Разработка приложений для iPhone на языке Objective-C

Второе издание

iPhone

Разработка приложений с открытым кодом

O'REILLY® bhv®

Джонатан Здзиарски

SECOND EDITION

iPhone Open Application Development

Jonathan Zdziarski

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

Джонатан Здзиарски

iPhone

Разработка приложений с открытым кодом

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.06
ББК 32.973.26-018.2
З-46

Здзиарски Дж.

З-46 iPhone. Разработка приложений с открытым кодом: Пер. с англ. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 368 с.: ил.

ISBN 978-5-9775-0397-6

Книга посвящена разработке приложений для iPhone на языке Objective-C с помощью iPhone API, используя последние версии инструментария с открытым кодом, обновленного для программного обеспечения iPhone 2.x и iPhone 3G. Рассматриваются настройка и работа с приложениями iPhone. Описана разработка пользовательских интерфейсов с помощью графической оболочки UIKit. Показана обработка событий. Рассмотрено программирование графики, включая анимацию и трехмерную трансформацию поверхностей. Уделено большое внимание вопросам записи и воспроизведения звуковых файлов. В приложении описаны различные приемы программирования и классы открытого кода для создания собственных приложений для iPhone.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Перевод с английского	*Александры Маленковой*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.02.09.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 29,67.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.002108.02.07
от 28.02.2007 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-0-596-15519-3 (англ.)
ISBN 978-5-9775-0397-6 (рус.)

Оглавление

Предисловие

Итак, вы хотите писать приложения для iPhone. Первое, что вы должны знать, — даже после выпуска первой версии SDK от Apple iPhone остается по большей части достаточно закрытым устройством, а Apple предпринимает шаги, чтобы не подпускать разработчиков близко к операционной системе. Тем не менее, это не удерживает людей от постижения всех глубин iPhone и создания приложений. С момента первой версии iPhone от Apple это популярное устройство прослыло самым трудновзламываемым мобильным устройством из когда-либо существовавших. Еще до конца 2007 года Web-узел **http://jailbreakme.com** Николаса Пенри (Nicholas Penree) сообщал, что более 30% рынка вломилось в их устройства, чтобы запускать несанкционированные приложения сторонних фирм на своих iPhone, с тех пор сообщество разработчиков неуклонно растет.

Команда хакеров iPhone, известная как iPhone Dev team, в течение первых 30 дней с момента выпуска iPhone 3G зафиксировала свыше шести миллионов посещений своего Web-узла. Большинство других узлов, хранящих средства взлома iPhone, имеют схожий рекордный по своей насыщенности трафик, и даже сотрудники Apple были пойманы на рекламе своих "настроенных" устройств.

В течение нескольких месяцев с момента выхода первой версии iPhone сообщество разработчиков открытого кода совершило геройский поступок, который привел к серьезному беспокойству Apple: это сообщество спроектировало бесплатный компилятор с открытым кодом и пакет инструментов для создания приложений iPhone, развернутые в специальном хранилище, и построило значительную базу для разработчиков. Все это было сделано для устройства, которое должно было быть закрытым. В 2008 году стало очевидно, что сообщество разработчиков программного обеспечения для iPhone не только существует, но и преуспевает в разработке выдающихся сторонних приложений для популярного устройства, и без какой-либо помощи от Apple. Чувствуя потенциально большой доход, Apple стремился реализовать собственную версию того, что сообщество разработчиков открытого кода уже создало: компилятор и пакет инструментов, специальное хранилище и большую базу для разработчиков. Можно только догадываться о том, явилось ли это результатом давления сообщества, но одно известно совершенно точно: со-

общество разработчиков открытого кода доказало факт того, что все это можно вполне успешно реализовать.

В марте 2008 года был представлен Apple SDK. Этот SDK позволил разработчикам писать приложения с помощью санкционированных Apple инструментов и наградил возможностью продаж очень большой аудитории потенциальных покупателей через интернет-рынок Apple — AppStore. Однако многие разработчики посчитали такое предложение не очень интересным. Разрабатываемые таким образом приложения имеют уровень детских развлечений, нет возможности использовать многие компоненты операционной системы. Это не позволяло приложениям выполнять простые задачи, такие как работа в фоне, эффективное прекращение работы приложений отправки мгновенных сообщений и любых других служебных программ, работающих в фоне в режиме реального времени. Произвольные ограничения в Apple SDK также не давали разработчикам писать более сложные приложения, например, видеопроигрыватели, высококачественные 2D-игры и любые другие приложения, которые, по мнению Apple, можно было реализовать с помощью их собственного пакета программного обеспечения.

Помимо отсутствия возможности доступа к некоторым известным и ценным функциям iPhone, SDK ввел набор новых еще более ограничивающих объектов, которые "скрывали" от разработчика множество более низкоуровневых API. Хотя эти API и продолжали существовать на платформе iPhone (и использовались в действительности многими собственными приложениями Apple), ограничения этих интерфейсов давали, по мнению многих, нечестное преимущество Apple. Дополнительно к жалобам разработчиков добавилось соглашение Apple для разработчиков, раскритикованное за запрет создания определенных видов приложений, включая пошаговое навигационное программное обеспечение и байт-кодовые интерпретаторы (такие как Java и Flash).

