3부 실 제 편

Chapter 11 안드로이드 아키텍처의 이해 및 분석
Chapter 12 에뮬레이터 개발 지원 환경 및 분석
Chapter 13 실제 타깃으로 포팅하기
Chapter 14 타깃 장치로 고급 포팅하기
Chapter 15 안드로이드 마켓

11장 안드로이드 아키텍처의 이해 및 분석

- **목차**

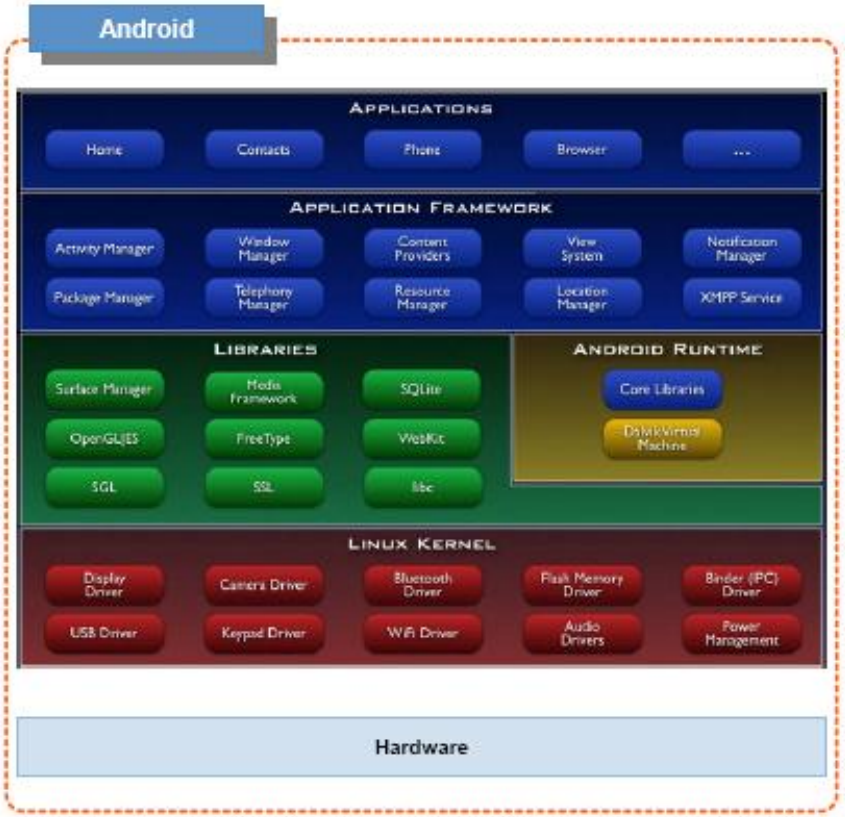
11.1 안드로이드 소프트웨어 플랫폼

플랫폼 구조

- 소프트웨어 플랫폼이란 ? 소프트웨어의 주요구성요소, 구성요소간의 인터페이스, 중요동작방식 등의 주요 특징들을 결정짓는 모든 설계 구조
- 소프트웨어 플랫폼은 소프트웨어 개발에 있어 가장 영향력
- 소프트웨어 플랫폼 종류 : 윈도우, 리눅스, 닷넷, 자바
- 런타임(runtime) 이란?
- 안드로이드 소프트웨어 플랫폼?

모바일 장치를 위한 소프트웨어 집합체
계층적 구조 : 커널, 라이브러리,
런타임라이브러리,애플리케이션 프레임워크,
애플리케이션으로 적층

- 자바 플랫폼이라고 하지 않음

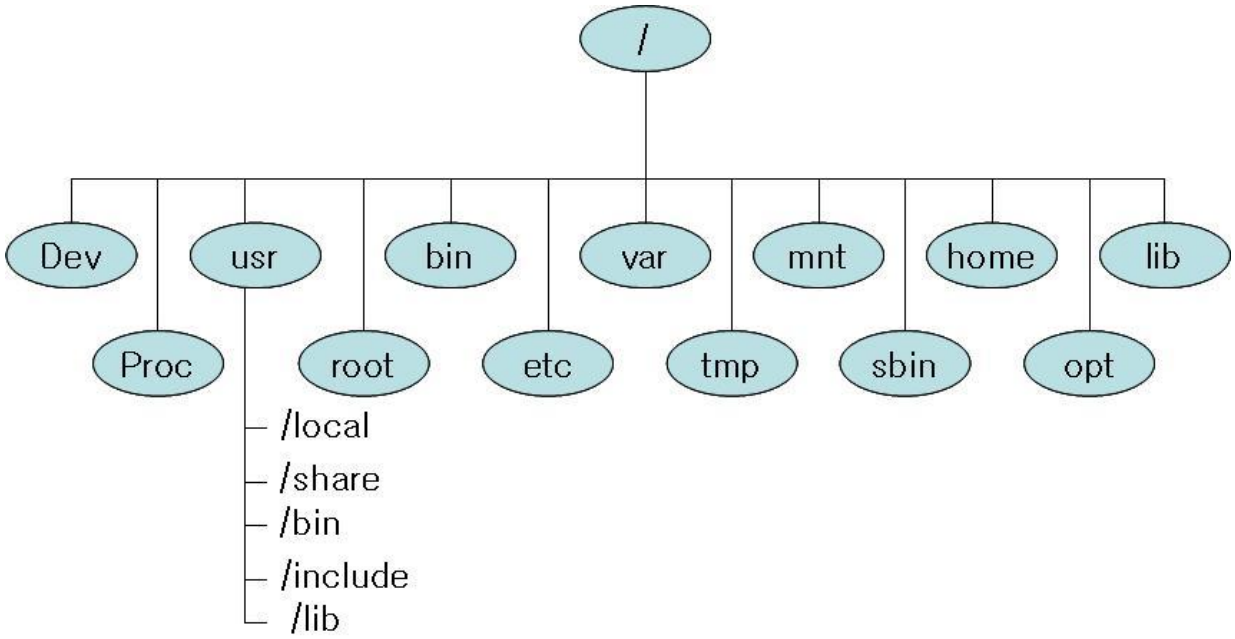


[그림 11-1] 안드로이드 플랫폼

11.1 안드로이드 소프트웨어 플랫폼

◉ 안드로이드 파일시스템 구조

- 일반 리눅스 파일시스템 구조와 다르다.

