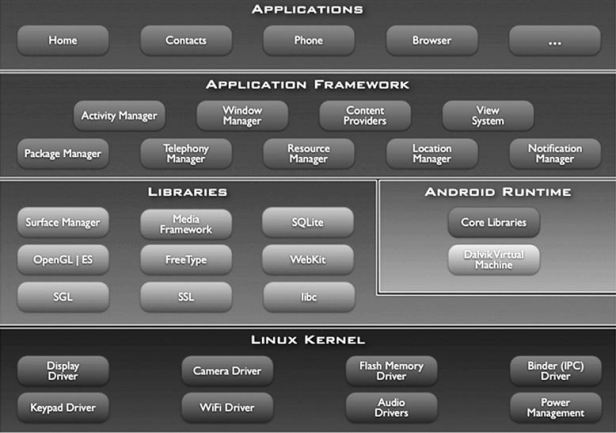
**Архитектура ОС Android**

Если представить компонентную модель *Android* в виде иерархии (рис. 1), то в самом низу, в самой основе будет располагаться *ядро* операционной системы. Оно обеспечивает функционирование системы и отвечает за *безопасность*, *управление памятью*, энергосистемой и процессами, а также предоставляет сетевой *стек* и модель драйверов. *Ядро* также действует как *уровень абстракции* между аппаратным и программным обеспечением.

"Выше" ядра, как *программное обеспечение промежуточного слоя*, лежит набор библиотек (Libraries), предназначенный для решения типовых задач, требующих высокой эффективности. То есть, именно этот уровень отвечает за предоставление реализованных алгоритмов для вышележащих уровней, поддержку файловых форматов, осуществление кодирования и декодирования информации (в пример можно привести мультимедийные кодеки), отрисовку графики и многое другое. Библиотеки реализованы на C/C++ и скомпилированы под конкретное *аппаратное обеспечение* устройства, вместе с которым они и поставляются производителем в предустановленном виде.

Перечислим некоторые из низкоуровневых библиотек:

1. Surface Manager - в ОС Android используется композитный менеджер окон, наподобие Compiz (Linux), но более примитивный. Вместо того, чтобы производить рисование графики напрямую в буфер дисплея, система посылает поступающие команды рисования в закадровый буфер, где они накапливаются вместе с другими, составляя некую композицию, а потом выводятся пользователю на экран. Это позволяет системе создавать интересные бесшовные эффекты, реализовать прозрачность окон и плавные переходы.
2. Media Framework - библиотеки, реализованные на базе PacketVideo OpenCORE. С их помощью система может осуществлять запись и воспроизведение аудио и видео данных, а также вывод статических изображений. Поддерживаются многие популярные форматы, включая MPEG4, H.264, MP3, AAC, AMR, JPG и PNG. В будущем на смену OpenCORE должен придти более простой фреймворк Stagefright.
3. SQLite - легковесная и производительная реляционная СУБД, используемая в Android в качестве основного движка для работы с базами данных.
4. 3D библиотеки - используются для высокооптимизированного рисования 3D-графики, при возможности используют аппаратное ускорение. Их реализации строятся на основе API OpenGL ES 1.0.
5. FreeType - библиотека для работы с битовыми картами, а также для растеризации шрифтов и осуществления операций над ними. Это высококачественный движок для шрифтов и отображения текста.
6. LibWebCore - библиотеки известного браузерного движка WebKit, используемого также в десктопных браузерах Google Chrome и Apple Safari.
7. SGL (Skia Graphics Engine) - открытый движок для работы с 2D-графикой. Графическая библиотека является продуктом Google и часто используется в других их программах.
8. SSL - библиотеки для поддержки одноименного криптографического протокола на базе OpenSSL.
9. libc - библиотека стандартных вызовов языка C, аналог glibc (GNU libc из Linux) для маленьких устройств. Носит название Bionic.



**Рис. 1.**Компонентная модель ОС Android

На этом же уровне располагается *Android* Runtime - среда выполнения прикладных программ. Ключевыми её составляющими являются набор стандартных библиотек и *виртуальная машина* Dalvik. Каждое *приложение* в ОС *Android* запускается в собственном экземпляре виртуальной машины Dalvik. Таким образом, все работающие процессы изолированы от операционной системы и друг от друга. *Архитектура* *Android* Runtime такова, что работа программ осуществляется строго в рамках окружения виртуальной машины. Благодаря этому осуществляется защита ядра операционной системы от возможного вреда со стороны других её составляющих. Поэтому код с ошибками или *вредоносное ПО* не смогут испортить ОС *Android* и устройство на её базе. Такая защитная *функция*, наряду с выполнением программного кода, является одной из ключевых для *Android* Runtime.