Имея в виду все эти неоднозначные вопросы, многие разработчики решили примкнуть к рядам тех, кто пишет программное обеспечение с открытым кодом, не связанным с лицензионными и техническими ограничениями Apple, программное обеспечение которого действительно использует те же самые API, что и собственные приложения Apple, и способно получать доступ к функциям на существенно более низком уровне, нежели заурядное программное обеспечение AppStore. На сегодняшний день вы можете найти множество отличных приложений только в сообществе разработчиков открытого кода. Установщик программного обеспечения данного сообщества Cydia распространяет много различных приложений, например, игровые

эмуляторы, видеопроигрыватели, инструменты UNIX, менеджеры тем и другое великолепное программное обеспечение. Сообщество разработчиков открытого кода разработало собственные удобные инструменты, поскольку разработчики посчитали SDK от Apple чересчур технически и политически ограничивающим. Кроме того, множество отличных коммерческих пакетов вне рамок AppStore превосходят свои аналоги с AppStore просто потому, что они имеют ничем не ограниченный доступ к низкоуровневым API iPhone. Даже простейшие приложения карманного фонарика могут превосходить аналоги с AppStore всего лишь из-за возможности настройки яркости экрана. Наконец, пакет инструментов с открытым кодом имеет определенный уровень стабильности, который, в частности, предоставляет значительную совместимость кода между основными редакциями фирменного программного обеспечения — предмета большого разочарования разработчиков на SDK от Apple, которые стали жертвами частого переписывания кода.

Таким образом, имея под рукой пакет инструментов и множество бессонных ночей, сообщество изучило то, как использовать низкоуровневые платформы и интерфейсы на iPhone, чтобы создавать эффектные сторонние приложения, которые обладают потенциалом дать шанс AppStore заработать свои деньги. В данной книге мы рассмотрим платформы, являющиеся ключевыми в разработке полнофункционального программного обеспечения на iPhone, и обозначим те инструменты, которые могут воспользоваться преимуществами других недокументированных здесь платформ.

Хотя эта книга посвящена компилятору и инструментам сообщества разработчиков открытого кода, и не посвящена Apple SDK, тем не менее, разработчики, использующие SDK от Apple, обнаружат, что бóльшая часть этой книги совпадает с объектами, напрямую или косвенно доступными в SDK, и что некоторые технические ограничения можно обойти, чтобы воспользоваться определенной функциональностью. Однако имейте в виду, что многие обсуждаемые здесь API официально запрещены для использования в SDK. Данная книга на самом деле предназначена для разработчиков открытого кода.

iPhone — великолепное устройство, и несмотря на политику, связанную с его недоступностью для разработчиков, сообщество его пользователей растет очень быстро. С помощью Apple или без нее iPhone дает рождение многим новым коммерческим рынкам и скоро превзойдет успех своих предшественников: PocketPC и Symbian, которые ранее владели рынком мобильных устройств.

По мере чтения этой книги вы, возможно, не будете осознавать то, как здорово, что она у вас есть. Простота, которую вы увидите в книге, является результатом тысяч часов работы активной части сообщества разработчиков, решающей практически невозможные задачи. Методы старой школы, использованные для работы с iPhone, были трудоемкими, если не сказать, изнурительными, и сами по себе могут составить тома книг.

Работа над открытием многих собственных интерфейсов на iPhone продолжается и сегодня. Мы приглашаем присоединиться к нам в нашей работе в сообществе всех, имеющих свои ноу-хау и огромное упорство в достижении поставленных целей.

Для кого предназначена эта книга

Чтобы эта книга была вам полезна, вы должны обладать определенными предварительными знаниями в программировании. Окружение iPhone использует Objective-C, с которым вы познакомитесь в *главе 2*. Положительным моментом является то, что вы в своих приложениях также можете использовать C и C++, поэтому все, кто имеет соответствующие познания, смогут быстро погрузиться в материал. Если вы не знаете C или C++, то данной тематике посвящена масса книг. Эта книга не является учебником для начинающих изучать эти языки, а раскрывает значение собственных классов и методов, необходимых для написания специализированных для iPhone приложений.

Структура книги

В *главе 1* объясняется, как проникнуть в ваш iPhone.

Глава 2 показывает состав приложения iPhone и то, как на вашем рабочем столе запускать пакет инструментов.

Глава 3 знакомит с UIKit, находящимся в центре разработки приложений iPhone и пользовательских интерфейсов.

Глава 4 описывает основные геометрические понятия, используемые в платформе Core Graphics, и уведомления о событиях.

Глава 5 погружается в процесс разработки для iPhone, предлагая изучить необработанные видеоповерхности и трансформации 3D.

Глава 6 посвящена множеству различных способов записи и воспроизведения звука и выходного потока цифрового аудио.

Глава 7 демонстрирует использование многих сложных компонентов пользовательского интерфейса из UIKit.

В *приложении* освещено несколько разнообразных приемов и классов открытого кода, позволяющих вам делать потрясающие вещи в ваших приложениях для iPhone.

Используемые в этой книге обозначения

В данной книге используются следующие соглашения об обозначениях:

- **полужирный текст** применяется для обозначения названий меню, пунктов меню и кнопок меню, URL;
- *курсив* используется для обозначения новых терминов;
- `моноширинный шрифт` служит для обозначения содержимого файлов, выходной информации команд, переменных, типов, классов, пространств имен, методов, объектов и всего того, что можно найти в программах.
- **`полужирный моноширинный шрифт`** используется для обозначения команд или другого текста, которые должны точно вводиться пользователем, и частей кода или файлов, выделенных для обсуждения.

В книге также есть совет, примечание общего плана и предупреждения.

Использование примеров кода

Эта книга написана, чтобы помочь вам выполнить свою работу. Вообще говоря, вы можете использовать код, приведенный в этой книге, в ваших программах и документации. Вам не нужно связываться с нами, чтобы получить разрешение на использование кода, если только вы не собираетесь воспроизводить существенную часть кода. Например, для написания программы, использующей несколько фрагментов кода из этой книги, не требуется разрешения. Для продажи или распространения компакт-дисков с примерами из книг издательства O'Reilly обязательно требуется разрешение. Для ответа на вопрос с использованием цитат или выдержек из этой книги не требуется разрешения. Для включения значительного количества кода из примеров,

приведенных в данной книге, в документацию вашего продукта обязательно требуется разрешение.

Мы будем признательны за ссылку на нашу книгу, хотя это и не обязательно. Ссылка обычно включает название книги, автора, издательство и ISBN. Например: "iPhone Open Application Development, Second Edition, by Jonathan Zdziarski. Copyright 2009 Jonathan Zdziarski, 978-0-596-15519-3".