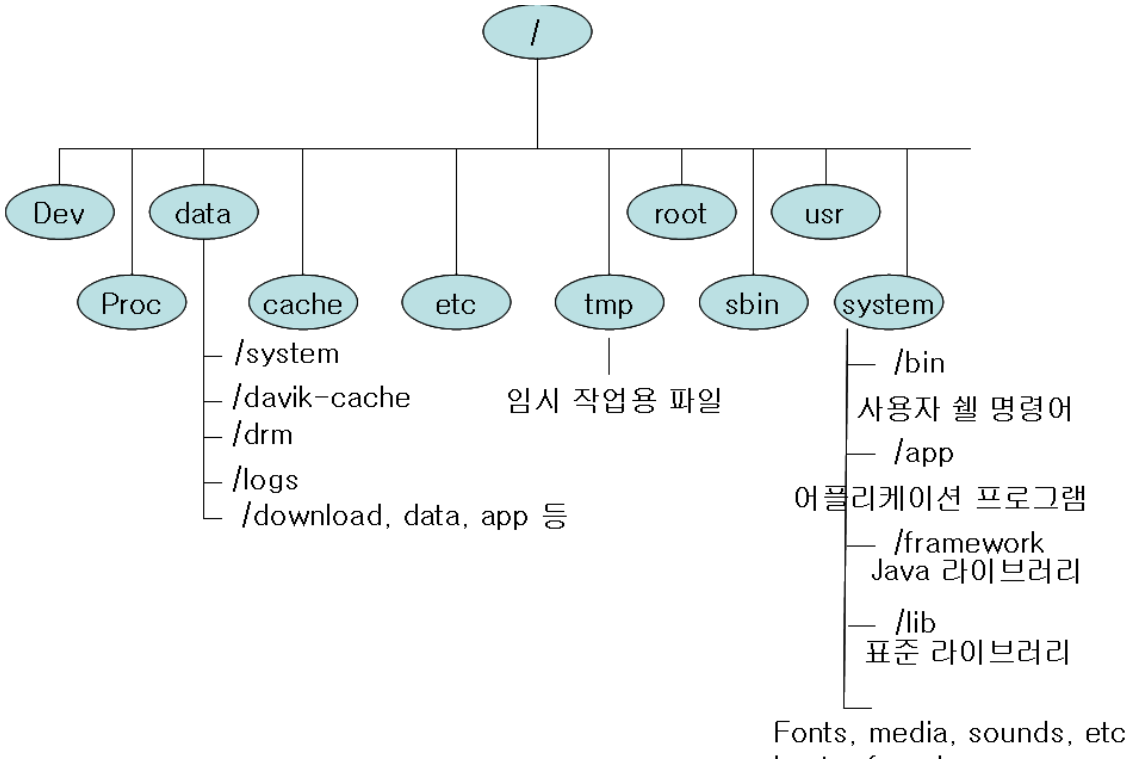


[그림 11-2] 일반 리눅스 파일시스템 구조

11.1 안드로이드 소프트웨어 플랫폼

안드로이드 파일시스템 구조

- 일반 리눅스 파일시스템에 안드로이드 플랫폼 위한 파일 저장 폴더 추가
- /data, /system 폴더 추가
- /recovery 추가 : 안드로이드 2.2 버전
- 각각의 추가된 폴더에 어떤 내용이 들어있는지 확인 필요

