интерфейсов Mach (Mach Interface Generator – MIG). MIG используется для компиляции процедурных интерфейсов в API на основе сообщений, основанных на описании этих API.

3.2. Задачи и потоки

OS X процессы и POSIX Threads (Pthreads) реализованы первыми для задач и потоков Mach, соответственно. Поток

является точкой управления потоком в задаче. Задача существует, чтобы предоставить ресурсы для потоков, которые он содержит. Это разделение сделано для обеспечения параллельности и совместного использования ресурсов.

Поток точки управления потоком в задаче имеет доступ ко всем элементам, содержащимся в задаче, выполняет (потенциально) параллельно с другими потоками даже потоки в пределах той же самой задачи, имеет минимальную информацию о состоянии на низких накладных расходах.

Задача – это набор системных ресурсов. На эти ресурсы, за исключением адресного пространства, ссылаются через порты. Эти ресурсы могут использоваться совместно с другими задачами, если права этим портам уже распределены, обеспечивает большое потенциально разбросанное адресное пространство, на которое ссылаются для виртуального адреса. Часть этого пространства может быть передана в порядке наследования или внешнего управления памятью, содержит некоторое количество потоков. Обратите внимание, что у задачи нет своей собственной жизни – только потоки выполняют инструкции. Когда говорят, что “задача Y выполняет X”, на самом деле означает, что “поток, содержащийся в задаче Y, выполняет X”.

Задача довольно дорогой объект. Она существует, чтобы быть набором ресурсов. Все потоки в задаче владеют совместно всем. Две задачи ничего совместно не используют без явных действий (хотя действия часто простые) и некоторые ресурсы (например, порт с правом получать) не могут быть общими для двух задач на всех.

Поток является довольно легким объектом. Это довольно дешево для создания и имеет низкие накладные расходы, чтобы работать. Это верно, поскольку поток имеет мало информации о состоянии (в основном своем регистре состояния). Его задачей владения является управление ресурсами. На компьютере с несколькими процессорами в задаче возможно несколько потоков выполнять параллельно. Даже когда параллелизм не является целью, несколько потоков имеют то преимущество, что каждый поток может использовать синхронный стиль программирования вместо асинхронного с одним потоком, пытающимся обеспечить различные сервисы. Поток является

основным вычислительным объектом. Поток принадлежит одной и только одной задаче, которая определяет ее виртуальное адресное пространство. Чтобы воздействовать на структуру адресного пространства или ссылаться на любые ресурсы, отличные от адресного пространства, поток должен выполнять специальную команду ловушки, которая заставляет ядро выполнять операции от имени потока или отправлять сообщение некоторому агенту от имени потока. В общем, эти ловушки манипулируют ресурсами, связанными с задачей, содержащей поток. Запросы могут быть сделаны из ядра для работы с этими объектами: создавать их, удалять и влиять на их состояние.

Mach обеспечивает гибкую структуру для потока – политики планирования. Ранние версии OS X поддерживают как политики разделения времени, так и с фиксированным приоритетом. Приоритет потока с разделением времени поднимается и опускается, чтобы сбалансировать свое потребление ресурсов по отношению к другим потокам в режиме разделения времени. Потоки с фиксированным приоритетом выполняются за определенный квант времени, а затем помещаются в конец очереди потоков с одинаковым приоритетом. Установка квантового уровня потока с фиксированным приоритетом на бесконечность позволяет потоку работать, пока он не заблокируется, или пока он не будет вытеснен потоком с более высоким приоритетом. Высокий приоритет потоков реального времени, как правило, является фиксированным.

OS X также предусматривает планирование временных ограничений для выполнения в реальном времени. Это планирование позволяет указать, что поток должен получить определенный квант времени в течение определенного периода времени.

3.3. Порты, права портов, наборы порта и пространство имен портов

За исключением виртуального адресного пространства задачи, все другие ресурсы Mach доступны через косвенный уровень, известный как порт. Порт является конечной точкой

однаправленного канала связи между клиентом, который запрашивает услугу, и сервером, оказывающим услугу. Если ответ должен быть предоставлен такому запросу на обслуживание, второй порт должен быть использован. Это сопоставимо с (однаправленный) каналом на языке UNIX.

В большинстве случаев ресурс, доступный через порт (названный им), называется *объектом*. Большинство объектов, названные *портами*, имеют один приемник и (потенциально) нескольких отправителей т. е. есть только один порт приема и по меньшей мере один передающий порт для типичного объекта, такого как очередь сообщений.

Службы, которые будут предоставлены объектом, определяются менеджером, который получает запрос, отправленный объекту. Отсюда следует, что ядро является приемником для порта, связанного с предоставляемыми ядром объектами и, что приемник для портов, связанных с предоставляемыми задачей объектами, является задачей обеспечения этих объектов.

Для портов, которые указывают предоставляемые задачей объекты, можно изменить приемник запросов на этот порт другой задачей, например, путем передачи порта для выполнения этой задачи в сообщении. Одна задача может иметь несколько портов, которые относятся к ресурсам, которые он поддерживает. Если на то пошло, любой объект может иметь несколько портов, которые его представляют, каждый подразумевая различные наборы допустимых операций. Например, многие объекты имеют имя порта и порт управления (иногда называемые *привилегированными портами*). Доступ к порту управления позволяет манипулировать объектом; доступ к имени порта просто называет объект, так что вы можете получить информацию о нем или выполнить другие непривилегированные операции от него.

Задачи имеют разрешения на доступ к портам определенным образом (отправка, получение, отправка один раз), которые называются *правами порта*. Порт может быть доступен только через право. Порты часто используются для предоставления клиентам доступа к объектам внутри Mach. Под правом направить в IPC порт объекта подразумевается право манипулировать объектом предписанным образом. Таким

образом, право собственности порта является основным механизмом безопасности в Mach. Обладание правом на объект должно иметь возможность доступа или манипулирования этим объектом.

Права порта можно копировать и перемещать между задачами через IPC. Это, в сущности, передача возможности некоторому объекту или серверу.

Один тип объекта, на который ссылается порт, является набором портов. Как подсказывает название, набор порта – это набор из прав порта, которые можно рассматривать как единое целое при приеме сообщения или события из любого из элементов набора. Наборы порта позволяют потоку ожидать ряд сообщений и источников событий, например, в работе петель.

Традиционно в Mach канал связи, обозначаемый как порт, всегда был очередью сообщений. Тем не менее, OS X поддерживает дополнительные типы каналов связи, и эти новые типы объектов IPC также представлены как порты и права портов.

Порты и права портов не имеют общесистемных имен, допускающих произвольные порты или права для непосредственного управления. Портами можно манипулировать с помощью задачи только, если задача имеет право порта в его пространстве имен порта. Право порта задается именем порта, числовой индекс в 32-битном пространстве имен порта. Каждая задача имеет связанное с ней единое пространство имен порта.

Задачи получают права порта, когда другая задача явно вставляет их в его пространство имен, когда они получают права в сообщениях, создавая объекты, которые возвращают право на объект, и через вызовы Mach для некоторых специальных портов (`mach_thread_self`, `mach_task_self` и `mach_reply_port`).

3.4. Управление памятью

Как и в большинстве современных операционных систем, Mach обеспечивает адресацию большого разбросанного виртуального адресного пространства. Доступ во время выполнения осуществляется через виртуальные адреса, которые

могут не совпадать с размещением в физической памяти на начальном периоде попыток доступа. Mach несет ответственность за принятие востребованного виртуального адреса, и присваивает ему соответствующее место в физической памяти. Он делает это через подкачки по требованию.

Диапазон виртуального адресного пространства заполняется данными, когда объект памяти отображается в этом диапазоне. Все данные в адресном пространстве в конечном счете доступны через объекты памяти. Мас запрашивает у владельца объекта памяти (пейджера) содержание страницы при ее создании в физической памяти и возвращает возможно измененные данные на пейджер до восстановления страницы. OS X включает в себя два встроенных пейджера – пейджер по умолчанию и vnode-пейджер.

По умолчанию пейджер управляет непостоянной памятью, известной как анонимная память. Анонимная память инициализирована нулями и существует только в течение “жизни” задачи. Vnode пейджер отображает файлы в объекты памяти. Mach экспортирует интерфейс для объектов памяти, чтобы позволить их содержанию быть внесенными задачами пользовательского режима. Этот интерфейс известен как Внешний интерфейс управления памятью, или EMMI.

Подсистема управления памятью экспортирует дескрипторы виртуальной памяти, известные как именованные записи или именованные записи памяти. Как и большинство ресурсов ядра они обозначены портами. Наличие дескриптора именованных записей памяти позволяет владельцу отображать основной объект виртуальной памяти или передавать право отображать основной объект другим. Отображение именованной записи в две различные задачи заканчивается в совместно используемой памяти окном между двумя задачами, тем самым обеспечивая гибкий способ установления совместно используемой памяти.

Начиная с OS X v10.1, система EMMI была расширена для поддержки “portless” EMMI. В традиционной EMMI два Mach порта были созданы для каждой области памяти и аналогично два порта – для каждого кэшированного vnode. Portless EMMI в своей начальной реализации заменяет это ссылками

непосредственной памяти (в основном указатели). В будущих версиях порты будут использоваться для связи с пейджерами вне ядра, используя прямые ссылки для связи с пейджерами, которые находятся в пространстве ядра. Конечный результат этих изменений состоит в том, что ранние версии portless EMMI не поддерживают пейджеры, работающие вне пространства ядра. Эта поддержка, как ожидается, будет восстановлена в будущих версиях.

Диапазоны адресов пространства виртуальной памяти также могут быть заполнены путем прямого распределения (с помощью `vm_allocate`). Основной виртуальный объект памяти является анонимным и поддерживается пейджером по умолчанию. Общие диапазоны адресного пространства могут также быть настроены через наследование. Когда новые задачи созданы, они клонированы от родителя. Это клонирование также принадлежит основному адресному пространству памяти. Отображаемые части объекта могут быть унаследованы как копии, или как совместно используемые, или вообще не на основе атрибутов, связанных с отображениями. Mach практикует формы отсроченных копий, известных как *copy-on-write* (копии при записи) для оптимизации производительности унаследованных копий при создании задачи.

Вместо прямого копирования диапазона *copy-on-write* оптимизация осуществляется путем защищенного разделения. Эти две задачи разделяют память, которая будет скопирована, но с доступом только для чтения. Когда одна задача пытается изменить часть диапазона, эта часть в это время скопирована. Эта оценка копий памяти – важная оптимизация, которая разрешает упрощения в нескольких областях, особенно в API обмена сообщениями.

Еще одна форма обмена обеспечивается Mach за счет экспорта именованных регионов. Именованный регион является одной из форм именованных записей, но вместо того, чтобы быть поддержанным объектом виртуальной памяти, оно поддержано фрагментом виртуального отображения. Этот фрагмент может содержать отображение многочисленных объектов виртуальной памяти. Это отображаемые в другие

виртуальные отображения, обеспечивающие способ наследования не только группы объектов виртуальной памяти, а также их существующих отношений отображения. Данная функция предлагает существенную оптимизацию в установке задачи, например, разделяя сложную область адресного пространства, используемого для общих библиотек.

3.5. Межпроцессное взаимодействие (IPC)

Связь между задачами является важным элементом философии Mach. Mach поддерживает системную структуру клиент/сервер, в которой задачи (клиенты) получают доступ к сервисам, отправляя запросы другим задачам (серверам) через сообщения, посланные по каналу связи.

Конечные точки этих каналов связи в Mach называют *портами*, в то время как права порта обозначают разрешение использовать канал. Формы IPC, предоставляемые Mach, включают:

- очереди сообщений;
- семафоры;
- уведомления;
- наборы блокировок;
- удаленный вызов процедуры (RPC).

Тип объекта IPC, обозначенный портом, определяет допустимые операции на этом порту, и как (и будет ли) происходит передача данных.

Существуют два принципиально различных API Mach для манипуляции портов – семейства `mach_ipc` и семейства `mach_msg`.

В пределах разумного оба семейства могут быть использованы с любым объектом IPC, однако вызовы `mach_ipc` являются предпочтительными в новом коде. Вызовы `mach_ipc` сохраняют информацию о состоянии в случае необходимости в целях поддержки понятия транзакции. Вызовы `mach_msg` поддерживаются для унаследованного кода, но не рекомендуются; они являются не сохраняющими состояния.

3.5.1. Транзакции IPC и диспетчеризация события

Когда поток вызывает `mach_ipc_dispatch`, он неоднократно обрабатывает события, поступающие на зарегистрированный набор портов. Эти события могут быть блоком аргументов от объекта RPC (как результат вызова клиента), принятого объекта блокировки (в результате снятия блокировки некоторого другого потока), отправленного уведомления или семафора, или пришедшего сообщения от традиционных очередей сообщений.

Эти события обработаны через `callouts` (выноски) от `mach_msg_dispatch`. Некоторые события подразумевают транзакции при жизни выноски. В случае блокировки состояние является владельцем блокировки. Когда `callout` возвращается, блокировка снимается. В случае удаленного вызова процедур состояние означает личность клиента, блок аргумента и порт ответа. Когда возвращается `callout`, ответ отправляется.

Когда возвращается `callout`, транзакция (если таковая имеется) будет выполнена, и поток ожидает следующего события. `Mach_ipc_dispatch`-средство предназначено для поддержки работы петель.

3.5.2. Очереди сообщений

Первоначально единственным стилем взаимодействия процессов в Mach был очередь сообщений. Только одна задача может получить право на порт, обозначающий очереди сообщений. Эта задача позволяла получить(читать) сообщения из порта очереди. Несколько задач могут содержать права на порт и позволяют им отправлять (запись) сообщения в очередь.

Задача общается с другой задачей путем создания структуры данных, которая содержит набор элементов данных, и затем выполняет операцию по отправке сообщения в порт, на что она имеет право. Через некоторое время данная задача будет выполнять операцию сообщение – прием.

Сообщение может включать некоторые или все из следующих действий:

- чистые данные;
- копии диапазонов памяти;
- права портов;

– неявные атрибуты ядра, таких как маркера безопасности отправителя.

Передача сообщений является асинхронной операцией. Сообщение логически копируется в получающую задачу, возможно, с оптимизацией копирования при записи. Несколько потоков внутри принимающей задачи могут пытаться получать сообщения от данного порта, но только один поток может получить любое сообщение.

3.5.3. Семафоры

Семафор IPC объектов поддерживает ожидания, размещение и размещение всех операций. Считающие семафоры в этих размещениях сохраняются (учитываются), если нет потоков, ожидающих в очереди ожидания семафора. Размещения всех операций будят все ожидающие потоки.

3.5.4. Уведомления

Как семафоры объекты уведомлений также поддерживают размещения и операции ожидания, но с добавлением поля состояния, состояние фиксированного размера, поля фиксированного формата, которое определяется при создании объекта уведомления. Каждое размещение обновляет поле состояния. Есть одно состояние, которое переписывается после каждого размещения.

3.5.5. Блокировки

Блокировка является объектом, который обеспечивает взаимно исключающий доступ к критической секции. Основной интерфейс для блокировок ориентирован на транзакции. Во время транзакции поток удерживает блокировку. Когда он вернется из транзакции, блокировка снимется.

3.5.6. Объекты удаленного вызова процедур (RPC)

Как следует из названия, объект RPC предназначен для облегчения и оптимизации удаленного вызова процедур.

Основной интерфейс для RPC-объектов ориентирован на транзакции.

Когда объект RPC создается, набор форматов блока аргументов определяется. Когда RPC (пересылка объекта) проводится клиентом, то он вызывает сообщение в одном из предопределенных форматов, которые должны быть созданы и в очереди на объект, и в конечном итоге передается на сервер (приемник). Когда сервер возвращается из транзакции, ответ будет возвращен отправителю. Mach пытается оптимизировать транзакции, выполнив сервер с помощью ресурсов клиента, что называется *миграцией потоков*.

3.6. Тайм-менеджмент

Традиционная абстракция времени в Mach-часы, которые предоставляет набор асинхронных служб сигнализации на основе `mach_timespec_t`. Есть один или более объектов часов, каждый из которых образует монотонно возрастающее значение времени, выраженное в наносекундах. Часы реального времени являются встроенным и самым важным, но могут быть и другие часы для других представлений о времени в системе. Часы поддерживают операции, чтобы получить текущее время, спать в течение данного периода, установить сигнал (уведомление, которое посылается в данный момент времени) и т. д.

Новые и предпочтительные API основаны на объектах таймера, которые в свою очередь используют `AbsoluteTime` в качестве основного типа данных. `AbsoluteTime` является машинно-зависимым типом, обычно основанным на базе времени родной платформы. Процедуры предусмотрены для преобразования `AbsoluteTime` значений и от других типов данных, таких как наносекунды. Объекты Таймера поддерживают асинхронный без дрейфа уведомлений, отмены и преждевременной тревоги. Они более эффективны и обеспечивают высокое разрешение, чем часы.

Глава 4

ПАМЯТЬ И ВИРТУАЛЬНАЯ ПАМЯТЬ

Виртуальная память OS X. Система виртуальной памяти использовала в OS X потомок Mach VM (виртуальная память ядра Mach), который был создан в Университете Карнеги-Меллон в 1980-е годы.

Дизайн виртуальной памяти Mach сосредотачивается вокруг концепции физической памяти, являющейся кэшем для виртуальной памяти. На его высшем уровне виртуальная память Mach состоит из адресных пространств и способов управлять содержанием этих адресных пространств снаружи пространства. Эти адресные пространства разбросаны и имеют понятие защиты, чтобы ограничить, какие задачи могут получить доступ к их содержимому. На более низком уровне, уровне объекта, виртуальная память можно рассматривать как коллекцию объектов виртуальной памяти и объектов памяти, где каждый объект имеет особого владельца и защиту. Эти объекты могут быть изменены вызовом объекта, которые доступны и задаче и пейджерам.

В то время как объекты памяти и объекты виртуальной памяти тесно связаны, эти термины не эквивалентны. Объект виртуальной памяти может быть поддержан одним или более объектами памяти, которыми, в свою очередь, управляет пейджер. Объект виртуальной памяти может также быть частично поддержан другими объектами виртуальной памяти.

Объект виртуальной памяти является внутренним для системы виртуальной памяти и включает основную информацию о доступе к памяти. Объект памяти, в отличие от этого, обеспечен пейджером. Содержимое памяти связано с тем объектом памяти, который может быть получен с диска или некоторого другого запоминающего устройства, обмениваясь сообщениями с объектом памяти. Неявно каждый объект виртуальной памяти связан с данным пейджером через его объект памяти.

Объекты виртуальной памяти кэшируются с системными страницами (RAM), которая может быть любой степени двойки, кратного размеру аппаратной страницы. В ядре ОС X системные

страницы имеют тот же размер что и аппаратные страницы. Каждая системная страница представлена в данном адресном пространстве записью отображения. У каждой записи отображения есть своя собственная защита и наследование. Данная запись отображения может быть унаследована от shared (совместно используемых), сору или попе. Если страница отмечена shared в данном отображении, дочерние задачи совместно используют эту страницу для чтения и записи. Если страница отмечена сору, дочерние задачи получают копию этой страницы (использование копии при записи). Если страница отмечена попе, страницу дочернего элемента оставляют нераспределенной.

Объектами виртуальной памяти управляет машинно-независимая система виртуальной памяти, обрабатывая машинно-зависимой rmap system в базовую виртуальную в физическое отображение. Rmap system фактически обрабатывает таблицы страниц, буфера быстрого преобразования адреса, сегменты и т.д. в зависимости от вида используемого оборудования. Когда объект виртуальной памяти дублирован (например, страницы данных от процесса, который только что вызвал ветвление), то объект создается. Теневой объект первоначально пуст и содержит ссылку на другой объект.

