
Android Framework 개요



Android Platform 소개 및 구조

Android Platform

- 안드로이드(Android)는 휴대 전화를 비롯한 휴대용 장치를 위한 운영 체제와 미들웨어, 사용자 인터페이스 그리고 표준 응용 프로그램(웹 브라우저, 이메일 클라이언트, 단문 메시지 서비스(SMS), 멀티미디어 메시지 서비스(MMS)등)을 포함하고 있는 소프트웨어 스택이자 모바일 운영 체제이다.
- 안드로이드는 개발자들이 자바 언어로 응용 프로그램을 작성할 수 있게 하였으며, 컴파일된 바이트코드를 구동할 수 있는 런타임 라이브러리를 제공한다.
- 안드로이드 소프트웨어 개발 키트(SDK)를 통해 응용 프로그램을 개발하기 위해 필요한 각종 도구들과 API를 제공한다.

Android Platform

- 안드로이드는 리눅스 커널 위에서 동작하며, 다양한 안드로이드 시스템 구성 요소에서 사용되는 C/C++ 라이브러리들을 포함하고 있다.
- 안드로이드는 기존의 자바 가상 머신과는 다른 가상 머신인 달빅 가상 머신을 통해 자바로 작성된 응용 프로그램을 별도의 프로세스에서 실행하는 구조로 되어 있다

Android Platform 특징

1. Open Source 기반

- OSA 단체에서 Google이 주도적으로 확산시킴
- Open Source에도 "License"가 있음
- **License** (Open Source 를 받았을 때 License를 확인해야 함)

구 분	GPL	LGPL	BSD	MPL
코드의 무료이용	○	○	○	○
코드의 자유배포	○	○	○	○
소스코드의 공개	○	○	○	○
소스코드의 수정	○	○	○	○
수정코드의 소스공개	○	○	X	○
상용소프트웨어와의 링크	X	○	○	○

2. Android Kernel – Linux Kernel OS

Original Linux Kernel을 사용하는 것이 아니라 Android용으로 patch해서 사용하는 것임

patch 내용 – 알람, 디버거, Low Memory Killer(전원 관리를 위해)의 사용, 크고 무거운 기능은 제거

Android Platform 특징

3. JAVA 언어 사용

생산성이 높고, 하드웨어 추상 층을 제공하여 전문 지식이 없어도 개발 가능 (성능, 섬세 함에서는 다소 불편함)

4. 검증된 많은 라이브러리를 대거 포함하여 웬만한 기능은 별도의 외부 라이브러리를 사용할 필요가 없음

5. BUILT-IN 프로그램과 사용자가 만든 프로그램이 동일한 API를 사용하여 모든 프로그램이 평등함. 사용자가 교체할 수 있음

Android Platform 구조



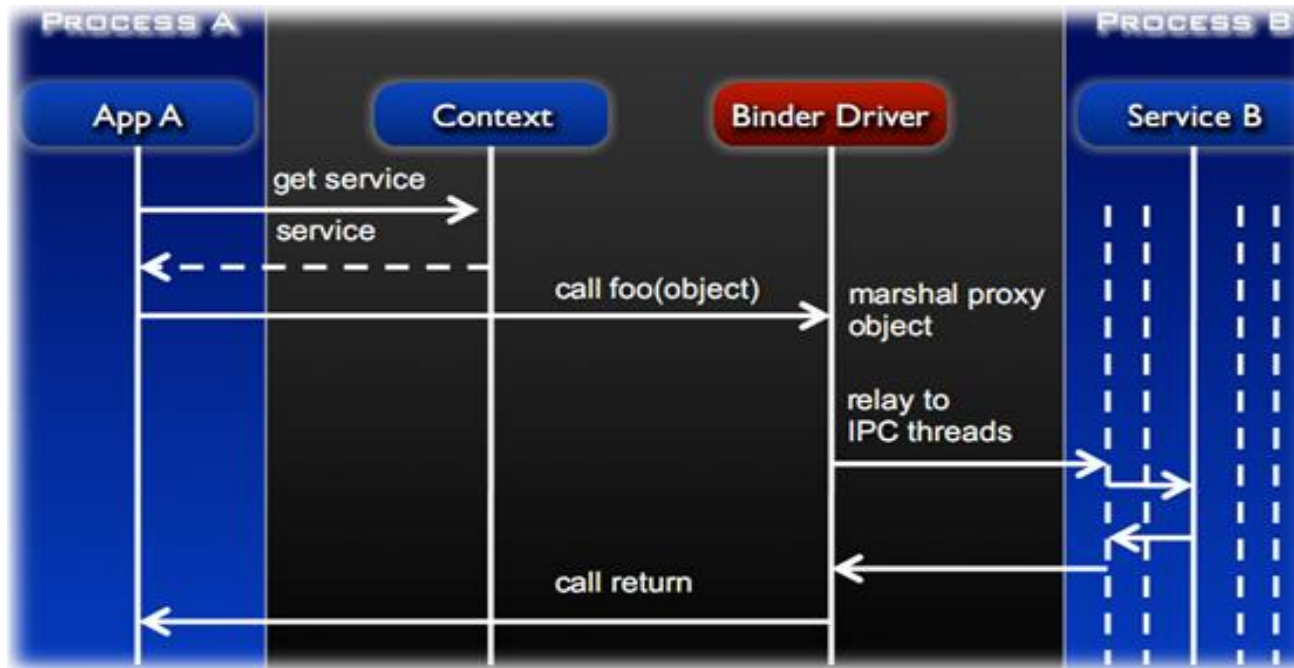
Android Platform 구조 – Kernel Layer



- 안드로이드는 리눅스 커널을 기반으로 하고 있으나, 안드로이드는 리눅스는 아니다.
- 안드로이드는 X-Window와 같은 내장 윈도우 시스템을 포함하지 않는다.
- 안드로이드는 glibc를 지원하지 않는다.
- 안드로이드는 표준 리눅스 유틸리티 전체를 포함하고 있지 않는다.
- 안드로이드는 리눅스 커널 버전 2.6.23, 2.6.24, 2.6.25, 2.6.27 을 사용해 왔다.
- 안드로이드를 지원하기 위해, 리눅스 커널 확장을 위한 패치를 포함하고 있다.
- 안드로이드에서 리눅스를 사용하는 이유는 메모리 및 프로세스 관리, 인가 (Permission) 기반의 보안 모델, 검증된 드라이버 모델, 공유 라이브러리 지원, 오픈 소스 기반 등의 장점 때문이다.
- 안드로이드를 위해 확장된 리눅스 커널 영역은, Alarm, Ashmem, Binder, Power Management, Low Memory, Killer, Kernel Debugger, Logger 이다.

Android Platform 구조 – Kernel Layer

Binder : Binder Driver는 기존의 전통적인 Kernel 구조가 객체지향 처럼 Component 중심으로 양쪽간 서로 다른 Application이나 Service간의 Message를 주고 받는 구조



- ✓장점 : 보안과 안정성 그리고 응답속도를 높일 수 있는 구조
- ✓단점 : 각각의 Service마다 Binder가 있어야 하므로 메모리 낭비가 발생

Android Platform 구조 – Kernel Layer

- Low Memory Killer

- smartphone의 주요 특징 : 적은 메모리와 배터리 사용
- Linux에서는 각 프로세스마다 우선순위를 부여하여 메모리 부족 시 우선순위별 종료
- 안드로이드는 Binder기능 때문에 프로세스마다 우선순위를 부여하지 않고 그룹마다 우선순위 부여, 메모리 부족 시 우선순위가 낮은 그룹부터 제거
- Low Memory Killer는 리소스가 부족 시 자동으로 실행되며 안드로이드는 그룹당 주어 해당되는 그룹을 한번에 해제
- 안드로이드에는 원칙적으로 프로그램 종료 기능이 없다.

- Power Management

- 전원 관리 기능은 기존 Linux의 3단계에서 2단계 추가한 5단계로 관리
- 전원이 부족하면 안드로이드에서는 키보드 -> 액정 -> CPU순으로 전원이 차단되며 제일 마지막에 D램에만 전원을 공급하는 상태가 된다

Android Platform 구조 – Library Layer

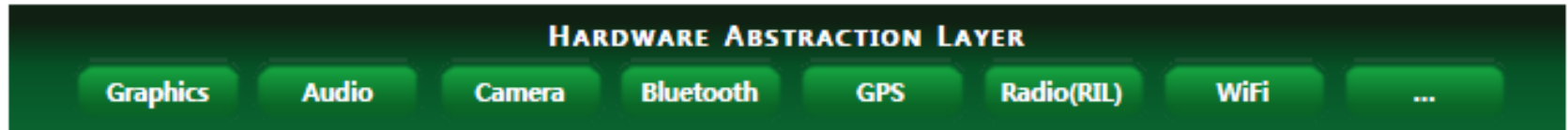


- 안드로이드 내장 라이브러리는, Bionic Libc, Function Library, Native Server, Hardware Abstraction Library로 구성된다.
- Bionic 은 임베디드에서 사용을 위해 최적화된 직접 구현된 libc 이다.
libc를 직접 구현한 이유는 다음과 같다.
 - 라이센스: user application에서 GPL 문제회피
 - size: 약200k, glibc(GNUversion of libc)의 절반크기
 - speed: 제한된CPU power에서 동작
- System C library : 임베디드 리눅스 기반 기기를 위한, 표준 C 시스템 라이브러리(libc)의 BSD 상속 구현체

Android Platform 구조 – Library Layer

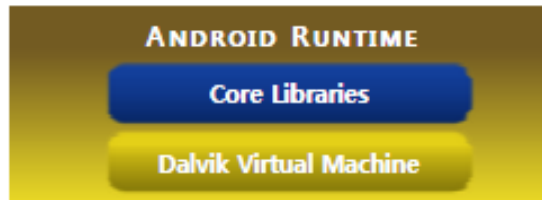
- Media Libraries : PacketVideo의 OpenCORE 기반이며, 인기있는 오디오 및 비디오 포맷, MPE4 / H.264 / MP3 / AAC / AMR / JPG / PNG를 포함하는 정적 이미지 파일의 재생 및 녹음(녹화)
- Surface Manager : 디스플레이 서브시스템 및 다수의 응용프로그램의 2D, 3D 그래픽 레이어
- LibWebCore : 안드로이드 브라우저 및 Embeddable 웹 뷰와 같은 최신의 웹 브라우저 엔진
- SGL : 2D graphics 지원
- 3D libraries : OpenGL ES 1.0 API 기반을 기반으로 하며, 하드웨어 3D 가속 또는 최적화된 3D S/W rasterized
- FreeType : 비트맵과 벡터 폰트 렌더링
- SQLite : 모든 응용프로그램에서 사용 가능한 강력하고 경량인 관계형 데이터베이스 엔진

Android Platform 구조 – Library Layer



- Hardware Abstraction Library는 User space의 C/C++ 라이브러리 계층으로써, 안드로이드에서 요구되는 하드웨어 드라이버의 구현에 대한 인터페이스를 정의한다. 더불어 하드웨어 인터페이스로부터 안드로이드 플랫폼의 로직을 분리하는데 사용된다.
- User-space의 HAL이 필요한 이유는, 모든 컴포넌트들이 표준화된 리눅스 커널 드라이버 인터페이스를 가지고 있지 않기 때문이며, 리눅스 드라이버들은 사적인 지적소유권을 공개할 수 밖에 없는 GPL 기반이란 이유 때문이다. 또한 안드로이드는 하드웨어 드라이버들을 위한 별도의 요구사항을 가지고 있기 때문이기도 하다.
- 안드로이드 동작을 위해 구현해야 하는 하드웨어 드라이버로 개발자가 구현해야 하는 API의 집합으로 보면 될 것 같습니다.

Android Platform 구조 – Android Runtime



- 안드로이드 런타임은, 안드로이드에서 사용되는 Dalvik 가상 머신과 Core 라이브러리들로 구성된다.
- Android의 개발은 Eclipse의 ADT Plug-in을 통해서 Java로 컴파일되고 class와 resource가 Dx컨버터를 통해서 Android App(.apk)로 만들어지며 이 apk가 Dalvik VM 위에서 동작하게 된다.
- 자바어플리케이션에서 호출할때 libc기반의 C/C++ library를 호출할 때 runtime library를 호출
- dex.(Dalvik Executables (DEX)) 실행파일의 구조를 가지게 된다. 실행파일이 symbolic resolution을 통하여 함수를 가지고 있다가 library를 호출한다.

Android Platform 구조 – Android Runtime

- 모든 안드로이드 응용프로그램은 각자의 프로세스상에서 실행되며, 고유의 Dalvik 가상머신의 인스턴스를 가지고 있다. Dalvik은 기기가 다수의 가상머신에서 효율적으로 실행될 수 있도록 제작되었으며, 최소의 메모리 영역에 최적화된 Dalvik Executable(.dex) 포맷 파일을 실행시킨다.
- 가상머신은 레지스터 기반이며, 자바 컴파일러로 컴파일된 클래스들을 "dk"툴을 이용하여 .dex 포맷으로 변경한 클래스들을 실행한다.
- Dalvik 가상머신은 스레딩과 저수준 메모리 관리와 같은 리눅스 커널 기능을 사용한다

Android Platform 구조 – Android Runtime

- 안드로이드 Core 라이브러리는 Java Standard Edition과 Java Mobile Edition과는 다르지만, 중복되는 부분이 상당히 있다

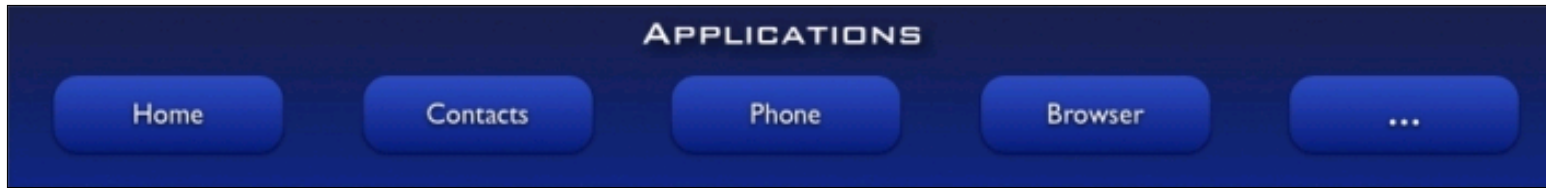
J2SE 5.0 Supported	J2SE 5.0 Not supported	3rd Party libraries	Android Specific libraries	
java.awt.font java.io java.lang java.math java.net java.nio java.security java.sql java.text java.util javax.crypto javax.microedition.khronos javax.net javax.security javax.sql javax.xml.parsers org.w3c.dom org.xml.sax	java.applet java.awt java.beans java.lang.management java.rmi javax.accessibility javax.activity javax.imageio javax.management javax.naming javax.print javax.rmi javax.security.auth.kerberos javax.security.auth.spi javax.security.sasl javax.sound javax.swing javax.transaction javax.xml org.ietf.* org.omg.* org.w3c.dom.*	org.apache.http org.json org.xml.sax org.xmlpull.v1	android android.app android.content android.content.pm android.content.res android.database android.database.sqlite android.graphics Provides android.graphics.drawable android.graphics.drawable.shapes android.hardware android.location android.media android.net android.net.http android.net.wifi android.opengl android.os	android.preference android.provider android.sax android.telephony android.telephony.gsm android.test android.test.mock android.test.suitebuilder android.text android.text.method android.text.style android.text.util android.util android.view android.view.animation android.webkit android.widget com.google.android.maps dalvik.bytecode dalvik.system

Android Platform 구조 – Application Framework Layer



- 안드로이드 애플리케이션 프레임워크는 Java 기반의 Framework 이며, 대부분이 JNI(Java Native Interface) 통해 native C/C++ 코드로 작성되어 있다.
- 응용프로그램 아키텍처는 컴포넌트 재사용을 손쉽게 할 수 있도록 디자인 됨
(단, 프레임워크의 보안 제약을 따라야 한다.)
- 이같은 메카니즘은 컴포넌트를 사용자에게 의해 교체할 수 있도록 한다.
 - Activity Manager : Application의 생명 주기를 제어, 사용자 Navigation을 위한 back-stack을 유지한다.
 - Content Provider : Application 간의 공유 되는 정보를 요약한다.
 - Resource Manager : Resource는 프로그램에서 코드를 제외한 모든 부분이다.
 - Location Manager : Android 폰은 항상 자신의 위치를 파악한다.
 - Notification Manager : 메시지 도착 등이 사용자에게 방해되지 않도록 전달한다.

