**Android NDK (native development kit)** – это набор инструментов, которые позволяют реализовать часть вашего приложения используя такие языки как С/С++.

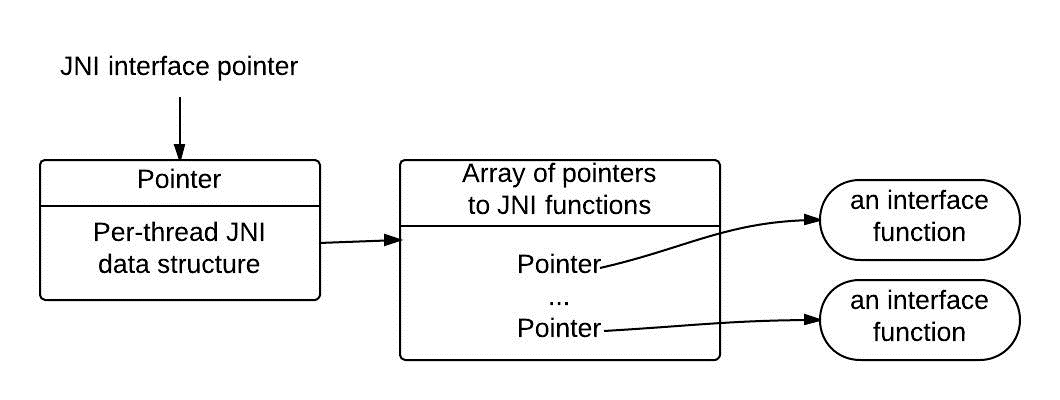
Google рекомендует прибегать к использованию NDK только в редчайших случаях. Зачастую это такие случаи:

* Нужно увеличить производительность (например, сортировка большого объема данных);
* Использовать стороннюю библиотеку. Например, много уже чего написано на С/С++ языках и нужно просто заиспользовать существующий материал. Пример таких библиотек, как, Ffmpeg, OpenCV;
* Программирование на низком уровне (например, всё что выходит за рамки Dalvik);

Java Native Interface – стандартный механизм для запуска кода, под управлением виртуальной машины Java, который написан на языках С/С++ или Assembler, и скомпонован в виде динамических библиотек, позволяет не использовать статическое связывание. Это даёт возможность вызова функции С/С++ из программы на Java, и наоборот.

Основное преимущество перед аналогами (Netscape Java Runtime Interface или Microsoft’s Raw Native Interface and COM/Java Interface) является то что JNI изначально разрабатывался для обеспечения двоичной совместимости, для совместимости приложений, написанных на JNI, для любых виртуальных машин Java на конкретной платформе (когда я говорю о JNI, то я не привязываюсь к Dalvik машине, потому как JNI был написан Oracle для JVM который подходит для всех Java виртуальных машин). Поэтому скомпилированный код на С/С++ будет выполнятся в не зависимости от платформы. Более ранние версии не позволяли реализовывать двоичную совместимость.

Двоичная совместимость или же бинарная совместимость – вид совместимости программ, позволяющий программе работать в различных средах без изменения её исполняемых файлов.



JNI таблица, организована как таблица виртуальных функций в С++. VM может работать с несколькими такими таблицами. Например, одна будет для отладки, вторая для использования. Указатель на JNI интерфейс действителен только в текущем потоке. Это значит, что указатель не может гулять с одного потока в другой. Но нативные методы могут быть вызваны из разных потоков.

Примитивные типы копируются между VM и нативным кодом, а объекты передаются по ссылке. VM обязана отслеживать все ссылки, которые передаются в нативный код. Все переданные ссылки в нативный код не могут быть освобождены GC. Но нативный код в свою очередь должен информировать VM о том, что ему больше не нужны ссылки на переданные объекты.

JNI делит ссылки на три типа: локальные, глобальные и слабые глобальные ссылки. Локальные действительны пока не завершиться метод. Все Java объекты которые возвращает функции JNI являются локальными. Программист должен надеется на то что VM сама подчистит все локальные ссылки. Локальные ссылки доступны лишь в том потоке в котором были созданы. Однако если есть необходимость то их можно освобождать сразу методом JNI интерфейса DeleteLocalRef.

Глобальные ссылки остаются пока они явно не будут освобождены. Что бы зарегистрировать глобальную ссылку следует вызвать метод NewGlobalRef. Если же глобальная ссылка уже не нужна, то её можно удалить методом DeleteGlobalRef.

JNI не проверяет ошибки такие как NullPointerException, IllegalArgumentException. Причины:

* снижение производительности;
* в большинстве функций C библиотек очень и очень трудно защитится от ошибок.

JNI позволяет использовать Java Exception. Большинство JNI функций возвращают код ошибок а не сам Exception, и поэтому приходится обрабатывать сам код, а в Java уже выбрасывать Exception. В JNI следует проверять код ошибки вызываемых функций и после них следует вызвать ExceptionOccurred(), которая в свою очередь возвращает объект ошибки.

В JNI существуют свои примитивные и ссылочные типы данных.