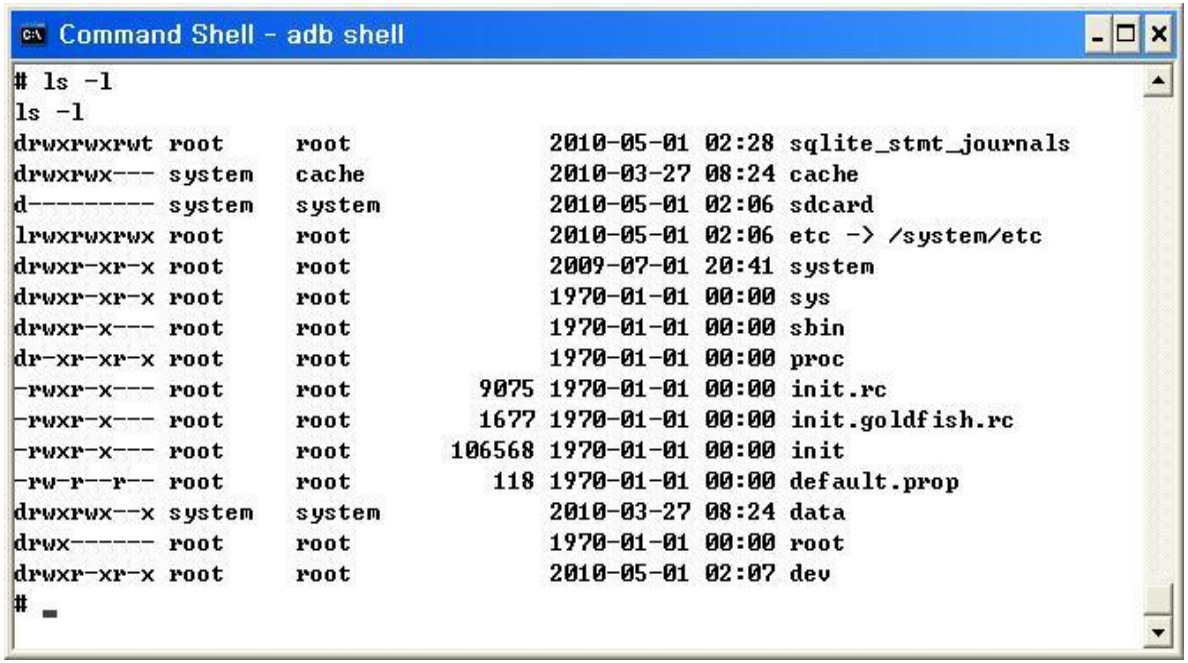


[그림 11-3] 안드로이드 리눅스 파일시스템 구조

11.1 안드로이드 소프트웨어 플랫폼

◉ 안드로이드 파일시스템 구조

- adb를 사용한 파일시스템 내용



[그림 11-4] 안드로이드 파일시스템 구조 확인

11.1 안드로이드 소프트웨어 플랫폼

안드로이드 파일시스템 구조

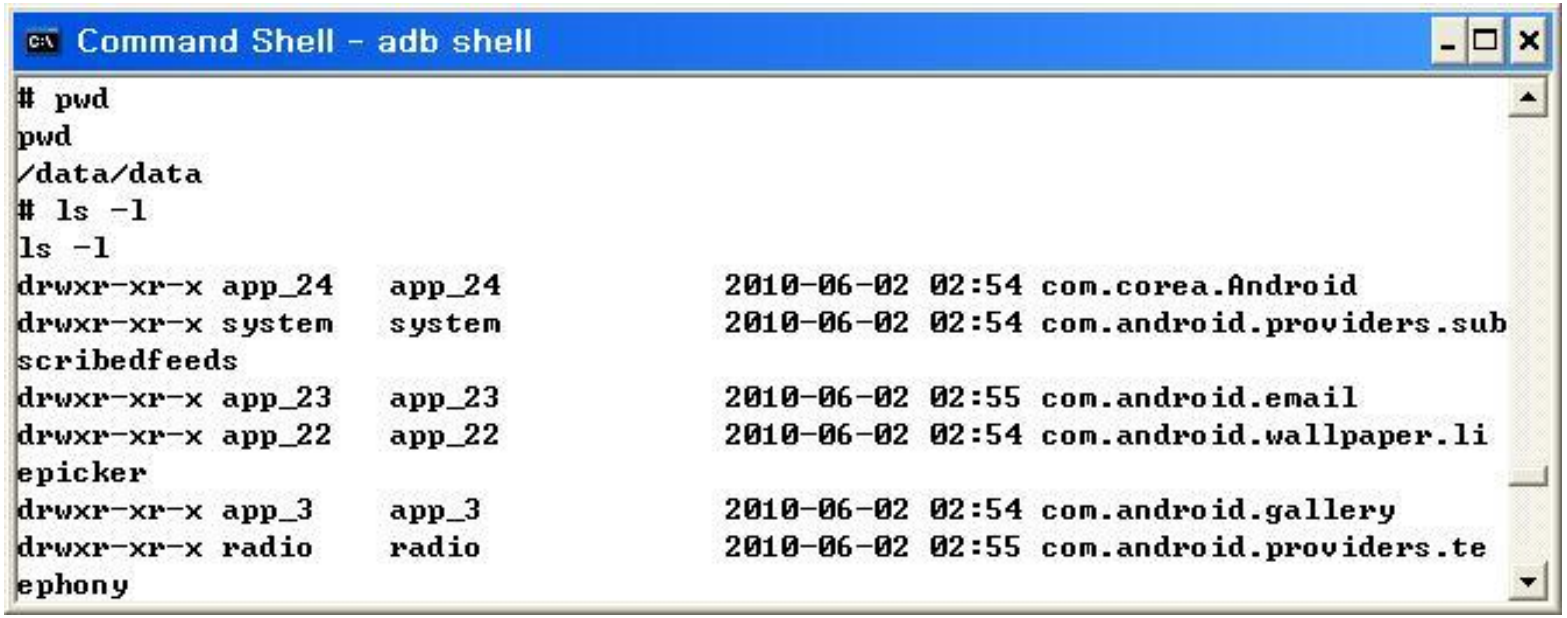
- adb를 사용한 /data/app/ 폴더 내용

[그림 11-5] /data/app 디렉토리 내용

11.1 안드로이드 소프트웨어 플랫폼

◉ 안드로이드 파일시스템 구조

- adb를 사용한 /data/data/ 폴더 내용



[그림 11-6] 애플리케이션용 데이터 저장 파일

11.2 안드로이드 커널

- **커널 특징**
  - **Open source**
  - **리눅스 커널 2.6.x 에 안드로이드 커널 패치**
  - **내부 메모리관리, 프로세스 관리, 네트워킹, 운영체계**
  - **상위층 : 라이브러리, 달빅 런타임**
  - **하위층: 하드웨어 (기본장치 외의 하드웨어를 모듈형식으로 동작)**

사용자 모드로 동작하면 커널 버전이 갱신될 때마다
플러그인 방식으로 배포/사용이 편리
속도 느려지는 단점
라이선스
안정성 향상

Command Shell - adb shell

```
crw-rw-rw- root     root       5,   2 2010-05-01 04:41 ptmx
crw------- root     root       5,   1 2010-05-01 02:06 console
crw-rw-rw- root     root       5,   0 2010-05-01 02:06 tty
drwxr-xr-x root     root              2010-05-01 02:06 graphics
crw------- root     root      10,  54 2010-05-01 02:06 network_throughput
crw------- root     root      10,  55 2010-05-01 02:06 network_latency
crw------- root     root      10,  56 2010-05-01 02:06 cpu_dma_latency
crw-rw-r-- system   radio     10,  57 2010-05-01 02:06 alarm
crw------- root     root      10,   1 2010-05-01 02:06 psaux
drwxr-xr-x root     root              2010-05-01 02:06 log
crw-rw-rw- root     root      10,  61 2010-05-01 02:06 binder
crw-rw-rw- root     root      10,  62 2010-05-01 02:06 ashmem
crw-rw---- root     audio     10,  63 2010-05-01 02:06 eac
crw------- root     root       1,  11 2010-05-01 02:06 kmsg
crw-rw-rw- root     root       1,   9 2010-05-01 02:06 urandom
crw-rw-rw- root     root       1,   8 2010-05-01 02:06 random
crw-rw-rw- root     root       1,   7 2010-05-01 02:06 full
crw-rw-rw- root     root       1,   5 2010-05-01 02:06 zero
crw-rw-rw- root     root       1,   3 2010-05-01 02:06 null
crw------- root     root       1,   2 2010-05-01 02:06 kmem
crw------- root     root       1,   1 2010-05-01 02:06 mem
crw------- root     root     254,   0 2010-05-01 02:06 rtc0
drwxr-xr-x root     root              2010-05-01 02:06 socket
drwxr-xr-x root     root              1970-01-01 00:00 pts
# _
```