Android Platform 구조 – Application Layer



- Android Architecture 최상위 계층은 Application 계층이다. 최종 사용자는 오직 Application만 보게 된다.
- Android phone을 구입하면 다음의 기본 System Application이 미리 설치되어 제공된다.
 - 전화 다이얼 장치
 - E-mail
 - 주소록
 - Web Browser
 - Android Market

Android Platform 구조 – Application Layer

- Android에는 foreground Application이 하나이며 상태줄을 제외한 전체 화면을 사용
- 사용자 Application을 실행 시키면 Android는 그 program을 시작한 후, foreground에 위치 시킨다. 이후 사용자는 다른 Application을 호출하거나 같은 Application내의 다른 화면을 호출한다. 이와 같은 모든 program과 화면은 Activity Manager에 의해 Application Stack에 기록된다. 언제나 이전 button을 누르면 stack에 저장된 이전 화면으로 이동한다.

Android Filesystem

Name	Size	Date	Time	Permissions	Info
acct		1970-02-19	23:25	drwxr-xr-x	
cache		1970-02-19	23:25	drwxrwx---	
charger	272364	1970-01-01	00:00	-rwxr-x---	
config		1970-02-19	23:25	dr-x----	
d		1970-02-19	23:25	lrwxrwxrwx	-> /sys/kernel/debug
data		1970-02-19	23:25	drwxrwx-x	
default.prop	286	1970-01-01	00:00	-rw-r--r--	
dev		1970-02-19	23:25	drwxr-xr-x	
etc		1970-02-19	23:25	lrwxrwxrwx	-> /system/etc
file_contexts	8983	1970-01-01	00:00	-rw-r--r--	
firmware		1970-01-01	00:00	dr-xr-x---	
fstab.hammerhead	2653	1970-01-01	00:00	-rw-r----	
init	179484	1970-01-01	00:00	-rwxr-x---	
init.envron.rc	919	1970-01-01	00:00	-rwxr-x---	
init.hammerhead.rc	16583	1970-01-01	00:00	-rwxr-x---	
init.hammerhead.usb.rc	5710	1970-01-01	00:00	-rwxr-x---	
init.rc	19672	1970-01-01	00:00	-rwxr-x---	
init.trace.rc	1795	1970-01-01	00:00	-rwxr-x---	
init.usb.rc	3915	1970-01-01	00:00	-rwxr-x---	
mnt		1970-02-19	23:25	drwxrwxr-x	
persist		1970-01-01	00:00	drwxr-xr-x	
proc		1970-01-01	00:00	dr-xr-xr-x	
property_contexts	2161	1970-01-01	00:00	-rw-r--r--	
res		1970-01-01	00:00	drwxr-xr-x	
root		2013-11-20	22:40	drwx----	
sbin		1970-01-01	00:00	drwxr-x---	
sdcard		1970-02-19	23:25	lrwxrwxrwx	-> /storage/emulated/legacy
seapp_contexts	656	1970-01-01	00:00	-rw-r--r--	
sepolicy	74842	1970-01-01	00:00	-rw-r--r--	
storage		1970-02-19	23:25	drwxr-x-x	
sys		1970-02-19	23:25	dr-xr-xr-x	
system		1970-01-01	00:00	drwxr-xr-x	
ueventd.hammerhead.rc	2204	1970-01-01	00:00	-rw-r--r--	
ueventd.rc	4024	1970-01-01	00:00	-rw-r--r--	
vendor		1970-02-19	23:25	lrwxrwxrwx	-> /system/vendor

- /directory에는 크게 /system, /data, /mnt directory로 구성
- /proc, /dev, /sys directory는 booting시 생성

Android Filesystem - /system

▼ system		1970-01-01	00:00	drwxr-xr-x	
▶ app		2014-01-26	19:41	drwxr-xr-x	
▶ bin		2014-01-26	19:22	drwxr-xr-x	
build.prop	3348	2014-01-26	14:38	-rw-r--r--	
▶ etc		2014-01-26	19:41	drwxr-xr-x	
▶ fonts		2014-01-26	17:55	drwxr-xr-x	
▶ framework		2014-01-26	19:41	drwxr-xr-x	
▶ lib		2014-01-26	19:41	drwxr-xr-x	
▶ lost+found		1970-01-01	00:00	drwx---	
▶ media		2014-01-26	16:51	drwxr-xr-x	
▶ priv-app		2014-01-26	19:40	drwxr-xr-x	
▶ tts		2014-01-26	16:52	drwxr-xr-x	
▶ usr		2014-01-26	16:52	drwxr-xr-x	
▶ vendor		2014-01-26	16:52	drwxr-xr-x	
▶ xbin		2014-01-26	19:18	drwxr-xr-x	

- /system directory는 root 권한을 부여 받아야 접근 가능
- /system directory는 Android Platform이 포함되어 있는 directory
- /system/app : key application 저장
- /system/framework : application framework layer와 library layer의 system service 내용이 저장
- /system/lib : library layer의 library가 저장

Android Filesystem - /data

```
▼ data
  ► app
  ► app-asec
  ► app-lib
  ► app-private
  ► backup
  ► bugreports
  ► cam_socket1
  ► cam_socket2
  ► dalvik-cache
  ► data
  ► dontpanic
  ► drm
  ► local
  ► lost+found
  ► media
  ► mediadr
  ► misc
  ► nfc
  ► property
  ► resource-cache
  ► security
  ► ssh
  ► system
  ► user
```

1970-02-19	23:25	drwxrwx-x
1970-02-19	23:25	drwxrwx-x
1970-02-19	23:25	drwx---
1970-02-19	23:25	drwxrwx-x
1970-02-19	23:25	drwxrwx-x
1970-02-19	23:25	drwx---
1970-02-19	23:25	lrwxrwxrwx
1970-02-19	23:25	srwxrwx---
1970-02-19	23:25	srwxrwx---
1970-02-19	23:26	drwxrwx-x
1970-02-19	23:25	drwxrwx-x
1970-02-19	23:25	drwxr-x---
1970-02-19	23:25	drwxrwx---
1970-02-19	23:25	drwxr-x-x
1970-01-01	00:00	drwxrwx---
1970-02-19	23:25	drwxrwx---
1970-02-19	23:25	drwxrwx---
1970-02-19	23:25	drwxrwx-t
1970-02-19	23:25	drwxrwx---
1970-02-19	23:25	drwx---
1970-02-19	23:25	drwxrwx-x
1970-02-19	23:25	drwx-x-x
1970-02-19	23:25	drwxr-x---
1970-02-19	23:55	drwxrwxr-x
1970-02-19	23:25	drwx-x-x

-> /data/data/com.android.shell/files/bugreports

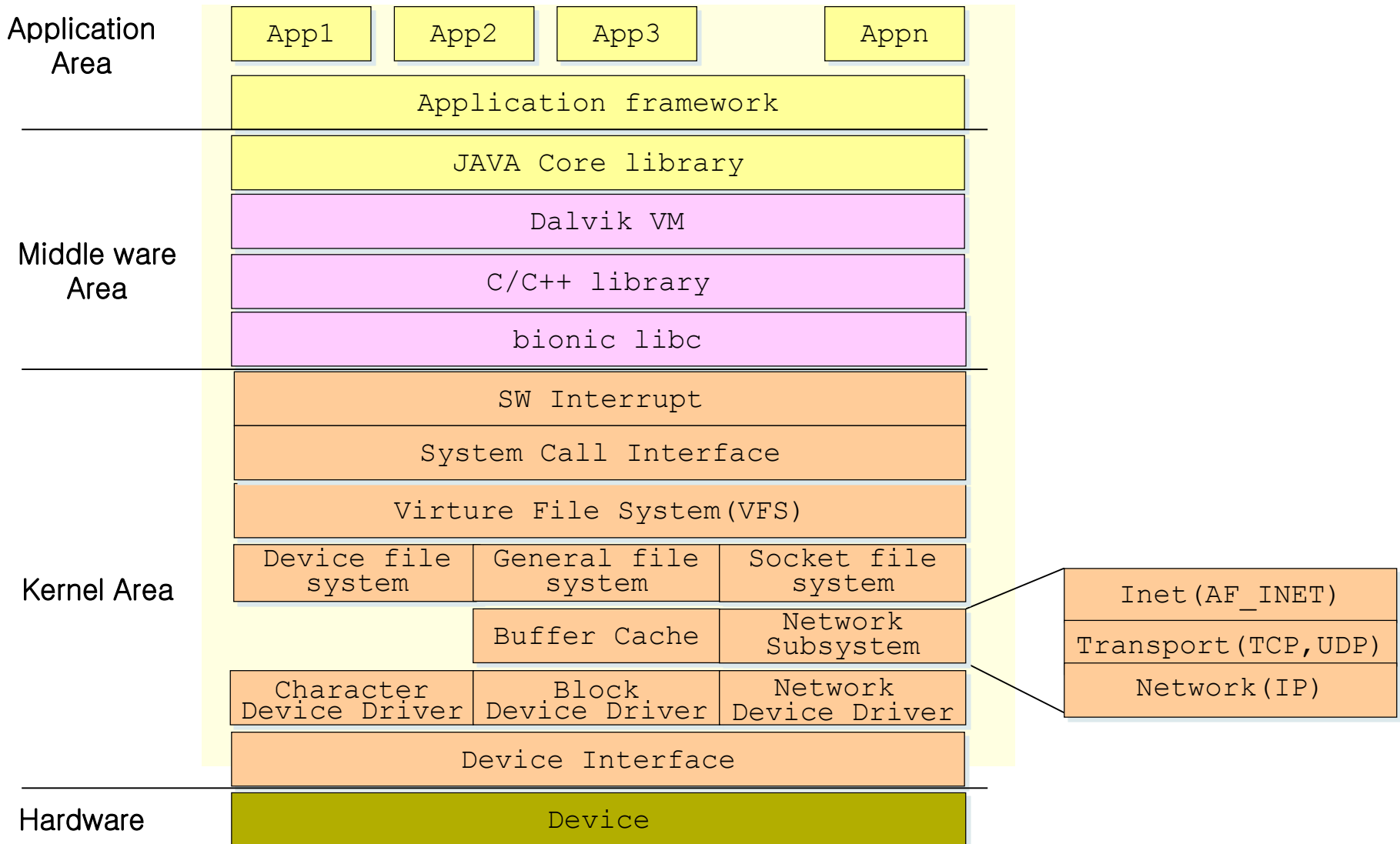
- /data directory는 일반 user가 사용
- /data directory는 Android Platform이 포함되어 있는 directory
- /data/app : 사용자 설치 application 저장
- /data/app-lib : application 별 native library 저장
- /data/data : application directory

Android Filesystem - /mnt

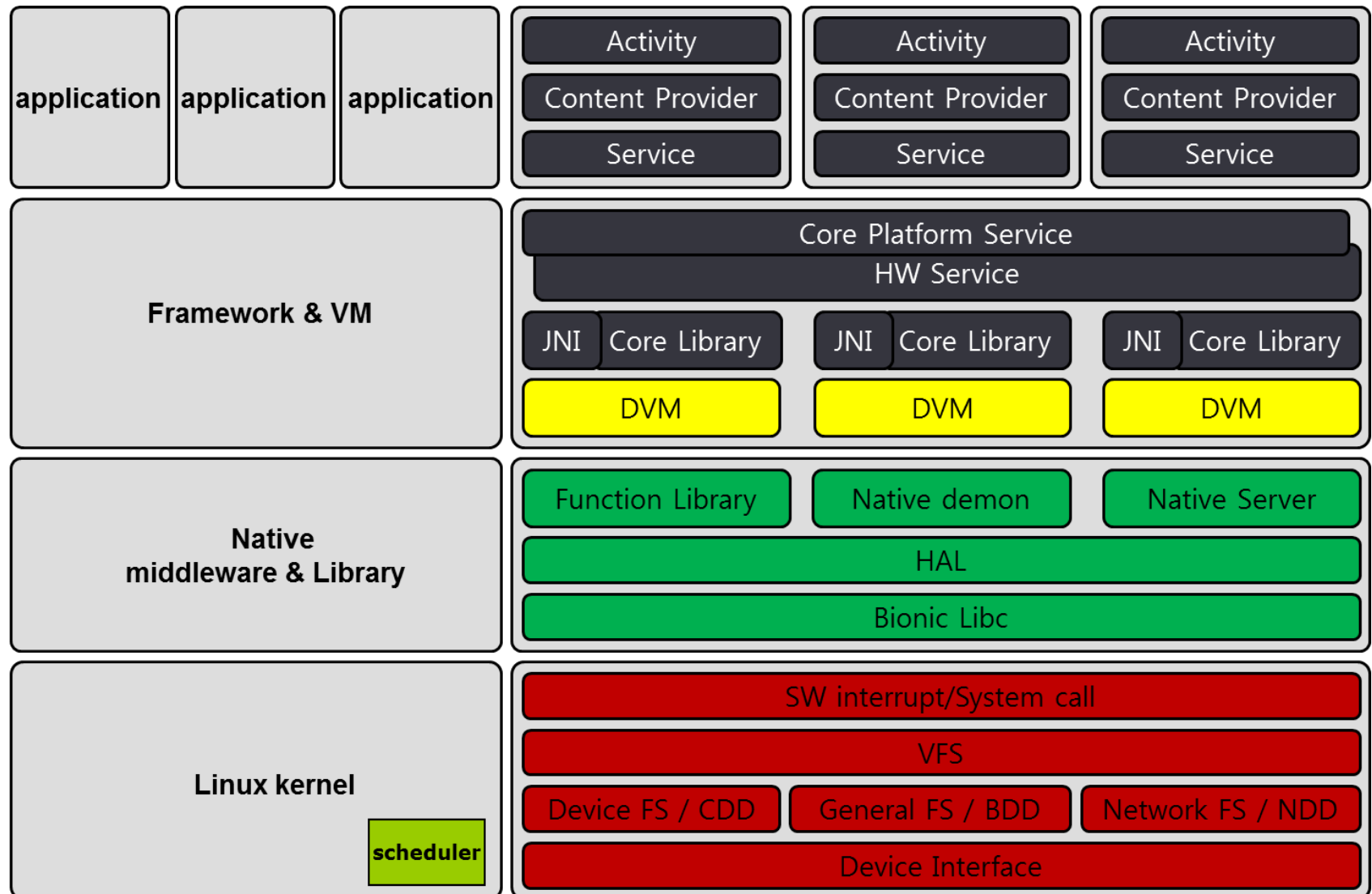
▼ 📁 mnt	1970-02-19	23:25	drwxrwxr-x	
▶ 📁 asec	1970-02-19	23:25	drwxr-xr-x	
▶ 📁 media_rw	1970-02-19	23:25	drwx---	
▶ 📁 obb	1970-02-19	23:25	drwxr-xr-x	
📁 sdcard	1970-02-19	23:25	lrwxrwxrwx	-> /storage/emulated/legacy
▶ 📁 secure	1970-02-19	23:25	drwx---	
▶ 📁 shell	1970-02-19	23:25	drwx---	
...