Если вы считаете, что ваше использование примеров кода из этой книги выходит за рамки выданных выше разрешений, то не стесняйтесь связаться с нами по адресу: **permissions@oreilly.com**.

Благодарности

Выражаем особую благодарность Патрику Валтону (Patrick Walton), Джею Фриман (Jay Freeman), Брайану Вайтману (Brian Whitman), Джону Баффорду (John Bafford), Николасу Пенри (Nicholas Penree), Элиоту Кру (Elliot Kroo), Дино Пастосу (Dino Pastos), Нейт Тру (Nate True), Стиву Данхаму (Steve Dunham), Николасу Бакка (Nicolas Bacca), Даниэлю Пиблсу (Daniel Peebles), Александру Пику (Alexander Pick), Аарону Александру (Aaron Alexander), Ричарду Талли (Richard Thally), Джастину Лазарову (Justin Lazarow), Крису Зимману (Chris Zimman), Эрику Макдональду (Eric McDonald) и многим другим из сообщества разработчиков для iPhone, пожелавшим остаться анонимными, кто пожертвовал бессонными ночами и наличными из собственных карманов ради исследования iPhone и создания прочного основания для разработки приложений.

Глава 1

Знакомство с iPhone и его настройка

iPhone — достаточно закрытое устройство. Но нас это не устраивает. Программное обеспечение iPhone до версии 2.*x* включительно наглухо закрывало перед пользователями дверь в операционную систему, а разработчики вынуждены были довольствоваться игрой в куличики в строго ограниченной песочнице, созданной в пользовательском пространстве. Хотя эти ограничения не отпугивают большинство пользователей iPhone, но существенно затрудняют начало серьезной работы с ним. Прежде чем можно будет приступить к какому-либо доскональному изучению, iPhone, без преувеличения говоря, должен быть извлечен из своего заточения.

Взаимодействие iPhone с программным обеспечением, например, iTunes, происходит в chroot-окружении, в котором ни один пользователь и ни одно прикладное приложение — даже iTunes — не может видеть операционную систему; что в мире UNIX известно под названием chroot-тюрьмы *(chroot jail)*. Эта тюрьма (и тот факт, что вы не можете просто вытащить жесткий диск) — единственная преграда, не позволяющая iPhone функционировать как полноценному переносному компьютеру Mac OS X. К счастью, существует множество бесплатных программных инструментов, упрощающих процесс извлечения iPhone из этого заточения.

В этой главе вы настроите ваш iPhone для разработки программного обеспечения таким образом, что сможете получать доступ к файлам вне этой тюрьмы, а ваши приложения смогут выполняться за рамками ограничивающей их песочницы. Для этого необходимо произвести освобождение из chroot-тюрьмы (называемое *jailbreaking*), тем самым вы получите доступ к файловой системе. Кроме того, вы установите BSD-мир UNIX, являющийся набо-

ром бинарных файлов UNIX, таких как `ls` и `cp`. Это позволит вам взаимодействовать и управлять операционной системой iPhone, которая, как предполагается, является версией Mac OS X 10.5 (Leopard) для процессора ARM. Наконец, у вас появится безопасное командное окружение входа в систему, SSH. Это удобно для копирования файлов с вашего iPhone и на него, а также для установки приложений и выполнения примеров.

Процедуры взлома (jailbreak)

То, как вы будете освобождать свой iPhone из заточения (jailbreak), сильно зависит от используемой вами версии программного обеспечения. Существует временной разрыв в несколько недель, между выпуском нового программного обеспечения iPhone и появлением общедоступных инструментов для его взлома. Как правило, в новых версиях содержатся незначительные изменения, что каждый раз несколько затрудняет их взлом. Положительным моментом является то, что как только написан новый инструмент для взлома, то все свободно распространяемые программные средства обновляются, позволяя практически каждому осуществить данную процедуру.

Программное обеспечение для взлома от сторонних фирм

Существует множество бесплатных инструментов, предназначенных для взлома iPhone, одни более надежные, другие менее. Наилучшими средствами являются полнофункциональные программы, позволяющие вам помимо всего прочего легко и просто настраивать командный процессор и устанавливать программное обеспечение сторонних фирм. Перечислим лучшие инструменты.

- iNdependenece, **http://code.google.com/p/independence/** (версии 1.0.0—1.1.4).

  iNdependenece — это программа для Mac OS X, осуществляющая взлом, активацию, установку SSH, и даже установку на iPhone звуков (ringtones), фоновых рисунков (wallpapers) и приложений сторонних фирм. iNdependenece работает под GPL, а автор создал библиотеку под названием *libPhoneInteraction*, позволяющую разработчикам писать другие инструменты для взаимодействия с iPhone.

- AppSnapp, **http://www.jailbreakme.com** (только версия 1.1.1).

  Пользователи, применяющие встроенное программное обеспечение iPhone версии 1.1.1, могут посетить этот Web-узел с помощью своего iPhone и выполнить весь процесс взлома удаленно. Для проникновения в телефон AppSnap пользуется уязвимостью в одной из библиотек изображений iPhone. Что особенно здорово на этом узле, так это то, что он не только взламывает ваш телефон, но еще и устраняет эту уязвимость, предотвращая тем самым возможность его злонамеренного использования. Кроме того, AppSnap от версии 1.1.1 и далее исправляет программное обеспечение iPhone, чтобы разрешить выполнение приложений сторонних фирм, а также устанавливает AppTapp, установщик NullRiver, который затем может использоваться для создания условий в вашем iPhone для разработки программного обеспечения.

- AppTapp, **http://iphone.nullriver.com** (версии 1.0.0—1.0.2).

  Nullriver — разработчик программного обеспечения из Онтарио (Канада), спроектировавший пакетный установщик для iPhone. Данный установщик позволяет вам с помощью нескольких касаний устанавливать на ваш iPhone любые приложения, имеющиеся в его хранилище. Само по себе программное обеспечение установщика работает с большинством версий программного обеспечения iPhone, но программа установки может взламывать встроенное программное обеспечение iPhone версии 1.0.*x*. Предыдущий инструмент в этом списке, AppSnapp, автоматически устанавливает AppTapp на устройства с версией 1.1.1. AppTapp также полезен для проведения процедуры уменьшения версии программного обеспечения, описанной далее.