Когда содержание страницы изменено, страница скопирована с родительского объекта в теневой объект и изменена. При чтении данных из страницы, если та страница существует в теневом объекте, страница, перечисленная в теневой объект, используется. Если у теневого объекта нет копии той страницы, с исходным объектом консультируются. Ряд из теневых объектов, указывающих на теневые объекты или исходные объекты, известен как *теневая цепочка*.

Теневые цепочки могут стать произвольно длинными, если объект многократно используется способом копии при записи. Однако так как ветвление часто сопровождается ехес, который заменяет весь материал, являющийся затененными, длинными цепочками пользуются редко. Далее Mach автоматически собирает “мусор” теневых объектов, удаляя любые промежуточные теневые объекты на чьи страницы больше не ссылается никакой (не существенный) теневой объект. Это даже

возможно для оригинала объекта, который будет удален, если больше не содержит страницы, которые относятся к цепочке.

Вызовы виртуальной памяти, доступные приложению, включают `vm_map` и `vm_allocate`, которые могут использоваться, чтобы отобразить данные файла или анонимную память в адресное пространство. В целом приложение может или отобразить файл в свое адресное пространство (через примитивы отображения файла, абстрагированные BSD), или отобразить объект (передав дескриптор к тому объекту). Кроме того, задача может изменить защиту объектов в его адресном пространстве и может совместно использовать те объекты с другими задачами.

В дополнение к аспектам отображения и выделения виртуальной памяти система виртуальной памяти содержит много других подсистем. Они включают бэкэнд (пейджеры) и подсистемы разделяемой памяти. Есть также другие подсистемы, связанные к виртуальной памяти, включая сервер разделяемой виртуальной памяти.

4.1. Понятие отображения памяти

Каждая задача Mach имеет свою собственную память отображения. В Mach эта память отображения образует упорядоченный двунаправленный список.

Никогда не нужно обращаться к памяти отображения непосредственно, если только вы не изменяли что-нибудь в пределах системы виртуальной памяти. `Vm_map_entry` структура содержит специфическую для задачи информацию об отдельном отображении наряду с ссылкой на объект поддержки. В сущности, это клей между объектом виртуальной памяти и отображением виртуальной памяти.

Несколько полей этой структуры имеют особое значение.

Поле `is_submap` является нулевым значением, которое говорит, является ли эта запись отображения нормальным объектом виртуальной памяти или `submap`. *Submap* – набор отображений, который является частью увеличенной карты. *Submap* часто используются, чтобы собрать группу отображений в целях совместного использования их среди многочисленных

задач Mach, но они могут использоваться в различных целях. То, что делает submap особенно мощной, – то, что, когда несколько задач отобразили submap в свое адресное пространство, они видят изменения друг друга, не только к содержанию объектов в карте, но к самим объектам. Это означает, что, поскольку дополнительные объекты добавлены или удалены из submap, они появляются или исчезают из адресных пространств всех задач, что разделяют эти submap.

Поле behavior управляет ссылочным поведением разбивки на страницы указанного диапазона в данной карте. Это значение изменяется, как кластеризируются *pageins*. Возможные значения – VM_BEHAVIOR_DEFAULT, VM_BEHAVIOR_RANDOM, VM_BEHAVIOR_SEQUENTIAL и VM_BEHAVIOR_RSEQNTL, для значения по умолчанию, случайного, последовательного или обратного-последовательного упорядочивания *pagein*.

Защита и max_protection-области управляют разрешениями объекта. Область защиты указывает на то, какие права задачи в настоящее время имеются для объекта, в то время как max_protection-область содержит максимальный доступ, который текущая задача может получить для объекта.

Урегулировав защиту, которая доступна только для чтения, любые записи к разделенной памяти вызвали бы исключение. Однако, когда задаче фактически нужно писать в этом выделенном регионе, оно может увеличить его разрешения в защищенном поле, чтобы позволить записи.

Можно использовать поле защиты при отладке разделяемой памяти. Устанавливая защиту только для чтения, любые непреднамеренные записи к разделяемой памяти вызвали бы исключение. Однако, когда задача фактически должна записать в ту совместно используемую область, она могла увеличить свои полномочия в поле защиты, чтобы позволить запись.

Это была бы дыра безопасности, если задача смогла увеличить свои собственные разрешения объекту памяти произвольно. Для того, чтобы сохранить необходимую модель механизма защиты, задача, которая владеет объектом памяти, должна быть способна ограничить права, предоставленные подчиненной задаче. Поэтому задаче не позволено увеличить

свою защиту после разрешений, предоставленных в `max_protection`.

Наконец, `use_rmap`-поле указывает, должны ли низкоуровневые отображения `submap` быть совместно использованы среди всех задач, в которые отображена `submap`. Если отображения не совместно используемое, то структура карты совместно использована среди всех задач, но фактического содержания страниц нет. Например, совместно используемые библиотеки обработаны с двумя подкартами. У совместно используемого раздела кода только для чтения есть набор `use_rmap true`. У чтения-записи (несовместно использованный) раздел есть набор `use_rmap false`, вынуждая чистую копию сегмента данных библиотеки быть отображенными на диске для каждой новой задачи.

4.2. Именованные записи

ОС X система виртуальной памяти обеспечивает абстракцию, известную как именованная запись. Именованная запись – не что иное, как дескриптор к совместно используемому объекту или подкарте.

Поддержка разделяемой памяти в ОС X достигается, совместно используя объекты между картами памяти различных задач. Объекты разделяемой памяти должны быть созданы из существующих объектов виртуальной памяти, вызывая `vm_allocate`, чтобы выделить память в вашем адресном пространстве и затем вызывая `mach_make_memory_entry_64`, чтобы получить дескриптор к базовому объекту виртуальной памяти.

Дескриптор, возвращенный `mach_make_memory_entry_64`, может быть передан к `vm_map`, чтобы отобразить тот объект в адресное пространство данной задачи. Дескриптор может также быть передан через IPC или другие средства для других задач так, чтобы они могли отобразить его в свои адресные пространства. Это обеспечивает возможность совместно использовать объекты с задачами, которые не находятся в вашем прямом распоряжении, и также позволяет вам совместно

использовать дополнительную память с задачами после того, как те задачи создаются.

Другая форма именованной записи, подкарты, используются, чтобы сгруппировать ряд отображений. Наиболее популярный способ использования подкарты должен совместно использовать отображения среди многочисленных задач Mach. Подкарта может быть создана с `vm_region_object_create`.

То, что делает подкарту особенно мощной, – то, что, когда несколько задач отобразили подкарту в свое адресное пространство, они видят изменения друг друга и в данных, и в структуре карты. Это означает, что одна задача может отобразить или неотобразить объект виртуальной памяти в адресном пространстве другой задачи, просто отображая или неотображая тот объект в подкарте.

Глава 5

ОБЗОР BSD

BSD часть OS X ядра происходит, прежде всего, от FreeBSD, версия 4.4BSD, который предлагает расширенные возможности сетей, производительности, безопасности и совместимости. BSD предоставляет множество дополнительных возможностей, включая следующие:

- *Преимущественная многозадачность с динамическим регулированием приоритетов.* Равномерное и справедливое распределение компьютера между приложениями и пользователями обеспечивается даже при тяжелейших нагрузках.

- *Многопользовательский доступ.* Многие люди могут использовать OS X-систему одновременно для различных вещей. Это означает, например, что системная периферия, такая как принтеры и диски, правильно разделяется между всеми пользователями в системе или сети, и что лимиты каждого ресурса могут быть размещены на пользователей или групп пользователей, защищая критические системные ресурсы от чрезмерного использования.

- *Мощный TCP / IP-стек с поддержкой промышленных стандартов:* SLIP, PPP, и NFS, OS X могут взаимодействовать с другими системами, а также выступать в качестве сервера предприятия, предоставляя жизненно важные функции, такие как NFS (удаленный доступ к файлам), услуги электронной почты или интернет-услуги: HTTP, FTP, маршрутизации и услуги межсетевого экрана (брандмауэра).

- *Защита памяти.* Приложения не могут мешать друг другу. Сбой одного приложения не влияет на работоспособность всей системы.

- *Виртуальная память и динамическое распределение памяти.* Приложения с огромными аппетитами к памяти удовлетворены, в то же время сохраняется интерактивность пользователей. С помощью системы виртуальной памяти в OS X каждое приложение имеет доступ к собственным 4 ГБ адресного пространства памяти. Это должно удовлетворить даже самые многопотребляющие память приложения.

- *Поддержка потоков в ядре на основе Mach потоков.* Пакеты потоков на уровне пользователя реализованы поверх

потоков ядра. Каждый поток ядра является независимо планируемым объектом. Когда поток из блоков пользовательского процесса вызывается в системе, другие потоки от того же процесса могут продолжать выполняться на тех или иных процессорах. По умолчанию процесс в обычном смысле имеет один поток – основной. Пользовательский процесс может использовать нить API POSIX для создания других пользовательских потоков.

– *Поддержка SMP.* Поддержка включена для компьютеров с несколькими процессорами.

– *Исходный код.* Разработчики получить наибольший Уровень контроля над средой программирования BSD, потому что источник включен.

– *Многие из POSIX API.*

5.1. BSD-возможности.

Возможности, которые доступны для пользовательского процесса, логически делятся на две части: возможности ядра и системные возможности, реализуемые или в сотрудничестве с процессом сервера.

Возможности, реализованные в ядре, определяют виртуальную машину, в которой каждый процесс работает. Как и многие реальные машины, эта виртуальная машина имеет управление памятью, средства прерывания, таймеры и счетчики.

Виртуальная машина также позволяет получить доступ к файлам и другим объектам через набор дескрипторов. Каждый дескриптор напоминает контроллер устройства и поддерживает набор операций. Как устройства на реальных машинах, некоторые из которых являются внутренними и некоторые – внешними, части дескрипторных структур встроены в операционную систему, в то время как другие часто реализуются в серверных процессах.

Компонент BSD предоставляет следующие услуги ядра:

– процессы и защита:

- хост и идентификаторы процесса;
- создание процесса и удаление;
- идентификаторы пользователя и группы;

- группы процессов;
- управление памятью;
- тексты, данные, стек и динамические разделяемые библиотеки;
- отображаемые страницы;
- управление защитой страницы:
 - примитивы синхронизации POSIX;
- POSIX разделяемой памяти;
- сигналы:
 - типы сигналов;
 - обработчики сигналов;
 - передача сигналов;
- сроки и статистика:
 - в реальном времени;
 - время интервала;
- дескрипторы
 - файлов;
 - каналов;
 - сокетов;
- управления ресурсами:
 - приоритеты процессов;
 - использование ресурсов и ограничения ресурсов;
 - квоты;
- поддержки системных операций:
 - операции начальной загрузки;
 - операции завершения работы;
 - учет.

Системные возможности BSD (объектов, которые могут взаимодействовать с пространством пользователя) включают:

- общие операции ввода/вывода, такие как *читать и писать, неблокирующие и асинхронные операции файловой системы*;
- межпроцессное взаимодействие;
- обслуживание терминалов и других устройств;
- управления процессом;
- сетевые операции.

5.2. Различия между OS X и BSD

Хотя BSD-часть в OS X в первую очередь происходит от FreeBSD, были внесены некоторые изменения.

Системный вызов `sbrk()` для управления памятью является устаревшим, его использование не рекомендуется в OS X.

Модель выполнения OS X использует другой формат объектного файла для исполняемых файлов и общих объектов и другой механизм для выполнения некоторых из этих исполняемых файлов. Основным родным форматом является Mach-O. Этот формат поддерживается редактором динамической компоновки (`dyld`). PEF-формат двоичного файла поддерживается менеджером фрагментов кода (`Code Fragment Manager`, CFM). Ядро поддерживает `execve()` с двоичными файлами Mach-O. Отображение и управление Mach-O динамических разделяемых библиотек, как и запуск приложений основанных на PEF-основе, выполняются в пространстве пользователя.

OS X не поддерживает устройства, отображаемые в памяти с помощью функции `mmap()`. (Поддержка графических устройств и других подсистем обеспечивает аналогичную функциональность, но с использованием различных API.) В OS X этот интерфейс будет осуществляться за счет клиентов пользователей. Смотреть документы комплекта I/O Apple для получения дополнительной информации. Вызов `swapon()` не поддерживается; `masx_swapon()` является эквивалентом вызова от пейджера Mach. Единая реализация в буферный кэш в OS X отличается от найденной в FreeBSD. Mach предоставляет ряд IPC-примитивов, которые традиционно не найдены в UNIX. Некоторые изменения были внесены в модель безопасности BSD для поддержки однопользовательских конфигураций и с многочисленными администраторами, в том числе возможностью отключения владельца и разрешений на основе *том-на-том*. Механизм блокировки, используемый по всему ядру, существенно отличается от механизма, используемого в FreeBSD. Механизм расширения ядра, используемый OSX, совершенно иной. В OSX слой драйвера, комплект ввода/вывода, является объектно-ориентированным стеком

драйверов, написанном на C ++. Программные интерфейсы общего ядра, или KPIs, используются для написания недрайверных расширений ядра.

Кроме того, несколько новых функций, которые были добавлены, являются специфическими для OS X (Darwin), реализациями BSD. Эти особенности не применяются в FreeBSD. Усовершенствования в буферном кэше файловой системы и кластеризации файлового ввода/вывода:

- адаптивное и умозрительное упреждающее чтение;
- процесс пользователя, контролируемый упреждающим чтением;
- время старение буферного кэша файловой системы;
- усовершенствования поддержки файловой системы;
- реализация расширений Apple, для ISO-9660 файловой системы;
- многопоточный асинхронный ввод/вывод для NFS.

Добавление системных вызовов для поддержки семантики файловых систем Mac OS Extended (HFS +):

- дополнения к соглашениям именования для имен путей, как это требуется для доступа к нескольким forks в файловых системах Mac OS Extended.

Глава 6

УЧЕТНЫЕ ЗАПИСИ ПОЛЬЗОВАТЕЛЕЙ И СЕТЕВЫЕ ВОЗМОЖНОСТИ

Учетные записи пользователей

Одним из отличительных признаков современных операционных систем является поддержка множества учетных записей пользователей. Lion поставляется с надежной, защищенной и тщательно отлаженной многопользовательской рабочей средой. Лежащая в основе Lion система UNIX отвечает за предоставление такой сложной многопользовательской рабочей среды.

6.1. Типы и атрибуты учетных записей пользователей

Mас с абсолютной точностью определяет пользователя через его учетную запись. Каждый отдельный файл или папка на жестком диске компьютера Мас также, как и каждый элемент и процесс на компьютере, принадлежат учетной записи какого-либо пользователя.

Типы учетных записей пользователей. OS X поддерживает несколько учетных записей пользователей, а также несколько типов учетных записей пользователей для обеспечения разных уровней доступа. По сути, тип учетной записи выбирают для предоставления определенного уровня доступа, который удовлетворяет требованиям пользователя. Учетные записи пользователей бывают пяти типов: стандартные пользователи, администраторы, пользователь guest (гость), пользователи только для общего доступа (sharing only users) и пользователь root.

Стандартные пользователи. Учетные записи стандартных пользователей должны использоваться при повседневной работе с компьютером. Они также широко применяются, когда компьютер используется несколькими людьми. Это обусловлено тем, что учетная запись стандартного пользователя обеспечивает оптимальный баланс между удобством использования и безопасностью. Стандартные пользователи являются членами группы staff и имеют доступ для чтения к

большинству объектов, параметров, приложений, а также к папкам Общие (Public) и Сайты (Sites) других пользователей. Они могут вносить изменения в личные настройки и объекты в своих домашних папках. Стандартным пользователям не разрешается изменять только системные параметры, системные файлы и то, что может повлиять на других пользователей.

Администраторы. Администраторы являются частью группы admin и имеют полный доступ ко всем приложениям, настройкам системы и большинству системных файлов. По умолчанию администраторы не имеют доступа к защищенным системным файлам или файлам других пользователей вне папок Общие (Public) и Сайты (Sites). Несмотря на это, они могут в случае необходимости обойти эти ограничения как в графической среде, так и в командной строке. Например, они могут обновлять системное программное обеспечение, если успешно пройдут проверку, когда приложение установки запрашивает авторизацию. Этот тип учетной записи используется по умолчанию для начальной учетной записи, которая создается при первичной настройке Lion с помощью Ассистента настройки (Setup Assistant). На Mac должна быть, по крайней мере, одна учетная запись администратора.

Пользователь guest аналогичен пользователю без полномочий администратора и не имеющему пароля. Однако, когда пользователь *guest* выходит из системы, его домашняя папка удаляется. Сюда включаются файлы настроек, журнал просмотра браузера и любые другие следы, которые пользователь мог оставить в системе. Пользователи только для общего доступа имеют доступ только к общим папкам и файлам. Пользователи только для общего доступа не имеют домашней папки и не могут войти в систему локально. Они изначально имеют права, похожие на права пользователя *guest*, но при этом имеют доступ только к папкам Общие (Public). Однако администраторы могут разрешить конкретному пользователю доступ к любой папке через панель настроек Общий доступ (Sharing).

Пользователь root. Учетная запись пользователя *root*, называемого также *Администратор системы* (System Administrator), дает неограниченный доступ к любым объектам на компьютере. Пользователь *root* может прочесть, изменить и

удалить любой файл, а также изменить любой параметр и установить любое программное обеспечение. Чтобы избежать злоупотребления этой учетной записью, по умолчанию никому не разрешается входить в систему под учетной записью root, так как пароль для пользователя root не задан. Поскольку многие системные процессы выполняются только от лица пользователя root, он должен существовать в системе, иначе OSX не смогла бы загрузиться. За вход пользователя в систему отвечает процесс loginwindow, а обслуживанием информации об учетных записях занимается фоновый процесс Open Directory. Процесс Open Directory хранит информацию об учетных записях пользователей в наборе XML-файлов, которые располагаются в папке /var/db/dslocal/nodes/Default/users. Эта папка доступна для чтения только учетной записи SystemAdministrator (root). Файлы организованы как списки атрибутов пользователей и соответствующих значений. Каждый пользователь имеет множество атрибутов, определяющих детали его учетной записи.

Атрибуты учетных записей пользователей:

- Полное имя (Full Name) может быть достаточно длинным и содержать практически любые символы и должно быть уникальным в системе. Вы можете легко изменить полное имя в любой момент позднее.

- Псевдонимы (Aliasnames) используются для связывания учетной записи локального пользователя с учетными записями других служб. Например, с локальной учетной записью можно связать идентификатор Apple ID пользователя. Этот атрибут в ОС Lion является опциональным, но он требуется для интеграции с такими интернет-сервисами компании Apple, как iCloud и BacktoMyMac.

- Учетная запись (Account Name) применяется для уникальной идентификации учетной записи и по умолчанию используется в качестве имени домашней папки пользователя. Пользователь при входе в систему может использовать как полное имя, так и имя учетной записи. Однако каждая учетная запись должна иметь свое уникальное имя; оно не может содержать никакие специальные символы или пробелы.

- Идентификатор пользователя (User ID) – числовой атрибут, используемый для отождествления учетной записи с

владением файлами и папками. Это число практически всегда уникально. Идентификаторы учетных записей пользователей начинаются с 501, а большинство идентификаторов системных учетных записей находятся ниже 100.