Уровнем выше располагается *Application* Framework, иногда называемый уровнем каркаса приложений. Именно через каркасы приложений разработчики получают *доступ* к *API*, предоставляемым компонентами системы, лежащими ниже уровнем. Кроме того, благодаря архитектуре фреймворка, любому приложению предоставляются уже реализованные возможности других приложений, к которым разрешено получать *доступ*. В базовый набор сервисов и систем, лежащих в основе каждого приложения и являющихся частями фреймворка, входят:

1. Богатый и расширяемый набор представлений (Views), который может быть использован для создания визуальных компонентов приложений, например, списков, текстовых полей, таблиц, кнопок или даже встроенного web-браузера.
2. Контент-провайдеры (Content Providers), управляющие данными, которые одни приложения открывают для других, чтобы те могли их использовать для своей работы.
3. Менеджер ресурсов (Resource Manager), обеспечивающий доступ к ресурсам, не несущим кода, например, к строковым данным, графике, файлам и другим.
4. Менеджер оповещений (Notification Manager), благодаря которому все приложения могут отображать собственные уведомления для пользователя в строке состояния.
5. Менеджер действий (Activity Manager), который управляет жизненными циклами приложений, сохраняет данные об истории работы с действиями, а также предоставляет систему навигации по ним.
6. Менеджер местоположения (Location Manager), позволяющие приложениям периодически получать обновленные данные о текущем географическом положении устройства.

На вершине программного стека *Android* лежит уровень приложений (Applications). Сюда относится набор базовых приложений, который предустановлен на ОС *Android*. Например, в него входят *браузер*, *почтовый клиент*, *программа* для отправки *SMS*, карты, календарь, *менеджер* контактов и многие другие. *Список* интегрированных приложений может меняться в зависимости от модели устройства и версии *Android*. И помимо этого базового набора к уровню приложений относятся все прикладные приложения под платформу *Android*, в том числе и установленные пользователем.

Как правило, приложения под *Android* пишутся на языке *Java*, но существует возможность разрабатывать программы и на C/C++ (с помощью *Native* *Development Kit*). Экзотикой можно назвать использования *Basic* (с помощью Simple) и других языков. Также можно создавать собственные программы с помощью конструкторов приложений, таких как *App* Inventor.

**1.6. Особенности ядра**

*Ядро* является самой важной частью ОС Linux, и в отличие от других его частей, было перенесено в ОС *Android* почти полностью. Тем не менее, в процессе переноса на *ядро* было наложено около 250 патчей.

В ядре ОС *Android* было решено отказаться от средств межпроцессного взаимодействия ОС Linux и вместо них создать единый механизм, названный *Binder*. *Binder* позволяет вызывать методы одного процесса из другого процесса, передавая им аргументы и получая результат, подобно тому, как методы вызываются внутри одного процесса. *Binder* делает эту работу с минимальным использованием памяти.

Для обеспечения отладки на маленьких устройствах в *ядро* добавлен *вывод* отладочной информации в *последовательный порт* и реализована *поддержка* команды logcat.

Большие изменения коснулись работы с памятью. Традиционная разделяемая *память* Linux shmem была заменена на ashmem. Та же задача, но на уровне физической памяти, решается с помощью драйвера pmem. Добавлен специальный обработчик нехватки памяти (out of *memory*), названный Viking *Killer*, в простейшем случае он просто убивает процесс, но могут быть заданы более сложные правила.

В сетевой *стек* добавлены новые настройки безопасности, *поддержка* файловой системы для флеш-носителей YAFFS2 включена в *ядро*.

**1.7. Java-машина Dalvik**

Dalvik *Virtual Machine* является частью мобильной платформы *Android*. Это *виртуальная машина*, автором которой является Дэн Бронштейн. Она распространяется как *свободное программное обеспечение* под *BSD*-совместимой лицензией *Apache* 2.0. Во многом именно этот факт сыграл свою роль в решении Google отказаться от JME (*Java* *Micro* *Edition*), на которую необходимо было бы получить лицензию от *Sun*. Поэтому *корпорация*, целью которой было создание открытой операционной системы, разработало свою собственную виртуальную машину.

В отличие от большинства виртуальных машин (той же *Java* *Virtual Machine*), которые являются *стек*-ориентированными, Dalvik является регистр-ориентированной, что нельзя назвать стандартным решением. С другой стороны, оно очень хорошо подходит для работы на процессорах RISC-архитектуры, к которым относятся и процессоры *ARM*, очень широко применяемые в мобильных устройствах.

Dalvik проектировалась специально под платформу *Android*. Учитывался тот факт, что платформа строит все процессы как изолированные, выполняющиеся каждый в своём адресном пространстве. *Виртуальная машина* оптимизирована для низкого потребления памяти и работы на мобильном аппаратном обеспечении. Начиная с версии *Android* 2.2., Dalvik использует *JIT* (Just-in-Time) компиляцию. В результате этих особенностей, получилась быстрая и производительная *виртуальная машина*, что не может не сказываться на работе приложений в целом.

Dalvik использует собственный байт-код. При разработке приложения под *Android* переводятся компилятором в специальный машинно-независимый низкоуровневый код. При выполнении на платформе именно Dalvik интерпретирует и выполняет такую программу.