- /mnt directory는 외부 장치가 mount된 directory

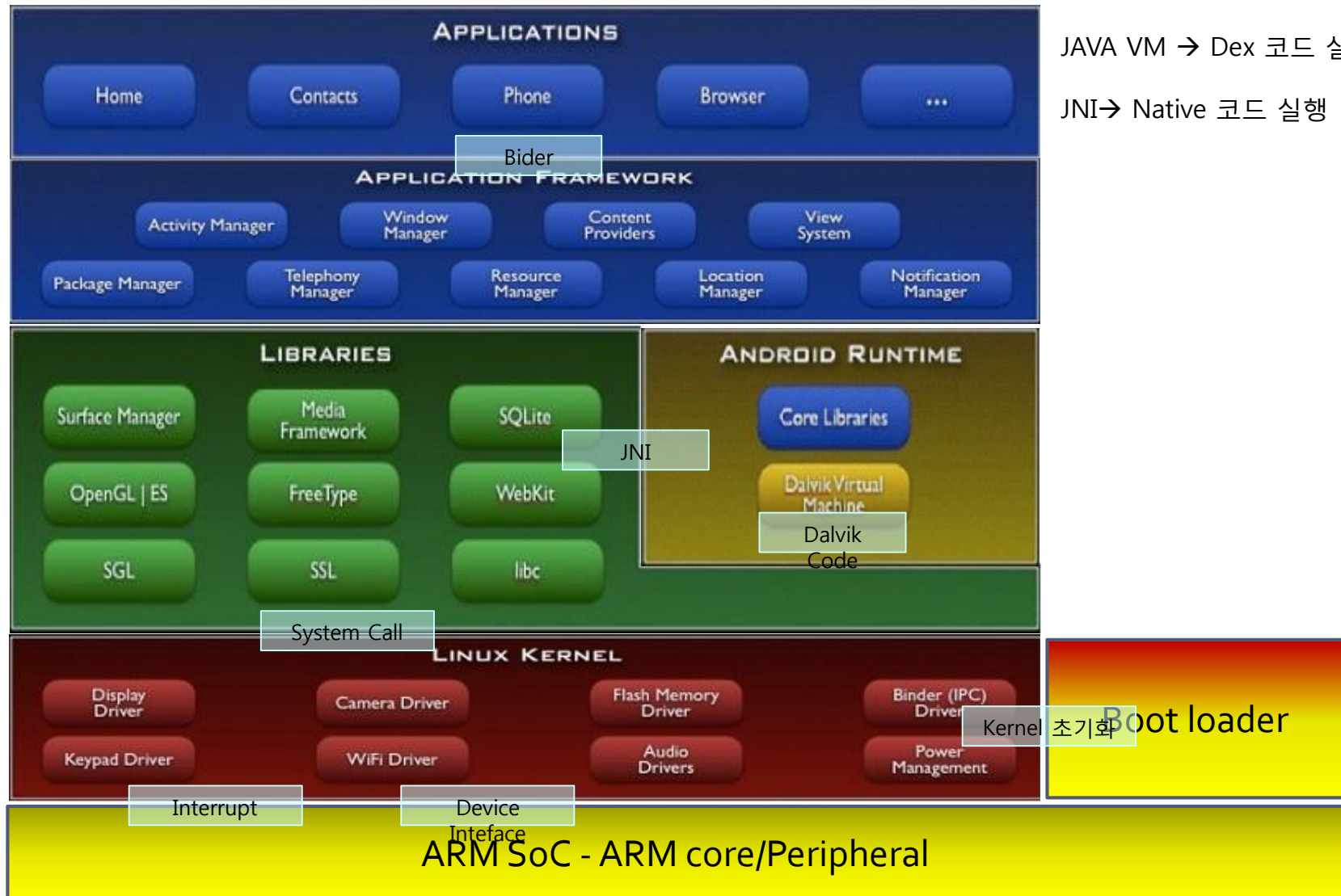
Android System 구조



Android Platform Model : multi-process



Android for ARM

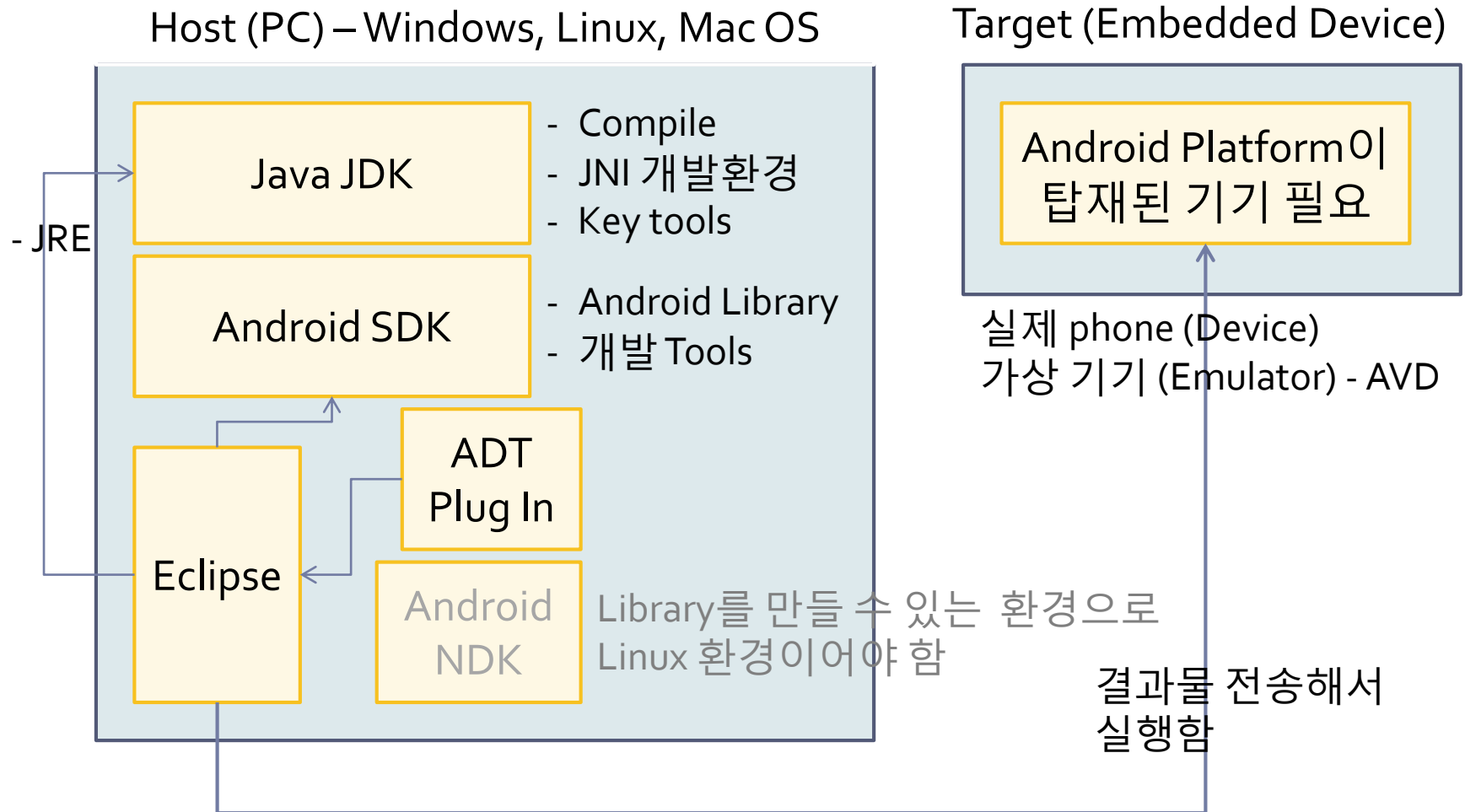


Android platform 개발 환경

Program 개발 유형별 개발 환경

개발 유형	개발 환경
Android Application	Java 기반, Android Library 사용 (Java, Android 환경) Software Development Kit
Android Application용 C 기반 Library	Linux 기반 Embedded Device 개발환경 Native Development Kit
Android Platform Porting Android Platform System Service Linux System Program Device Driver	Linux 기반 Android Platform Build 환경 Embedded Device 개발환경 Platform Development Kit

Software Development Kit



Software Development Kit

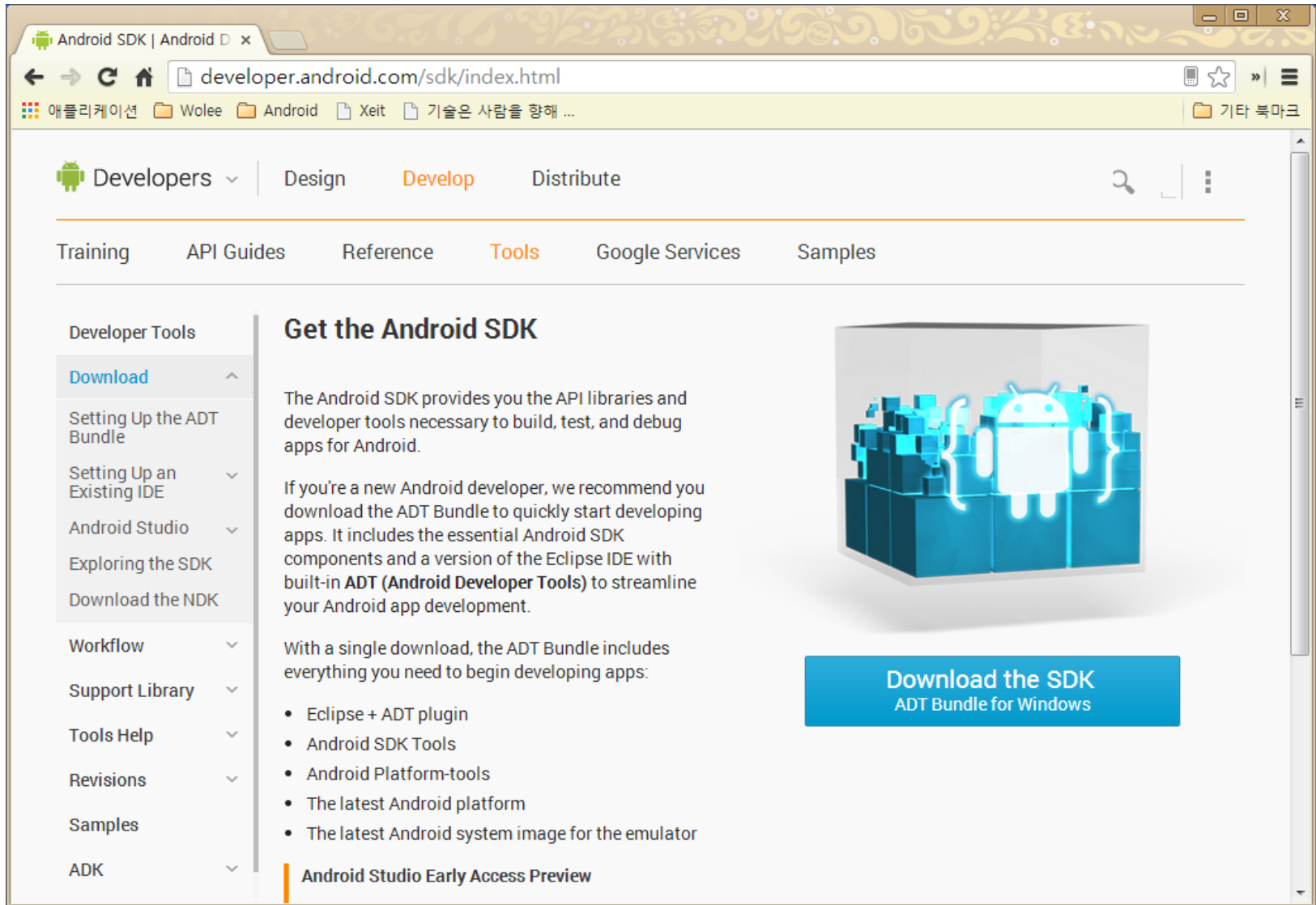
1. 일반적인 SDK 환경 구축방법

1. JDK 설치
2. Android SDK
3. Eclipse 설치
4. ADT Plug In 설치 (Eclipse의 Preference에서)

2. ADT Bundle 설치

3. Android Studio 설치

SDK Download 사이트 : <http://developer.android.com/sdk/index.html>



The screenshot shows a web browser window with the URL developer.android.com/sdk/index.html. The page features a navigation bar with 'Developers', 'Design', 'Develop', and 'Distribute' tabs. Below this is a secondary navigation bar with 'Training', 'API Guides', 'Reference', 'Tools' (highlighted), 'Google Services', and 'Samples'. A left sidebar lists various developer tools and resources, with 'Download' selected under 'Developer Tools'. The main content area is titled 'Get the Android SDK' and provides information about the ADT Bundle, including a list of components like the Eclipse + ADT plugin, Android SDK Tools, and the latest Android platform. A large blue button labeled 'Download the SDK' is prominently displayed on the right side of the main content area.

Android SDK | Android D x

developer.android.com/sdk/index.html

애플리케이션 Wolee Android Xeit 기술은 사람을 향해 ... 기타 북마크

Developers Design Develop Distribute

Training API Guides Reference Tools Google Services Samples

Developer Tools

- Download
- Setting Up the ADT Bundle
- Setting Up an Existing IDE
- Android Studio
- Exploring the SDK
- Download the NDK
- Workflow
- Support Library
- Tools Help
- Revisions
- Samples
- ADK

Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

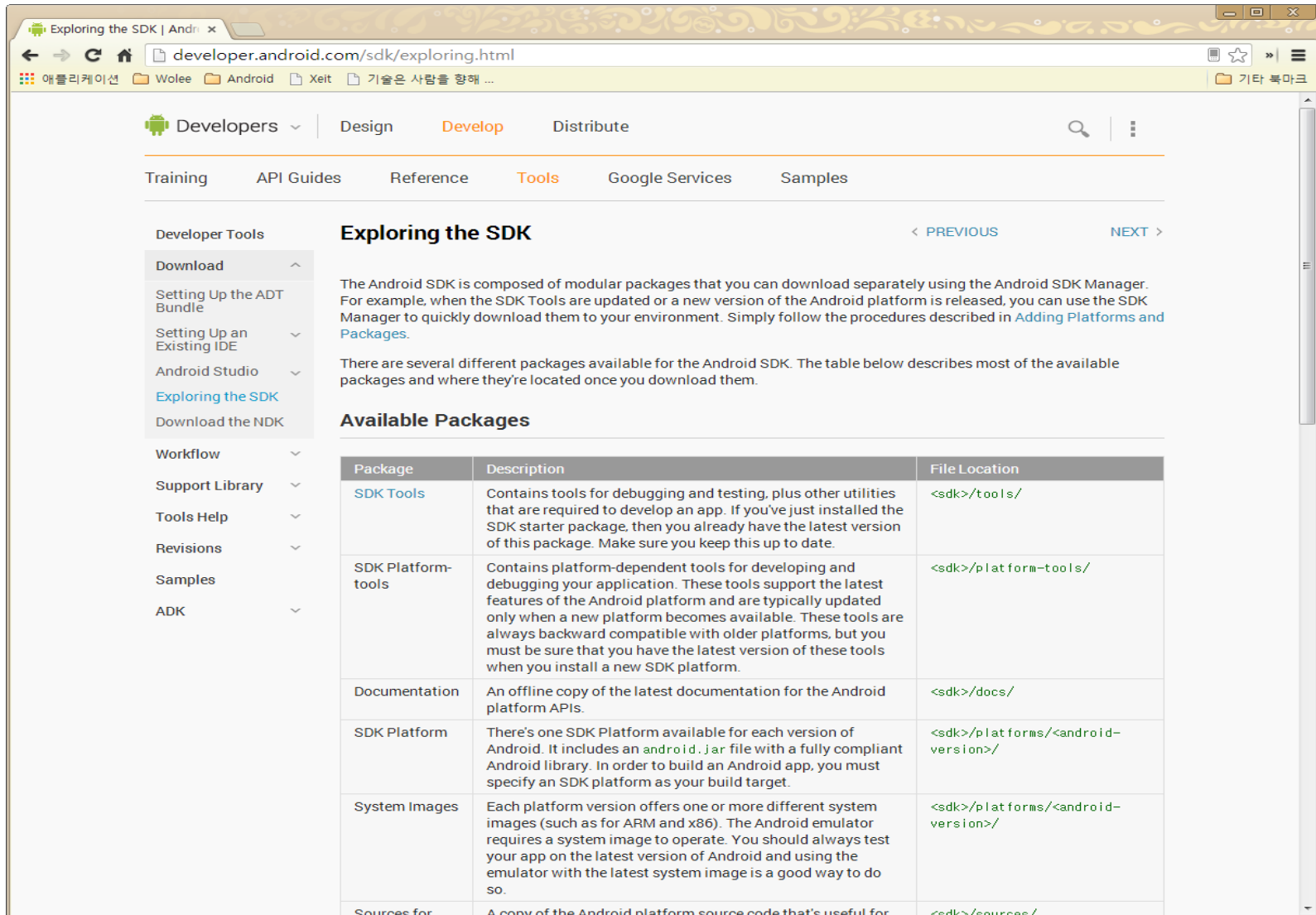
If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

Download the SDK
ADT Bundle for Windows

SDK Download 사이트 : <http://developer.android.com/sdk/index.html>



The screenshot shows a web browser window displaying the 'Exploring the SDK' page on the Android developer website. The browser's address bar shows the URL 'developer.android.com/sdk/exploring.html'. The page has a navigation bar with 'Developers', 'Design', 'Develop', and 'Distribute' tabs. Below this is a secondary navigation bar with 'Training', 'API Guides', 'Reference', 'Tools' (highlighted), 'Google Services', and 'Samples'. The main content area is titled 'Exploring the SDK' and includes a left sidebar with a 'Download' section containing links like 'Setting Up the ADT Bundle', 'Setting Up an Existing IDE', 'Android Studio', 'Exploring the SDK' (active), and 'Download the NDK'. The main text explains that the Android SDK is composed of modular packages and provides a table of available packages. The table has three columns: 'Package', 'Description', and 'File Location'. The packages listed are SDK Tools, SDK Platform-tools, Documentation, SDK Platform, System Images, and Sources for.