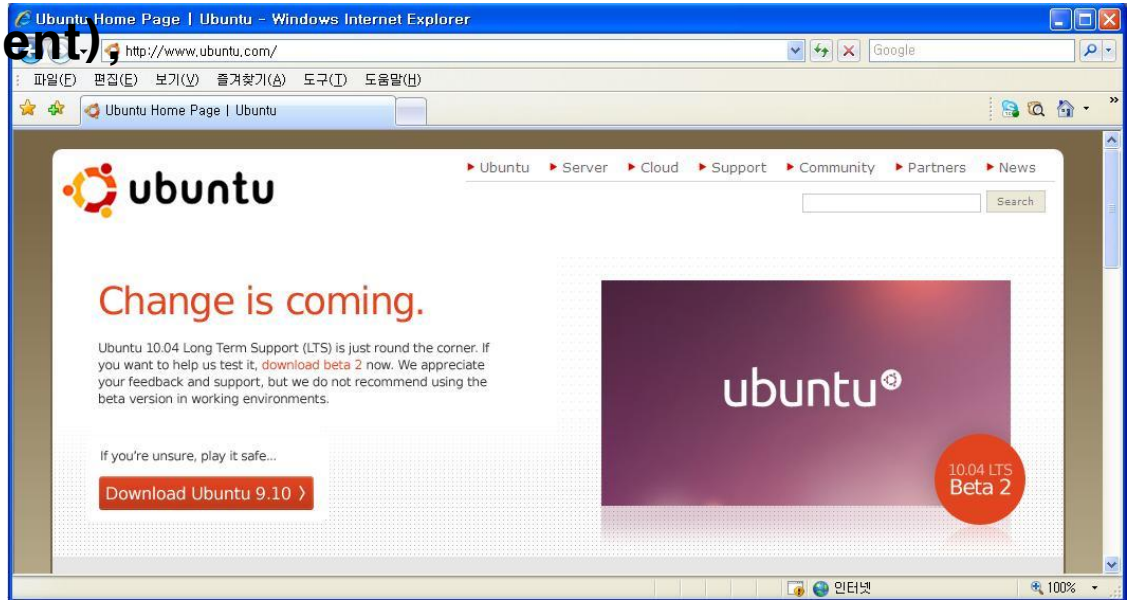
[그림 11-7]
/dev의 하드웨어
드라이버 확인

11.2 안드로이드 커널

우분투 리눅스 + 안드로이드 기능추가

- Debian GNU/Linux 기본
- 무료로 다운로드
- GPL, LGPL 라이선스
- 레드햇 리눅스 상업화 이후 독보적 마켓쉐어
- 커널 설정에서 다음 기능 추가

Alarm 기능,
공유메모리 (ashmem),
low memory-killer,
전력관리 (Power management),
바인더,
logger 등



[그림 11-8] 우분투 홈 사이트

11.2 안드로이드 커널

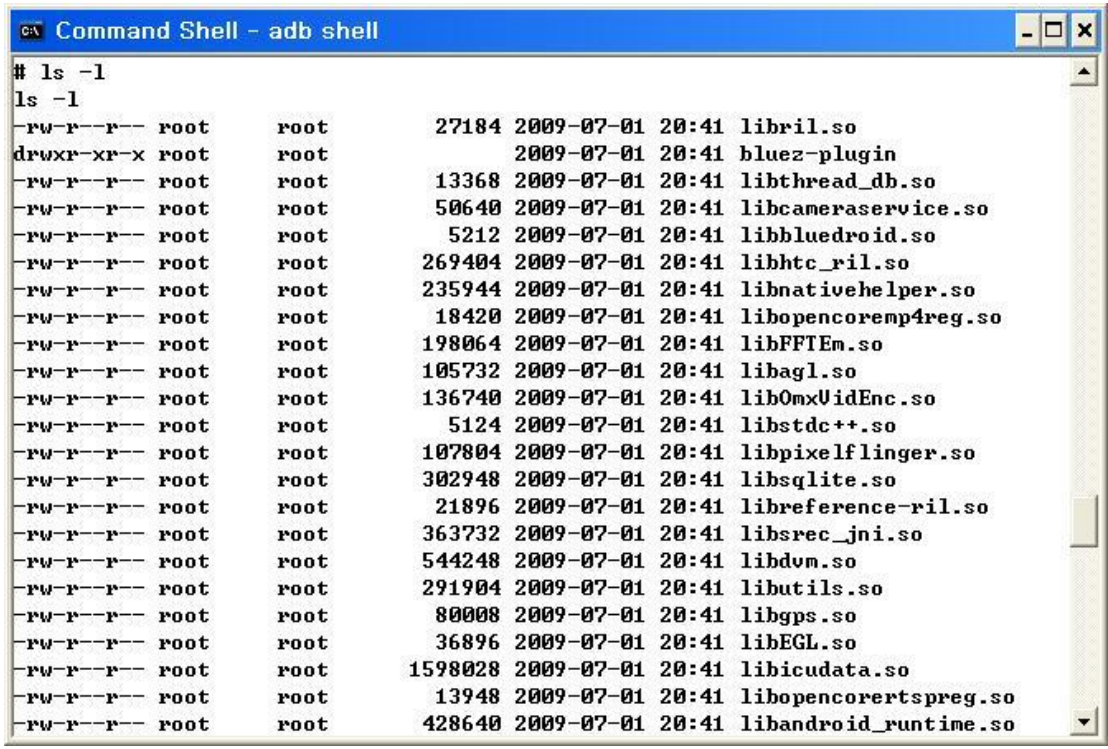
커널 빌드 도구 변경

- 기존 임베디드 시스템은 ARM 사의 ABI (Application Binary Interface) 기반 gcc 컴파일러 사용
- 안드로이드에서는 ARM EABI (Embedded Application Binary Interface) 기반의 툴체인 사용
- EABI?