Кроме того, Dalvik способна переводить байт-коды *Java* в коды собственного формата и также исполнять их в своей виртуальной среде. Программный код пишется на языке *Java*, а после компиляции все .*class* файлы конвертируются в формат .dex (пригодный для интерпретации в Dalvik) с помощью специальной утилиты dx, входящей в состав *Android* *SDK*.

**1.8. Bionic**

*Bionic* - библиотека стандартных вызовов языка C, распространяемая под лицензией *BSD* (Berkeley *Software* *Distribution* ? система распространения программного обеспечения в исходных кодах, созданная для обмена опытом между учебными заведениями) и разработанная Google для *Android*. В *bionic* отсутствуют некоторые не используемые в *Android* функции *POSIX*, доступные в полной реализации glibc.

Основные отличия *bionic* :

1. BSD лицензии: Android использует Linux ядро, которое находится под GNU General Public License (GPL), но Google пожелал изолировать приложения для Android от последствий GPL. GNU libc, который обычно используется с ядром Linux находится под лицензией GNU LGPL, как альтернативный uClibc.
2. малые размеры: объектный код bionic намного меньше (примерно в 2 раза), чем glibc и несколько меньше, чем uclibc.
3. bionic предназначена для процессоров c относительно низкими тактовыми частот.
4. усеченная, но эффективная реализация нитей POSIX.

Рекомендуемый способ непосредственного использования и расширения *bionic* это использование *Android* *Native* *Development Kit*.

**1.9. Обзор Java-интерфейсов прикладного программиста**

Для прикладного программиста *Android* - набор интерфейсов на языке *Java*. Рассмотрим, как он организован. В основе набора - пакеты, входящие в *стандарт языка* *Java*, такие как java.util, java.lang, java.io. Они есть на любой платформе, где могут быть запущены *java*-приложения, и неспецифичны для *Android*. К ним примыкают расширения, которые в *стандарт языка* не входят, но де-факто давно являются стандартными - пакеты javax.net, javax.xml.

Также в *Android* включены менее распространенные расширения *Java* - пакет org.apache.http, самая солидная реализация протокола *HTTP*. Пакет org.json отвечает за сериализацию объектов JavaScript и поддержку технологии *AJAX*. Пакет org.w3c.dom обеспечивает объектную модель документа *HTML*, а пакет org.xml.sax - работу с *XML*. Такой подбор компонентов свидетельствует об ориентации на веб-разработку, веб-приложения. Одновременно использование ставших классическими библиотек облегчает перенос приложений на *Android*.

Наконец, самым большим и интересным является набор интерфейсов, созданных специально для *Android*. Рассмотрим некоторые из его пакетов.

Пакеты android.view и android.widget отвечают за графический *интерфейс* пользователя (*GUI*). Они содержат набор встроенных виджетов, таких как кнопки и поля ввода, компоновки (*layout*) для расположения виджетов на экране, взаимодействие виджета с пользователем. С их помощью можно создать простейшее *приложение* для *Android*.

Для работы с примитивами рисования и графическими файлами предназначен пакет android.graphics. С помощью android.animation можно создавать несложную анимацию. Начиная с версии *Android* 3.0 стала доступна мощная и универсальная система *Property* *Animation*, в более ранних версиях *анимация* была либо привязана к *GUI*, либо просто представляла собой набор кадров.

Пакет android.opengl предоставляет движок *OpenGL* ES 2.0, android.gesture - *поддержка* управления жестами на сенсорном экране, позволяет распознавать жесты и создавать новые.

Большое количество интерфейсов предназначено для коммуникации. Пакет android.net включает стеки сетевых протоколов высокого уровня, таких как *HTTP* и *SIP*, поддержку WiFi. Пакет droid.webkit - популярный движок веб-браузера, позволяет легко отображать веб-страницы в приложении. Пакеты android.bluetooth и android.nfc предоставляют стеки протоколов связи на коротких расстояниях *BlueTooth*.и Near *Field* *Communication* соответственно. Пакет *android*.*telephony* дает *доступ* к телефонной функциональности - например, *информация* о соте или *отправка* *SMS*. Пакет *android*.drm позволяет контролировать защищенный *контент* с помощью системы *Digital Rights* *Management*.

Для управления прикладными приложениями предназначен пакет android.app. Пакет android.os - *Java*-обертка для некоторых системных библиотек, например, для *Binder*. Пакет *android*.*hardware* позволяет обращаться к камере и датчикам, а пакет android.location предоставляет информации о географических координатах устройства, в т. ч. с помощью датчика *GPS*.

Пакет *android*.*media* отвечает за *кодирование* звуковых и видео потоков, для маленьких устройств это до сих пор вычислительно сложная задача, требующая качественной оптимизации. Пакет android.database предоставляет *доступ* к базам данных, включая SQLite.