Exploring the SDK

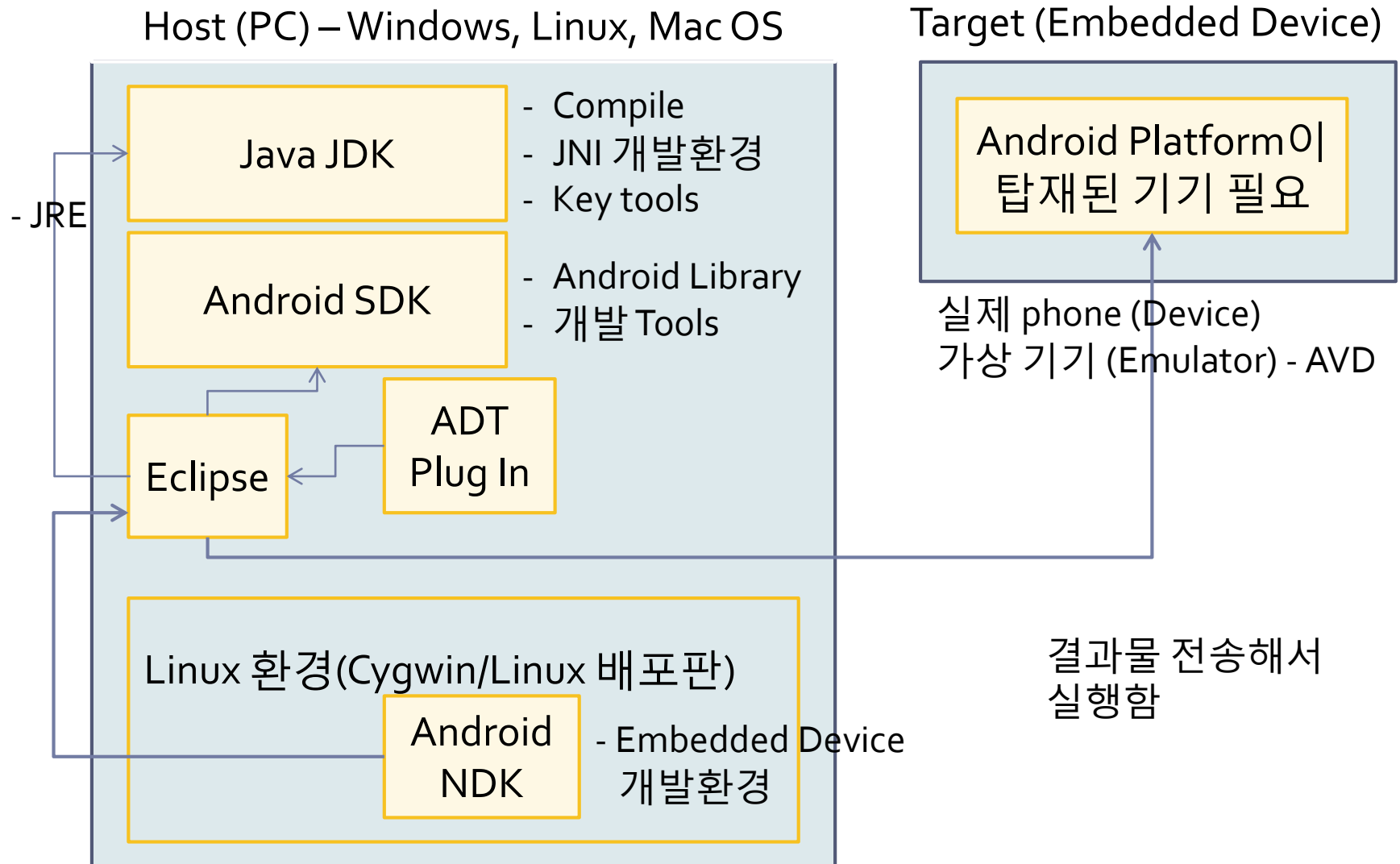
The Android SDK is composed of modular packages that you can download separately using the Android SDK Manager. For example, when the SDK Tools are updated or a new version of the Android platform is released, you can use the SDK Manager to quickly download them to your environment. Simply follow the procedures described in [Adding Platforms and Packages](#).

There are several different packages available for the Android SDK. The table below describes most of the available packages and where they're located once you download them.

Available Packages

Package	Description	File Location
SDK Tools	Contains tools for debugging and testing, plus other utilities that are required to develop an app. If you've just installed the SDK starter package, then you already have the latest version of this package. Make sure you keep this up to date.	<sdk>/tools/
SDK Platform-tools	Contains platform-dependent tools for developing and debugging your application. These tools support the latest features of the Android platform and are typically updated only when a new platform becomes available. These tools are always backward compatible with older platforms, but you must be sure that you have the latest version of these tools when you install a new SDK platform.	<sdk>/platform-tools/
Documentation	An offline copy of the latest documentation for the Android platform APIs.	<sdk>/docs/
SDK Platform	There's one SDK Platform available for each version of Android. It includes an <code>android.jar</code> file with a fully compliant Android library. In order to build an Android app, you must specify an SDK platform as your build target.	<sdk>/platforms/<android-version>/
System Images	Each platform version offers one or more different system images (such as for ARM and x86). The Android emulator requires a system image to operate. You should always test your app on the latest version of Android and using the emulator with the latest system image is a good way to do so.	<sdk>/platforms/<android-version>/
Sources for	A copy of the Android platform source code that's useful for	<sdk>/sources/

Native Development Kit



Native Development Kit

1. 일반적인 NDK 환경 구축방법

1. Cygwin(<http://cygwin.com>) 설치

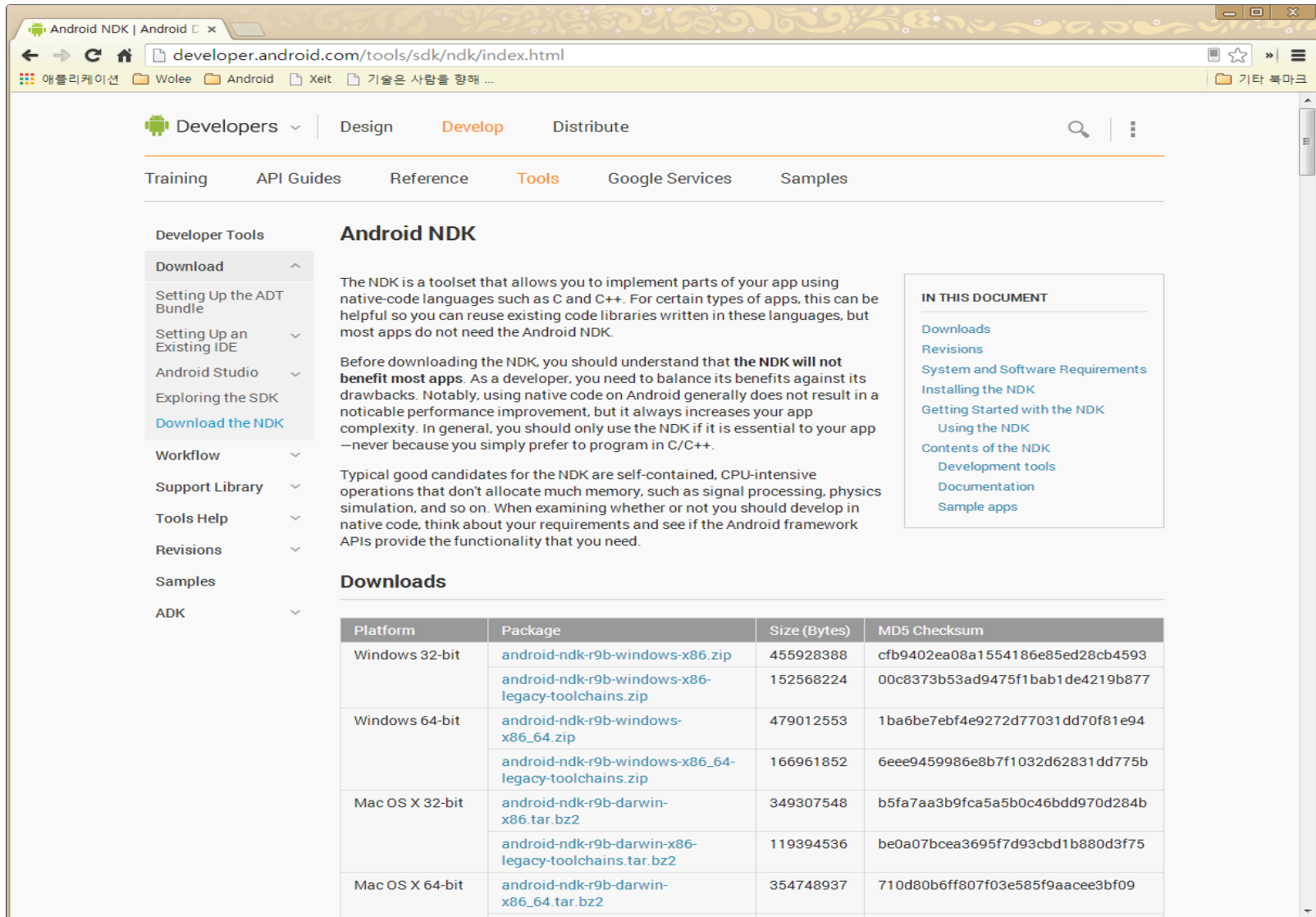
Windows에서 Linux 환경을 제공하는 프로그램

2. Android NDK 설치

Android System에서 Java가 아닌 C code를 cross compile하기 위한 toolchain

2. Linux 환경에서 Android NDK 설치

NDK Download 사이트 : <http://d.android.com/tools/sdk/ndk/index.html>



Android NDK

The NDK is a toolset that allows you to implement parts of your app using native-code languages such as C and C++. For certain types of apps, this can be helpful so you can reuse existing code libraries written in these languages, but most apps do not need the Android NDK.

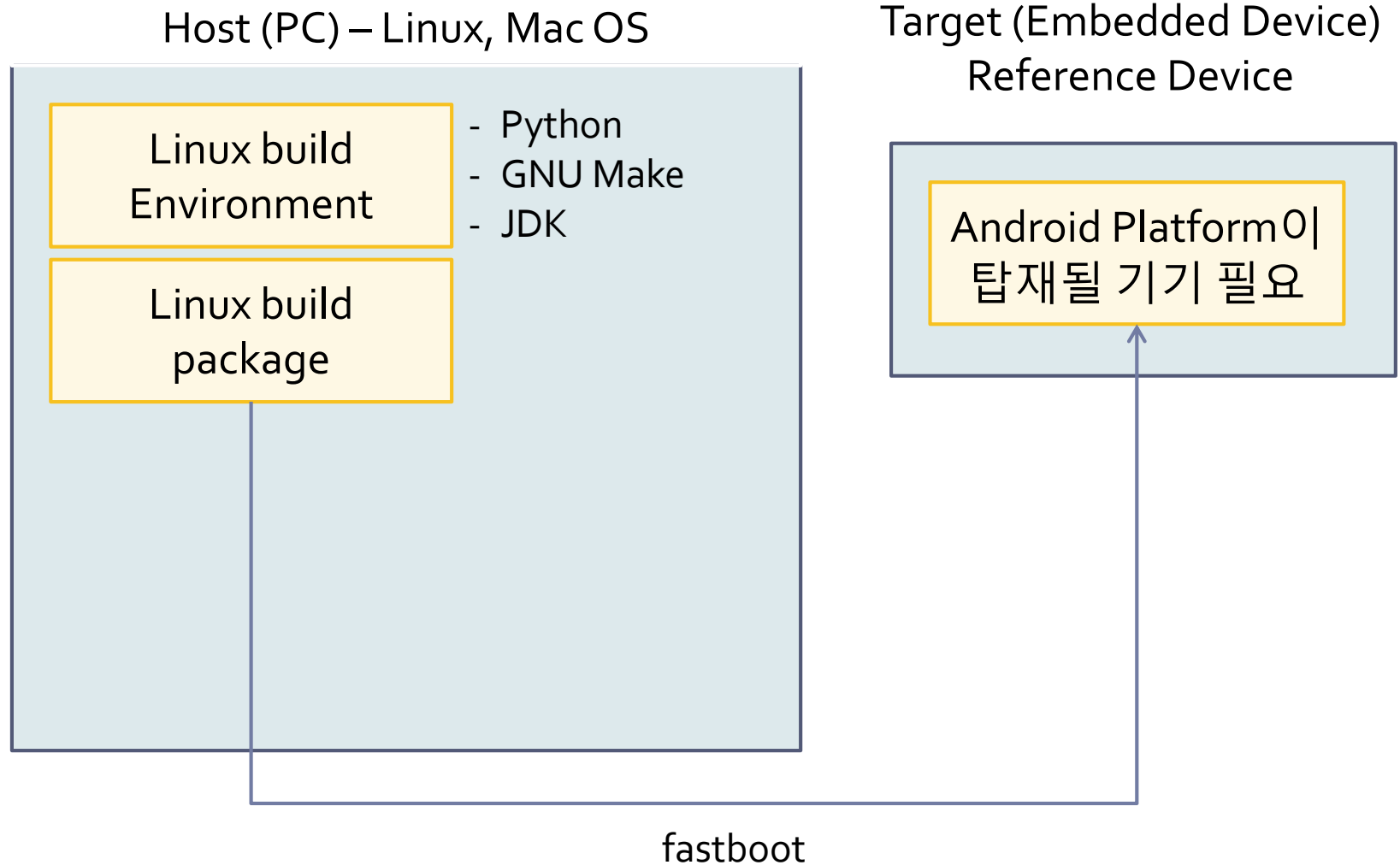
Before downloading the NDK, you should understand that **the NDK will not benefit most apps**. As a developer, you need to balance its benefits against its drawbacks. Notably, using native code on Android generally does not result in a noticeable performance improvement, but it always increases your app complexity. In general, you should only use the NDK if it is essential to your app—never because you simply prefer to program in C/C++.

Typical good candidates for the NDK are self-contained, CPU-intensive operations that don't allocate much memory, such as signal processing, physics simulation, and so on. When examining whether or not you should develop in native code, think about your requirements and see if the Android framework APIs provide the functionality that you need.