– Универсальный уникальный идентификатор (UniversallyUniqueID (UUID)), иногда также называемый «GeneratedUID» (GUID). Данный атрибут генерируется компьютером в процессе создания учетной записи и является уникальным. UUID используется для ссылки на пароль пользователя, который хранится в отдельном защищенном месте. Он используется также для определения членства в группах и прав доступа к файлам.

Группа (Group) – это числовой атрибут, используемый для связывания пользователя с группой по умолчанию. По умолчанию для большинства пользователей данный атрибут имеет значение 20, которое указывает на группу staff. Это означает, что когда пользователь создает новый файл, последний будет принадлежать учетной записи этого пользователя и группе staff.

Оболочка при входе в систему (LoginShell) – этот путь указывает на файлы оболочки командной строки (*command-lineshell*), используемой по умолчанию учетной записью. Для каждого пользователя, которому разрешено использовать командную строку, данный атрибут по умолчанию устанавливается в значение /bin/bash. По умолчанию как администраторам, так и стандартным пользователям разрешен доступ к командной строке.

Домашний каталог (HomeDirectory) – путь, который определяет расположение домашней папки пользователя. Для всех пользователей, за исключением пользователей только для общего доступа, не имеющих домашних папок, используется значение /Пользователи/<пате>, где <пате>– это имя учетной записи.

6.2. Основы обеспечения безопасности учетных записей

Основным назначением многопользовательской операционной системы является предоставление всем пользователям безопасной рабочей среды. Уязвимости учетных

записей для большей гибкости в управлении доступом пользователей компания Apple сделала эти различные типы учетных записей доступными. Так как каждый тип учетной записи предназначен для обеспечения различных уровней доступа, необходимо знать о потенциальных рисках безопасности каждого типа учетной записи.

Стандартные пользователи. Данный тип учетных записей является максимально безопасным, если задан подходящий пароль. Эти пользователи могут использовать почти все ресурсы и средства компьютера Mac, но не могут изменить ничего, что может повлиять на операционную систему. Можно еще больше ограничить эту учетную запись, используя управляемые настройки средств родительского контроля.

Администраторы. Это учетная запись создается при начальной настройке компьютера Mac с помощью Ассистента миграции (Setup Assistant), позволяет изменять все, что требуется для управления системой. Недостатком является то, что пользователь может изменить или установить такое программное обеспечение, которое сделает систему незащищенной или неустойчивой. Для повседневного использования можно применять дополнительные учетные записи администраторов. Они создаются равноценными в правах. Все учетные записи администраторов могут вносить любые изменения в систему, включая удаление или изменение пароля других учетных записей администраторов. Администраторы могут также изменять права для любой другой учетной записи, либо отключая текущих администраторов, либо превращая стандартных пользователей в администраторов. Более того, запуск вредоносных программ от имени администратора может нанести серьезный вред операционной системе. Любой администратор может включить учетную запись root или изменить пароль учетной записи root с помощью утилиты Служба каталогов (DirectoryUtility), расположенного в папке Система/Библиотеки/CoreServices (/System/Library/CoreServices). По этим причинам необходимо ограничивать число учетных записей администраторов.

Пользователь guest. Пользователям guest по умолчанию разрешен доступ к компьютеру через сетевой общий доступ без пароля, и им можно разрешить входить в систему локально, не вводя пароль. Очевидный риск безопасности состоит в том, что любой пользователь получает доступ, эквивалентный доступу стандартной учетной записи (в том числе доступ к папкам Общие [Public], Почтовый ящик [Drop Box], Сайты [Sites] и Общий доступ [Shared]). Такой пользователь может выполнять некоторые потенциально опасные приложения или заполнять жесткий диск нежелательными файлами. Пользователь guest может также перезапустить или выключить Mac, что позволяет скомпрометировать систему во время запуска.

6.3. Типы паролей

OSX использует пароли в качестве основного метода проверки подлинности пользователя. Большинство пользователей знакомо только со своим паролем учетной записи, но Mac может иметь также пароль прошивки, главный пароль, множество паролей ресурсов и несколько паролей для keychain.

Каждый тип пароля предназначен для конкретной цели.

Пароль учетной записи (accountpassword). Пароль учетной записи – это атрибут, который используется для аутентификации пользователя, чтобы он мог войти в систему. По соображениям безопасности пароль учетной записи пользователя хранится как зашифрованные данные в записи пользователя. Только пользователь root может обращаться к зашифрованным данным в записи пользователя.

Пароль прошивки (firmwarepassword). Используется для защиты компьютера Mac во время загрузки. По умолчанию любой может обойти систему защиты путем простого использования одной из широко известных комбинаций клавиш перехвата загрузки. Например, по умолчанию любой может удерживать клавишу [Options] при загрузке, чтобы выбрать альтернативную операционную систему, тем самым обходя систему защиты. Задание пароля прошивки не позволит

несанкционированным пользователям использовать какие-либо клавиатурные комбинации перехвата загрузки.

Главный пароль (*masterpassword*). Используется для сброса паролей учетных записей стандартных пользователей, администраторов и учетных записей, использующих систему шифрования FileVault, если пользователь забыл свой пароль учетной записи.

Пароль ресурса (*resourcepassword*). Общий термин для описания пароля, применяемого практически любой службой, которая требует аутентификации. Пароли ресурсов включают пароли электронной почты, сайта, файлового сервера, приложения и зашифрованного образа – диска. Многие пароли ресурсов автоматически сохраняются для пользователя системой keychain.

Пароль *keychain-файла*. OS X защищает важные аутентификационные активы пользователя (кроме пароля учетной записи) в зашифрованных *keychain-файлах*. Каждый keychain-файл защищается с помощью собственного пароля. Система попытается синхронизировать пароли keychain-файлов с паролем учетной записи пользователя. Однако эта синхронизация не обязательна, и для keychain-файлов могут использоваться свои пароли.

6.4. Управление keychain-файлами

OSX автоматически защищает все данные, необходимые для аутентификации, в зашифрованных keychain-файлах. Так как большое количество важных объектов оказывается в файлах keychain, то сами эти файлы шифруются с помощью устойчивого алгоритма. Они будут недоступны, если вы не знаете пароль keychain-файла. Если вы забудете пароль keychain-файла, его содержимое будет утрачено навсегда. Самым важным свойством системы keychain-файлов является то, что она работает полностью автоматически при использовании настроек по умолчанию.

Keychain-файлы. В системе хранятся различные файлы keychain для разных пользователей и ресурсов.

- Пользователи/<имя_пользователя>/Библиотеки/Keychains/login.keychain (/Users/<имя_пользователя>/Library/Keychain/login.keychain).
 - /Библиотеки/Keychains/FileVaultMaster.keychain (/Library/Keychain/FileVaultMaster.keychain). Этот keychain-файл содержит главный пароль системы FileVault.
 - Библиотеки/Keychains/System.keychain (/Library/Keychain/System.keychain).
 - Система/Библиотеки/Keychains/ (/System/Library/Keychains/).
- Доступ к keychain-файлам. Основным инструментом для управления keychain-файлами является приложение Связка ключей (KeychainAccess). С помощью этого приложения можно просматривать и модифицировать любой объект keychain-файла, включая сохраненные пароли ресурсов, сертификаты, ключи, формы сайтов и защищенные заметки. Можно также создавать и удалять keychain-файлы, изменять их настройки и пароли, а также исправлять поврежденные keychain-файлы.

6.5. Сброс паролей учетных записей

Ошибка пользователя при вводе или забытый пароль являются основными причинами большинства проблем входа в систему в любой ОС. Второй наиболее распространенной проблемой, специфичной для Lion, является рассогласование паролей keychain-файлов пользователя с его паролем входа в систему.

Использование мастера-пароля. Мастер-пароль может использоваться для восстановления паролей обычных учетных записей и необходим для сброса паролей учетных записей, защищенных системой File Vault. Установка единого мастера-пароля – это самая доступная гарантия того, что у вас всегда будет «*черный ход*» в систему без необходимости создавать на каждом компьютере свою локальную учетную запись.

Задание или изменение мастера-пароля. В двух случаях мастер-пароль уже может быть задан на системе OSX. Первый – когда вы обновили предыдущую версию MacOS X, в которой мастер-пароль уже задан, до ОС Lion. Второй – если вы использовали Ассистент миграции (Migration Assistant) для

переноса учетной записи, защищенной старой версией системы FileVault, в систему Lion. Если вы знаете мастер-пароль, то можете использовать следующие инструкции, чтобы изменить его. Однако если мастер-пароль неизвестен, тогда следует выполнить инструкции в разделе «Сброс утерянного мастера-пароля». Чрезвычайно важно, чтобы мастер-пароль был правильно настроен и известен администратору. Если он по какой-то причине утрачен, администратору нужно немедленно его сбросить, чтобы можно было сбрасывать пароли обычных пользователей. Однако возможность задать новый мастер-пароль для Mac не означает, что можно будет восстановить учетные записи, защищенные системой Традиционный File Vault (Legacy File Vault), которые были созданы с использованием старого мастера-пароля. Только мастер-пароль, используемый при включении системы защиты Традиционный File Vault (Legacy File Vault), может разблокировать недоступную учетную запись.

Существуют две ситуации, когда требуется сбросить мастер-пароль. Первая – когда мастер-пароль известен и администратор хочет изменить его. Изменить известный мастер-пароль легко. Кроме того, этот метод сохраняет вашу способность сбрасывать пароли, которые связаны как с обычными учетными записями, так и с учетными записями, защищенными системой «Традиционный File Vault» (Legacy File Vault).

Вторая ситуация – текущий мастер-пароль утрачен, и требуется создать новый. В этом случае новый мастер-пароль не позволит сбрасывать пароли учетных записей, защищенных системой Традиционный FileVault (Legacy File Vault). Это объясняется тем, что система Lion не поддерживает перешифрование учетных записей, защищенных системой Традиционный FileVault (Legacy File Vault), а этот шаг необходим для изменения утраченного мастера-пароля. Поэтому, если утрачен мастер-пароль, который использовался при включении системы Традиционный FileVault (Legacy File Vault), и пользователь забывает свой пароль, восстановить данные пользователя нельзя. По этой причине настоятельно

рекомендуется перейти от системы Традиционный File Vault (Legacy File Vault) к системе полного шифрования диска File Vault 2 в OS X

Сброс пароля при входе в систему. Для сброса пароля в окне входа в систему используются два метода – мастер-пароль или идентификатор AppleID пользователя. Первый метод требует знания текущего мастера-пароля компьютера Mac и позволяет сбрасывать пароли как для обычных учетных записей пользователей, так и для учетных записей, которые защищены системой Традиционный FileVault (LegacyFileVault). Второй метод предполагает, что учетная запись пользователя была связана с идентификатором AppleID. Однако AppleID нельзя использовать для сброса пароля учетной записи пользователя, защищенной системой Традиционный FileVault (LegacyFiieVault) или FileVault 2.

Эти два способа обычно используются, когда нет доступа к учетной записи администратора на компьютере Mac. Если же у вас есть доступ к учетной записи локального администратора, тогда для сброса утерянного пароля следует использовать инструкции.Сброс пароля администраторомЕсли у вас есть доступ к учетной записи администратора, то на панели Пользователи и группы (Users&Groups) можно легко сбросить пароли учетных записей, не защищенные системой Традиционный FileVault (LegacyFileVault). Этот способ обычно используется, только когда пользователь забывает свой пароль и нужно создать новый.

Сброс паролей учетных записей, защищенных системой FileVault. Учетные записи, защищаемые системой FileVault, являются уникальными, так как домашняя папка пользователя сохраняется в зашифрованном образе диска, защищенном паролем учетной записи этого пользователя. Следовательно, для администратора крайне важно иметь возможность сбросить пароль учетной записи пользователя, защищаемой системой FileVault, если пользователь захочет снова получить доступ к файлам своей домашней папки.Обычной учетной записи администратора недостаточно для восстановления забытого пароля учетной записи, защищаемой с помощью FileVault.

Система FileVault была бы не слишком надежна, если бы любой администратор смог это сделать. Поэтому если пользователь, защищаемый системой FileVault, забыл свой пароль учетной записи, для сброса этого пароля понадобится главный пароль.

Если главный пароль будет утрачен вместе с паролем пользователя, защищаемого системой FileVault, то сделать ничего будет нельзя. Для восстановления учетной записи, защищаемой FileVault, необходим, как минимум, один из этих двух паролей. В противном случае восстановить данные пользователя невозможно. Даже компания Apple помочь ничем не сможет – ее специалисты разработали систему FileVault таким образом, чтобы она была максимально защищена, и поэтому имеется только один способ сброса пароля учетной записи, защищенной FileVault– главный пароль.

Сброс из панели настроек Пользователи и группы (Users & Groups). Если вы знаете мастер-пароль, Lion предоставляет два метода для простого сброса паролей пользователей, защищенных Традиционным FileVault. Первый используется, как рассмотрено в разделе «Сброс пароля при входе в систему» ранее. Второй метод включает в себя сброс учетной записи, защищенной Традиционным FileVault, из панели настроек Пользователи и Группы (Users&Groups).

Сброс паролей keychain-файлов. Система изменит пароль файла login.keychain, только если пользователь знает свой текущий пароль учетной записи и решает изменить его. Для обеспечения максимальной безопасности пароли keychain-файлов нельзя изменить никаким внешним процессом сброса паролей. Apple создала систему keychain-файлов без «черного хода», иначе система стала бы значительно менее безопасной. Следовательно, всякий раз, когда администратор сбрасывает пароль учетной записи пользователя, пароль keychain-файла изменяться не будет, а сам файл login.keychain не будет автоматически открываться при входе пользователя в систему. Однако по умолчанию в Lion, когда пользователь с недавно сброшенным паролем входит в систему, ему выдается диалоговое окно с предложением обновить пароль файла login.keychain или создать новый keychain-файл. Выбранный по умолчанию вариант *Обновить пароль (UpdateKeychainPassword)*

будет работать, если только пользователь знает предыдущий пароль keychain-файла. Это отличается от ситуации, когда пользователь только что сбросил пароль. Здесь пользователь может щелкнуть на кнопке *Новая связка* ключей (CreateNewKeychain), чтобы создать новый файл login.keychain. Система переименует старый файл login.keychain и оставит его в папке ~/Библиотеки/Keychains (~/Library/Keychains) пользователя на случай, если пользователь вспомнит свой старый пароль. Наконец, пользователь может просто проигнорировать предупреждение, щелкнув на кнопке *Продолжить вход* (ContinueLogin). Если диалоговое окно автоматической синхронизации keychain-файлов не появилось, вы все равно можете сбросить пароль файла login, keychain из приложения KeychainAccess (если конечно, помните предыдущий пароль keychain-файла). Если предыдущий пароль keychain-файла пользователя неизвестен, то содержимое этого keychain-файла будет потеряно навсегда.

Изменение пароля основной учетной записи. Типична ситуация, когда выполняется автоматический вход под учетной записью основного пользователя и мастер-пароль для системы не задан. Все это с большой вероятностью приводит к тому, что владельцы компьютеров Mac в конечном счете забывают свой пароль основной учетной записи администратора и не имеют никакой возможности восстановить его, так как никогда не создавали мастера-пароля или другую учетную запись администратора. Компания Apple предусмотрела возможность такого сценария и включила утилиту Сброс пароля (ResetPassword) в систему восстановления LionRecovery.

Совет. Можно использовать утилиту ResetPassword для восстановления прав доступа к домашней папке и списков контроля доступа (ACL) для выбранной учетной записи, щелкнув на кнопке *Reset*. Утилита ResetPassword является довольно опасным приложением, которое может полностью уничтожить любой из параметров безопасности, которые вы настроили для защиты Mac. Поэтому она не станет работать, будучи скопированной с оригинальных носителей, однако любой пользователь с доступом к OSXRecovery может воспользоваться этой утилитой. Apple предусмотрел эту

ситуацию, предоставив еще одну утилиту по восстановлению OS X – Утилиту пароля прошивки (FirmwarePasswordUtility). Эта утилита предотвращает проникновение неавторизованных пользователей в обход нормального запуска системы.

Сетевые возможности

6.6. Конфигурация сети

Сегодня почти каждая сеть – от небольших локальных до Интернета – основана на пакете TCP/IP. OS X несет в себе надежную реализацию TCP/IP. Первыми компьютерными системами, начавшими использовать TCP/IP, были системы UNIX. Именно поэтому большая часть встроенного в Mac OS X программного обеспечения TCP/IP основывается на программном обеспечении UNIX с открытым исходным кодом.

Фундаментальные сетевые концепции. Для правильной настройки и решения сетевых проблем необходимо разбираться в фундаментальных сетевых концепциях. Фундаментальная сетевая терминология. Существует устоявшаяся семиуровневая модель, используемая для описания сетевых технологий – Open System Interconnection Reference Model (известная, как модель OSI). Концепция сети в упрощенном представлении из трех основных элементов:

Сетевой интерфейс

Это та среда, через которую происходит обмен сетевыми данными. Сетевые интерфейсы могут быть *физическими* или *виртуальными*. Наиболее распространенным физическими сетевыми компьютерными интерфейсами являются Ethernet и 802.11 беспроводной Ethernet, который известен как Wi-Fi. Виртуальные сетевые интерфейсы могут использоваться для защиты незащищенных иным образом сетевых подключений путем создания виртуальной частной сети (virtual private network, VPN) поверх стандартного сетевого интерфейса.

Сетевой протокол

Определяет множество стандартных правил для представления данных, сигнализации, аутентификации, обнаружения ошибок на сетевых интерфейсах. Протоколы

предназначены для обеспечения правильной и полной передачи данных. Специфические протоколы имеют ограниченное применение, поэтому часто несколько протоколов объединяются или накладываются друг на друга, чтобы создать законченное сетевое решение. Например, объединенный пакет протоколов TCP/IP обеспечивает только адресацию и пересылку данных между двумя точками через Интернет; для таких действий, как проверка электронной почты или просмотр сайта, требуются десятки других протоколов.

Сетевая служба

В контексте настроек сети термин сетевая служба (network service) описывает настройки сетевого интерфейса, которые требуются для определения сетевого подключения. Фундаментальным свойством Linux является возможность поддерживать несколько сетевых служб, или подключений, для каждого индивидуального сетевого интерфейса.

MAC-адрес (Media Access Control) MAC-адрес (Media Access Control – Управление доступом к среде). Используется для уникальной идентификации физического сетевого интерфейса в локальной сети. Каждый физический сетевой интерфейс имеет, по крайней мере, один связанный с ним MAC-адрес.

Наиболее распространенным сетевым интерфейсом является Ethernet, поэтому адрес MAC часто называют «адресом Ethernet». Тем не менее, практически каждый другой тип сетевого интерфейса также использует некий тип MAC-адреса для уникальной идентификации. К таким интерфейсам относятся Wi-Fi, Bluetooth, Fire Wire и Fibre Channel.

Адрес MAC является обычно 48-битным числом, представленным шестью группами из двузначных шестнадцатеричных чисел (называемых *октетами*), разделенными двоеточием. Например, обычный адрес MAC может выглядеть примерно так: 00:1C:B3:D7:2F:99. Три первых октета являются Уникальным идентификатором организации (Organizationally Unique Identifier, OUI), а три последних определяют само сетевое устройство. Другими словами, первые три октета MAC-адреса можно использовать для определения изготовителя сетевого устройства. Адрес IP (Internet Protocol) Коммуникация с компьютерами в локальных и удаленных сетях

требует IP-адреса (адреса протокола Интернета). IP-адреса, в отличие от адресов MAC, связаны с сетевым интерфейсом непостоянно. IP-адреса назначаются сетевому интерфейсу на основе локальной сети, с которой он соединен. Это означает, что если используется портативный компьютер, то при каждом подключении к новой сети ему будет выдаваться новый IP-адрес. Можно задать несколько IP-адресов для каждого сетевого интерфейса, но такой подход используется только для компьютеров, которые обеспечивают сетевые службы.