- ZiPhone, **http://www.ziphone.org** (версии 1.0.0—1.1.4).

  ZiPhone — это метод взлома, разработанный iPhone Dev Team. Он держался в величайшей тайне в преддверии выхода Apple SDK, но в итоге был выдан одним из бывших членов команды разработки. С тех пор ZiPhone был серьезно доработан и вышел за рамки всего лишь простого метода взлома, также к нему было добавлено множество других программ, включая полную разблокировку всех iPhone вплоть до OTB (Out-of-the-Box) версии 1.1.4.

- Pwnage, **http://www.iphone-dev.org** (версии 1.0.0—2.*x*).

  Pwnage был первым инструментом, поддерживающим встроенное программное обеспечение версии 2.0 и iPhone 3G. Pwnage позволяет пользователю создавать собственный встроенный пакет, содержащий общий

установщик программного обеспечения под названием Cydia, и другие программные пакеты сторонних фирм. Pwnage пользуется слабыми местами iPhone и загрузочного ROM iPhone для загрузки в устройство неподписанного встроенного программного обеспечения. Для пользователей Windows существует Windows-версия Pwnage под названием WinPwn.

Установка SSH

После того как вы взломали ваш iPhone, установка Secure Shell позволит вам получить доступ к UNIX-окружению вашего iPhone и без труда копировать файлы на телефон и с него через беспроводное подключение WiFi.

Для использования SSH необходимо, чтобы ваш iPhone был подключен к той же беспроводной сети WiFi, что и ваш настольный компьютер. Если у вас нет доступа к беспроводной сети WiFi, то для установки приложений на ваш iPhone вам придется воспользоваться таким инструментом, как iNdependence, поэтому вы можете пропустить этот раздел. Однако вы можете попробовать установить MobileTerminal — бесплатную программу терминала для iPhone. По меньшей мере, это позволит вам работать в UNIX-окружении iPhone, что является необходимым условием для выполнения небольшого количества примеров. MobileTerminal может быть загружен прямо на iPhone с помощью Cydia или с адреса: **http://code.google.com/p/mobileterminal/**.

Независимо от инструмента, использованного вами для взлома вашего iPhone, общий установщик программного обеспечения Cydia должен быть добавлен на экран рабочего стола вашего iPhone. Чтобы установить SSH из Cydia:

1. Коснитесь значка Cydia, чтобы запустить само приложение. В начале Cydia может предложить вам обновить свое программное обеспечение. В этом случае пройдите процедуру обновления и перезагрузите установщик.
2. Коснитесь кнопки разделов, расположенной внизу, прокрутите сетевые пакеты (Networking packages), найдите и установите OpenSSH. Или же для его нахождения можете воспользоваться встроенной функцией поиска Cydia. Коснитесь кнопки **Install** и подтвердите установку OpenSSH.

Теперь на iPhone должен быть запущен SSH, но прежде чем вы сможете подключиться к нему, вам необходимо узнать IP-адрес вашего iPhone в вашей локальной беспроводной сети WiFi. Для этого:

1. На вашем iPhone коснитесь приложения **Settings**.

2. Выберите вкладку **General**, затем **Network**, затем **WiFi**.
3. В списке справа от вашей беспроводной сети WiFi должна быть отображена голубая стрелка.
4. Коснитесь голубой стрелки. Появится окно, содержащее ваш IP-адрес.

Чтобы упростить подключение, укажите ваш IP-адрес в файле host на вашем рабочем столе. Если вы пользуетесь Mac OS или UNIX, то можете отредактировать ваш файл /etc/hosts. Если вы пользуетесь Windows XP, то можете отредактировать файл C:\Windows\System32\drivers\etc\hosts. Добавьте в ваш файл следующую строчку:

```
x.x.x.x iphone
```

где `x.x.x.x` является IP-адресом вашего iPhone.

Теперь вы готовы подключиться к вашему iPhone с помощью клиента SSH. Если вы пользуетесь Mac OS или Linux с предустановленным SSH, то можете это сделать из терминального окна:

```
$ ssh –l root iphone
```

Если вы пользуетесь Windows XP, то вам придется загрузить клиента SSH. Наиболее распространенным бесплатным клиентом является PuTTY, доступный по адресу: **http://www.chiark.greenend.org.uk/~sgtatham/putty**.

В зависимости от того, какую версию программного обеспечения iPhone вы используете, заданным по умолчанию корневым паролем будет либо `dottie`, либо `alpine`. Как только вы войдете, то сразу же попадете в командную строку командного процессора.

Установка дополнительных компонентов UNIX

Сама по себе возможность доступа к командному процессору вашего iPhone ничего не дает без UNIX, предоставляющей основные команды. Приложение Cydia содержит базовую подсистему UNIX BSD Subsystem, но вам может потребоваться какой-либо конкретный инструмент, который не был установлен. Просмотрите список пакетов в Cydia и найдите те дополнительные инстру-

менты UNIX, которые вам требуются. Выберите и установите эти пакеты, коснувшись их, а затем коснувшись кнопки **Install**.

Дополнительные ресурсы

Apple периодически обновляет программное обеспечение iPhone, поэтому мы не можем описать то, как будет вести себя та или иная версия программного обеспечения — особенно новые версии, которые будут выпущены после публикации этой книги. Неоценимым ресурсом с точки зрения получения самой свежей информации о взломе вашего iPhone или установке приведенных в этой главе инструментов является следующий Web-узел команды разработчиков: iPhone Dev Team (**http://www.iphone-dev.org**). Это официальный узел команды разработчиков iPhone, отвечающей на сегодняшний день за большинство взломов версия 1.*x* и все известные версии 2.*x*.