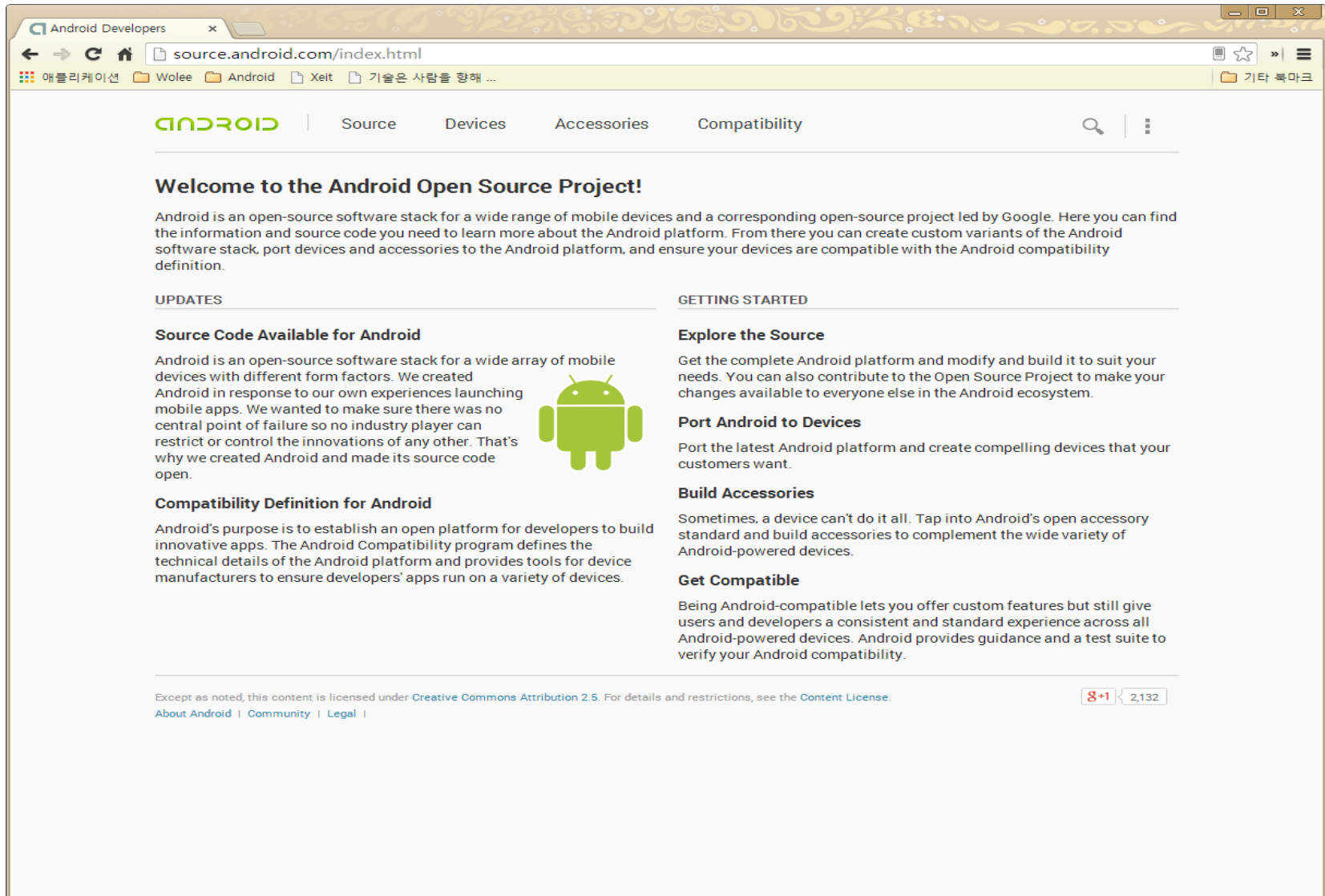
Downloads

Platform	Package	Size (Bytes)	MD5 Checksum
Windows 32-bit	android-ndk-r9b-windows-x86.zip	455928388	cfb9402ea08a1554186e85ed28cb4593
	android-ndk-r9b-windows-x86-legacy-toolchains.zip	152568224	00c8373b53ad9475f1bab1de4219b877
Windows 64-bit	android-ndk-r9b-windows-x86_64.zip	479012553	1ba6be7ebf4e9272d77031dd70f81e94
	android-ndk-r9b-windows-x86_64-legacy-toolchains.zip	166961852	6eee9459986e8b7f1032d62831dd775b
Mac OS X 32-bit	android-ndk-r9b-darwin-x86.tar.bz2	349307548	b5fa7aa3b9fca5a5b0c46bdd970d284b
	android-ndk-r9b-darwin-x86-legacy-toolchains.tar.bz2	119394536	be0a07bcea3695f7d93cbd1b880d3f75
Mac OS X 64-bit	android-ndk-r9b-darwin-x86_64.tar.bz2	354748937	710d80b6ff807f03e585f9aacee3bf09

Platform Development Kit



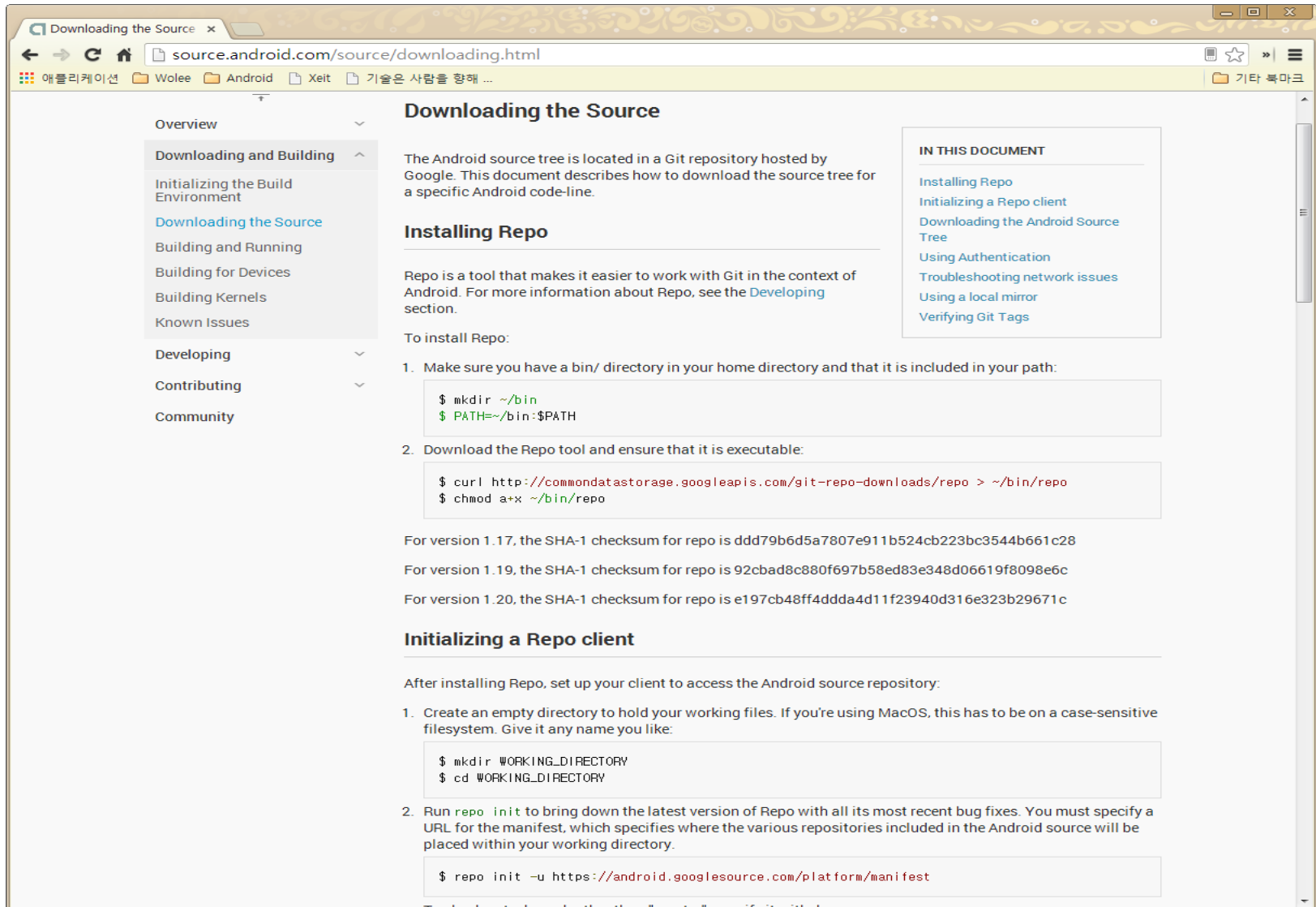
PDK Download 사이트 : <http://source.android.com>



PDK Download 사이트 : <http://source.android.com>

The screenshot shows a web browser window with the address bar displaying source.android.com/source/initializing.html. The page title is "Initializing a Build Environment". The left sidebar contains a navigation menu with the following items: Overview, Downloading and Building (expanded), Initializing the Build Environment (selected), Downloading the Source, Building and Running, Building for Devices, Building Kernels, Known Issues, Developing, Contributing, and Community. The main content area has the heading "Initializing a Build Environment" and the text: "This section describes how to set up your local work environment to build the Android source files. You will need to use Linux or Mac OS. Building under Windows is not currently supported." Below this is a note: "Note: The source download is approximately 8.5GB in size. You will need over 30GB free to complete a single build, and up to 100GB (or more) for a full set of builds." Further down is the section "Choosing a Branch" with text about requirements and a link to "Build Numbers". Below that is "Setting up a Linux build environment" with instructions for Ubuntu LTS and Gingerbread. A "Detailed instructions for Ubuntu and MacOS follow" section lists requirements, including "Python 2.6 – 2.7, which you can download from python.org". On the right side, there is a box titled "IN THIS DOCUMENT" containing a list of links: Installing the JDK, Installing required packages (Ubuntu 12.04), Installing required packages (Ubuntu 10.04 -- 11.10), Configuring USB Access, Setting up ccache, Using a separate output directory, Creating a case-sensitive disk image, Master branch, Branch 4.2.x and earlier branches, Branch 4.0.x and all earlier branches, Installing required packages, Reverting from make 3.82, and Setting a file descriptor limit.

PDK Download 사이트 : <http://source.android.com>



Downloading the Source

The Android source tree is located in a Git repository hosted by Google. This document describes how to download the source tree for a specific Android code-line.

Installing Repo

Repo is a tool that makes it easier to work with Git in the context of Android. For more information about Repo, see the [Developing](#) section.

To install Repo:

1. Make sure you have a `bin/` directory in your home directory and that it is included in your path:

```
$ mkdir ~/bin
$ PATH=~/.bin:$PATH
```
2. Download the Repo tool and ensure that it is executable:

```
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

For version 1.17, the SHA-1 checksum for repo is ddd79b6d5a7807e911b524cb223bc3544b661c28
For version 1.19, the SHA-1 checksum for repo is 92cbad8c880f697b58ed83e348d06619f8098e6c
For version 1.20, the SHA-1 checksum for repo is e197cb48ff4ddda4d11f23940d316e323b29671c

Initializing a Repo client

After installing Repo, set up your client to access the Android source repository:

1. Create an empty directory to hold your working files. If you're using MacOS, this has to be on a case-sensitive filesystem. Give it any name you like:

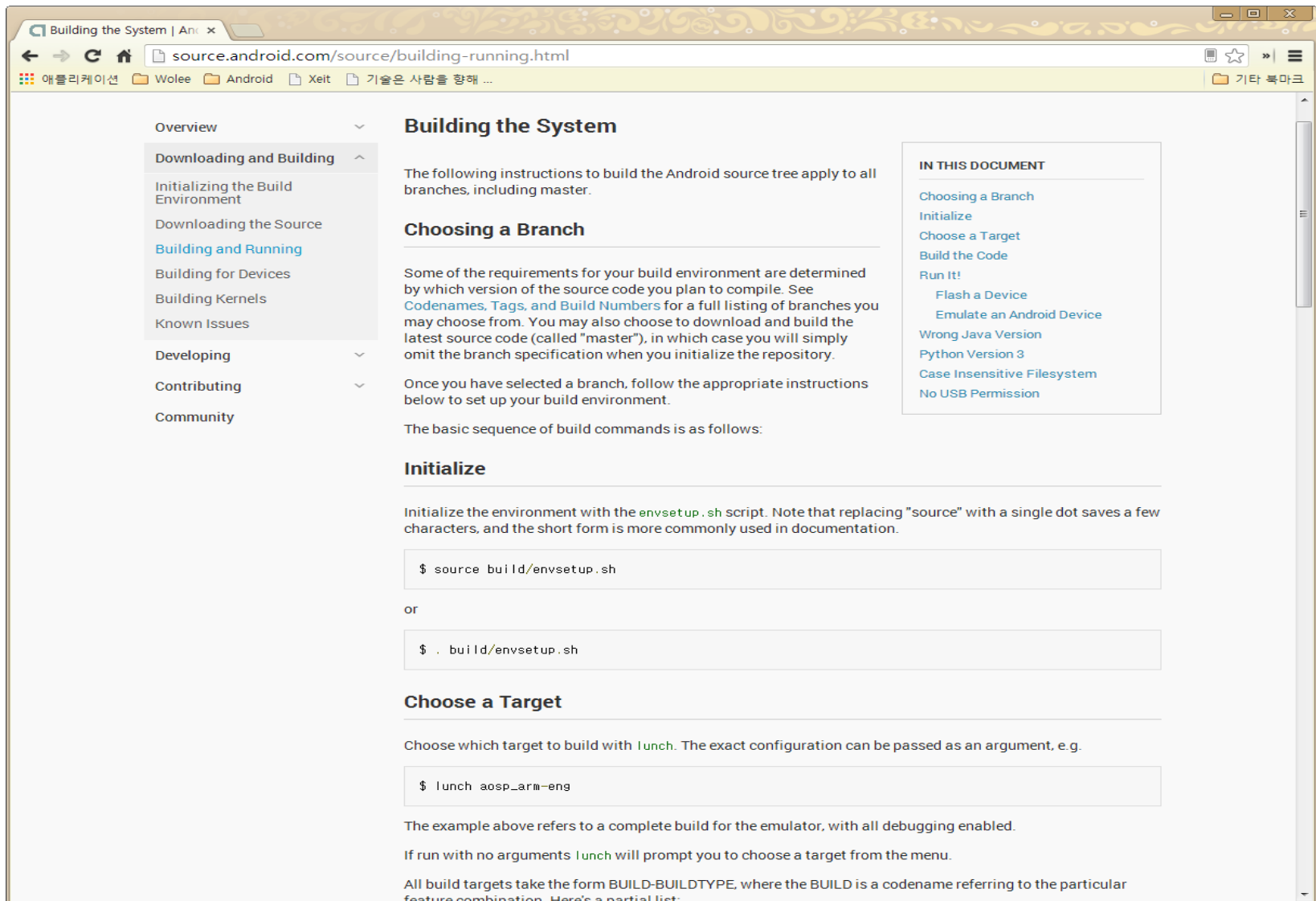
```
$ mkdir WORKING_DIRECTORY
$ cd WORKING_DIRECTORY
```
2. Run `repo init` to bring down the latest version of Repo with all its most recent bug fixes. You must specify a URL for the manifest, which specifies where the various repositories included in the Android source will be placed within your working directory.

```
$ repo init -u https://android.googlesource.com/platform/manifest
```

IN THIS DOCUMENT

- [Installing Repo](#)
- [Initializing a Repo client](#)
- [Downloading the Android Source Tree](#)
- [Using Authentication](#)
- [Troubleshooting network issues](#)
- [Using a local mirror](#)
- [Verifying Git Tags](#)

PDK Download 사이트 : <http://source.android.com>



The screenshot shows a web browser window with the address bar displaying `source.android.com/source/building-running.html`. The page title is "Building the System | Android". The left sidebar contains a navigation menu with the following items: Overview, Downloading and Building (selected), Initializing the Build Environment, Downloading the Source, Building and Running, Building for Devices, Building Kernels, Known Issues, Developing, Contributing, and Community. The main content area is titled "Building the System" and contains the following sections:

- Building the System**
The following instructions to build the Android source tree apply to all branches, including master.
- Choosing a Branch**
Some of the requirements for your build environment are determined by which version of the source code you plan to compile. See [Codenames, Tags, and Build Numbers](#) for a full listing of branches you may choose from. You may also choose to download and build the latest source code (called "master"), in which case you will simply omit the branch specification when you initialize the repository.
Once you have selected a branch, follow the appropriate instructions below to set up your build environment.
The basic sequence of build commands is as follows:
- Initialize**
Initialize the environment with the `envsetup.sh` script. Note that replacing "source" with a single dot saves a few characters, and the short form is more commonly used in documentation.