11.2 안드로이드 커널

안드로이드 일반 라이브러리

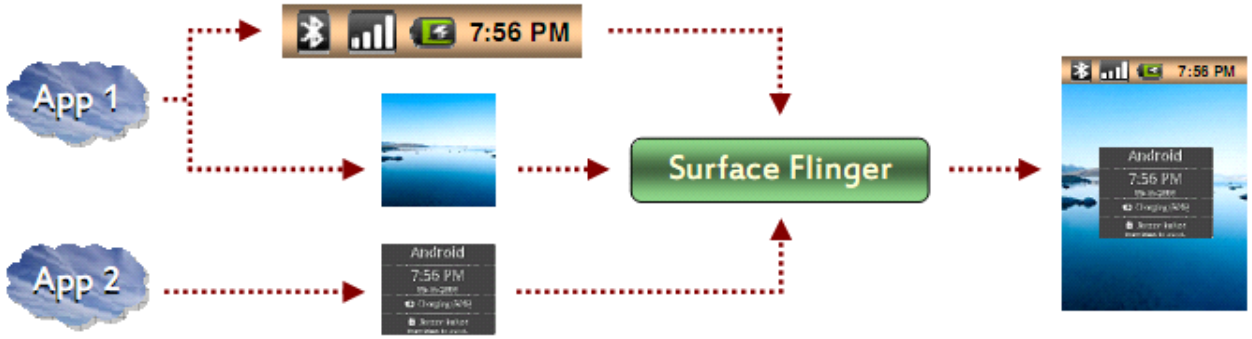
- 라이브러리 : C/C++로 제작
- C/C++ 라이브러리 변경 : glibc -> BSD Unix의 libc->Bionic C
- 라이브러리가 동적 라이브러리 확장자(so)/system/lib 폴더에 위치
- surface manager, media framework, sqlite, webkit, libc등으로 구성



[그림 11-9]
안드로이드 라이브러리 종류

11.2 안드로이드 커널

- **안드로이드 일반 라이브러리 : surface manager**
  - 애플리케이션 생성화면 -> 프레임버퍼 저장(RAM) -> LCD 화면표시
  - LCD와 CPU 데이터 처리속도 차이: 프레임버퍼 저장후 일시에 LCD 출력
  - 2D, 3D 표현
  - 화면합성, 결합, 반투명 효과



[그림 11-10] 서피스 매니저

11.2 안드로이드 커널

안드로이드 일반 라이브러리 :

- 미디어 프레임워크 : PocketVideo OpenCORE 기반, MPEG4, H264, AAC
- Sqlite : 개방형 관계형 데이터베이스
- WebKit : 웹브라우저 기반 엔진, 화면 웹 내용, 오픈소스, 웹 관련 클래스 제공

- glibc : GPL, 소스공개의무
  libc : BSD, 공개의무 없음,
  코드 경량,
  libc를 플랫폼에 맞게 수정한 Bionic C lib,
  프로세스마다 동적으로 적재, libc.so: 226KB의 경량 쓰레드
- 모든 라이브러리 코드가 bionic과 함께 컴파일
- LibWebCore
- OpenGL ES
- FreeType
- Audio Manager

[그림 11-11] 오디오 매니저

11.2 안드로이드 커널

- **안드로이드 런타임 라이브러리 :**
  - 컴퓨터 프로그램을 실행하는 동안의 기능을 추가하기 위하여 사용되는 특별한 프로그램 라이브러리
  - 입출력, 메모리 관리, 산술함수 기능
  - 일반 라이브러리와 차이는 컴파일러 제조사에 따라 다름
  - 런타임 라이브러리는 컴파일러와 플랫폼에 종속
  - 달빅(Dalvik) 가상머신의 지원으로 하드웨어 종속성 해결,
  - 달빅은 405KB의 livdvm.so 라이브러리로 지원

11.2 안드로이드 커널

◉ 안드로이드 런타임 라이브러리 : DalVik VM

- 달빅 가상머신
- 가장 신비스러운 곳 :미공개 소스
- Interpreter, register based, 적은 메모리에 최적화된 가상머신, banked 레지스터 기반 데이터 이동으로 속도가 빠름
- 일반자바 가상머신: stack based(PUSH, POP 사용)
- 자바 언어 사용하지만 선마이크로시스템스의 자바와는 다름(특허문제 때문)
- 선 마이크로시스템스 Java : 1-2바이트 길이 코드
- 안드로이드 Java : 4 바이트 코드, ARM 프로세서에 최적화, 4바이트 레지스터로 명령수행, 빠른 성능

```
000b: iload 05
000d: iload 04
000f: if_icmpge 0024
0012: aload_3
0013: iload 05
0015: iaload
0016: istore 06
0018: lload_1
0019: iload 06
001b: i2l
001c: ladd
001d: lstore_1
001e: iinc 05, #+01
0021: goto 000b
```

가) 자바 바이트코드

```
0007: if-ge v0, v2, 0010
0009: aget v1, v8, v0
000b: int-to-long v5, v1
000c: add-long/2addr v3, v5
000d: add-int/lit8 v0, v0, #int 1
000f: goto 0007
```

나) 달빅 바이트코드

[그림 11-12] 자바와 달빅 가상머신 바이트코드 비교

11.2 안드로이드 커널

- **안드로이드 런타임 라이브러리 : Dalvik VM**
  - **일반 자바 : 소스코드(.java) -> 자바 컴파일러 -> 클래스생성 (.class)**
    -> 가상머신에서 실행
  - **안드로이드 자바 : 소스코드(.java) -> 자바 컴파일러 -> 클래스생성 (.class)**
    -> dx 사용 -> dex (최소 메모리로 최적화)
    -> Dalvik 가상머신에서 실행