Глава 2

Начало работы с приложениями

Если вы новичок в мире Mac, то будете удивлены тем фактом, что приложения не являются exe-файлами. Потрясающая архитектура аппаратного обеспечения и графики, которой славится Apple, распространяется и на архитектуру программного обеспечения, и на то, каким способом организованы приложения в файловой системе. Стратегия, используемая в настольных системах Apple, перенесена и на iPhone.

Apple практикует создание модульных независимых приложений с собственными внутренними файловыми ресурсами. В итоге процесс установки большинства приложений сводится всего лишь к простому перетаскиванию их в вашу папку приложений; а удаление их осуществляется путем перетаскивания их в корзину. В этой главе будет объяснена структура приложений iPhone. Кроме того, вы познакомитесь с набором бесплатных инструментов с открытым кодом, используемых для создания исполняемых файлов, а также узнаете, как устанавливать приложения на ваш iPhone. Наконец, вы познакомитесь с языком Objective-C, с его спецификой, достаточной для того, чтобы без труда перейти на него с C или C++.

Анатомия приложения

Apple использует элегантный способ содержать приложения в их операционных системах. Поскольку OS X является платформой на базе UNIX, то Apple хотелось, чтобы она придерживалась основных файловых соглашений UNIX, поэтому использование ветвей ресурсов стало недостаточным (перестало

быть эффективным в этих условиях). Суть задачи была в том, чтобы спроектировать структуру, которая бы позволила приложению оставаться независимым, сохраняя работоспособность в файловой системе. В результате приложение стали рассматривать как *пакет* (bundle) внутри *каталога* (directory) и использовать стандартные API для получения доступа к ресурсам, выполнению бинарных файлов и чтению информации о приложении.

Если вы взглянете внутрь любого приложения Mac, то обнаружите, что расширение app указывает не на файл, а на каталог. Это *программный каталог* (program directory). Внутри он является организованной структурой, содержащей ресурсы, которые необходимы приложению для выполнения, информацию о приложении и исполняемые бинарные файлы приложения. Такую структуру программного каталога компилятор не генерирует, а создает только исполняемые бинарные файлы. Поэтому, чтобы построить цельное приложение, разработчик сам создает скелет структуры, которая, в конечном счете, будет содержать бинарный файл и все его ресурсы.

Программный каталог для приложения iPhone гораздо менее структурирован, нежели настольные приложения Mac. На самом деле все файлы, используемые приложением, находятся в корне программной папки .app:

```
drwxr-xr-x root admin Terminal.app/
       -rw-r--r-- root admin Default.png
       -rw-r--r-- root admin Info.plist
       -rwxr-xr-x root admin Terminal
       -rw-r--r-- root admin icon.png
       -rw-r--r-- root admin pie.png
```

Приведенный выше пример отражает самое основное приложение iPhone под названием MobileTerminal. MobileTerminal — это терминальный клиент с открытым кодом для iPhone, позволяющий пользователю улучшить командный процессор и работать в окружении UNIX (которое тоже должно быть установлено как программное обеспечение сторонних фирм). MobileTerminal иллюстрирует все основные компоненты приложения iPhone:

- MobileTerminal.app — каталог, в котором находятся все ресурсы приложения;
- Default.png — изображение в формате PNG (Portable Network Graphics). Когда пользователь запускает приложение, iPhone анимирует его, чтобы показать, как оно переходит на передний план экрана. Это делается с помощью загрузки файла Default.png и *масштабирования* его до размера всего экрана. Это изображение размером 320×480 пикселов перемещается

на передний план и остается на экране до тех пор, пока приложение не закончит свою загрузку, после чего оно становится фоновым рисунком для любых элементов пользовательского интерфейса, отображаемых на экране. Как правило, для фонового рисунка приложения используют сплошной черный или белый цвета;

- Info.plist — список свойств, содержащий информацию о приложении. Он содержит название исполняемого бинарного файла и идентификатор пакета, используемый приложением SpringBoard для его загрузки. Позднее в этом разделе будет приведен пример списка свойств;
- Terminal — фактически исполняемый бинарный файл, вызываемый при запуске приложения. Это то, что выдает ваш компилятор при построении приложения. При создании конечной производственной версии ваш сборочный файл проекта (make-файл) может копировать ваш бинарный файл в папку приложения. В этой главе будет представлен пример такого процесса;
- icon.png — изображение, создающее значок приложения в SpringBoard (приложение рабочего стола iPhone). SpringBoard не волнует размер файла, и оно будет пытаться отобразить изображение за пределами области значка, если то достаточно велико. Большинство значков, как правило, имеет размер 60×60 пикселов;
- pie.png — изображение, используемое приложением MobileTerminal. Окружение iPhone предоставляет множество методов для выбора ресурсов, большинство из них принимают только имя файла и не принимают путь к файлу. Поэтому файл, передаваемый в эти методы, должен храниться непосредственно в программном каталоге. Такой подход соответствует попыткам Apple делать приложения независимыми.

Создание скелета приложения

Первое, что вы должны сделать прежде, чем приступить к созданию приложения, — это составить скелет каталога .app, который будет его содержать.[1] Скелет будет предоставлять всю информацию, необходимую iPhone для под-

[1] Технически вполне возможно запускать приложение напрямую из командной строки iPhone, но это рушит многие функции уровня приложения. Само по себе SpringBoard тесно интегрировано с платформой пользовательского интерфейса, поэтому чтобы ваше приложение стало полностью работоспособным, оно должно быть соответствующим образом смонтировано и вызвано.

твержденияя существования вашего приложения, что позволит запускать его из SpringBoard.