```
$ source build/envsetup.sh
```

or

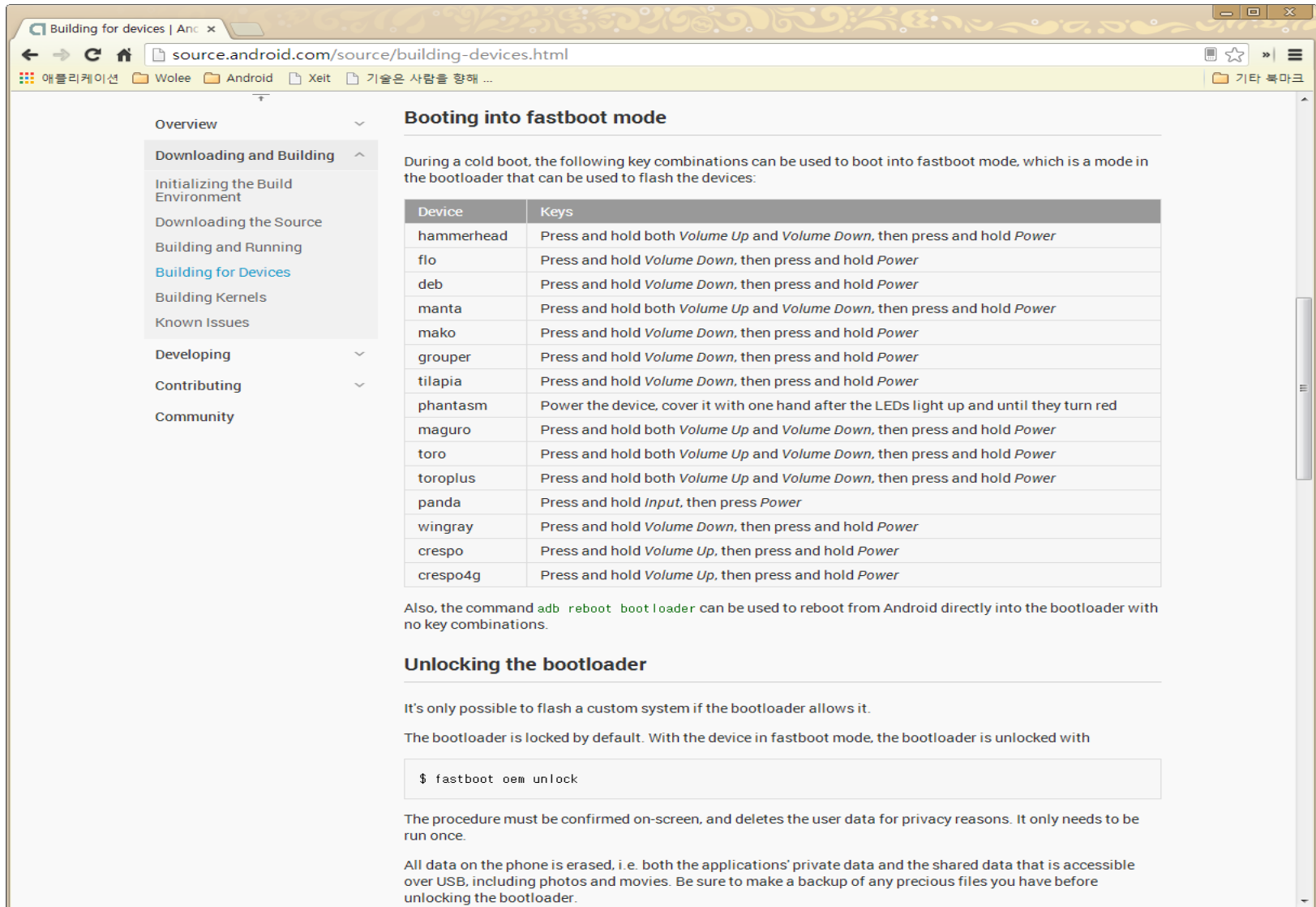
```
$ . build/envsetup.sh
```
- Choose a Target**
Choose which target to build with `lunch`. The exact configuration can be passed as an argument, e.g.