В настоящее время существуют два стандарта IP-адресов: IPv4 и IPv6. IPv4 – первая широко используемая схема IP-адресации и до сих пор наиболее распространенная. Адрес IPv4 является 32-битным числом, которое представляют четырьмя группами трехзначных чисел (называемых также *октетами*), разделенных точками. Каждый октет имеет значение между 0 и 255. Например, типичный IPv4-адрес может выглядеть следующим образом: 10.1.45.186.

При использовании IPv4 имеется около четырех миллиардов уникальных адресов. Если учесть, что многие пользователи владеют несколькими поддерживающими сеть устройствами и количество пользователей постоянно растет, то это число оказывается недостаточно большим. В настоящее время мы расширяем адреса IPv4, используя сетевые маршрутизаторы, которые могут использовать один реальный адрес IPv4 для диапазона повторно используемых частных сетевых адресов. Так сконфигурировано большинство домашних сетей, однако это только временное решение. Преемником IPv4 является IPv6, но поскольку IPv4 прочно закрепился в Интернете, то переход на IPv6 идет медленно. Основное преимущество IPv6 состоит в значительно большем адресном пространстве – таком большом, что каждый человек на Земле мог бы иметь примерно $1,2 \times 10^{19}$ копий всего адресного диапазона IPv4. Задача создания IPv6 состояла в том, чтобы исключить необходимость в частных адресах и облегчить переназначение и изменение адресов в новой сети. Адрес IPv6 является 128-битным числом, которое представлено в восьми группах четырехзначных шестнадцатеричных чисел, разделенных двоеточием. Шестнадцатеричные числа

используют систему счисления с основанием 16, поэтому после цифры 9 используются буквы от А до F. Например, типичный адрес IPv6 будет выглядеть примерно следующим образом: 2C01:0EF9:0000:0000:0000:0000:142 D:57AB. Большие строки нулей в адресе IPv6 можно сократить с помощью двойного двоеточия, что приводит к адресу вроде 2C01:0EF9::142D:57AB.

Маска подсети. Компьютер использует маску подсети для определения диапазона IPv4- адресов в локальной сети. Сети на основе протокола IPv6 не требуют масок подсетей. Маска подсети аналогична адресу Ipv4 тем, что также является 32-битным числом, организованным в четыре группы октетов. Компьютер применяет маску подсети к своему IP-адресу для определения диапазона адресов локальной сети. Ненулевые биты в маске подсети (обычно 255) соответствуют части IP-адреса, которая определяет, к какой сети относится адрес. Нулевые биты соответствуют части IP-адреса, которая различается между хостами в той же сети. Например, предположим, что компьютер имеет IP-адрес 10.1.5.3 и используется самая распространенная маска подсети – 255.255.255.0. Тогда локальная сеть определяется как хосты с IP-адресами в диапазоне от 10.1.5.1 до 10.1.5.254.

Когда компьютер собирается осуществить взаимодействие с другим сетевым устройством, он применяет маску подсети к целевому IP-адресу другого устройства, чтобы определить, находится ли оно в той же локальной сети. Если это так, то компьютер пытается напрямую обратиться к другому сетевому устройству. Если это не так, то другое устройство принадлежит другой сети, и компьютер будет отправлять все данные для этого устройство на маршрутизатор.

Адрес маршрутизатора. Маршрутизаторы являются сетевыми устройствами, которые управляют подключениями между отдельными сетями. Маршрутизаторы направляют сетевой трафик между сетями, мостами между которыми они являются. Для определения того, куда перенаправляется трафик, в маршрутизаторах используются таблицы маршрутизации. Даже если маршрутизатор получит трафик, предназначенный для сети, о которой ничего не знает, он направит этот трафик другому маршрутизатору, который может быть ближе к месту

назначения. Маршрутизаторы являются «разумом» Интернета. Для того чтобы компьютер мог обращаться к другим компьютерам за пределами локальной сети, он должен быть сконфигурирован с указанием IP-адреса маршрутизатора. Последний соединяет эту локальную сеть с другой сетью или (в жилых домах) с поставщиком услуг Интернета. Обычно адрес маршрутизатора находится в начале диапазона локальных адресов и всегда в той же подсети. Используя предыдущий пример и предполагая, что компьютер имеет IP-адрес 10.1.5.3 и используется самая распространенная маска подсети 255.255.255.0, диапазон IP-адресов локальной сети будет от 10.1.5.0 до 10.1.5.255, и адрес маршрутизатора будет, скорее всего, 10.1.5.1.

Протокол управления передачей (TCP). Протокол управления передачей (TCP) является основным протоколом для передачи данных между двумя IP-устройствами. TCP является преимущественным транспортным механизмом для многих служб Интернета, так как гарантирует надежную доставку данных в нужном порядке. IP обеспечивает сетевую адресацию и маршрутизацию данных, а TCP гарантирует, что данные полностью доставляются по месту назначения. Комбинация этих протоколов предоставляет пакет TCP/IP, обычно называемый *пакетом интернет-протоколов*. Пакет протоколов TCP/IP разбивает непрерывные потоки данных на множество отдельных пакетов информации перед пересылкой по сети. Это связано с тем, что IP-сети используют технологию коммутации пакетов для маршрутизации и передачи данных. Почти все цифровые технологии сетей основываются на пакетах, что обеспечивает эффективный транспорт для сетевых соединений. Протокол TCP/IP был первоначально разработан для военных, так что сетевая технология на основе пакетов создавалась для решения проблем отказа линий связи. Сложное оборудование маршрутизации было первоначально разработано для сетей TCP/IP, чтобы данные могли перенаправляться и посылаться повторно при отказе сетевой линии связи. Реже используется протокол, называемый *Универсальным протоколом датаграмм (UDP)*. UDP является более простым протоколом, но не гарантирует надежности или порядка данных, посланных через

сети. В некоторых случаях UDP является предпочтительным, так как обеспечивает лучшую производительность, чем TCP. Примерами сетевых служб, которые используют UDP, являются Система доменных имен (DNS), потоковая передача мультимедиа, передача голоса через IP (VoIP) и сетевые игры. Эти службы были созданы с учетом возможности потери данных или нарушения порядка их доставки с тем, чтобы получить выгоду от более высокой производительности протокола UDP.

Сети в действии. Пакеты TCP/IP упаковываются в кадры Ethernet для передачи по локальной сети. Пакет TCP/IP включает IP-адреса источника и места назначения вместе с посылаемыми данными. Сетевое устройство применяет настройки маски подсети, чтобы определить, что IP-адрес места назначения находится в локальной сети. Если это так, устройство обратится к своей таблице протокола разрешения адресов (Address Resolution Protocol, ARP), чтобы посмотреть, знает ли оно MAC-адрес, соответствующий целевому IP-адресу. Каждый хост в сети поддерживает и постоянно обновляет ARP-таблицу известных MAC-адресов, которые соответствуют IP-адресам в локальной сети. Если MAC-адрес отсутствует в этой таблице, сетевое устройство проводит широковещательную рассылку ARP-запроса в локальную сеть, запрашивая целевое устройство ответить и указать свой MAC-адрес и добавляет ответ в свою ARP-таблицу для последующего обращения. Когда MAC-адрес определен, исходящий пакет Ethernet, содержащий пакет TCP/IP, будет послан по целевому MAC-адресу. Другое сетевое устройство, скорее всего, вернет информацию с помощью той же техники передачи пакетов TCP/IP в пакетах Ethernet с адресами MAC. Это продолжается вновь и вновь с тысячами пакетов каждую секунду для передачи потока данных. Размер стандартного пакета Ethernet составляет только 1500 байтов (это примерно 1,5 килобайта или 0,0015 мегабайта), поэтому можно представить, сколько пакетов необходимо для передачи даже небольшого файла.

Система имен доменов (DNS). Для сетей TCP/IP система имен доменов (DNS) делает сетевую адресацию значительно более доступной для восприятия человеком. По сути DNS

является всемирной сетью серверов доменов с единственной задачей поддержания удобных для людей имен хостов и облегчения поиска конкретных сетевых IP-адресов. Если вы провели некоторое время в Интернете, то знакомы с соглашением об именах DNS. Любое сетевое устройство может иметь имя хоста, но только сетевым устройствам, предоставляющим службу, которую нужно легко находить в Интернете, должны задаваться имена на DNS-сервере. Сайты и почтовые серверы являются наиболее распространенными устройствами, имеющими записи DNS. Иерархическое соглашение об именах DNS связано с иерархической структурой архитектуры доменов DNS. Как известно, имена DNS разбиты на метки, разделенные точками. Каждая метка представляет различный уровень, или домен иерархии DNS. На самом верху иерархии DNS находится «корневой» или «.» домен. Имена, которые являются частью корневого домена, — хорошо знакомые сокращения в конце почти каждого ресурса Интернета. Распространенные примеры: com, edu, .gov и другие, включая двухбуквенные коды стран. Эти домены верхнего уровня (TLD - *top-level domains*) управляются консорциумом коммерческих и правительственных организаций. Под TLD-доменами отдельные организации или пользователи размещают или арендуют собственный домен DNS. Например, компания Apple поддерживает несколько серверов DNS, которые известны серверам TLD и используются для поддержки домена apple.com. Компания Apple может разместить неограниченное число имен хостов в домене apple.com. Она может также создать неограниченное число доменов, помещая любой текст перед apple.com. Когда локальному сетевому устройству необходимо разрешить DNS-имя в соответствующий IP-адрес, оно посылает запрос имени по IP-адресу сервера DNS. IP-адрес для сервера DNS обычно конфигурируется вместе с другой адресной информацией TCP/IP для сетевого устройства. DNS-сервер сначала выполнит поиск в своих локальных и кэшированных записях имен. Если запрошенное имя не будет найдено локально, сервер запросит другие серверы доменов в иерархии DNS. Этот процесс может занять некоторое время, поэтому серверы DNS будут временно кэшировать все имена, которые

они недавно разрешали, чтобы предоставлять более быстрый ответ будущим запросам. Запрос DNS-серверу на разрешение IP-адреса по заданному имени хоста называется *прямым просмотром (forward lookup)*, в то время как запрос DNS-серверу на разрешение имени хоста по заданному IP-адресу называется *обратным просмотром (reverse lookup)*. При начальной конфигурации сетевые клиенты будут запрашивать на сервере DNS обратный поиск своего IP-адреса, чтобы определить, имеет ли сетевой клиент собственное имя DNS.

Протокол динамической настройки хоста (DHCP). Протокол динамической конфигурации узла-хоста (DHCP) используют почти все клиенты сети для автоматизации предварительной конфигурации TCP/IP. В некоторых ситуациях администратор может ввести информацию о конфигурации сети TCP/IP вручную. Обычно это делается для сетевых устройств, которые предоставляют сетевые службы. Однако конфигурирование множества сетевых клиентов вручную – трудоемкая работа, которая не застрахована от ошибок, поэтому даже в строго управляемых сетях DHCP широко используется для конфигурирования сетевых клиентов. Для обеспечения этой службы требуется сервер DHCP. Во многих сетях службу DHCP предоставляют маршрутизаторы, но в некоторых случаях для этой цели может выделяться отдельный сервер. Когда сетевое устройство становится активным в сети, оно сначала устанавливает связь с аппаратным интерфейсом, а затем отправляет широковещательный сигнал в локальную сеть, запрашивая DHCP-информацию. Так как новый клиент сети еще не имеет IP-адреса, он использует для идентификации MAC-адрес сетевого интерфейса. Если сервер DHCP, который получает сообщение, имеет доступные адреса, то он пошлет клиенту ответ с информацией о конфигурации TCP/IP. Эта информация как минимум содержит IP-адрес, маску подсети, маршрутизатор и значение минимального времени, которое определяет, как долго клиент может сохранять адрес, прежде чем вернет его. Вспомогательная информация DHCP может включать информацию DNS, информацию о службе каталогов и информацию NetBoot. Если сервер DHCP не имеет свободных сетевых адресов или служба DHCP отсутствует (как в случае

небольших или специальных сетей), клиент автоматически сгенерирует случайный адрес локальной связи. Адреса локальной связи всегда находятся в диапазоне IP-адресов 169.254jсхх.ххх. Сетевой клиент будет автоматически генерировать случайный адрес локальной связи и затем проверять локальную сеть, чтобы убедиться, что ни одно другое сетевое устройство не использует этот адрес. Когда уникальный адрес локальной связи установлен, сетевой клиент сможет устанавливать соединения только с другими сетевыми устройствами в локальной сети.

Базовая конфигурация сети. Изначальная конфигурация сети выполняется приложением Ассистент настройки (Setup Assistant), которое запускается при первой загрузке нового компьютера Mac или после «чистой» установки системы Lion. Ассистент настройки (Setup Assistant) делает процесс конфигурации настроек сети легким даже для новичка. Если после начальной настройки необходимо изменение сети, можно использовать утилиту Ассистент настройки сети (Network Setup Assistant), которая поможет выполнить процесс конфигурации сети. Доступ к этой утилите можно получить, щелкнув на кнопке Ассистент (Assist me) внизу окна системных параметров Сеть (Network). Вы должны быть знакомы со всеми параметрами сетевой конфигурации, чтобы справиться с любыми потенциальными сетевыми проблемами. С выпуском одной из предыдущих версий, Mac OS X v10.5, компания Apple произвела консолидацию ранее отдельных окон настройки сети в новую унифицированную панель настроек Сеть (Network). Таким образом, теперь все настройки сети находятся на одной панели. Панель настроек Сеть (Network) является заблокированной, тем самым показывая, что только администраторы имеют доступ к данным настройкам и что DHCP включен по умолчанию для интерфейсов Ethernet и Wi-Fi.

Использование сетевых интерфейсов.Использование аппаратных сетевых интерфейсов. Устройства Mac имеют давнюю историю использования встроенных сетевых подключений. Компания Apple начала поставлять Ethernet на Mac в 1991 году.

OS X содержит встроенную поддержку для следующих аппаратных сетевых интерфейсов:

- **Bluetooth.** Этот относительно низкоскоростной беспроводный интерфейс стал популярным стандартом связи на близких расстояниях. Каждый современный Mac, который включает поддержку Wi-Fi, также включает Bluetooth. Lion поддерживает Bluetooth в качестве сетевого моста с некоторыми мобильными телефонами и точками доступа, например, со смартфонами iPhone, которые могут предоставлять доступ в Интернет через сеть мобильного оператора связи.

- **Ethernet.** Ethernet является семейством стандартов IEEE 802.3, которые определяют самые современные проводные LAN. Каждый Mac с 1997 г. включает стандартное встроенное соединение Ethernet, некоторые модели обладают несколькими портами Ethernet. Единственным исключением из этого правила является современный MacBook Air, для которого при желании понадобится приобрести сетевой адаптер Apple USB Ethernet Adapter.

- **Внешний (аналоговый) модем.** Ни один поставляемый в настоящее время Mac не содержит аналоговый модем, но он доступен через дополнительный адаптер USB и по-прежнему поддерживается в Mac OS X. В течение многих лет аналоговый модем был стандартным методом соединения с Интернетом для домашних пользователей. С распространением высокоскоростных и беспроводных соединений с Интернетом аналоговые модемы остались только в самых удаленных местах или же используются для факсимильной связи.

- **FireWire.** FireWire является торговым названием Apple для стандарта связи IEEE 1394. Хотя это и не часто встречающийся сетевой стандарт, Mac OS X включает программное обеспечение, которое позволяет создавать небольшие одноранговые сети с помощью соединенных гирляндой кабелей FireWire. FireWire является стандартом на многих моделях Mac.

- **USB.** Хотя технически это не стандарт подключения к сети, ОС Lion поддерживает массу USB-адаптеров, предоставляющих доступ в Интернет через оператора мобильной связи. Пример такого устройства – смартфон iPhone.

– **Wi-Fi.** Ранее называемый *AirPort* (маркетинговое название, используемое компанией Apple), Wi-Fi – это более распространенное название для семейства стандартов беспроводной связи IEEE 802.11, которые используются в большинстве беспроводных сетей. Название *AirPort* продолжает использоваться для семейства базовых станций Apple, *AirPort Express* и *AirPort Extreme*. Каждый настольный или портативный компьютер Mac с 2006 г. включает стандартный встроенный адаптер Wi-Fi. Wi-Fi также остается опцией для компьютеров Mac Pro.

Использование виртуальных сетевых интерфейсов.

Виртуальные сетевые интерфейсы являются логическими сетями в аппаратном сетевом интерфейсе. Виртуальный сетевой интерфейс позволяет определить другой уникальный сетевой интерфейс, выделив часть установленного сетевого соединения. Одни виртуальные сетевые службы увеличивают безопасность, шифруя данные перед отправкой по IP-сети, другие разделяют или объединяют сетевой трафик в соединениях LAN. OS X включает клиентское программное обеспечение, которое позволяет соединяться со многими распространенными виртуальными сетевыми службами и создавать виртуальный сетевой интерфейс.

OS X включает встроенную поддержку для следующих виртуальных сетевых интерфейсов:

– ***Протокол точка-точка через Ethernet (PPPoE)***– используется в основном для непосредственного соединения Mac с модемом, обеспечивающим высокоскоростное соединение DSL с Интернетом.

– ***Виртуальная частная сеть (VPN)*** – наиболее часто используемая виртуальная сетевая служба. VPN-сети используются в основном для создания защищенных виртуальных подключений к частным локальным сетям через Интернет.

– ***Виртуальная локальная вычислительная сеть (VLAN)*** – реализация VLAN в Lion позволяет определить отдельные независимые LAN на одном сетевом аппаратном интерфейсе.

- **Агрегация каналов** – позволяет определить один виртуальный интерфейс LAN, использующий несколько аппаратных сетевых интерфейсов.

- **6 to 4**– создает VPN для передачи пакетов IPv6 через сеть IPv4. При использовании соединения 6 to 4 безопасность не повышается, но создается впечатление, что Mac непосредственно соединен с удаленной IPv6 LAN.

Использование сетевых протоколов. Каждый аппаратный или виртуальный сетевой интерфейс предоставляет соединение для ряда стандартных сетевых протоколов. Окно системных настроек Сеть (Network) показывает настройки основного протокола, когда вы выбираете службу из списка служб, однако многие параметры конфигурации протоколов доступны только при щелчке на кнопке Дополнительно (Advanced).

OS X включает встроенную поддержку следующих сетевых протоколов:

- **TCP/IP**, сконфигурированный через DHCP. TCP/IP, является основным сетевым протоколом для сетей LAN и WAN, а DHCP – популярной сетевой службой, которая будет автоматически конфигурировать клиентов TCP/IP.

- **TCP/IP**, сконфигурированный вручную. Если в локальной сети нет службы DHCP или если настройки TCP/IP не должны изменяться, можно вручную сконфигурировать настройки TCP/IP.

- **DNS**. Предоставляет имена хостов для устройств IP-сети. Настройки DNS часто конфигурируют вместе с настройками TCP/IP либо с помощью DHCP, либо вручную. Mac OS X поддерживает несколько серверов DNS и доменов поиска.

- **Wireless Ethernet (Wi-Fi)**. Беспроводной характер Wi-Fi часто требует дополнительной настройки для облегчения выбора сети и аутентификации.

- **Аутентифицированный Ethernet через 802. IX**. Протокол 802. IX используется для защиты сетей Ethernet, разрешая только правильно аутентифицированным сетевым клиентам соединяться с LAN.