В этой книге представлено множество полнофункциональных примеров кода. Для корректного запуска их вы должны построить пример скелета под названием MyExample.app. Создать каталог достаточно просто:

```
$ mkdir MyExample.app
```

Далее, напишите список свойств, чтобы описать само приложение и то, как его запустить. Файл Info.plist отображает список свойств в формате XML. Он должен выглядеть следующим образом:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
 "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>English</string>
    <key>CFBundleExecutable</key>
    <string>MyExample</string>
    <key>CFBundleIdentifier</key>
    <string>com.oreilly.www.iphone.examples</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleSignature</key>
    <string>????</string>
    <key>CFBundleVersion</key>
    <string>1.0</string>
</dict>
</plist>
```

В приведенном примере большинство наиболее важных параметров выделено полужирным шрифтом. Это значения свойств `CFBundleExecutable` и `CFBundleIdentifier`. Свойство `CFBundleExecutable` задает имя исполняемого бинарного файла в пределах папки. Это именно тот файл, который начинает исполняться при запуске вашего приложения, т. е. то, что выдает ваш

компилятор. В данном примере имя файла совпадает с именем приложения, но это совершенно необязательно.

Свойство `CFBundleIdentifier` определяет уникальный идентификатор, под которым будет находиться ваше приложение. Уровень приложения iPhone больше заботится об адресации вашего приложения целиком, нежели самого бинарного файла. Всякий раз, когда SpringBoard (или другое приложение) запускает приложение MyExample, ссылаться на него нужно будет с использованием этого идентификатора. Его имя должно быть уникальным среди всех других приложений на iPhone. Наиболее распространенным способом, гарантирующим уникальность идентификатора, является включение в его имя URL вашего Web-узла.

Файлы приложения icom.png и Default.png тоже должны быть скопированы. Если это будет упущено, то iPhone использует вместо них по умолчанию совершенно неприглядные изображения. Чтобы показать, что имеется в виду, мы не будем включать эти файлы в наш пример. Но чтобы ваше приложение выглядело вполне профессионально, убедитесь, что при публикации своего приложения вы создали изображения и включили файлы этих изображений с такими именами.

Теперь наш скелет вполне подходит для выполнения примера. В следующем разделе вы установите пакет инструментов на ваш рабочий стол, после чего сможете компилировать примеры приложений. В последующих главах вы создадите множество примеров. После того как каждый из них будет построен, исполняемый бинарный файл `MyExample` нужно будет скопировать в вашу программную папку. Законченное ваше приложение будет выглядеть следующим образом:

```
drwxr-xr-x   root   admin   MyExample.app/
-rw-r--r--   root   admin   Info.plist
-rwxr-xr-x   root   admin   MyExample
```

Примеры, приводимые в этой книге, как правило, не требуют никаких дополнительных ресурсов, поэтому изображения и звуки будут необходимы только при вызове их из примеров. Однако в этих случаях вы скопируете требуемые файлы в каталог MyExample.app. Большинство примеров старается использовать существующие на iPhone файлы, чтобы не переполнять эту книгу большим количеством бинарного кода.

Создание бесплатного пакета инструментов

Как мы уже говорили в *главе 1*, iPhone зарождался как закрытая платформа. Изначально это означало, что никакой инструментарий разработчика для создания приложений для iPhone не будет общедоступен. Было много разговоров о том, что Apple втайне надеялась на то, что сообщество взломает телефон, подтверждая тем самым свой статус среди помешанных на этом сообществ. В течение первых нескольких месяцев жизни iPhone именно это и происходило. Сообщество открытого кода успешно взломало телефон и начало написание пакета инструментов для создания приложений. С тех пор оно уже выпущено как бесплатное программное обеспечение. Пакет инструментов состоит из кросс-компилятора, компоновщика, транслятора, блока C в компоновщике под названием Csu и заголовков классов для платформы Objective-C, генерируемых инструментом под названием class-dump.

К сегодняшнему дню данный пакет инструментов претерпел множество изменений и усовершенствований, сделанных Джеем Фриманом, также известным как saurik, и доступен в двух видах. Настольная версия пакета инструментов использует кросс-компилятор, выполняющийся на одной машине (а именно на вашей настольной), но создает исполняемые файлы, которые могут исполняться на другой машине (в iPhone это процессор ARM). Родной компилятор также может быть установлен напрямую на iPhone, позволяя тем самым вам создавать приложения без необходимости установки на ваш рабочий стол какого-либо специализированного программного обеспечения.

Приводя в этой книге команды и пути к файлам, мы исходим из того, что для создания и/или установки данного пакета инструментов вы использовали процедуры из этой главы. Обновление пакета инструментов происходит периодически в виде выпуска новых версий, поэтому процесс его установки может иногда меняться. Самые последние инструкции по созданию пакета инструментов на рабочем столе можно найти на Web-узле Джея Фримана по адресу: **http://www.saurik.com**.

По умолчанию пакет инструментов создается и устанавливается в папку /toolchain. Все приводимые в этой книге примеры предполагают, что именно в нее был установлен пакет. Если же вы создали пакет инструментов ранее или же только что обеспокоились модификацией файлов в ней, то вы наверняка захотите переместить вашу текущую папку /toolchain в другое место и начать с чистого каталога.

Внимание!

Если у вас имеется предыдущая версия пакета инструментов, то более новая версия может создаться некорректно. Чтобы убедиться в том, что вы начинаете с чистой установки, переместите вашу старую копию в другое место.

Что вам потребуется

Если вы ищете способ установки пакета инструментов напрямую в iPhone, то единственное, что вам потребуется, — это взломанный iPhone с запущенным на нем установщиком Cydia. Чтобы установить пакет инструментов, просто запустите Cydia, а затем в списке **Sections** выберите раздел **Development**. Найдите и выделите в списке iPhone 2.0 Toolchain, затем коснитесь кнопки **Install**. На ваш iPhone будет установлен пакет инструментов, и вы сможете вызывать его с помощью стандартной команды `gcc`. Теперь вы можете пропустить оставшуюся часть данного раздела.

Создание пакета на рабочем столе — это более сложный и запутанный процесс. Хотя в Интернете существует несколько неофициальных бинарных дистрибутивов пакета инструментов, вы создадите его из источников, приводимых в этом разделе.

Поддерживаемые настольные платформы

Первое, что вам потребуется, — это поддерживаемая настольная платформа. На текущий момент пакетом инструментов поддерживаются следующие платформы:

- Mac OS X 10.4 Intel или PPC;
- Mac OS X 10.5 Intel;
- Ubuntu Feisty Fawn, Intel;
- Ubuntu Gutsy Gibbon, Intel;
- Fedora Core, Intel;
- Gentoo Linux 2007.0, *x*86_64;
- Debian 2.6.18;
- CentOS 4.

Другие платформы следуют тем же основным шагам, что и приведенные выше. Официальные инструкции пакета инструментов можно найти на Web-узле **http://www.saurik.com**.

Высокоскоростное подключение к Интернету

Пакет инструментов имеет достаточно большой объем, даже в виде исходных файлов. Если только вы не хотите провести в ожидании несколько дней, то вы, скорее всего, загрузите исходные файлы по высокоскоростному каналу Интернета. Если у вас нет такого канала, то лучше произвести установку из библиотеки или в местном интернет-кафе.

Инструменты с открытым кодом

Следующее, что вам потребуется, — это набор обязательных инструментов с открытым кодом, установленных на вашем рабочем столе:

- bison (версия 1.28 или позднее);
- flex (версия 2.5.4 или позднее);
- gcc (компилятор GNU, который обрабатывает команды C, C++ и Objective-C);
- git (утилита контроля исходного кода).

Если у вас нет каких-либо из этих инструментов, то прежде чем продолжить, загрузите и установите их. На Mac все эти инструменты, кроме git, входят в состав набора инструментов Xcode, но прежде чем продолжить, лучше установить или обновить Xcode на самую последнюю версию. Большинство других операционных систем предлагает эти инструменты в своих дистрибутивах как необязательные компоненты.

> ***Примечание***
>
> Инструменты Xcode могут быть загружены с Web-узла Apple по адресу: **http://developer/apple.com/tools/xcode**.

Файловая система iPhone

Вам необходимо сделать копию файловой системы вашего iPhone, особенно библиотек и оболочек. Из-за возможных необратимых последствий и в связи

с тем, что наши юристы заставляют нас сделать это, мы публикуем этот отказ от ответственности:

> ***Внимание!***
>
> Установка пакета инструментов требует от вас создания копии библиотек с вашего iPhone на ваш рабочий стол. Убедитесь в том, что это не противоречит местному и федеральному законодательству, а также законодательству штата, в котором вы находитесь.

После установки SSH на iPhone *(см. главу 1)* для загрузки в папку /toolchain/sys необходимых вам файлов воспользуйтесь следующими командами:

```
$ sudo -s
# mkdir -p /toolchain/sys/
# cd /toolchain/sys/
# mkdir -p ./System/Library ./usr
# scp -r root@iphone:/System/Library/Frameworks/ ./System/Library
# scp -r root@iphone:/System/Library/PrivateFrameworks/ ./System/Library
# scp -r root@iphone:/usr/lib ./usr
```

Компиляция пакета инструментов

Прежде чем вы начнете, вы должны задать несколько переменных окружения, чтобы определить, куда устанавливается пакет инструментов. Чтобы установить в /toolchain, используйте приведенные далее пути. Конечно, вы можете изменить их согласно своим предпочтениям. Убедитесь в том, что вы поместили загруженные библиотеки и оболочки туда, куда указано в `${sysroot}`:

```
# export target=arm-apple-darwin9
# export prefix=/toolchain/pre
# export sysroot=/toolchain/sys
# export PATH="${prefix}/bin":$PATH
# export cctools=/toolchain/src/cctools
# export gcc=/toolchain/src/gcc
# export csu=/toolchain/src/csu
# export build=/toolchain/bld
```

Как только все переменные будут экспортированы, вы готовы к созданию пакета.

Шаг 1. Установка Csu

Csu предоставляет C-вход (C hooks) в точку входа "start" сборки и задает стек, чтобы могла быть вызвана функция `main()` вашей программы. Это, по сути, связующий код:

```
# mkdir -p ${csu}
# cd "${csu}"
# svn co http://iphone-dev.googlecode.com/svn/trunk/csu .
# cp -R *.o "${sysroot}"/usr/lib
# cd "${sysroot}"/usr/lib
# chmod 644 *.o
# cp -Rf crt1.o crt1.10.5.o
# cp -Rf dylib1.o dylib1.10.5.o
```

Шаг 2. Построение и установка инструментов кросс-компилятора

Приведенные далее команды создают и устанавливают компоненты кросс-компилятора пакета инструментов.[1] Это весьма специфично для Mac OS X, поэтому если вы используете другую платформу, то обратитесь к ее официальной документации:

```
# rm -rf "${cctools}"
# svn co http://iphone-dev.googlecode.com/svn/branches/odcctools-9.2-ld
"${cctools}"
# mkdir -p "${build}"
# cd "${build}"
# mkdir cctools-iphone
# cd cctools-iphone
# CFLAGS=-m32 LDFLAGS=-m32 "${cctools}"/configure ↳
    --target="${target}" ↳
```

[1] Здесь и далее в коде встречается символ ↳, означающий перенос строки. То есть код между этим символом на самом деле надо набирать в одну строку. — *Ред.*

```
    --prefix="${prefix}" ↳
    --disable-ld64
# make
# make install
```

Шаг 3. Установка системных заголовков

Системные заголовки, имеющиеся в вашем Xcode SDK, являются общими для вашего рабочего стола и платформы iPhone, однако некоторые макросы прекомпилятора сильно привязаны к архитектуре. Поскольку архитектура iPhone отличается от архитектуры рабочего стола, то для того чтобы эти заголовки работали для iPhone, их необходимо установить. Чтобы установить специфичный для iPhone набор заголовков в ваш недавно созданный пакет инструментов, выполните приведенные далее команды. Все это основано на заголовках Xcode:

```
# cd "${build}"
# svn co http://iphone-dev.googlecode.com/svn/branches/include-1.2-sdk
include
# cd include
# ./configure --prefix="${sysroot}"/usr
# bash install-headers.sh
```

Шаг 4. Построение и установка LLVM

После установки заголовков последним шагом является создание компилятора LLVM. Оболочка LLVM (Low Level Virtual Machine) обеспечивает стандартную инфраструктуру для создания компиляторов. Она предоставляет необходимые входы и API для создания стандартизованного компилятора без необходимости переписывать все основные компоненты компилятора. Чтобы скомпилировать и установить версию компилятора LLVM, выполните следующие команды:

```
# rm -rf "${gcc}"
# git clone git://git.saurik.com/llvm-gcc-4.2 "${gcc}"
# mkdir -p "${build}"
# cd "${build}"
# mkdir gcc-4.2-iphone
# cd gcc-4.2-iphone
```

```
# "${gcc}"/configure ↳
    --target="${target}" ↳
    --prefix="${prefix}" ↳
    --with-sysroot="${sysroot}" ↳
    --enable-languages=c,c++,objc,obj-c++ ↳
    --with-as="${prefix}"/bin/"${target}"-as ↳
    --with-ld="${prefix}"/bin/"${target}"-ld ↳
    --enable-wchar_t=no ↳
    --with-gxx-include-dir=/usr/include/c++/4.0.0
# make -j2
# make install
# mkdir -p "${sysroot}"/"$(dirname "${prefix}")"
# ln -s "${prefix}" "${sysroot}"/"$(dirname "${prefix}")"
```

Создание и установка приложений

Теперь, когда установлен пакет инструментов, пришло время узнать, как им пользоваться. Существуют два основных способа создания исполняемых файлов: командная строка и make-файл.

Приводимые в этой книге примеры достаточно просты, поэтому они могут быть созданы с помощью командной строки. Пакет инструментов совместим со стандартными аргументами компилятора и должен быть вам знаком, если вы когда-либо пользовались gcc. Прежде чем попытаться воспользоваться кросс-компилятором, убедитесь в том, что среди ваших путей имеется /toolchain/pre/bin:

```
$ export PATH=$PATH:/toolchain/pre/bin
```

Анатомия типичного компилятора командной строки следующая:

```
$ arm-apple-darwin9-gcc -o MyExample MyExample.m -lobjc ↳
  -framework CoreFoundation -framework Foundation ↳
  -march=armv6 -mcpu=arm1176jzf-s
```

Здесь:

- `arm-apple-darwin9-gcc` — само название кросс-компилятора. Оно располагается в /toolchain/pre/bin, поэтому убедитесь в том, что эта папка добавлена в ваши пути;

- `-o MyExample` — сообщает компилятору выводить скомпилированный исполняемый фай в файл с названием MyExample;
- `MyExample.m` — имена исходных файлов, включенных в программу, разделяемых пробелами. Расширения m сообщает компилятору о том, что исходники написаны на Objective-C;
- `-lobjc` — указывает компилятору скомпоновать библиотеки сообщений Objective-C, необходимые всем приложениям iPhone, в пакет инструментов. Помимо всего прочего, эта библиотека склеивает вызовы функций C-стиля с сообщениями Objective-C.
- `-framework CoreFoundation -framework Foundation` — компоновка двух основных оболочек в приложение. В зависимости от того, какие компоненты операционной системы используются в коде, разные оболочки предоставляют различные уровни функциональности. В этой книге вы познакомитесь со множеством различных оболочек;
- `-march=armv6 -mcpu=arm1176jzf-s` — задает корректную архитектуру и тип процессора исполняемого файла.

Командной строки будет более чем достаточно для большинства небольших приложений и примеров, но для больших приложений имеет смысл писать *make-файл* (makefile). Make-файл — это простой текстовый файл, который выступает в роли декларации при построении приложений. Он используется программой `make`, являющейся портативной утилитой построения, поставляемой с множеством наборов средств разработки. Программа `make` отвечает за вызов компилятора (и компоновщика) и передачу им необходимых флагов и параметров. Make-файлы являются логическими способами спланировать структуру приложения. Кроме того, они позволяют разработчику легко наводить порядок среди объектных файлов в каталоге, создавать пакеты приложений, а также выполнять целый ряд других задач, весьма полезных при создании приложений.

Предыдущий пример командной строки может быть переписан в make-файл так, как показано далее. Make-файл назван Makefile и помещен в исходный каталог:

```
CC = /toolchain/pre/bin/arm-apple-darwin9-gcc
LD = $(CC)
LDFLAGS = -lobjc ↳
          -framework CoreFoundation ↳
          -framework Foundation
CFLAGS = -march=armv6 -mcpu=arm1176jzf-a
```

```
all:      MyExample

MyExample: MyExample.o
           $(LD) $(LDFLAGS) -o $@ $^

%.o:       %.m
           $(CC) -c $(CFLAGS) $(CPPFLAGS) $< -o $@

%.o:       %.c
           $(CC) -c $(CFLAGS) $(CPPFLAGS) $< -o $@

%.o:       %.cpp
           $(CC) -c $(CFLAGS) $(CPPFLAGS) $< -o $@
```

> ***Внимание!***
>
> Все отступы являются в действительности табуляциями. Табуляции необходимо использовать для корректной работы make-файла.

Как только make-файл создан, исполняемый файл приложения может быть создан с помощью всего одной простой команды:

```
$ make
```

Кроме создания приложения можно добавить функциональность для копирования исполняемого файла приложения в созданный вами скелет программной папки:

```
package:
           cp -p MyExample ./MyExample.app/
```

Добавив это в make-файл, вы сможете запустить `make package` для автоматического задания вашего каталога .app.

Другой весьма распространенный вариант использования make-файлов — это очистка каталога, чтобы его можно было отправить кому-либо другому. Вы даже можете указать make-файлу удалить исполняемый файл, который был скопирован в программную папку:

```
clean:
           rm -f *.o *.gch
           rm -f ./MyExample.app/MyExample
```