```
$ lunch aosp_arm-eng
```

The example above refers to a complete build for the emulator, with all debugging enabled.
If run with no arguments `lunch` will prompt you to choose a target from the menu.
All build targets take the form BUILD-BUILDTYPE, where the BUILD is a codename referring to the particular feature combination. Here's a partial list:

On the right side of the page, there is a box titled "IN THIS DOCUMENT" containing a list of links: Choosing a Branch, Initialize, Choose a Target, Build the Code, Run It!, Flash a Device, Emulate an Android Device, Wrong Java Version, Python Version 3, Case Insensitive Filesystem, and No USB Permission.

PDK Download 사이트 : <http://source.android.com>



Building for devices | Android

source.android.com/source/building-devices.html

애플리케이션 | Wollee | Android | Xelit | 기술은 사람을 향해 ...

기타 북마크

Booting into fastboot mode

During a cold boot, the following key combinations can be used to boot into fastboot mode, which is a mode in the bootloader that can be used to flash the devices:

Device	Keys
hammerhead	Press and hold both <i>Volume Up</i> and <i>Volume Down</i> , then press and hold <i>Power</i>
flo	Press and hold <i>Volume Down</i> , then press and hold <i>Power</i>
deb	Press and hold <i>Volume Down</i> , then press and hold <i>Power</i>
manta	Press and hold both <i>Volume Up</i> and <i>Volume Down</i> , then press and hold <i>Power</i>
mako	Press and hold <i>Volume Down</i> , then press and hold <i>Power</i>
grouper	Press and hold <i>Volume Down</i> , then press and hold <i>Power</i>
tilapia	Press and hold <i>Volume Down</i> , then press and hold <i>Power</i>
phantasm	Power the device, cover it with one hand after the LEDs light up and until they turn red
maguro	Press and hold both <i>Volume Up</i> and <i>Volume Down</i> , then press and hold <i>Power</i>
toro	Press and hold both <i>Volume Up</i> and <i>Volume Down</i> , then press and hold <i>Power</i>
toroplus	Press and hold both <i>Volume Up</i> and <i>Volume Down</i> , then press and hold <i>Power</i>
panda	Press and hold <i>Input</i> , then press <i>Power</i>
wingray	Press and hold <i>Volume Down</i> , then press and hold <i>Power</i>
crespo	Press and hold <i>Volume Up</i> , then press and hold <i>Power</i>
crespo4g	Press and hold <i>Volume Up</i> , then press and hold <i>Power</i>

Also, the command `adb reboot bootloader` can be used to reboot from Android directly into the bootloader with no key combinations.

Unlocking the bootloader

It's only possible to flash a custom system if the bootloader allows it.

The bootloader is locked by default. With the device in fastboot mode, the bootloader is unlocked with

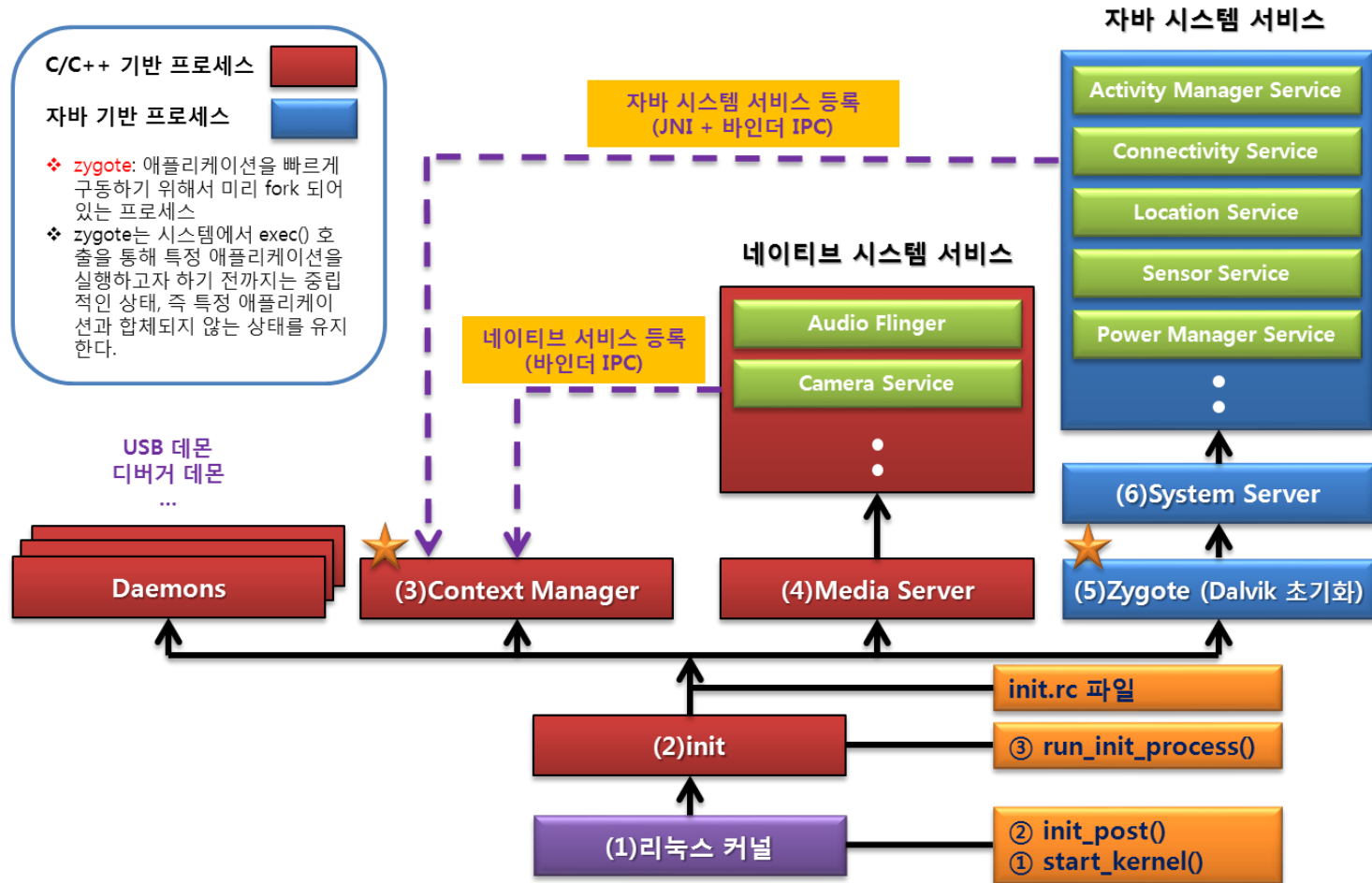
```
$ fastboot oem unlock
```