- **Network Basic Input/Output System (NetBIOS) и Windows Internet Naming Service (WINS)**. NetBIOS и WINS

являются протоколами, которые чаще всего используют компьютеры на основе Windows для предоставления сетевой идентификации и обнаружения служб.

- **Прокси IP.** Серверы прокси действуют как посредники между сетевым клиентом и запрошенной службой и используются для улучшения производительности или предоставления дополнительного слоя безопасности.

- **Аппаратные вариации Ethernet.** Lion поддерживает как автоматическую, так и ручную настройку оборудования Ethernet.

- **Подключения к сетям операторов мобильной связи.** Это не столько конкретный сетевой протокол, сколько тип подключения, который использует различные протоколы. ОС Lion поддерживает несколько методов получения к Интернету через сотовые сети. К ним относятся коммутируемые сетевые соединения Bluetooth (dial-up networking, DUN), персональные сети Bluetooth (personal area network, PAN) и USB-адаптеры для подключения к сети оператора мобильной связи. Примером устройства, которое может предоставлять доступ в Интернет через сеть оператора мобильной связи, является iPhone.

- **Внешний (аналоговый) модем с PPP.** Аналоговые модемы требуют конфигурации вручную для активации соединения.

- **Протокол точка-точка (PPP).** Так как PPP используется для соединения аналогового модема и соединения Bluetooth DUN, требуется дополнительная конфигурация.

Решение сетевых проблем. Устранение неполадок сети с помощью панели настроек. Самое важное, что следует запомнить при устранении проблем с сетью, что это зачастую не вина компьютера. Существует множество точек отказа при рассмотрении проблем с подключением к локальной сети и Интернету. Поэтому второе, что следует помнить: необходимо сначала определить причину проблемы, а лишь затем пытаться решить ее.

Проблемы с сетью можно разделить на три общие категории:

– **Локальные проблемы.** Обычно связаны с неправильно сконфигурированными сетевыми настройками или разъединенными сетевыми соединениями.

– **Сетевые проблемы.** Самые трудные для решения, так как могут иметься сотни точек отказа. В этом случае всегда помогает знакомство с физической топологией сети. Начните с проверки устройств, предоставляющих доступ в сеть, которые находятся ближе всего к вашему Мас. Проблему может вызывать что-нибудь простое типа неработающего порта Ethernet на сетевом коммутаторе. Исследуя устройства, вы перемещаетесь все дальше от своего Мас и скоро обнаружите, что лучше всего начать исследование, используя сетевые диагностические утилиты, поставляемые вместе с OS X.

– **Проблемы служб.** Связаны с реальным сетевым устройством или службой, к которой вы пытаетесь обратиться. Например, устройства, предоставляющие службы DHCP или DNS, могут быть временно выключены или неправильно сконфигурированы. Чтобы убедиться, что проблема связана только с конкретной службой, проверьте другие сетевые службы. Если другие сетевые службы работают, то вы, вероятно, имеете дело не с сетевой и не с локальной проблемой. В данном случае OS X также предоставляет несколько полезных диагностических инструментов для тестирования доступности служб. Решение проблем с сетевыми службами также рассматривается в уроке

Для диагностики сетевых проблем в OS X используются три основных инструмента: *Сетевые настройки*, *мастер Диагностика сети* и *Сетевая утилита*. Устранение неполадок с помощью панели настроек Сеть (Network). Инструментом диагностики, к которому всегда следует прибегать в первую очередь, является панель настроек Сеть (Network). На ней присутствует динамически обновляющийся список, в котором показывается текущее состояние всех сетевых интерфейсов. Если сетевое подключение не работает, то проще всего обнаружить это именно здесь.

Индикаторы состояния сети включают следующее:

– **Зеленый индикатор**– подключение активно и сконфигурировано с настройками TCP/IP. Однако это еще не

гарантирует, что служба использует правильные настройки. Например, на предыдущем рисунке интерфейс Ethernet отображается с зеленым индикатором, который показывает, что для него были заданы допустимые настройки TCP/IP. Тем не менее, этот интерфейс использует адрес IP, который назначил себе сам, не получив нужных данных от ДНСР. Если вы испытываете проблемы с этой службой, сделайте двойной щелчок на настройках сети. Если считаете, что эти настройки правильные, перейдите к другим диагностическим утилитам.

– **Желтый индикатор** – соединение активно, но настройки TCP/IP сконфигурированы неправильно. Дважды проверьте все сетевые настройки. Если вы исправили ошибку, значок статуса станет зеленым.

– **Красный индикатор** – проблемы обычно связаны либо с неправильно сконфигурированными сетевыми настройками, либо с отключенными сетевыми интерфейсами. Если это интерфейс с постоянным подключением, проверьте наличие самого подключения. Если это виртуальное соединение или соединение PPP, дважды проверьте настройки и попробуйте соединиться снова. Использование мастера Диагностика сети (Network Diagnostics).

OS X включает мастер Диагностика сети (Network Diagnostics), который помогает решать типичные сетевые проблемы. Некоторые сетевые приложения автоматически открывают этот мастер, когда сталкиваются с проблемой сети. Можно также открыть его вручную, щелкнув на кнопке Ассистент (Assist Me) внизу панели настроек Сеть (Network), а затем на кнопке Диагностировать (Diagnostics). Мастер Диагностика сети (Network Diagnostics) задаст несколько простых вопросов о настройке сети и выполнит ряд тестов для определения, где могла возникнуть проблема. Результаты тестов выводятся с помощью цветных индикаторов на левой стороне окна. Если действительно имеются проблемы, мастер предложит способы их решения.

Использование Сетевой утилиты (Network Utility).

Системные настройки Сеть (Network) и мастер Диагностика сети (Network Diagnostics) хороши для решения понятных сетевых проблем, однако наиболее мощным приложением в OS

X для их диагностики являются Программы/Утилиты/Сетевая утилита (/Applications/Utilities/ Network Utility). Сетевая утилита (Network Utility) предоставляет набор популярных сетевых инструментов идентификации и диагностики. Фактически большинство инструментов в Сетевой утилите (Network Utility) основывается на сетевых утилитах командной строки UNIX, которые используются сетевыми администраторами много лет.

Сетевая утилита (Network Utility) разбита на следующие разделы.

- **Свойства (Info)**– позволяет проанализировать детали аппаратных сетевых интерфейсов.

- **Netstat** – показывает информацию о маршрутизации и сетевую статистику.

- **Ping** – фундаментальный инструмент решения сетевых проблем, позволяет протестировать подключение к сети и задержку.

- **Lookup** – очень важный инструмент, который позволяет протестировать разрешение доменных имен через сервер DNS.

- **Traceroute** – этот мощный инструмент позволяет анализировать, как сетевые соединения маршрутизируются в свое место назначения.

- **Whois** – позволяет запрашивать серверы базы данных whois и определять владельца имени домена DNS или IP-адреса.

- **Finger** – позволяет собрать информацию из сетевой службы на основе имени учетной записи пользователя.

- **Port Scan**– наиболее важный инструмент для определения того, имеет ли сетевое устройство доступные службы.

Программу Сетевая утилита (Network Utility) можно также открыть, когда Mac загружен с раздела восстановления системы Lion Recovery, как рассматривалось в уроке 1. Чтобы сделать это, нужно в меню Утилиты (Utilities) выбрать команду Сетевая утилита (Network Utility). Однако при работе в системе восстановления OS X Recovery нет доступа к панели настроек Сеть (Network). Это означает, что Mac автоматически активирует встроенные проводные Ethernet-подключения и попытается получить конфигурацию через службу DHCP.

В качестве альтернативы доступно меню Wi-Fi, которое позволяет пользователю временно подключаться к беспроводным сетям.

– **Интерфейс Сетевой утилиты (Network Utility)**

При запуске Сетевой утилиты (Network Utility) вы сначала увидите раздел Свойства (Info), который позволяет увидеть подробный статус любого аппаратного сетевого интерфейса. Даже при запуске Сетевой утилиты (Network Utility) для использования другого раздела всегда уделяйте некоторое время, чтобы проверить, правильно ли активирован сетевой интерфейс. Начните с выбора из всплывающего меню интерфейса, с которым возникла проблема. Вы заметите, что варианты выбора здесь не всегда совпадают с именами служб, заданными на панели настроек Сеть (Network). Вместо этого в данном меню интерфейсы показываются с использованием типа интерфейса и заданного UNIX-имени. При правильной работе интерфейс его должен быть первым внутренним портом Ethernet и в большинстве случаев интерфейс en1 является интерфейсом Wi-Fi. Если есть компьютер Mac с двумя внутренними Ethernet-портами, второй будет иметь имя en1, а интерфейс Wi-Fi будет иметь имя en2. Интерфейс Fire Wire будет иметь имя fw0. Выбрав интерфейс, можно увидеть общую информацию интерфейса слева и статистику передачи справа. Нас интересуют параметры Статус подключения (Link Status), Скорость подключения (Link Speed) и IP-адрес (IP Address(es)). Будут отображаться только активные аппаратные сетевые интерфейсы, а скорость связи будет показывать, правильно ли подключен интерфейс. Очевидно, что для установления соединения TCP/IP требуется правильный IP-адрес. Можно также определить MAC-адрес выбранного интерфейса, который используется для его идентификации в локальной сети.

В качестве последней проверки выбранного сетевого интерфейса можно посмотреть статистику последних передач. Если вы запускаете другие сетевые приложения для получения некоторого сетевого трафика, то сможете убедиться, что на этом интерфейсе пакеты отправляются и принимаются. Если вы видите здесь активность, однако по-прежнему испытываете проблемы, это связано, скорее всего, с проблемой сетевой

службы, а не с реальным сетевым интерфейсом. Если этот интерфейс демонстрирует ошибки передачи, то источником проблемы может быть локальное соединение сетевого оборудования. Для решения проблем аппаратного сетевого интерфейса всегда начинайте с проверки физического соединения. Для проводных сетей проверьте различные сетевые порты или кабели, чтобы исключить аппаратную составляющую проблемы. Для беспроводных сетей дважды проверьте настройки Wi-Fi и конфигурацию всех беспроводных базовых станций. В исключительных случаях может оказаться, что сетевое оборудование Mac работает неправильно, тогда необходимо доставить Mac в авторизованный сервисный центр компании Apple.

Сетевая утилита ping. Если вы определили, что сетевые настройки верно сконфигурированы и аппаратный сетевой интерфейс работает правильно, но тем не менее испытываете сетевые проблемы, следующий шаг состоит в тестировании сетевого соединения с помощью инструмента ping. Инструмент ping является наиболее фундаментальным сетевым тестом для определения того, может ли Mac успешно посылать и получать данные по сети. Mac отправит пакет данных ping по целевому IP-адресу, а другое устройство должно вернуть пакет ping, подтверждая тем самым наличие подключения.

Для использования утилиты ping выполните следующие шаги:

1. Запустите /Программы/Утилиты/Сетевая утилита (/Applications/Utilities/Network Utility) и затем щелкните на вкладке Ping сверху.

2. Введите IP-адрес или имя хоста сетевого устройства для тестирования соединения с этим устройством.

Начните с ввода IP-адреса постоянно доступного устройства в LAN, т. е. сетевого маршрутизатора.

Имейте в виду, что использование имени домена предполагает, что компьютер Mac правильно взаимодействует с DNS-сервером. Это, однако, может не соответствовать действительности при проблемах с подключением.

3. Щелкните на кнопке Ping, чтобы инициировать процесс ping.

Если команда `ping` выполнялась успешно, она должна вернуть значение времени, которое потребовалось пакету `ping` для перемещения к сетевому устройству и обратно. Это время обычно составляет миллисекунды, поэтому получение времени `ping` более одной секунды является необычным.

4. После выполнения успешного обмена пакетами `ping` с локальными устройствами можно перейти к адресам WAN или Интернета. Инструмент `ping` позволит вам убедиться, что работает все, за исключением той службы, которая и стала причиной проблемы в сети.

Сетевая утилита *Lookup*. Если вы можете успешно выполнить `ping` других сетевых устройств по их IP-адресу, но попытка соединиться с другим устройством по имени хоста не работает, то проблема связана с DNS. Просмотр в сети (`network lookup`) позволяет протестировать разрешение имени на сервере DNS. Для использования *Lookup* выполните следующие шаги:

1. Запустите Программы/Утилиты/Сетевая утилита (`/Applications/Utilities/Network Utility`) и затем щелкните на вкладке *Lookup* сверху.

2. Введите IP-адрес или имя хоста сетевого устройства, чтобы протестировать разрешение DNS. Начните с введения имени хоста устройства или службы в локальном домене. Если удастся разрешить имена локальных хостов, но не имена хостов Интернета, то это указывает, что локальный сервер DNS разрешает локальные имена, но не соединен правильно со всемирной сетью DNS. Если локальный домен отсутствует, можно использовать любое имя хоста Интернета.

3. Щелкните на кнопке *Lookup*, чтобы инициировать процесс просмотра в сети.

Успешный прямой просмотр вернет IP-адрес введенного имени хоста. Успешный обратный просмотр вернет имя хоста для введенного IP-адреса.

4. Если не удастся получить ответ любого из просмотров, то это значит, что Mac не подключен к DNS-серверу. Можно протестировать соединение, отправив пакет `ping` по IP-адресу DNS-сервера.

Сетевая утилита *Traceroute*. Если вы можете соединиться с одними сетевыми ресурсами, а с другими – нет, утилита

Traceroute позволит определить, где разрывается соединение. Помните, что соединения WAN и Интернет требуют пересылки данных через множество сетевых маршрутизаторов до места назначения. Инструмент Traceroute будет проверять каждый сетевой переход между маршрутизаторами с помощью инструмента ping, чтобы определить, где соединения отказывают или замедляются.

Для использования Traceroute выполните следующие шаги:

1. Запустите Программы/Утилиты/Сетевая утилита (/Applications/Utilities/Network Utility) и затем щелкните на вкладке Traceroute сверху.

2. Введите IP-адрес или имя хоста сетевого устройства, чтобы протрассировать соединение с этим устройством. Начните с ввода IP-адреса постоянно доступного устройства в LAN, такого как сетевой маршрутизатор.

Имейте в виду, что использование имени домена предполагает, что компьютер Mac правильно взаимодействует с DNS-сервером. Это может не соответствовать действительности при проблемах с подключением.

3. Щелкните на кнопке Trase, чтобы инициировать процесс traceroute.

Если трассировка (traceroute) прошла успешно, она должна вернуть список маршрутизаторов, которые требуются для выполнения соединения, и время, которое требуется пакету ping для перемещения до каждого сетевого маршрутизатора. Заметим, что на каждое расстояние отправляются три пробы, поэтому для каждого перехода будет перечислено по три значения времени. В данном случае задержка также обычно измеряется в миллисекундах, поэтому задержка больше одной секунды будет необычным значением.

4. Когда маршруты к локальным устройствам будут успешно проложены, можно переключиться на адреса WAN или Интернета. С помощью инструмента traceroute можно выяснить, что причиной проблемы является конкретный маршрутизатор.

Примечание: Некоторые сетевые администраторы рассматривают чрезмерное количество запросов ping как угрозу, поэтому многие из них настраивают свои сетевые экраны или устройства так, чтобы не отвечать на любые сетевые запросы ping.

Глава 7

ФАЙЛОВЫЕ СИСТЕМЫ

С момента появления первого Mac в 1984 г. частота процессора персонального компьютера увеличилась примерно в тысячу раз, а емкость устройств хранения – более чем в миллион раз. Рассмотрим технологии хранения данных в OS X.

7.1. Компоненты файловой системы

Для управления хранилищем данных OS X, важно понимать различие между запоминающим устройством (*storage*), разделами (*partitions*) и томами (*volumes*). Традиционно запоминающее устройство компьютера определялось накопителями на дисках. Кроме накопителей на дисках популярными стали и другие, более удобные форматы съемных запоминающих устройств вследствие увеличения их емкости. К ним относятся оптические носители (CD- и DVD-диски) и твердотельные запоминающие устройства (SSD-диски, USB-диски и карты CompactFlash). Все они в равной степени поддерживаются в OSX. Однако без правильного форматирования любая технология хранения будет не более чем большим пустым набором из единиц и нулей. Форматирование – это процесс применения к запоминающему устройству некоторой логической схемы в виде разделов и томов. Разделы используются для задания границ на запоминающем устройстве. Можно определить несколько разделов, чтобы физическое запоминающее устройство выглядело как несколько отдельных хранилищ. Внутри созданных разделов система может создать тома (логические разделы). Тома определяют, как файлы и папки реально хранятся на физических устройствах. В конечном счете именно том монтируется файловой системой и представляется в виде значка хранилища в приложении Finder. Очевидно, что запоминающее устройство с несколькими разделами, каждый из которых содержит отдельный том, будет отображаться в приложении Finder в виде нескольких значков хранилищ.

Схемы разбиения. Диски должны разбиваться на разделы. На каждом диске должен быть, как минимум, один раздел, однако OS X поддерживает до 16 разделов на диске. OS X поддерживает три типа схем разбиения:

- GUID Partition Table (GPT) – схема разбиения по умолчанию, используемая компьютерами Mac на базе процессоров Intel. Это единственная схема разделов, которая позволяет загружаться компьютерам на базе процессоров Intel, использующим диски в качестве запоминающего устройства.

- Apple Partition Map (APM) – схема разбиения, используемая по умолчанию компьютерами Mac на базе процессоров PowerPC. Это единственная схема разбиения, с которой может загружаться компьютер Mac на базе PowerPC. Однако все компьютеры Mac на базе Intel также могут использовать этот тип разбиения.

- Master Boot Record (MBR) – схема разбиения, используемая по умолчанию большинством неMac-компьютеров, в том числе совместимыми с Windows. Данная схема разбиения по умолчанию применяется к дискам, которые продаются предварительно отформатированными, и используется периферийными устройствами, в которых данные хранятся на флэш-дисках, такими как цифровые фотоаппараты или смартфоны. Хотя Mac не может загрузиться с этого типа разбиения, все компьютеры Mac могут обращаться к данным на разделах, созданных с помощью схемы MBR.

Компьютеры Mac на базе Intel могут загружаться с внешних дисков, подключенных через интерфейсы USB и FireWire.

Форматы томов. Формат тома определяет, как файлы и папки сохраняются на диск. OS X поддерживает чтение и запись следующих форматов:

- **Mac OS Standard (HFS)** – устаревший формат тома, используемый классической операционной системой Mac OS. Этот формат не может использоваться в качестве загрузочного тома для OS X Lion.

- **Mac OS Extended** – является унаследованным форматом тома, созданным и поддерживаемым компанией Apple для компьютеров Macintosh. Mac OS Extended поддерживает все

дополнительные возможности, требуемые OS X, включая имена файлов в кодировке Unicode, развитые метаданные, разрешения POSIX, списки контроля доступа (ACL), ссылки в стиле UNIX и алиасы (псевдонимы).

– **Mac OS Extended (учет пер. клавиатуры).** Этот вариант формата MacOSExtended добавляет в файловую систему учет регистра клавиатуры. Обычно формат MacOSExtended сохраняет, но не различает регистр символов. Это означает, что отформатированный обычным образом том помнит, какой регистр был выбран для символов имени файла, но не различает имена файлов, отличающиеся только регистром символов. Вариант с учетом регистра букв не рекомендуется использовать в качестве стандартного для клиентских систем Lion.