11.2 안드로이드 커널

안드로이드 런타임 라이브러리 : Dalvik VM

- dex 파일 구조 실습

```
tools> dexdump -d classes.dex
```

<실습 11-1> dexdump의 이용

1. cmd 명령으로 명령창 띄운다.
2. D 드라이버로 이동하기 위하여 d:을 입력한다.
3. 안드로이드 SDK에 대한 환경 변수 설정이 되어 있지 않다면 "cd android\android-sdk\platforms\android-3\tools" 명령을 실행한다.
4. "dexdump -f D:\android\workspace\android\bin\classes.dex" 명령을 실행

11.2 안드로이드 커널

◉ 안드로이드 런타임 라이브러리 : Dalvik VM

- dex 파일 구조 실습

DEX 버전: 최초 8 바이트의 Magic Value 값은 '035'이다.

```
Command Shell

D:₩android₩android-sdk₩platforms₩android-3₩tools>dexdump -f D:₩android₩workspace
₩android₩bin₩classes.dex
Processing 'D:₩android₩workspace₩android₩bin₩classes.dex'...
Opened 'D:₩android₩workspace₩android₩bin₩classes.dex', DEX version '035'
DEX file header:
magic               : 'dex
035'
checksum            : 6e030ca9
signature           : b445...4a56
file_size           : 1904
header_size         : 112
link_size           : 0
link_off            : 0 (0x000000)
string_ids_size     : 34
string_ids_off      : 112 (0x000070)
type_ids_size       : 14
type_ids_off        : 248 (0x0000f8)
field_ids_size      : 4
field_ids_off       : 340 (0x000154)
method_ids_size     : 11
method_ids_off      : 372 (0x000174)
class_defs_size     : 6
class_defs_off      : 460 (0x0001cc)
data_size           : 1252
data_off            : 652 (0x00028c)

Class #0            -
  Class descriptor  : 'Lcom/corea/Android/Android;'
  Access flags      : 0x0001 (PUBLIC)
  Superclass        : 'Landroid/app/Activity;'
  Interfaces        -
  Static fields     -
  Instance fields   -
  Direct methods    -
    #0              : (in Lcom/corea/Android/Android;)
      name          : '<init>'
      type          : '()V'
      access        : 0x10001 (PUBLIC CONSTRUCTOR)
      code          -
      registers     : 1
      ins           : 1
      outs          : 1
      insns size    : 4 16-bit code units
```

[그림 11-13]
dex 파일의 헤더 구조

11.2 안드로이드 커널

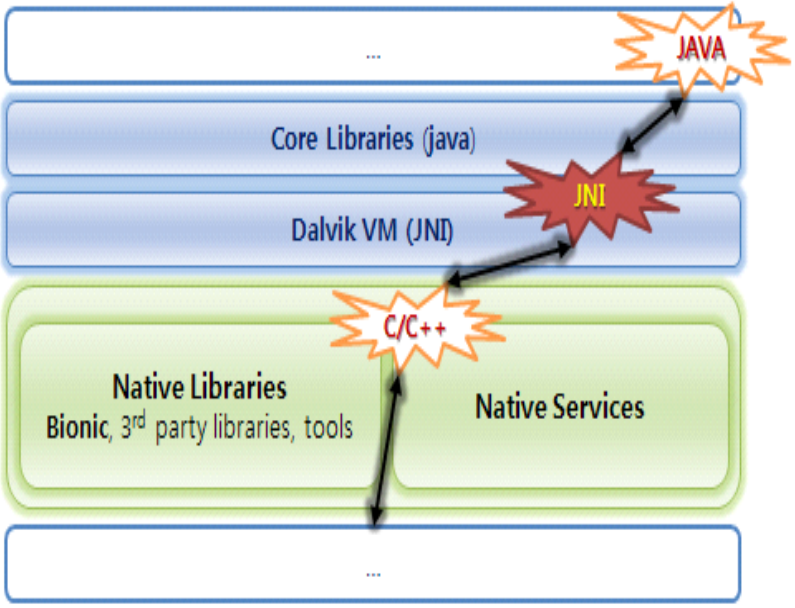
<표 11-1> Dex 파일 헤더구조

오프셋	크기 (바이트)	설 명
0x0	8	'Magic' value: "dex\n009\0"
0x8	4	Checksum
0xC	20	SHA-1 Signature
0x20	4	Length of file in bytes
0x24	4	Length of header in bytes (currently always 0x5C)
0x28	8	Padding (reserved for future use?)
0x30	4	Number of strings in the string table
0x34	4	Absolute offset of the string table
0x38	4	Not sure. String related
0x3C	4	Number of classes in the class list
0x40	4	Absolute offset of the class list
0x44	4	Number of fields in the field table
0x48	4	Absolute offset of the field table
0x4C	4	Number of methods in the method table
0x50	4	Absolute offset of the method table
0x54	4	Number of class definitions in the class definition table
0x58	4	Absolute offset of the class definition table

11.2 안드로이드 커널

안드로이드 런타임 라이브러리 : Dalvik VM

- 모든 애플리케이션은 각자의 프로세스 실행 –고유의 달빅 가상머신 인스턴스
- 그래서 안드로이드는 동시에 여러 개의 가상머신 실행
- 일반 자바는 1 개의 가상머신 실행
- 안드로이드에서 애플리케이션 프로세스 간의 독립과 메모리 관리 쓰레드는 리눅스 운영체제의 지원
- 애플리케이션(자바코드)에서 JNI를 통하여 C 라이브러리 (하드웨어 드라이버) 호출이 가능