The procedure must be confirmed on-screen, and deletes the user data for privacy reasons. It only needs to be run once.

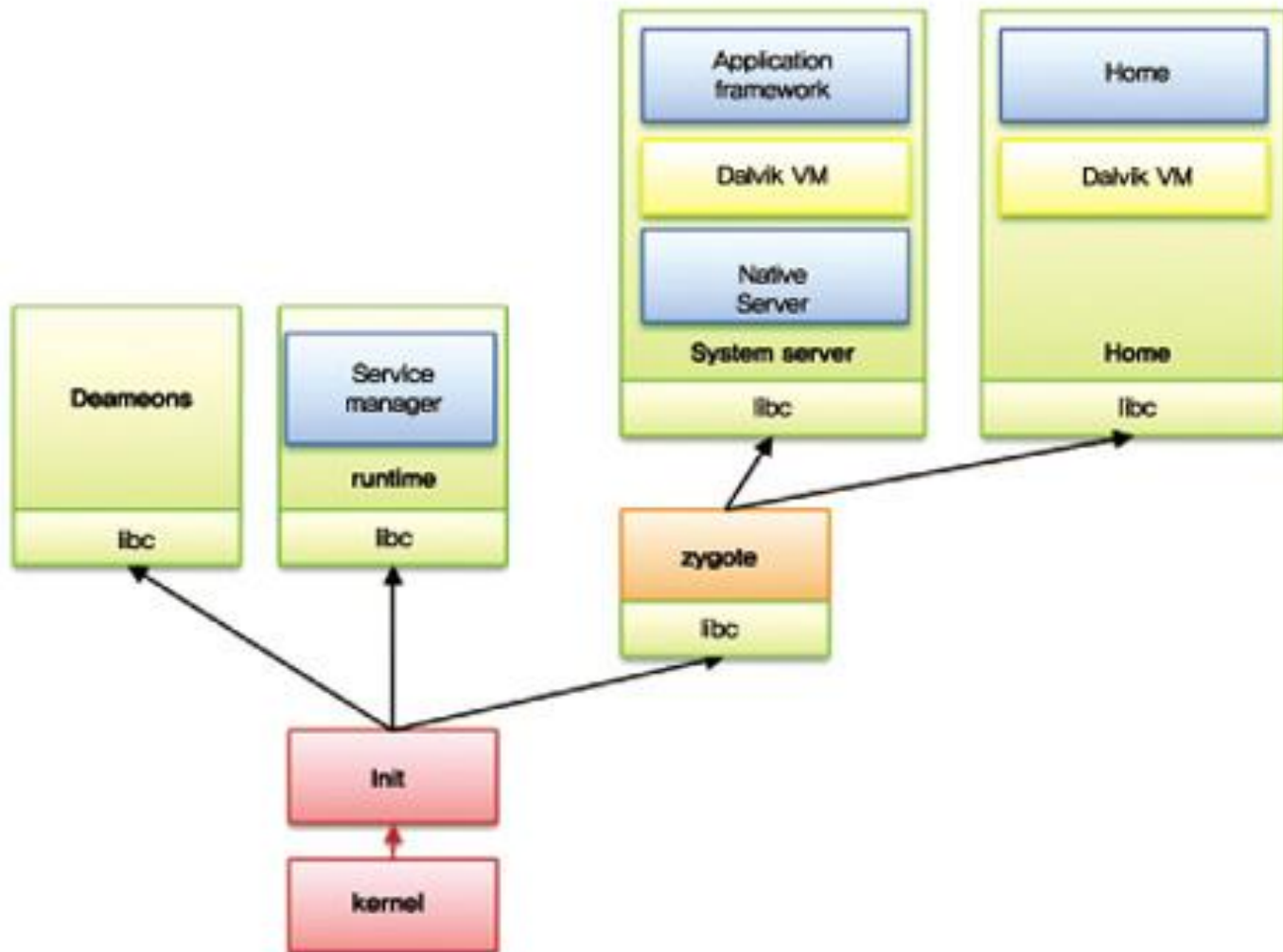
All data on the phone is erased, i.e. both the applications' private data and the shared data that is accessible over USB, including photos and movies. Be sure to make a backup of any precious files you have before unlocking the bootloader.

Android booting / Layer interaction

Android booting Sequence



Android booting Sequence

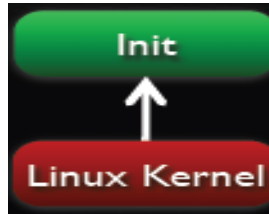


Android booting Sequence

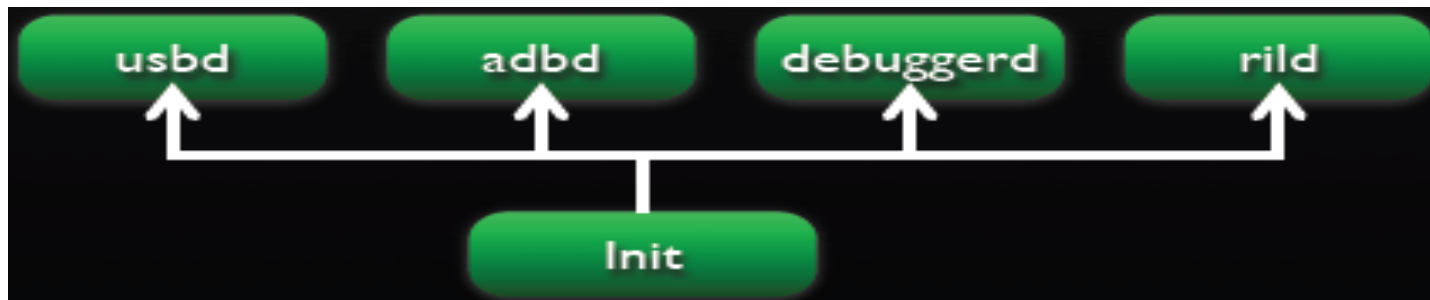
- 부팅 과정
 - 부트로더 : 시스템 초기화 및 리눅스 커널 로드/실행
 - 리눅스 커널 실행
 - init : 최초 생성 프로세스로 초기화 동작과 프로세스 생성 동작 수행
 - daemon process : 리눅스 서비스 프로세스
 - adbd : Android Debug Bridge 연결 관리 프로세스
 - usbd : USB 연결관리
 - debuggerd : debug system의 시작 관리
 - rild : 무선 통신 연결 관리
 - servicemanager : system service 관리하는 역할을 수행
 - mediaserver : audio, camera, mediaplayer system service 생성
 - serfaceflinger : surface flinger service 생성
 - zygote : android application 생성을 위한 프로세스
 - systemserver : android framework layer의 system service 생성
 - application process : 기본 실행 application 프로세스 생성

Android booting

- Linux Kernel booting 및 init process 생성



- init process에 의해서 Linux daemon process 생성

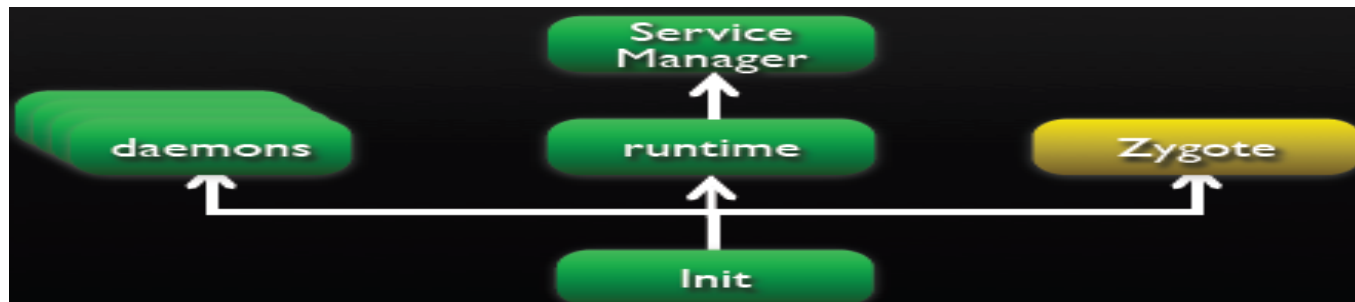


Android booting

- init process 생성에 의해서 zygote process 생성

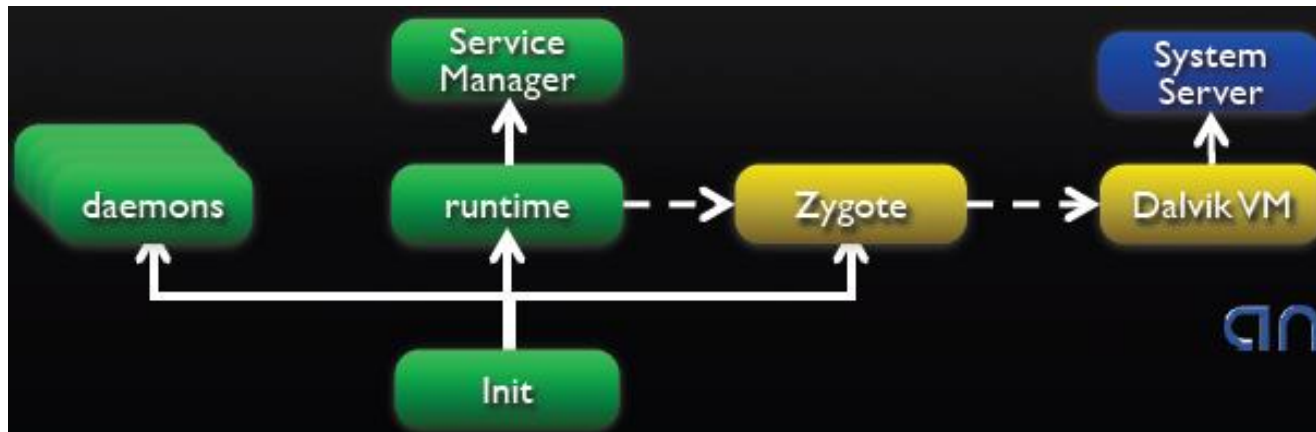


- init process에 의해서 runtime process 생성



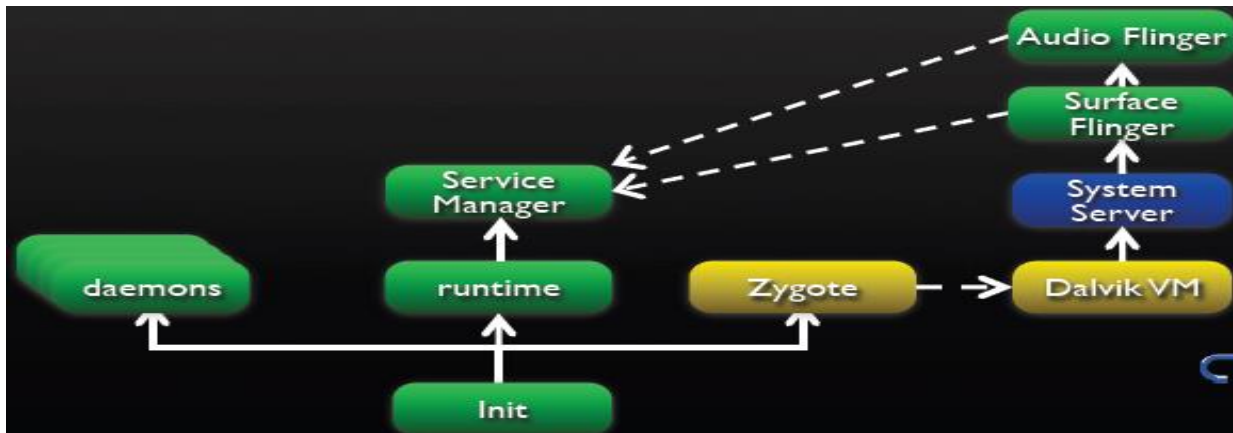
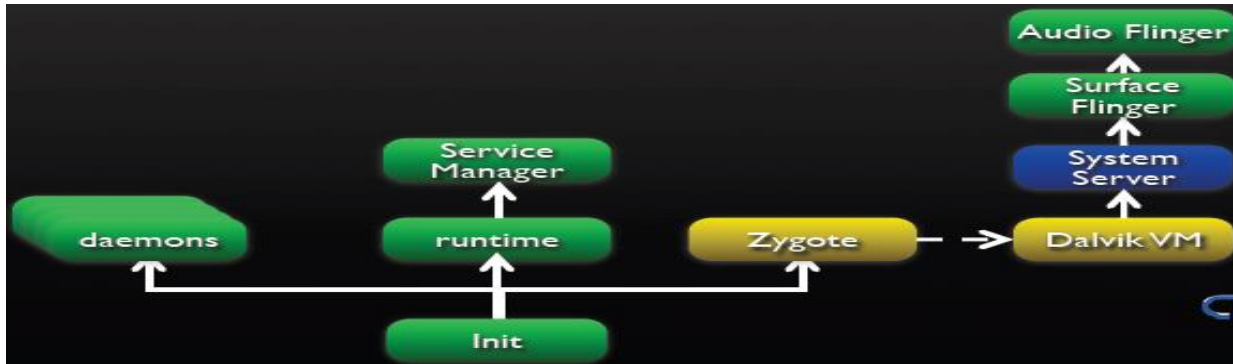
Android booting

- zygote process에 의해서 SystemServer process 생성



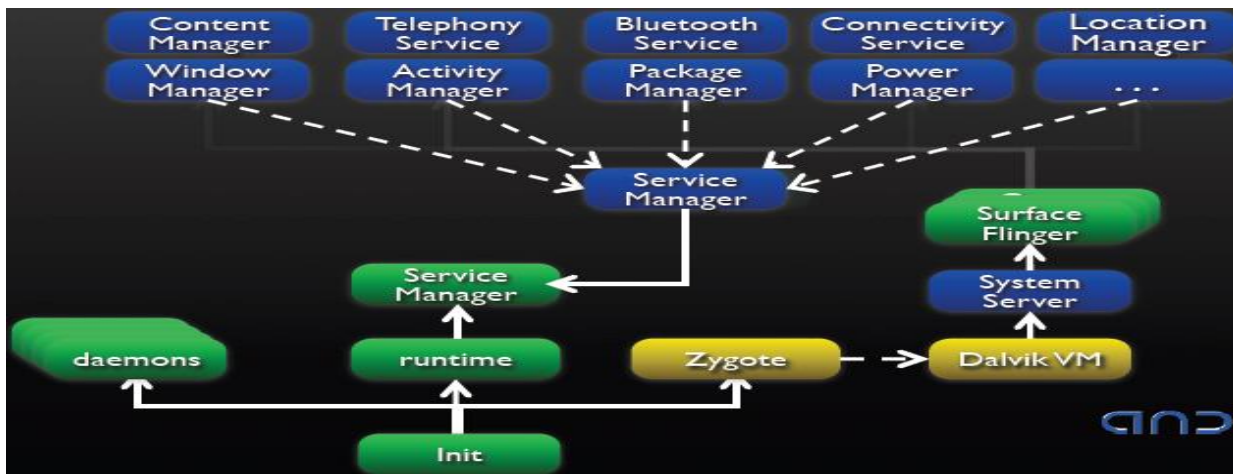
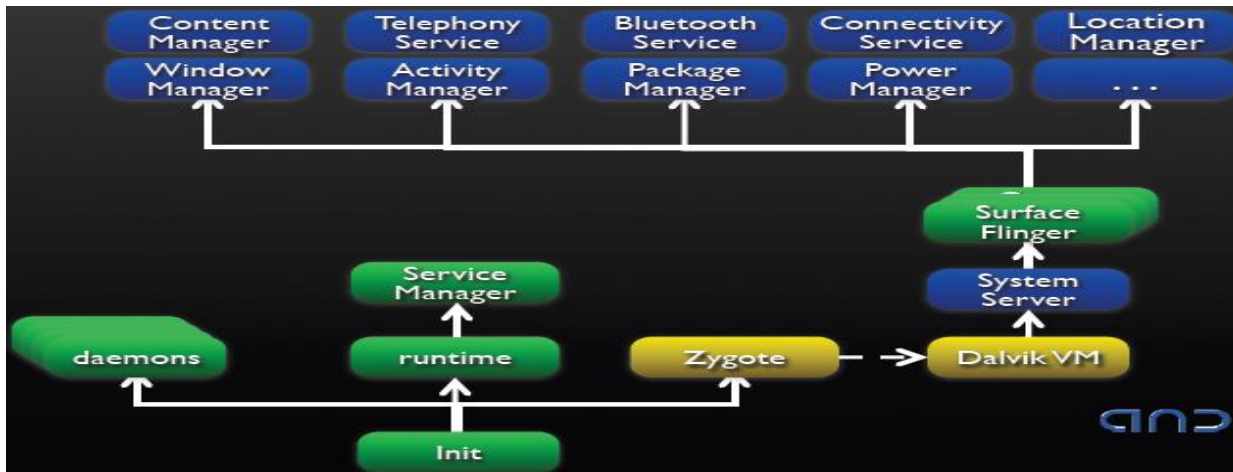
Android booting

- Native system service process 생성 및 Service Manager 등록



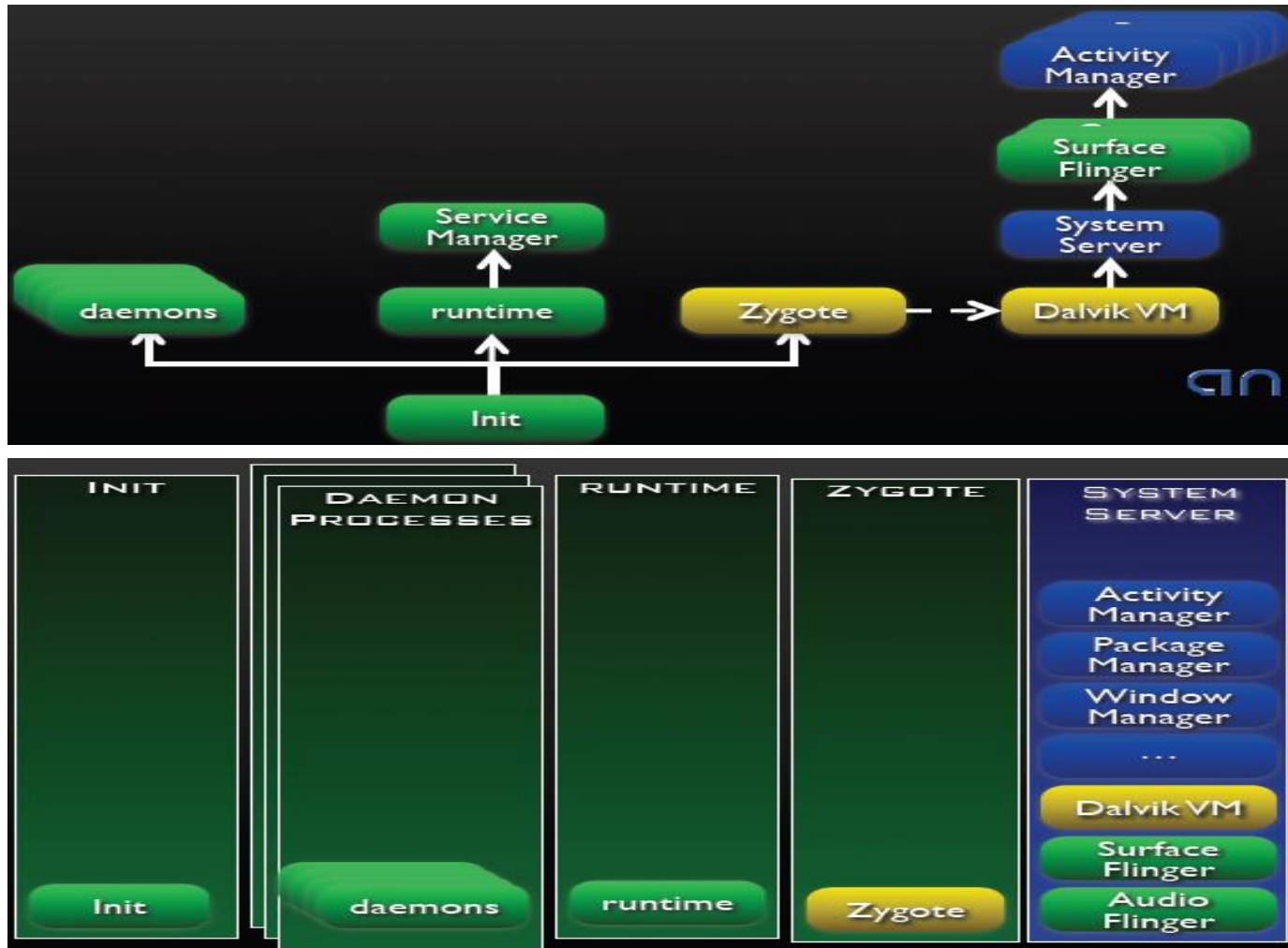
Android booting

- Java System Service 생성 및 Service Manager 등록



Android booting

- SystemServer process 생성 후 내용



Android booting

- zygote process에 의해 application process 생성



Android booting

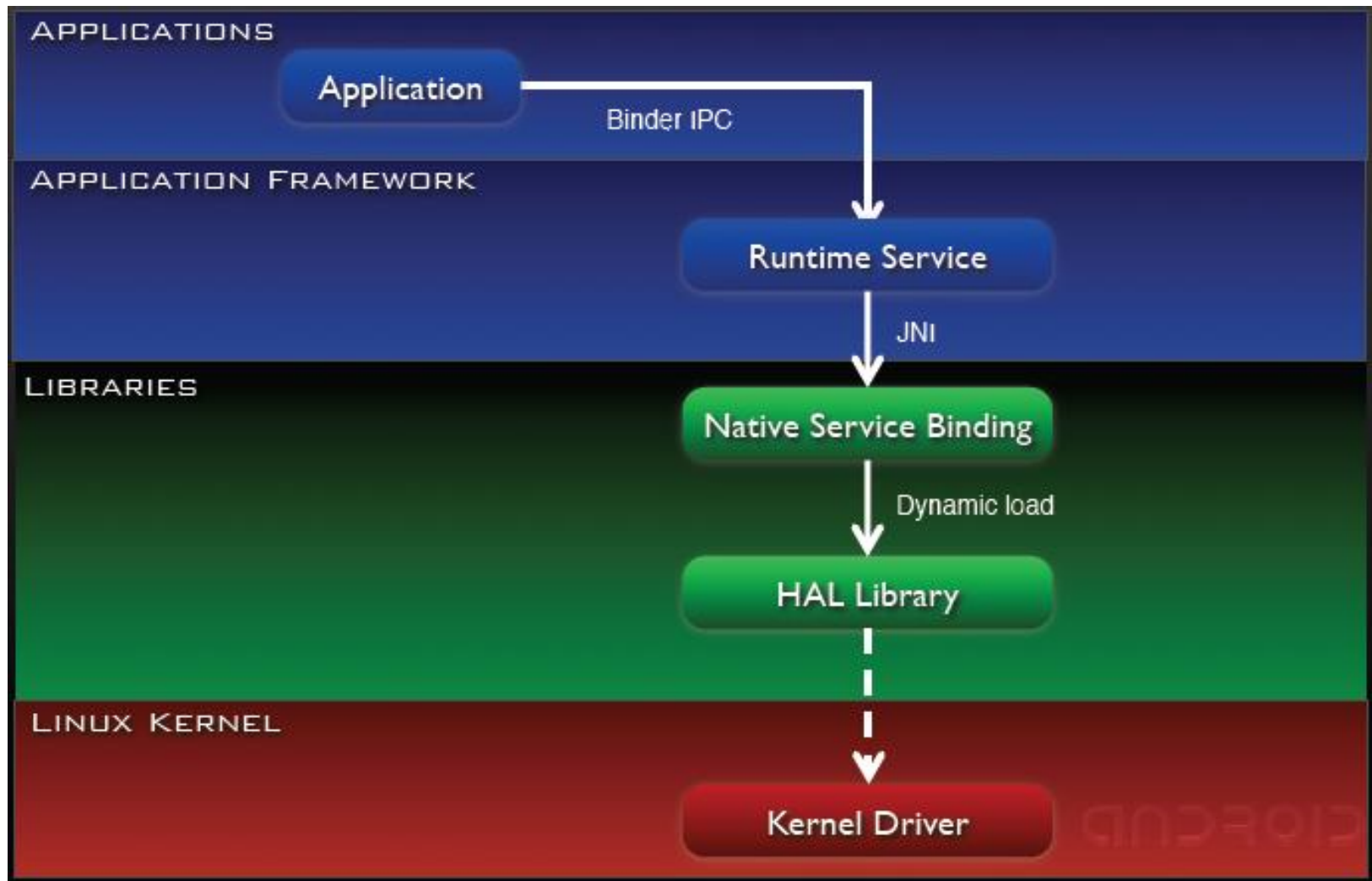
- zygote process에 의해 application process 생성



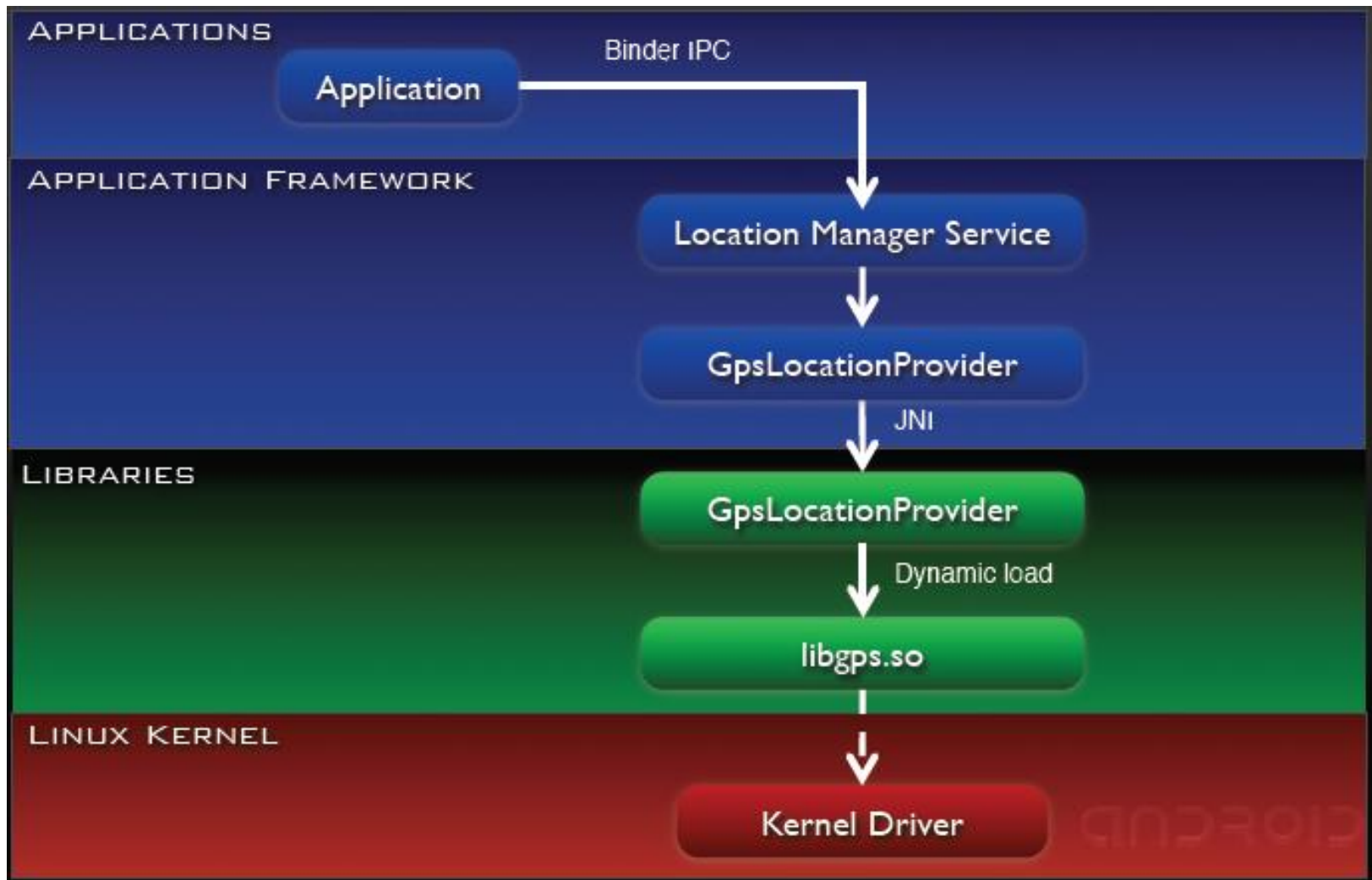
Layer Interaction

- App → Runtime Service → lib
 - Location Manager
- App → Runtime Service → Native Service → lib
 - Audio Flinger
- App → Runtime Service → Native Daemon → lib
 - Ril Daemon

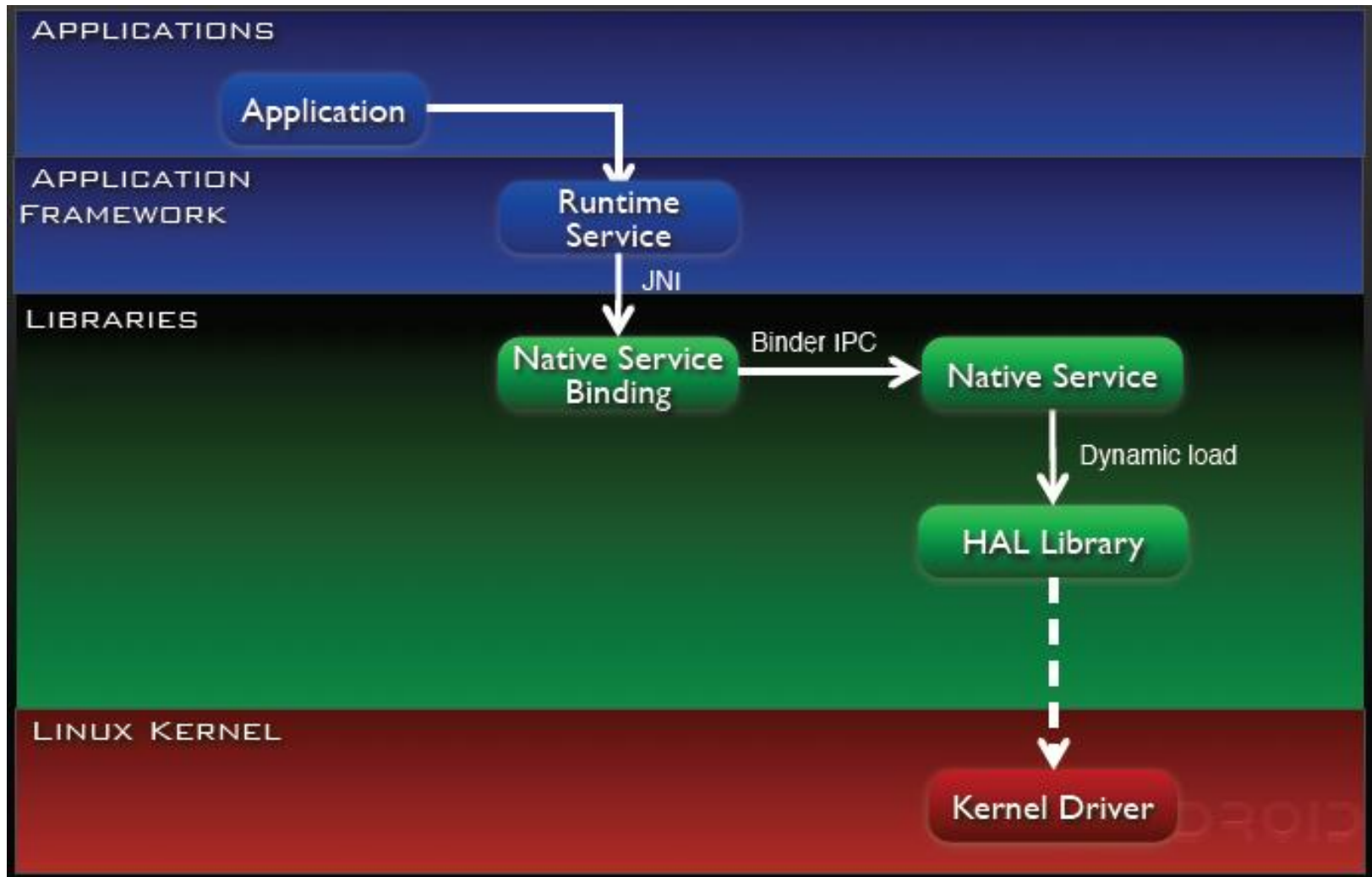
App -> Runtime Service -> lib



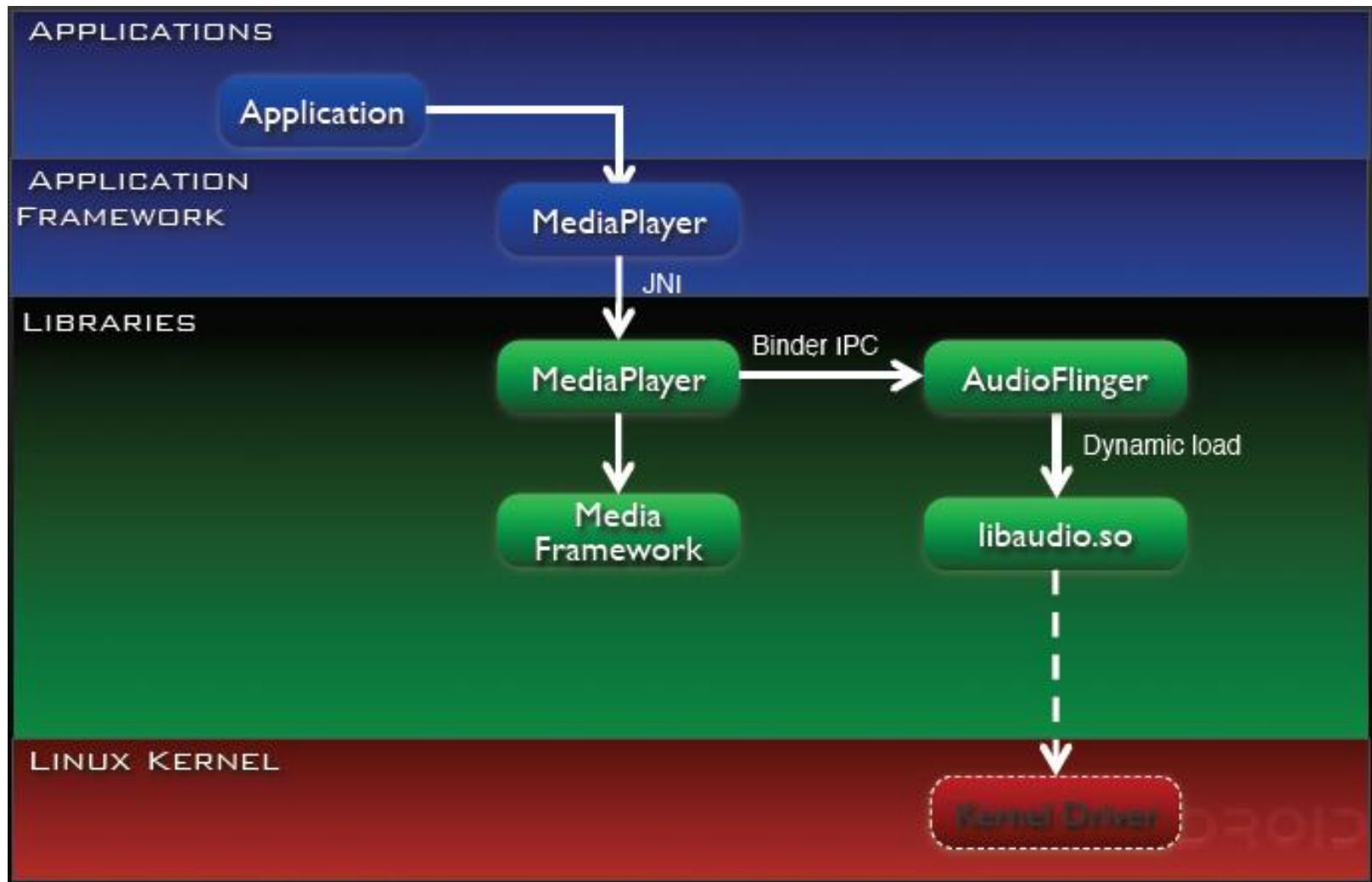
App -> Runtime Service -> lib (Location Manager)



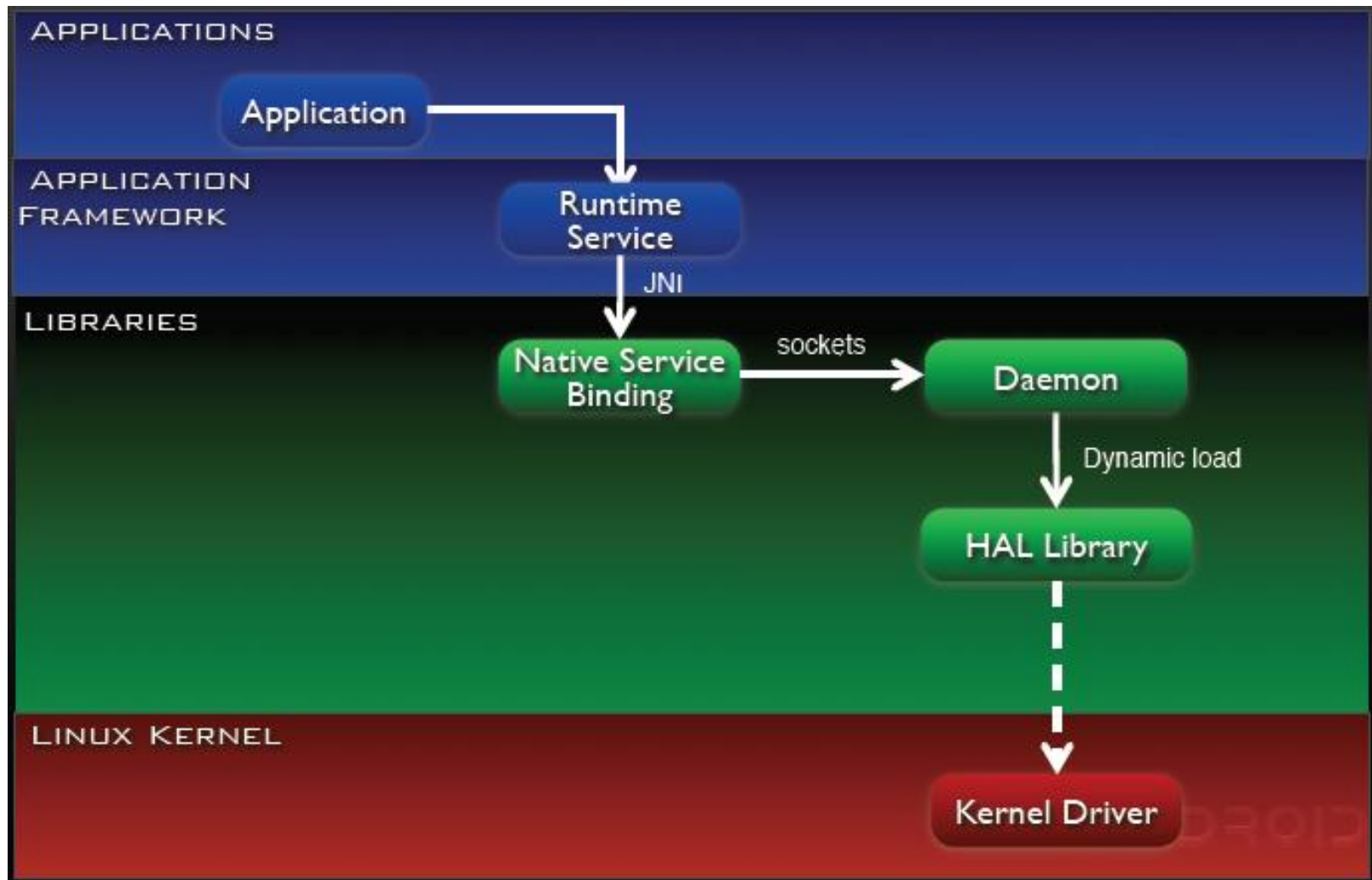
App -> Runtime Service -> Native Service -> lib



App -> Runtime Service -> Native Service -> lib(Audio Flinger)



App -> Runtime Service -> Native Daemon -> lib



App -> Runtime Service -> Native Daemon -> lib(Ril Daemon)