– **Mac OS SExtended (журнальный) или Mac OS Extended (учет пер. клавиатуры, журнальный).** Это вариант формата MacOSExtended с функцией журналирования файловой системы, которая помогает сохранить целостность структуры тома. В журнал записывается, какие операции с файлами (создание, расширение, удаление и т.д.) выполняются в любой заданный момент времени. Это позволяет избежать повреждения файлов и значительно сэкономить время, которое пришлось бы потратить после сбоя на выполнение процессов проверки и исправления тома. MacOSExtended (журнальный) в OS X используется по умолчанию в качестве системного тома.

– **Mac OS Extended (журнальный, зашифрованный) или Mac OS Extended (учет пер. клавиатуры, журнальный, зашифрованный).** Этот формат представляет собой вариант формата MacOSExtended, но с функцией полного шифрования диска по алгоритму XTS-AES 128. На данной технологии основана система шифрования всего диска FileVault 2.

– **File Allocation Table (FAT)** – устаревший формат тома, используемый компьютерами с операционными системами семейства Windows.

– **Extended File Allocation Table (ExFAT).** Созданная специально для больших по объему флэш-накопителей файловая система ExFAT расширяет устаревшую систему FAT для поддержки дисков, размер которых превышает 32 Гб. Многие цифровые видеокамеры используют файловую систему

ExFAT для поддержки дисков больших размеров, требуемых для записи видео высокой четкости.

- **UNIX File System (UFS)** – устаревший «родной» формат тома, поддерживаемый MacOS X. DiskUtility не поддерживает создания томов в формате UFS.

OS X поддерживает чтение следующих форматов.

- **NT File System (NTFS)**. BootCamp поддерживает загрузку Windows с NTFS-тома, но сама система OSX с такого тома загрузиться не может и не будет записывать данные на него. Более того, DiskUtility не поддерживает создание NTFS-томов.

- **ISO 9660 или Compact Disk File System (CDFS)** – общепринятый стандарт для CD-дисков, доступных только для чтения. Но следует заметить, что отформатированные на компьютере Mac CD-диски могут содержать тома в формате HFS.

- **Universa IDisk Format (UDF)** – общепринятый стандарт для DVD-дисков, доступных только для чтения. И снова следует заметить, что отформатированные на компьютере Mac DVD-диски могут содержать тома в формате HFS.

7.2. Управление файловой системой

Основным инструментом управления устройствами хранения данных в OS X является утилита /Applications/Utilities/DiskUtility.

Форматирование или переформатирование диска. При подключении неотформатированного диска Mac автоматически предложит открыть Disk Utility. Если диск уже был отформатирован, и вы хотите изменить схему разбиения или структуру тома, можно переформатировать диск, используя те же действия. Важно помнить, что переформатирование диска разрушит все форматирование, которое там уже имеется, и тем самым диск утратит все свое содержимое.

Переразбиение диска. Утилита Disk Utility имеет функцию, которая позволяет динамически переразбивать диск без утраты хранящихся на нем в данный момент данных. Эта функциональность была создана для облегчения процесса настройки Boot Camp. Установка обновления OS X также

позволяет извлечь выгоду из этой функциональности, создавая скрытую OSX Recovery HD на существующем системном диске. Если для системного диска включить систему защиты FileVault, то вы не сможете динамически переразбить диск.

Стирание диска или тома. Если требуется действительно «стереть» диск или том, необходимо записать новые несекретные данные поверх существующих. Дисковая утилита включает ряд возможностей, которые позволяют безопасно стереть старые данные. Можно безопасно стереть весь диск, весь том или только свободное пространство тома. Более того, интерфейс Очистить – это единственное место в Дисковой утилите, в котором можно создать том в формате Mac OS Extended (журналируемый, зашифрованный). Чтобы зашифровать системный диск, нужно включить систему шифрования File Vault 2. Программа Дисковая утилита (DiskUtility) не позволяет преобразовать файловую систему существующего тома на MacOSExtended (журнальный, зашифрованный). Однако команда diskutil позволяет преобразовать существующий том в зашифрованный без утраты данных. Стирание или форматирование диска не изменяет схему разбиения диска.

7.3. Использование системы FileVault 2

До внедрения ОС Lion для защиты пользовательских данных выполнялось шифрование только содержимого домашней папки пользователя. Эта система, которая теперь называется *Традиционный FileVault (LegacyFileVault)*, имела ряд недостатков. Систему шифрования FileVault 2 нельзя включить, если для кого-либо из локальных пользователей еще активна система Традиционный FileVault (LegacyFileVault).

Система FileVault 2 защищает весь системный диск путем его преобразования в формат MacOSExtended (журнальный, зашифрованный), который использует шифрование по стандарту XTS-AES 128. Этот переход к схеме шифрования всего диска устраняет все ограничения старой версии FileVault, поскольку шифрование происходит на уровне драйвера файловой системы. Решение FileVault 2 компании Apple – это

больше, чем просто новый формат диска; это система из новых технологий, которая позволяет компьютеру Mac перейти от стандартного системного диска к защищенному. Основные технологии FileVault 2: бесшовное преобразование форматов томов, синхронизация паролей учетных записей пользователей, защищенное хранилище ключей на серверах Apple для восстановления утерянных паролей, OSXRecovery для начального запуска системы и новое окно ввода пароля прошивки. Со многими этими особенностями читатель встретится при загрузке с зашифрованного системного диска. Раньше процесс загрузки системы был традиционно прямолинейной задачей. Теперь же все переходит на новый уровень сложности, когда не удастся напрямую прочесть системный загрузочный диск. Поэтому в компании Apple придумали способ проходить проверку подлинности и дешифровать защищенный системный диск при загрузке. В этом случае Mac на самом деле сначала загружается с раздела OSXRecoveryHD и отображает пользователю окно входа в систему. Пользователь вводит пароль своей учетной записи, который затем используется для доступа к ключу расшифровки, с помощью которого выполняется разблокировка и расшифровка защищенного системного тома. Когда Mac получает доступ к системному тому, загрузка продолжается обычным путем с одним исключением: пользователь, который уже ввел свои учетные данные для разблокировки шифрована автоматически, войдет в систему под своей учетной записью. Процесс загрузки FileVault 2 требует, чтобы системный диск содержал скрытый раздел OSXRecovery HD. Потому, если на системном диске нет скрытого раздела OSXRecovery HD, включить систему FileVault 2 нельзя. К счастью, OS X делает невозможном случайное включение FileVault 2, если не соблюдено это требование.

7.4. Разрешения файловой системы

Технологии, известные как *«разрешения файловой системы»*, используются в OS X для управления авторизацией файлов и папок. Разрешения файловой системы работают

совместно с технологиями учетных записей пользователей, которые управляют идентификацией и аутентификацией, предоставляя безопасную мультипользовательскую рабочую среду. Разрешения файловой системы, как и учетные записи пользователей, охватывают все уровни операционной системы, поэтому ее доскональное исследование необходимо для полного понимания OSX. Каждый отдельный объект на системном томе имеет заданные для него операционной системой правила разрешений. Только пользователи и процессы с доступом учетной записи root могут игнорировать эти правила. Эти правила используются для определения доступа к файлам и папкам для каждого стандартного пользователя, администратора, пользователя guest или пользователя только для общего доступа. Любой пользователь может определить разрешения файла или папки с помощью окна свойств (GetInfo) утилиты Finder. Принадлежность для разрешений. Каждый файл и папка принадлежат, как минимум, одному владельцу и одной группе, а также имеют параметр владения для всех остальных пользователей. Эта трехуровневая структура принадлежности создает основу для разрешений файловой системы.

– **Owner** – по умолчанию владельцем (owner) объекта является пользователь, который создал или скопировал объект на Mac. Например, пользователь владеет большинством объектов в своей домашней папке. Пользователь root почти всегда владеет объектами операционной системы, включая системные ресурсы и приложения. OS X предоставляет каждому администратору возможность изменять принадлежность и разрешения независимо от того, кто является владельцем объекта.

– **Group** – по умолчанию группа (group) объекта наследуется от папки, в которой он был создан. Поэтому большинство объектов принадлежат группам staff, wheel или admin. Принадлежность к группе позволяет пользователям, не являющимися владельцами, получить доступ к объекту. Например, хотя папкой /Программы (/Applications) владеет пользователь root, ее группа задана как admin, поэтому администраторы могут вносить изменения в содержимое этой папки.

– **Everyone** – параметр Все (Everyone) задает доступ для любого пользователя, который не является владельцем объекта и не является членом группы, которой принадлежит объект. Другими словами, он означает «все остальные». Сюда включаются локальные пользователи, пользователи только для общего доступа и пользователь guest.

Стандартные (POSIX) разрешения. Базовая структура разрешений файловой системы в OS X основывается на использующихся десятилетиями разрешениях в стиле UNIX. Данная система иногда также называется *разрешениями в стиле POSIX*. Для владельца, группы и всех остальных пользователей задаются отдельные конкретные права доступа для каждого файла или папки. Более того, в связи с собственной иерархией, встроенной в файловую систему, где папки могут располагаться внутри других папок, можно легко создать сложную файловую структуру, которая обеспечивает различные уровни доступа и безопасности. Однако компания Apple ограничила утилиту Finder наиболее распространенными вариантами разрешений. С помощью утилиты Finder можно назначить файлу следующие разрешения:

– **Чтение и запись (ReadandWrite)** – пользователь может открыть файл и сохранить изменения.

– **Только чтение (ReadOnly)** – пользователь может открыть файл, но не может сохранить никакие изменения.

– **Нет доступа (NoAccess)** – пользователь или члены группы не имеют доступа к файлу.

С помощью утилиты Finder можно назначить папке следующие разрешения.

– **Чтение и запись (ReadandWrite)** – пользователь или члены группы могут просматривать и вносить изменения в содержимое папки.

– **Только чтение (ReadOnly)** – пользователь или члены группы могут просматривать содержимое папки, но не могут вносить изменения в ее содержимое.

– **Только запись (Почтовый ящик) (WriteOnly [DropBox])** – пользователь или члены группы не могут просматривать папку, но могут копировать или перемещать в нее объекты.

– **Нет доступа (NoAccess)**– пользователь или члены группы не имеют доступа к содержимому папки.

Finder предполагает, что доступ «исполнение» идет одновременно с доступом «чтение» для папок.

Списки контроля доступа (ACL). Списки контроля доступа (ACL) были разработаны для расширения базовой архитектуры разрешений в стиле UNIX для дополнительного контроля над доступом к файлам и папкам. Хотя и не существует общего стандарта для ACL-списков, OS X использует стиль ACL, аналогичный тому, который используется в файловых системах NTFS, используемых в Windows и в UNIX-системах, которые поддерживают NFSv4. Эта реализация ACL является чрезвычайно гибкой, но она увеличивает сложность, добавляя более десяти уникальных типов атрибутов привилегий и наследования. Более того, она поддерживает неограниченное число ACL-атрибутов для любого пользователя или группы. Наконец, важно отметить, что ACL-правило, применяемое к пользователю или группе, преобладает над традиционными разрешениями UNIX. Однако пользователи или группы, к которым не применены ACL-правила, управляются текущими разрешениями UNIX. Обычным пользователям не нужны все возможности, доступные с помощью ACL, поэтому Finder предоставляет только наиболее распространенные конфигурации ACL. Фактически Finder позволяет назначить только те ACL-атрибуты, которые соответствуют перечисленным выше наиболее распространенным конфигурациям разрешений UNIX. Единственная возможность ACL-списков, которую Finder в действительности реализует, – это возможность иметь неограниченное число правил привилегий пользователя или группы. Другими словами, утилита Finder использует архитектуру ACL, чтобы позволить сконфигурировать уникальные привилегии для неограниченного числа пользователей или групп. Управление данными. Системные файлы OSX реорганизованы в легкую для понимания структуру, которая обеспечивает улучшенную безопасность и управляемость для администратора Mac.

7.5. Структура файловой системы

Структура системы OSX обеспечивает баланс между простотой использования и развитой функциональностью. Если обычный пользователь в утилите Finder посмотрит на корень или начало файловой системы, то он увидит только четыре стандартные папки: Программы (Applications), Библиотеки (Library), Система (System) и Пользователи (Users). Содержимое этих папок представляет все, что может когда-либо понадобиться большинству пользователей и администраторов. Тем не менее, если опытный пользователь посмотрит на корень системы из командной строки, то увидит гораздо больше объектов, которые приложение Finder обычно скрывает. Таким образом, сложность и гибкость операционной системы UNIX доступна именно для тех пользователей, кому это требуется.

Ниже описываются стандартные папки в корне системы, которые видны из Finder.

– **Программы (Applications).** Данная папка, часто называемая *локальной папкой Applications*, используется по умолчанию для приложений, доступных всем локальным пользователям. Только администраторы могут вносить изменения в содержимое этой папки.

– **Библиотеку (Library).** Данная папка, часто называемая *локальной папкой Library*, используется по умолчанию для вспомогательных ресурсов системы и приложений, доступных всем локальным пользователям. Только администраторы могут вносить изменения в содержимое этой папки.

– **Система (System).** Папка содержит ресурсы, необходимые для функционирования операционной системы. Пользователям очень редко бывает нужно вносить изменения в содержимое этой папки. Даже администраторы не могут вносить изменения в объекты в папке System от своего имени.

– **Пользователи (Users).** Используемое по умолчанию место для домашних папок локальных пользователей.

– **Developer Tools.** Она не входит в стандартную установку, но является фундаментальной частью системы. Этот пакет можно найти в Mac App Store. Как и в папки Applications и

Library, в папку Developer может вносить изменения только администратор.

Типы системных ресурсов. Все специфические для OS X системные ресурсы находятся в различных папках Библиотеки (Library), разбросанных по системному тому. Системными называют любые ресурсы, которые не являются приложениями общего назначения или файлами пользователя. Назначение папки Библиотеки – держать пользовательские и системные ресурсы упорядоченными и хранить их отдельно от объектов, которые используются повседневно. Это позволяет не загромождать папку Программы (Applications) и домашние папки пользователей системными ресурсами.

Открыв любую из папок Библиотеки, можно увидеть несколько десятков категорий объектов. Перечислим некоторые важнейшие из имеющихся системных ресурсов:

- **Application Support.** Эту папку можно найти как в пользовательских, так и в системной библиотеках. В ней могут находиться любые вспомогательные данные, требуемые приложению. Здесь часто хранятся файлы справки или шаблоны для приложения.

- **Расширения (Extensions) (также называемые расширениями или модулями ядра).** Данные объекты находятся только в системной и локальной папках Библиотеки (Library). Расширения являются драйверами низкого уровня, которые подключают себя к ядру операционной системы. Расширения обеспечивают поддержку драйверов для оборудования, сети и периферийных устройств. Расширения загружаются и выгружаются автоматически.

- **Шрифты (Fonts).** В каждой папке Библиотеки (Library) находится папка шрифтов. Это файлы с описанием гарнитуры шрифта для вывода на экран и печати.

- **Frameworks.** Инфраструктуры, находящиеся в каждой папке Library, являются хранилищами общего кода, используемого различными частями операционной системы или приложениями. Загруженные в Mac инфраструктуры можно просмотреть с помощью приложения Программы/Утилиты/Профили (/Applications/Utilities/System Profiler).

– **Keychains.** Keychain-файлы находятся в каждой папке Library и служат для безопасного хранения секретной информации, такой как пароли, сертификаты, ключи, формы веб-сайта и заметки.

– **LaunchDaemons и LaunchAgents.** Данные объекты могут находиться как в локальных, так и системных библиотеках, а LaunchAgents также могут находиться и в библиотеке пользователя. Launch-объекты используются для определения многочисленных фоновых процессов Mac OS X, которые запускаются автоматически с помощью процесса launchd. Более того, каждый отдельный процесс является потомком процесса launchd. LaunchAgents-объекты (агенты запуска) предназначены для процессов, которые должны запускаться, когда пользователь находится в системе, в то время как LaunchDaemons-объекты (демоны запуска) используются для запуска процессов, которые должны всегда выполняться в фоне.

– **Журналы (Logs).** Многие системные процессы и приложения записывают сообщения о ходе своего выполнения или ошибках в файлы журналов. Их можно найти в каждой локальной папке Библиотеки (Library). Файлы журналов просматриваются с помощью приложения /Applications/Utilities/Console.

– **Панели настроек (Preferences Panes).** Файлы настроек находятся в локальных и пользовательских библиотеках и используются для хранения настроек системы и приложений. Каждая новая настройка конфигурации любого приложения или системной функции сохраняется в файле настроек. Так как настройки играют критическую роль в функционировании системы, они часто являются причиной многих программных проблем.

– **Объекты запуска (Startup Items).** Объекты запуска находятся только в локальных и системных библиотеках и являются предшественниками LaunchAgents-объектов и LaunchDaemons-объектов. С выпуском Mac OS X 10.5 компания Apple официально посоветовала отказаться от использования объектов запуска (Startup Items).

– **Иерархия системных ресурсов.** Папки библиотек и, следовательно, системные ресурсы располагаются в каждом из

четырёх доменов: user, local, network и system. Разделение ресурсов на четыре домена обеспечивает повышенную гибкость в администрировании, безопасность ресурсов и надёжность системы.

Ниже приводится описание четырёх доменов ресурсов.

– **User (пользовательский).** Каждый пользователь имеет для ресурсов собственную папку Библиотеки (Library) в своей домашней папке. К помещённым туда ресурсам есть доступ только у самого пользователя. Он может иметь также собственную папку Программы (Applications) в своей домашней папке.

– **Local (локальный).** Корневые папки Программы (Applications) и Библиотеки (Library) являются частью домена локальных ресурсов. Именно поэтому их называют *локальными папками Программы (Applications) и Библиотеками (Library)*. Любые ресурсы, помещённые в эти папки, доступны всем учетным записям локальных пользователей. По умолчанию только администраторы могут вносить изменения в локальные ресурсы.

– **Network (сетевой).** OSX обращается к системным ресурсам и приложениям из общей сетевой папки. Для того чтобы можно было включить домен ресурсов Network, администраторы должны настроить автоматически подключаемую общую папку.

– **System (системный).** Домен системы содержит все объекты, необходимые для обеспечения базовой функциональности системы. В корне системного тома есть много скрытых объектов, которые составляют домен системных ресурсов, но в Finder можно видеть только корневую папку Система (System). В большинстве случаев сюда не требуется добавлять ресурсы или управлять ими. При наличии этих четырёх разных доменов, содержащих ресурсы, существует большая вероятность их наложения, т. е. наличия нескольких копий одинаковых ресурсов, доступных в любой заданный момент времени системе и пользователю. Система спроектирована с учетом этого и выполняет поиск ресурсов в порядке от самого специфичного (т. е. в домене пользователя) к наименее специфичному (т. е. в домене системы). Если

обнаруживается несколько одинаковых ресурсов, то система будет использовать ресурс, наиболее специфический для пользователя.

7.6. Управление скрытыми объектами

OS X является полностью совместимой с UNIX операционной системой, и ей требуется ряд файлов, которые никогда не понадобятся обычным пользователям. Корневой уровень системного тома Mac заполнен ресурсами, которые требуются процессам UNIX и нужны для ее администрирования. Компания Apple сконфигурировала Finder таким образом, чтобы скрыть эти объекты от рядового пользователя. Люди, которым может понадобиться работать с данными, обычно скрытыми ресурсами, в любом случае будут для выполнения своей работы использовать интерфейс командной строки. OS X, будучи гибридом технологий UNIX и Mac OS, использует два метода для сокрытия файлов и папок. Первый метод является традицией UNIX: простое использование точки вначале имени объекта скроет данный объект. Он будет скрыт как в приложении Finder, так и при использовании опций по умолчанию для перечисления объектов с помощью команды `ls` в командной строке. Второй метод является традицией Mac OS: задействуется флажок скрытого файла. Однако этот метод скрывает объект только в Finder. Показ скрытых объектов в Finder. Для того чтобы открыть обычно скрытые объекты в Finder, существуют два метода. В первом – для открытия любой папки используется меню Перейти (Go) приложения Finder, во втором – команда `open` из командной строки для открытия любого файла или папки. Скрытие или отмена скрытия объектов. Самый легкий способ скрыть объект – присвоить ему имя с точкой в начале. Однако, чтобы не дать новичкам случайно скрыть свои объекты, приложение Finder не позволит сохранить файл с точкой в начале его имени. Следовательно, нужно использовать командную строку. Просто используйте команду `mv` для переименования объекта.