[그림 11.14] 라이브러리 동작 개념

11.2 안드로이드 커널

Application Framework

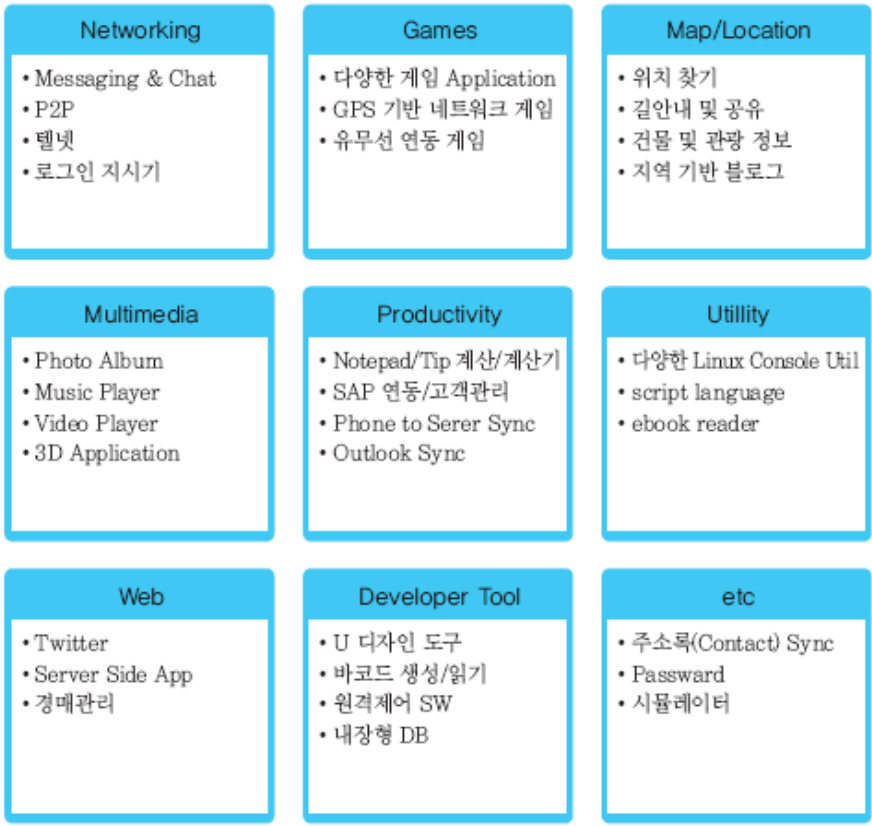
- 애플리케이션 기발
- Activity manager, Window manager, package manager, telephony, resource manager, location manager, content provider 로 구성
- 개발자는 프레임워크의 API 활용하여 애플리케이션 개발
- 자바로 구현
- 프레임워크와 라이브러리 사이에는 자바와 C 사이의 호출 규약을 매핑하는 JNI 바인딩 기술 존재

- 윈도우, 리눅스 : 다수의 애플리케이션 동시에 표시
- 안드로이드 : 1 개의 foreground 애플리케이션 표시,
  액티비티 매니저에 의해 관리
- 패키지 매니저 : 시스템에 동작중인 애플리케이션 관련 정보 관리,
  하나의 패키지 파일로 배포 (윈도우는 다수의 DLL로 구성,배포)
- 윈도우 매니저 : 애플리케이션과 관련된 화면 관리
- 컨텐트 프로바이더 : 데이터에 대한 접근 제어, 다른 애플리케이션과 데이터 공유,

11.2 안드로이드 커널

Application

- Activity, Broadcast receiver, service, content provider 4가지 components 조합으로 구성
- 애플리케이션간의 동일한 권한
- 애플리케이션에서는 사용 컴포넌트 목록을 AndroidManifest.xml 파일에 기록
- 하나의 애플리케이션 패키지 : apk
- 기본 Home 화면 내용: 전화 dial, e-mail, 주소록, web browser, 안드로이드 마켓
- 애플리케이션은 각각 독립된 메모리 공간
- 인텐트 : 애플리케이션 간의 메모리 공유