Java Type Native Type Description

boolean jboolean unsigned 8 bits

byte jbyte signed 8 bits

char jchar unsigned 16 bits

short jshort signed 16 bits

int jint signed 32 bits

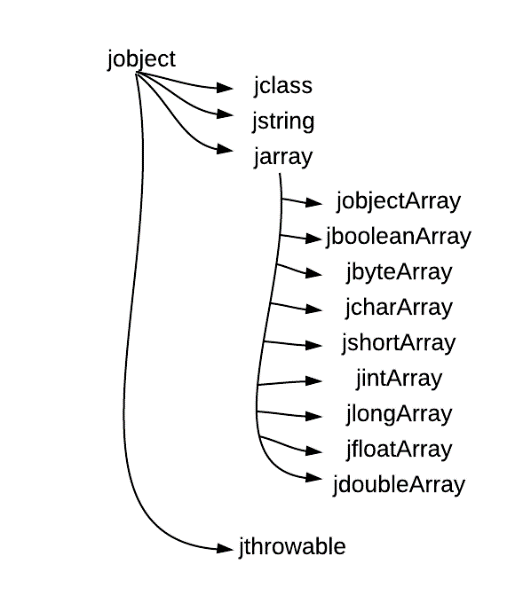
long jlong signed 64 bits

float jfloat 32 bits

double jdouble 64 bits

void void N/A

Ссылочные типы JNI



Модифицированный UTF-8

JNI использует модифицированную кодировку UTF-8 для представления строк. Java в свою очередь использует UTF-16. UTF-8 в основном используется в С, потому что он кодирует \u0000 в 0xc0, вместо привычной 0x00. Изменённые строки кодируются так, что последовательность символов, которые содержат только ненулевой ASCII символы могут быть представлены с использованием только одного байта.

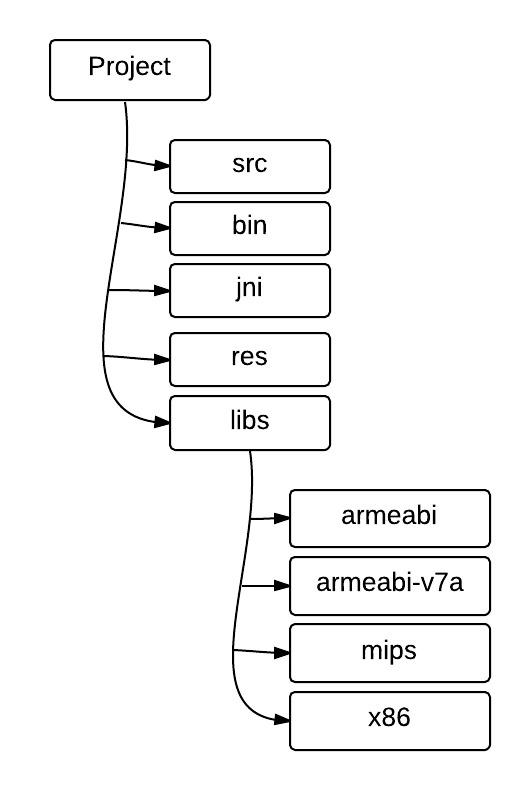
Функции JNI

Интерфейс JNI содержит в себе не только собственный набор данных, но и свои собственные функции. На их рассмотрение уйдёт много времени, так как их не один десяток. Ознакомится с ними вы сможете в официальной документации.

Потоки

Всеми потоками в Linux управляет ядро, но они могут быть прикреплены к JavaVM функциями AttachCurrentThread и AttachCurrentThreadAsDaemon. Пока поток не присоединён он не имеет доступа к JNIEnv. Важно, Android не приостанавливает потоки которые были созданы JNI, даже если срабатывает GC. Но перед тем как поток завершиться он должен вызвать метод

Структура проекта у вас должна выглядеть следующим образом:



Как мы видим из рисунка, весь нативный код находится в папке jni. После сборки проекта, в папке libs создастся четыре папки под каждую архитектуру процессора, в которой будет лежать ваша нативная библиотека (количество папок зависит от количество выбранных архитектур).

Для того, чтобы создать нативный проект, нужно создать обычный Android проект и проделать следующие шаги:

* В корне проекта нужно создать папку jni, в которую поместить исходники нативного кода;
* Создать файл Android.mk, который будет собирать проект;
* Создать файл Application.mk, в котором описываются детали сборки. Он не является обязательным условием, но позволяет гибко настроить сборку;
* Создать файл ndk-build, который будет запускать процесс сборки (тоже не является обязательным).

Как собрать проект?

Раньше это было мучением. Нужно было установить CDT плагин, скачать компилятор cygwin или mingw. Скачать Android NDK. Подключить это всё в настройках Eclipse. И как на зло это всё оказывалось не рабочим. Я первый раз когда столкнулся с Android NDK, то настраивал это всё 3 дня (а проблема оказалось в том что в cygwin нужно было дать разрешение 777 на папку проекта).

Сейчас с этим всё намного проще. Идёте по этой ссылке. Качаете Eclipse ADT Bundle в котором уже есть всё то что необходимо для сборки.