Если требуется скрыть объект из Finder, однако вы не можете изменить его имя так, чтобы оно начиналось с точки,

задействуйте флаг скрытого файла. Это также можно сделать и из командной строки.

Использование псевдонимов и ссылок. Другим примером того, что OSX является гибридом UNIX и классической Mac OS, является существование нескольких способов, используемых для указателей или ярлыков файловой системы. В сущности, ярлыки файловой системы – это файлы, которые ссылаются на другие файлы или папки. Это позволяет использовать один объект в нескольких местах или с несколькими именами без необходимости создавать несколько его копий. И операционная система, и пользователи используют преимущества ярлыков файловой системы для обращения к объектам в более удобных местах без необходимости дублировать эти объекты.

Примечание: Не путайте ярлыки в Dock и на боковой панели приложения Finder с настоящими ярлыками файловой системы. И Dock, и Finder сохраняют ссылки на исходные объекты в своих файлах конфигурации, в то время как ярлыки файловой системы являются отдельными файлами, которые могут размещаться в любом месте тома.

В OSX используются три основных типа ярлыков файловой системы:

– **Псевдонимы (Aliases).** Данные ярлыки достались в наследство от классической Mac OS, но были обновлены для текущей версии Mac OS X. Псевдонимы могут создаваться с помощью Finder, но они бесполезны в командной строке. Средства командной строки считают псевдонимы файлами данных и не знают, как переходить по ссылкам к исходным объектам. Однако псевдонимы являются более стойкими по сравнению с другими типами ярлыков. Если исходный объект будет заменен или перемещен, псевдоним практически никогда не потеряет исходный объект.

– **Символические ссылки (Symbolic Links).** Данные ярлыки являются частью традиционной файловой системы UNIX и простыми указателями на путь в файловой системе к исходному объекту. Таким образом, если переместить исходный объект, то символическая ссылка будет разорвана. Однако замена исходного объекта работает, поскольку путь остается тем же самым. Символические ссылки создаются с помощью

команды `ln`. Finder не позволяет создавать символические ссылки, однако позволяет переходить по ним к исходным объектам.

– **Жесткие ссылки (Hard Links).** Данные ярлыки также являются частью традиционной файловой системы UNIX и служат дополнительными ссылками на исходный объект. Представьте обычный файл в виде двух частей: первая – биты на физическом диске, которые составляют фактическое содержимое файла, вторая – имя, которое указывает на эти биты. Жесткая ссылка – это дополнительное имя, которое указывает на эти же биты на физическом диске. Для создания жестких ссылок также используется команда `ln`. Finder не позволяет создавать жесткие ссылки, но может переходить по ним. Примером использования жестких ссылок в OS X являются архивы Time Machine. Приложение Time Machine использует жесткие ссылки для ссылки на объекты, которые не изменились с момента предыдущей операции архивирования. Тем самым значительно экономится место на диске. Наконец, Mac OS X уникальна в своей способности использовать жесткие ссылки как для файлов, так и для папок. Это опять же используется для поддержки архивов Time Machine.

Метаданные файловой системы. Метаданные являются данными о данных, точнее, информацией, которая используется для описания содержимого. Самыми базовыми формами метаданных файлов и папок, которые используют почти все операционные системы, являются имена, пути в файловой системе, даты изменения и разрешения. Эти объекты метаданных не являются частью содержимого объекта, но они нужны для его описания в файловой системе. OS X использует несколько типов дополнительных метаданных файловой системы для различных технологий, которые расширяют возможности пользователя.

Ветви ресурсов. В OS X ветви ресурсов имеют длинную историю. Чтоб упростить работу пользователя, Apple создала ветвящуюся файловую систему, в которой сложные объекты (такие как приложения) отображаются как один значок. Ветвящиеся файловые системы, например, Mac OS Extended, позволяют нескольким фрагментам данных отображаться в

файловой системе как один объект. В данном случае файл будет отображаться как один объект, но в действительности он состоит из двух отдельных частей: ветви данных и ветви ресурсов. Mac OS поддерживает стандартные типы файлов в ветви данных, в то время как дополнительная, специфичная для Mac информация хранится в ветви ресурсов. В течение многих лет Mac OS использует ветвящиеся файлы для хранения как приложений, так и файлов. OS X не только продолжает, но и расширяет использование ветвей ресурсов, давая разработчикам возможность добавлять произвольное число дополнительных именованных ветвей. Это позволяет компании Apple и другим разработчикам реализовывать уникальные решения для файловой системы без необходимости изменять существующую файловую систему. Например, начиная с Mac OS X v10.6, стал поддерживаться сжатый программный код. То есть все исполняемые программные файлы были сжаты для экономии места и автоматически разархивируются на лету, когда нужно. Чтобы избежать неправильной обработки этих сжатых исполняемых файлов предыдущими версиями Mac OS X или более старыми приложениями, Apple предпочла скрыть сжатые биты в различных ветвях данных и ветви ресурсов.

Недостаток ветвей ресурсов и других типов дополнительных метаданных файловой системы заключается в том, что многие сторонние файловые системы, такие как FAT, не знают, как правильно разместить данные ветви ресурсов. Эта проблема решается с помощью формата файлов AppleDouble.

Флаги файлов и дополнительные атрибуты. OS X также использует флаги и дополнительные атрибуты файловой системы для реализации множества различных функций файловой системы. Флаги файловой системы унаследованы от оригинальной Mac OS и используются главным образом для контроля доступа пользователей.

С Mac OS X компании Apple нужно было расширить диапазон возможных атрибутов, связанных с файлом или папкой, где появились так называемые *дополнительные атрибуты (extended attributes)*. Любой процесс или приложение может добавить произвольное число пользовательских атрибутов к файлу или папке. Напомним, это позволяет

разработчикам создавать новые формы метаданных без необходимости изменять существующую файловую систему. Файловая система Mac OS Extended сохранит любые дополнительные атрибуты как другую ветвь, связанную с файлом. Finder использует дополнительные атрибуты для нескольких функций, включая задание цветовой метки для объекта, включение опции шаблона, включение опции скрытия расширения и комментарии Spotlight. Доступ к этим элементам можно получить из окна Свойства (Get Info) приложения Finder.

Связки и пакеты. Иногда ветвящиеся файлы являются не самым эффективным решением для скрытия данных, в особенности если имеется множество связанных файлов, которые требуется скрыть. Вместо создания новой технологии контейнеров Apple просто модифицировала существующий контейнер файловой системы – обычную папку. Связки (bundles) и пакеты (packages) – это ни что иное, как обычные папки, которые содержат связанные программное обеспечение и ресурсы. Это позволяет разработчикам программного обеспечения легко организовать все ресурсы, необходимые для сложного продукта, в один пакет или связку и не давать обычным пользователям обращаться к этим ресурсам. Анатомия пакета установки довольно проста; он обычно содержит только сжатый архив устанавливаемой программы и несколько конфигурационных файлов, используемых приложением установки. С другой стороны, иные программные связки и пакеты часто являются гораздо более сложными, поскольку содержат все ресурсы, необходимые для приложения или программного обеспечения.

Связки или пакеты программного обеспечения часто включают:

- исполняемый код для нескольких платформ;
- файлы с описаниями;
- мультимедиа-ресурсы, такие как изображения и звуковые файлы;
- файлы описания интерфейса пользователя;
- текстовые ресурсы;
- ветви ресурсов;
- ресурсы, локализованные для конкретных языков;

- частные программные библиотеки и инфраструктуры,
- подключаемые модули или другое программное обеспечение для расширения возможностей.

Несмотря на то, что Finder по умолчанию скрывает содержимое пакета, его все же можно просмотреть. Использование Spotlight. Утилита Spotlight в Mac OS X 10.4 совершила революцию в поиске пользователями файла на компьютерах Mac. Она позволяет выполнять почти мгновенный поиск, который проводится шире и глубже, чем предыдущими технологиями поиска на настольном компьютере. Spotlight выходит за рамки простого поиска по файловой системе и ищет соответствующие метаданные в файлах приложений и базах данных. Например, приложение Адресная книга (Address Book) хранит контактную информацию в базе данных, которая не просматривается файловой системой. А утилита Spotlight почти мгновенно может вернуть результаты поиска из базы данных Адресная книга (Address Book) вместе с результатами из десятков других баз данных и всей иерархии файловой системы. В дополнение к метаданным файловой системы, которые рассматривались ранее в этом уроке, многие файлы также содержат внутренние метаданные, используемые для описания содержимого файла.

Сетевые службы. Современные операционные системы предоставляют большое количество разнообразных сетевых служб и служб Интернета. Все они обладают похожей базовой сетевой архитектурой клиентского программного обеспечения, которое обращается к службам, и серверного программного обеспечения, предоставляемого этими службами. OS X обеспечивает поддержку многих популярных сетевых протоколов, позволяя обращаться к различным сетевым службам. Большая часть этой функциональности сетевых служб обусловлена использованием UNIX в качестве основы для OS X, что открывает дорогу программному обеспечению с открытым исходным кодом. Следовательно, компьютер Mac должен хорошо интегрироваться с любой другой современной операционной системой для доступа и предоставления сетевых служб. С точки зрения архитектуры, сетевые службы определяются клиентским программным обеспечением (созданным для доступа к этой службе) и серверным

программным обеспечением (созданным для предоставления службы). Коммуникация между клиентским и серверным программными обеспечениями осуществляется общеизвестными сетевыми протоколами или стандартами. При соблюдении таких стандартов разработчики программного обеспечения могут создавать уникальное, но при этом совместимое сетевое программное обеспечение клиента и сервера. Это позволяет выбрать программный инструмент, который лучше всего подходит для ваших потребностей. Программное обеспечение сетевых служб. Обычно клиентское программное обеспечение имеет форму самостоятельных приложений, как в случае со многими интернет-службами (такими как электронная почта и веб-браузеры). Иногда клиентское программное обеспечение интегрировано в операционную систему, как, например, службы файлов и печати. В любом случае, когда создается соединение с сетевой службой, настройки для этой службы сохраняются в файлах настроек на локальном компьютере. Эти клиентские настройки часто включают местоположения ресурсов и информацию аутентификации. С другой стороны этого тандема находится серверное программное обеспечение, правильная настройка которого – значительно более сложная задача. Администраторы серверов могут потратить месяцы на создание, конфигурирование и администрирование программного обеспечения, которое предоставляют сетевые службы. Серверные настройки включают параметры конфигурации, настройки протоколов и информацию учетной записи. Выбрав сетевую службу, к которой нужно подключиться, зачастую приходится удостоверить свою личность для поставщика службы. Этот процесс называется *аутентификацией* (или *проверкой личности*). Успешная аутентификация – это, как правило, последний шаг в установке подключения к этой службе. После установки подключения обычно вступают в действие технологии безопасности, которые гарантируют, что пользователю разрешено обращаться только к определенным ресурсам. Этот процесс называется *авторизацией*. Использование сетевых приложений в связи с широким распространением пакета интернет-протоколов TCP/ IP предназначено почти для всех коммуникаций LAN, WAN и Интернета, не существует разницы между тем, обращаетесь вы к

стандартной сетевой службе или к интернет-службе. Почти все сетевые службы работают в LAN точно так же, как и в Интернете. Основное различие между ними состоит в области действия. Такие службы, как электронная почта и мгновенный обмен сообщениями, могут, конечно, работать и на локальном уровне, но они созданы для коммуникации между разделенными сетями и серверами. OS X включает ряд клиентских приложений, созданных для доступа к различным сетевым службам. В OS X введена новая панель настроек Почта, адреса и календари (Mail, Contacts, & Calendars), обеспечивающая единый унифицированный интерфейс для настройки учетных записей сетевых служб, которые используются в программах Mail, iCal, Адресная книга (Address Book) и iChat. Другими словами, ввод одной учетной записи сетевой службы на панели настроек Почта, адреса и календари (Mail, Contacts, & Calendars) настроит все четыре основных сетевых приложения в OS X. Эти приложения могут настраиваться на использование учетных записей сетевых. Использование служб общего доступа к файлам OS X поддерживает множество сетевых протоколов, обеспечивающих общий доступ к файлам по сети. В этом разделе читатель исследует обе стороны служб обеспечения общего доступа к файлам – клиент и сервер. Для большинства пользователей компьютеры Mac функционируют исключительно как клиенты файловых серверов. Но любой Mac может также выступать в качестве сервера и предоставлять доступ к своим общим ресурсам. Кроме того, в ОС Lion введена новая функция AirDrop, которая является самым легким способом деления файлов между компьютерами Mac. Службы обеспечения общего доступа к файлам

При передаче файлов через сети и Интернет наиболее эффективными являются протоколы, созданные для обеспечения общего доступа к файловым системам. Сетевые файловые серверы могут делать доступными через сеть для клиентского компьютера целые файловые системы. Клиентское программное обеспечение, встроенное в утилиту Finder в OS X, может подключить службу сетевых файлов аналогично подключению локального тома запоминающего устройства. Когда служба сетевых файлов на Mac подключена, то можно манипулировать с файлами и папками, как с локальной файловой системой. Кроме того, полномочия доступа к

службам сетевых файлов определяются той же самой архитектурой прав владения и прав доступа, которая используется локальными файловыми системами.

Утилита Finder предоставляет два метода для соединения с сетевой файловой системой: автоматическое обнаружение общих ресурсов (путем их обзора в папке Network приложения Finder) и подключение вручную (путем ввода адреса сервера, предоставляющего файловую службу).

Использование служб общего доступа к хостам. Кроме служб общего доступа к файлам и веб, которые рассматривались в предыдущем разделе, OSX включает ряд сетевых служб, не предназначенных для предоставления общего доступа к файлам, которыми вы научитесь управлять в этом разделе. Данные службы «общего доступа к хостам» различаются по своей реализации и назначению, но все они позволяют пользователям удаленно обращаться к ресурсам на компьютере, предоставляющем службу. Все они также легко включаются и управляются из панели настроек Общий доступ (Sharing).

Защита сетевых служб. С точки зрения сетевых служб, Мас прекрасно защищен, поскольку по умолчанию на нем работают несколько ключевых служб, которые отвечают на внешние запросы. Даже когда вы предоставляете отдельные службы общего доступа, Мас отвечает на запросы только тех служб, которые в данный момент активированы. Кроме того, службы, способные вызвать проблемы при своей компрометации (такие как общий доступ к файлам или экрану), могут настраиваться таким образом, чтобы ими пользовался только ограниченный круг лиц (рассматривалось ранее в этом уроке). Тем не менее, пользователи могут открывать сторонние приложения или фоновые службы, которые делают Мас уязвимым для сетевой атаки.

Глава8

ОБЗОР КОМПЛЕКТА ВВОДА-ВЫВОДА

Во время создания OS X, Apple полностью переработал архитектуру ввода/вывода Macintosh, обеспечивая структуру для упрощенной разработки драйверов, которые поддерживают множество категорий устройств. Эта структура называется I/O Kit (*набором операций ввода-вывода*) [2]. С точки зрения программирования, набор ввода/вывода обеспечивает абстрактный вид системы оборудования в верхних слоях OS X. В наборе операции ввода-вывода используется модель объектно-ориентированного программирования, внедренная в ограниченном подмножестве C++, способствующая повторному использованию кода. Начиная с должным образом разработанных базовых классов, этим мы получаем преимущество с написанием нового драйвера; так как большая часть кода драйвера уже написана, Вам нужно лишь дописать конкретный код, который и отличает Ваш драйвер от остальных. Например, все контроллеры устройств SCSI обладают общим набором стандартных команд, но выполняют их с помощью различных низкоуровневых механизмов. Правильно используя методологию объектно-ориентированного программирования, с драйвером SCSI можно реализовать эти части низкоуровневого механизма без переделки кода протокола SCSI высокого уровня. Аналогичные возможности для повторного использования кода можно найти в большинстве типов драйвера. Все исходные коды комплекта ввода-вывода доступны как часть Darwin. Исходный код можно использовать как помощь в создании (и отладки) новых драйверов. Вместо того чтобы скрыть интерфейсы, дизайнеры Apple решили подать личный пример. Примеры кодов и классов показывают рекомендованный (простой) способ для записи драйвера.

Переделка модели ввода-вывода. Ядро OS X значительно более продвинутое, чем предыдущая архитектура системы Mac OS; OS X должен заниматься защитой памяти, выгрузкой, многопроцессорной обработкой и другими функциями, которые не существуют (или недостаточно глубоко проработаны) в предыдущих версиях Mac OS. Несмотря на то, что FreeBSD

поддерживает эти функции, модель драйверов BSD не предлагала автоматическую конфигурацию, укладку, управление питанием или динамические загружающие устройство функции, требуемые в современной, ориентированной на потребителя, операционной системе. Переделав архитектуру ввода-вывода, инженеры Apple могут воспользоваться преимуществами функций операционной системы в OS X. В OS X упростилось взаимодействие драйвера с VM. Возможности VM встроены в операционную систему OS X и не могут быть выключены пользователем. Таким образом, возможности VM могут быть абстрагированы в набор ввода-вывода, а код для обработки VM не должен быть написан для каждого драйвера. OS X предлагает беспрецедентные возможности повторного использования кода. В Mac OS 9, например, все комплекты разработчика программного обеспечения (SDKs) были независимы друг от друга, но при этом имели много схожего. В OS X набор ввода-вывода является частью основных инструментов разработчика, и код повсеместно используется в различных его частях.

В отличие от традиционных моделей ввода-вывода, модель повторно используемого кода, обеспеченная набором ввода-вывода, может существенно уменьшить ваше время разработки. В портировании драйверов из Mac OS 9 аналоги из OS X могут быть на 75% меньше. В целом, вся поддержка аппаратных средств обеспечена непосредственно объектами набора ввода-вывода. Исключением к этому правилу являются отображающие устройства, такие как принтеры, сканеры и цифровые фотоаппараты (несмотря на то, что некоторые используют функциональность набора ввода-вывода). В частности, несмотря на то, что связь с этими устройствами обработана набором ввода-вывода (например, под семейства FireWire или USB), поддержка конкретного устройства осуществляется кодом пространства-пользователя (user-space code). Если вам нужна поддержка отображающих устройств, то вы должны использовать соответствующий комплект разработчика программного обеспечения (SDK). Набор ввода-вывода стремится представить в программном обеспечении ту же иерархию, что и в аппаратных средствах. Кроме того, все

драйверы существуют для того, чтобы управлять аппаратными средствами; все аппаратные средства отличаются. Даже с допускающей повторное использование моделью, обеспеченной набором ввода-вывода, все еще необходимо знать о любых аппаратных особенностях, которое может повлиять на высокоуровневое представление устройства. Код, поддерживающий эти особенности все еще должен быть уникальным, от драйвера к драйверу. В любом случае исходный код всегда доступен. Можно заменить функциональность и изменить классы самостоятельно, если это требуется. В разработке набора ввода-вывода цель состояла в том, чтобы сделать жизнь разработчиков проще.

8.1. Архитектура I/O Kit

Набор ввода-вывода обеспечивает модель аппаратных средств системы на объектно-ориентированной платформе [2]. Каждый тип службы или устройства представлен классом C++; каждая дискретная служба или устройство – экземпляром (объектом) того класса. Есть три главных концептуальных элемента архитектуры набора ввода-вывода:

- “Семейства”;
- “Драйверы”;
- “Узлы”.

Семейство является набором высокоуровневых абстракций, характерных для всех устройств определенной категории, которая принимает форму кода C и классов C++. Семейства могут включать заголовки, библиотеки, примеры кода, тестовые использования и документацию. Они обеспечивают API, универсальный код поддержки и, по крайней мере, один драйвер в качестве примера (в документации). Семейства предоставляют услуги для многих различных категорий устройств. Например, есть семейство протокола (такие как SCSI, USB и FireWire), семейство хранения (диск), сетевые семейства и семья описания устройств интерфейса пользователя (мышь и клавиатура). Когда устройства имеют схожие черты, программное обеспечение, поддерживающее эти особенности, чаще всего можно найти в соответствующем семействе. Общие абстракции определены и реализованы семейством, позволяя всем драйверам в семействе

с легкостью использовать подобные функции совместно. Например, у всех контроллеров SCSI есть определенные вещи, которые они должны сделать, такие как сканирование шины SCSI. Семейство SCSI определяет и реализует функциональность, которая характерна для контроллеров SCSI. Поскольку эта функциональность была включена в семейство SCSI, то, например, код сканирования не должен включаться в ваш новый драйвер контроллера SCSI. Вместо этого вы можете сконцентрироваться на специфичных для устройства деталях, которые делают ваш драйвер отличным от других драйверов SCSI.

Семейства существуют для того, чтобы уменьшить количество кода, необходимого для написания вашего драйвера. Семьи динамически загружаемы; они загружаются при необходимости и выгружаются, когда в них больше нет необходимости.

8.2. Драйвер – объект

Набор ввода-вывода управляет определенным устройством или шиной, предоставляя более абстрактное представление этого устройства другим частям системы. Когда драйвер загружен, его требуемые семейства также загружены, чтобы обеспечить необходимую общую функциональность. Когда запрос о загрузке драйвера поступает в систему, перед самим драйвером загружаются его семейства (и их пререквизиты). После того, как все требования удовлетворены, требуемый драйвер загружается. Обратите внимание на то, что семейства загружаются по требованию драйвера, а не наоборот. Иногда, семейство может быть загружено в память, когда его потребует драйвер; однако вы никогда не должны принимать это как данность. Чтобы удостовериться, что все требования удовлетворены, каждый драйвер устройства должен перечислить все свои требования в его списке свойств. Большинство драйверов находится в отношении клиента-провайдера, где драйвер должен знать и о семействах, от которой он наследовался и о семействах, к которым он подключен. Драйвер контроллера SCSI, к примеру, должен быть в состоянии связаться и с семейством SCSI и с семейством PCI

(как клиент PCI и провайдер SCSI). Дисковый драйвер SCSI связывается, и с семейством SCSI, и с семейством хранения.

“Узел” (pub) – объект набора ввода-вывода, который представляет точку соединения для драйвера. Это предоставляет управляемый объект, такой как диск или шина. Узел загружен как часть семьи, которая описывает его. Каждый узел предоставляет доступ к устройству или службам, которые они представляют, и предоставляют такие функции как соответствие, арбитраж и управление питанием. Понятие узлов можно легко представить, вообразите телевизор. Есть провод, присоединенный к вашей стене, который обеспечивает телевизионную услугу от какого-либо провайдера. Для всех практических целей он перманентно ассоциирован с этим провайдером, в этом случае класс-экземпляр - кабельная компания, которая установила линию. Он может быть подсоединен к телевизору, чтобы предоставлять услугу (кабельное телевидение). Этот провод и есть узел. Каждый узел обеспечивает мост между двумя драйверами (и в дополнение к этому между двумя семьями). Нередко случается, что драйвер публикует один узел для каждого устройства или сервиса, которые он контролирует (в этом примере вообразите один провод, обслуживающий каждый дом от этой кабельной компании). Также для драйвера, который управляет только одним устройством или службой, возможно, играет роль своего собственного узла (вообразите антенну позади телевизора, у которого есть встроенный провод).

Пример соединения. Рис. 8.1 иллюстрирует архитектуру набора ввода-вывода, используя несколько драйверов в качестве примера и их соответствующие узлы. Обратите внимание на то, что возможно много различных комбинаций драйвера; эта схема показывает только одну возможность. В этом случае стек SCSI показан с контроллером PCI, диском и сканером SCSI. Диск SCSI управляет драйвер резидентного ядра. Сканером SCSI управляет драйвер, который является частью пользовательского приложения.

На рисунке 8.1 представлена архитектура набора ввода-вывода.

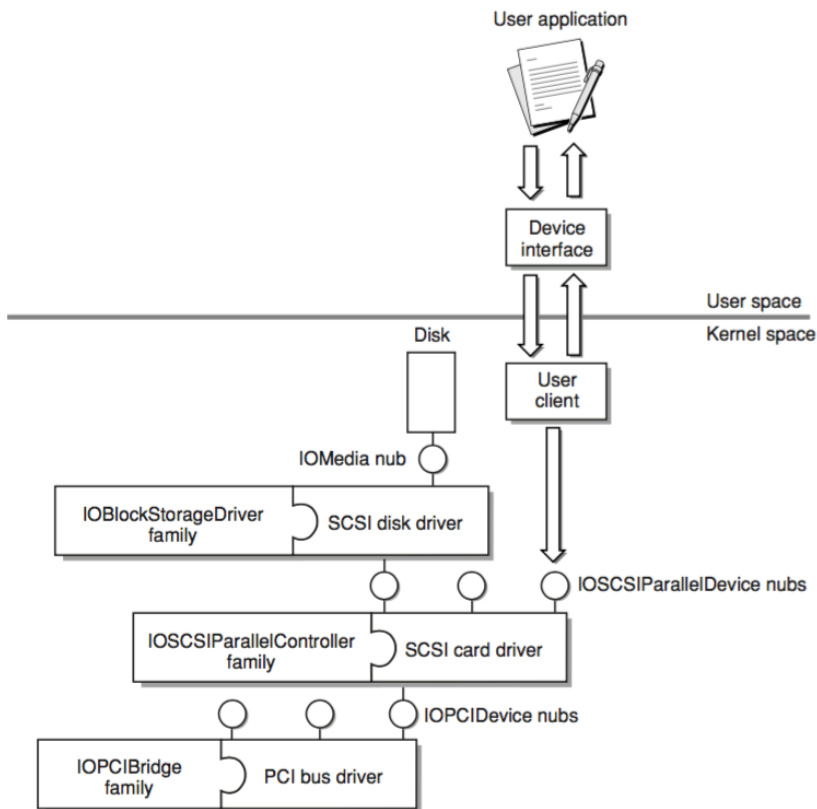


Рис. 8.1. Набор ввода-вывода

Этот пример иллюстрирует, как дисковый драйвер SCSI (семейство Storage) соединен с шиной PCI. Соединение выполнено по пунктам:

1. Драйвер шины PCI обнаруживает устройство PCI и объявляет о его присутствии, создавая узел (IOPCIDevice). Класс узла определен семейством PCI.

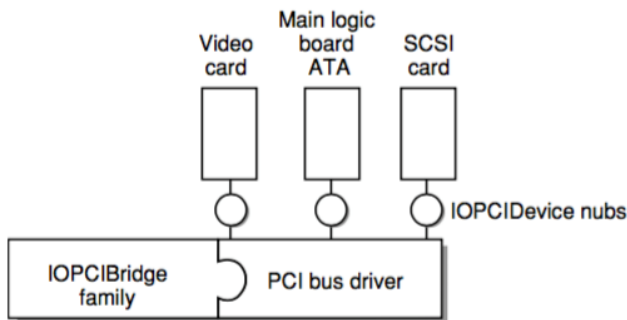


Рис. 8.2

2. Драйвер шины идентифицирует корректный драйвер устройства и запрашивает, чтобы драйвер был загружен. В конце этого процесса драйвер контроллера SCSI будет найден и загружен. Загрузка драйвера контроллера заставляет все требуемые семейства быть загруженными. В этом случае семейство SCSI загружено; семейство PCI (также требуемое) уже присутствует. Драйверу контроллера SCSI дают ссылку на узел IOPCIDevice.

3. Драйвер контроллера SCSI сканирует шину SCSI для устройств. После нахождения устройства он объявляет о присутствии устройства, создавая узел (IOSCSIDevice). Класс этого узла определен семейством SCSI [2].

4. Драйвер контроллера идентифицирует корректный драйвер устройства и запрашивает, чтобы драйвер был загружен. В конце этого процесса дисковый драйвер будет найден и загружен. Загрузка дискового драйвера заставляет все требуемые семейства быть загруженными. В этом случае семейство Storage загружено; семейство SCSI (также требуемое) уже присутствует. Дисковому драйверу дают ссылку на узел IOSCSIDevice.

Периферийные устройства. Apple является пионером концепции автоматической поддержки периферийных устройств. Эта возможность, известная как plug-and-play (подключи и играй), более или менее успешно поддерживается во всех современных операционных системах. Периферийное

оборудование также усовершенствовалось – в наши дни самые распространенные стандарты подключения поддерживают «горячее» подключение и даже беспроводные подключения. Нигде это стремление не проявилось так явно, как в выходе компании Apple на рынок потребительской электроники с такими продуктами, как iPod, Apple TV и iPhone. Ну и, конечно, OS X поддерживает все популярные современные стандарты периферийных устройств. Apple и Adobe совершили переворот в мире настольных издательских систем, представив первое высококачественное решение печати для персональных компьютеров. Систему печати PostScript создала Adobe, а Apple первой включила ее в операционную систему Macintosh и в самый первый принтер PostScript – Apple LaserWriter. Компания Apple продолжила свою деятельность по развитию программного обеспечения печати в Mac OS X, внедрив рабочий процесс печати на основе Portable Document Format (PDF) компании Adobe и Common Unix Printing System (CUPS).

8.3. Периферийные устройства

Любое несетевое устройство, которое может быть подключено непосредственно к компьютеру. Периферийное устройство напрямую управляется компьютером Mac, а сетевые устройства являются общедоступными. Способы подключения периферийных устройств. Большинство периферийных устройств взаимодействуют с Mac через механизм соединения, называемый *шиной*. Соединения через шину наиболее распространены для периферийных устройств, так как позволяют одновременно подключить несколько последних к компьютеру. Фактически только соединения для аудио- и видеосвязи не являются шинами, но даже в этом случае Mac соединяется через периферийную шину с промежуточным оборудованием, отвечающим за кодирование или декодирование аудио- и видеосигналов.

Периферийные соединения можно разделить на четыре типа:

- Периферийные шины – шины общего назначения, используемые в основном для соединения внешнего устройства с Mac;

- Шины расширения – созданы для расширения аппаратных возможностей Mac, обычно добавляют дополнительные возможности соединения;

- Шины памяти – используются только для доступа к устройствам памяти;

- Аудио- и видеосоединения – стандартные интерфейсы для пересылки аудио- и видеосигналов с одного устройства на другое.

Каждое соединение предназначено для определенного типа коммуникации, поэтому для работы периферийного устройства требуется комбинация технологий. Например, графическое оборудование Mac создано для вывода стандартного видеосигнала, но взаимодействует с процессором через шину расширения. При исследовании различных периферийных устройств вы увидите много примеров комбинированных типов соединения. OS X содержит приложение «Информация о системе» («System Information»), помогающее идентифицировать присоединенное периферийное оборудование и определить типы соединения. Приложение «Информация о системе» («System Information») можно запустить, выбрав команду Apple [Об этом Mac] (About This Mac) и щелкнув на кнопке [Дополнительно] (More Info) в диалоговом окне. Когда программа откроется, выберите интерфейс оборудования из списка содержимого, чтобы увидеть информацию о нем.

Периферийные шины. Периферийные шины обычно ассоциируются с периферийными устройствами компьютера. Они созданы для предоставления связи общего назначения между компьютером и периферийным устройством, и использовать их может множество устройств. За много лет были разработаны десятки стандартов соединения периферийных устройств, но в последнее десятилетие на рынке доминируют три типа периферийных соединений: *Универсальная последовательная шина (USB), FireWire и Bluetooth.*

Печать, сканирование и факсы. Конфигурирование. **Печать, сканирование и факсы.** Надежная печать всегда была важной составляющей операционной системы Mac в связи с ее популярностью у дизайнеров. OS X продолжает эту традицию. При этом функциональная часть системы печати была переработана, а интерфейсы печати стали еще проще.

Архитектура технологии печати и факсов. OS X Lion использует систему печати Common UNIX Printing System 1.5 (CUPS) с открытым исходным кодом для управления локальной печатью и работой с факсами. Первоначально независимый продукт, CUPS был куплен компанией Apple и остается проектом с открытым исходным кодом. CUPS управляет задачами печати и факсов на основе стандарта Internet Printing Protocol (IPP) и для драйверов печати использует файлы PostScript Printer Description (PPD).

Драйверы печати, сканирования и факсов. Перед тем как начать печатать или отправлять факсы, нужно сконфигурировать настройки принтера или факс-модема, включая связывание подходящего драйвера принтера с печатающим устройством. Эта связь проводится автоматически, когда пользователь настраивает новое печатающее устройство. Однако в системе, чтобы она могла использовать принтер, уже должен быть установлен его драйвер. Apple предоставляет драйверы принтеров для большинства популярных моделей, среди которых Brother, Canon, Epson, Fuji-Xerox, Hewlett-Packard, Lexmark, Ricoh и Samsung.

Программа установки OS X старается экономить место на системном томе и поэтому не устанавливает дополнительные драйверы печати. Новые установки OS X включают только общие драйверы печати и драйверы печати Apple. При обновлении ОС до текущей OS X будут установлены драйверы только для используемых в данный момент принтеров.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Keith Loeper*. Mach 3 Kernel Principles. Open Software Foundation and Carnegie Mellon University. OSF NORMA-MK12: July 15, 1992.
2. <https://developer.apple.com>. Kernel Programming Guide
3. <https://developer.apple.com>. Mac Technology Overview
4. *Kevin M. White* and *Gordon Davisson*. OS X Support Essentials 10.8. Apple Pro Training Series, 2013.
5. *Kevin M. White* and *Gordon Davisson*. OS X Support Essentials 10.9. Apple Pro Training Series, 2014.

ГЛОССАРИЙ

1. **Classic**– среда выполнения приложений в операционных системах (Mac OS 8 и Mac OS 9). Этот термин используется для обозначения приложений, предназначенных для выполнения в ОС, предшествовавших системе Mac OS X. Система Mac OS X не поддерживает приложения Classic, в отличие от версий системы Mac OS X, предшествовавших версии Snow Leopard.

2. **Carbon** – технология компании Apple, позволяющая разработчикам создавать приложения, которые могут выполняться под управлением Mac OS 9 и Mac OS X под видом “родных” приложений для обеих операционных систем.

3. **Cocoa**– приложения, специально разработанные для системы Mac OS X. Они не совместимы с более старыми ОС, такими как Mac OS 9. Приложения Cocoaобладают всеми современными возможностями, предоставляемыми системой Mac OS X, такими как расширенная память, вытесняющая многозадачность и интерфейс Aqua.

4. **Java (JDK)** – язык программирования, разработанный компанией Sun Microsystems. Язык Java является платформенно-независимым, что позволяет приложениям функционировать на любой платформе, на которой установлена Java-машина.

5. **BSD** (Berkeley Software Distribution – существует много разных версий или стилей операционной системы Unix, включая дистрибутивы Linux, Solaris и AIX. “BSD” – это название стиля системы Unix, разработанного в Калифорнийском университете в Беркли (University of California at Berkeley). Система Mac OS X основана именно на этой версии системы Unix.

6. **QuickTime** – лицензионный кросс-платформенный механизм авторизации и распространения компании Apple. Термин “Quick Time” относится и к формату файла, и к набору приложений. В системе Mac OS X Snow Leopard и более поздних версиях соответствующий механизм называется *Quick TimeX*.

ОГЛАВЛЕНИЕ

	Введение	3
Глава 1	Технологии Mac	4
1.1.	Виды программного обеспечения для Mac ..	5
1.2.	Слой приложений Cocoa	6
1.3.	Media Layer	8
1.4.	Core Services Layer	10
1.5.	Core OS Layer	12
1.6.	Kernel and Device Drivers Layer (Слой ядра и драйверов устройств)	14
1.6.1.	Высокоуровневые возможности	14
1.6.2.	Kernel (Ядро)	15
1.6.3.	BSD	18
Глава 2	Обзор архитектуры ядра	24
2.1.	Darwin	26
2.2.	Архитектура	28
2.3.	Kernel Extensions	31
Глава 3	Обзор Mach	32
3.1.	Абстракции ядра Mach	33
3.2.	Задачи и потоки	33
3.3.	Порты, права портов, наборы порта и пространство имен портов	35
3.4.	Управление памятью	37
3.5.	Межпроцессное взаимодействие (IPC)	40
3.5.1.	Транзакции IPC и диспетчеризация события.	41
3.5.2.	Очереди сообщений	41
3.5.3.	Семафоры	42
3.5.4.	Уведомления	42
3.5.5.	Блокировки	42
3.5.6.	Объекты удаленного вызова процедур (RPC).	42
3.6.	Тайм-менеджмент	43
Глава 4	Память и виртуальная память	44
4.1.	Понятие отображения памяти	46
4.2.	Именованные записи	48
Глава 5	Обзор BSD	50
5.1.	BSD-возможности	51
5.2.	Различия между OS X и BSD	53

Глава 6	Учетные записи пользователей и сетевые возможности	55
6.1.	Типы и атрибуты учетных записей пользователей	55
6.2.	Основы обеспечения безопасности учетных записей	58
6.3.	Типы паролей	60
6.4.	Управление keychain-файлами	61
6.5.	Сброс паролей учетных записей	62
6.6.	Конфигурация сети	67
Глава 7	Файловые системы	87
7.1.	Компоненты файловой системы	87
7.2.	Управление файловой системой	90
7.3.	Использование системы FileVault 2	91
7.4.	Разрешения файловой системы	92
7.5.	Структура файловой системы	96
7.6.	Управление скрытыми объектами	100
Глава 8	Обзор комплекта ввода-вывода	109
8.1.	Архитектура I/O Kit	111
8.2.	Драйвер – объект	112
8.3.	Периферийные устройства	116
	Библиографический список	119
	Глоссарий	120
	Содержание	121

Учебное издание

Гульнар Ешатаевна Сейдахметова
Бахытжан Мухамеджановна Мустафина

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ ДЛЯ ОС APPLE

Учебное пособие

Нач. РО УИЦ	<i>З. А. Губайдулина</i>
Редактор	<i>З. А. Губайдулина</i>
Компьютерная верстка	<i>А. С. Сайбулатова</i>

Подписано в печать.03.06.2016 г.

Тираж 300 экз. Формат 60х84х 1/16. Бумага типогр. № 1.
Уч.-изд.л. 7.5. Усл. п.л.7,0. Заказ № .732. Цена договорная.

Издание Казахского национального исследовательского технического
университета имени К. И. Сатпаева

Учебно-издательский центрКазННТУ имени К. И. Сатпаева,
г. Алматы, ул. Сатпаева, 22

ISBN 978-601-228-958-9