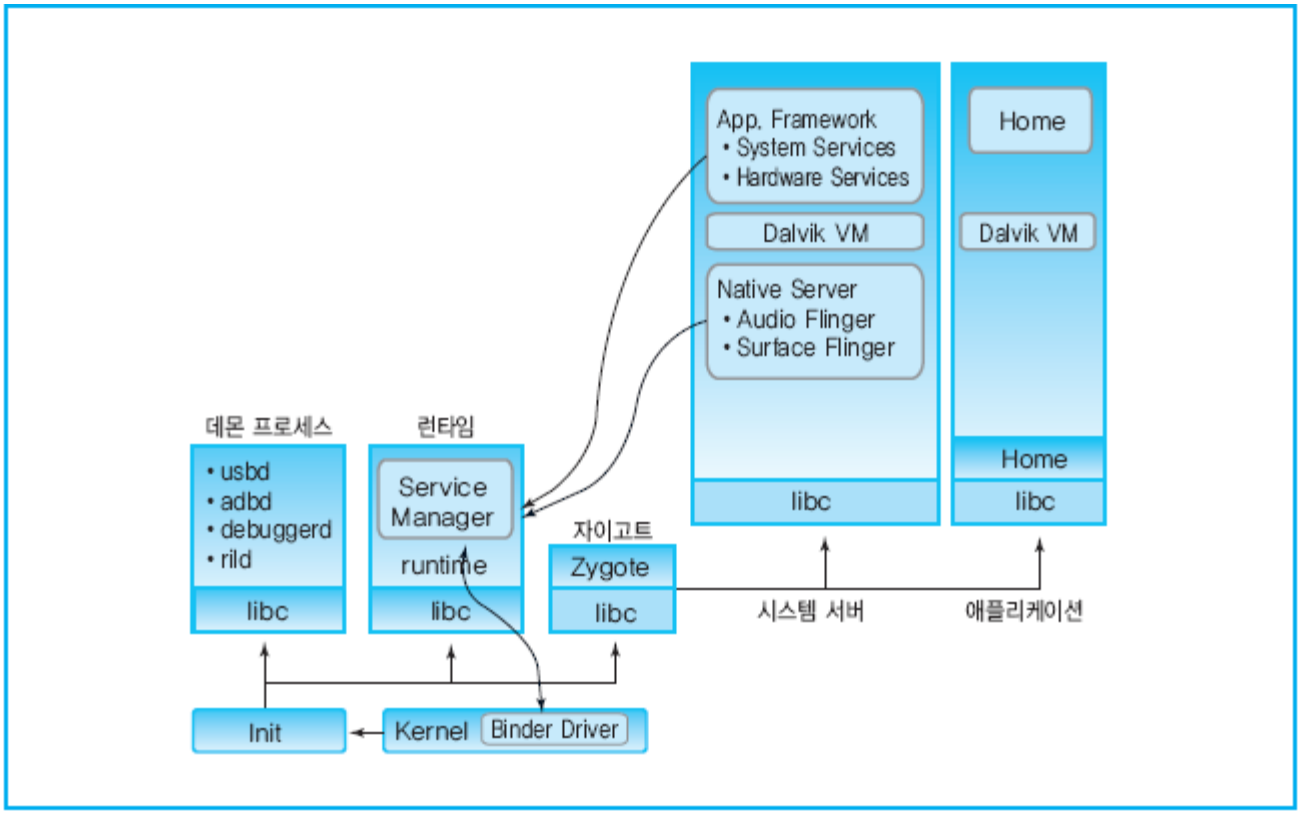


[그림 11-15] 애플리케이션 분류

11.2 안드로이드 커널

안드로이드 플랫폼 부팅

- 커널 부팅 -> init (파일시스템 마운팅, 폴더권한 설정, 시작프로그램 동작) -> 안드로이드 데몬 적재



출처: http://yotteum.tistory.com/1

[그림 11-16] 안드로이드 플랫폼 부팅 과정

11.2 안드로이드 커널

안드로이드 플랫폼 부팅

- 리눅스 커널 시작후 안드로이드의 고유 데몬 실행
- 소리볼륨 데몬-> 디버그시스템 데몬
  -> 자이고트

Command Shell - adb shell

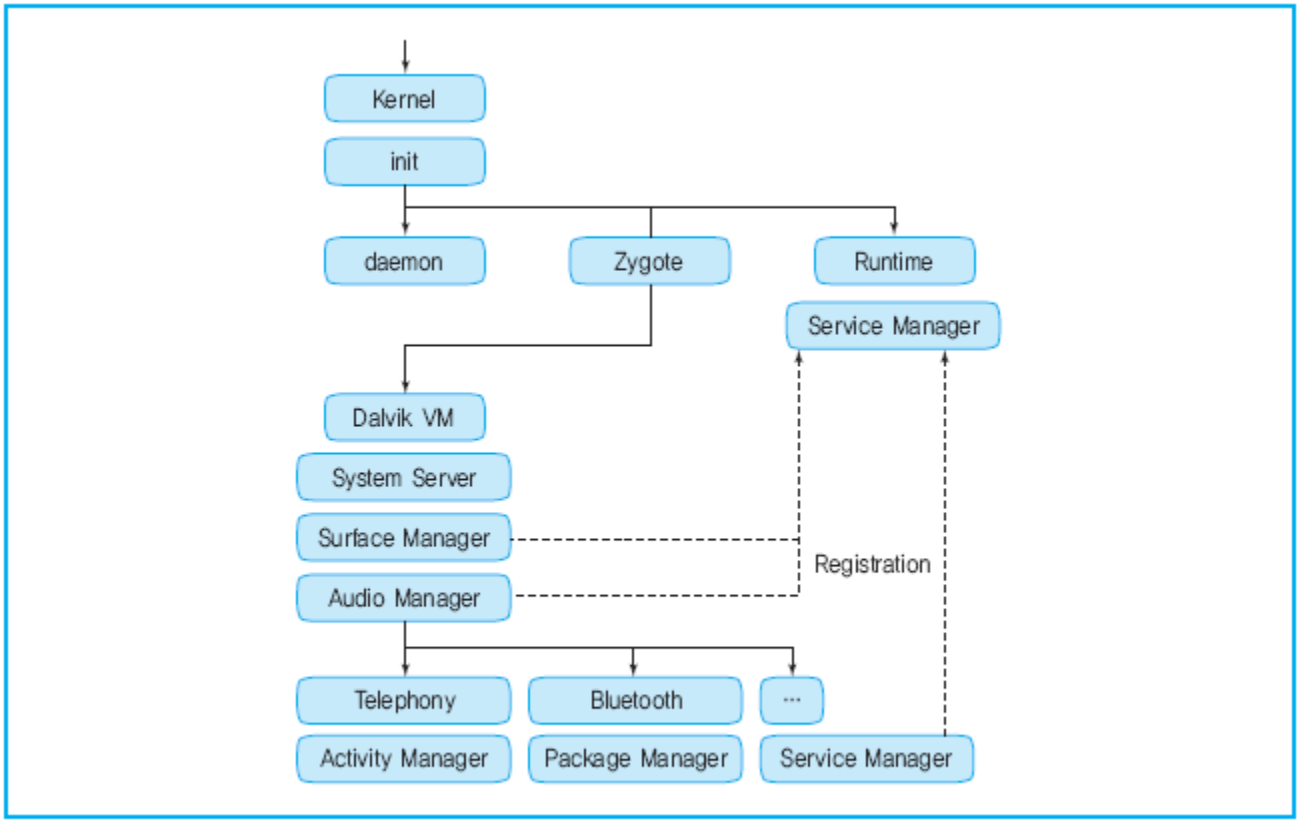
```
# ps -x
ps -x
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     280    188   c009a694 0000c93c S /init (u:9, s:472)
root      2     0     0      0     c004dea0 00000000 S kthreadd (u:0, s:1)
root      3     2     0      0     c003f778 00000000 S ksoftirqd/0 (u:0, s:1)
root      4     2     0      0     c004aaf4 00000000 S events/0 (u:0, s:13)
root      5     2     0      0     c004aaf4 00000000 S khelper (u:0, s:0)
root      6     2     0      0     c004aaf4 00000000 S suspend (u:0, s:0)
root      7     2     0      0     c004aaf4 00000000 S kblockd/0 (u:0, s:0)
root      8     2     0      0     c004aaf4 00000000 S cqueue (u:0, s:0)
root      9     2     0      0     c017bb3c 00000000 S kseriod (u:0, s:0)
root      10    2     0      0     c004aaf4 00000000 S kmmcd (u:0, s:0)
root      11    2     0      0     c006ecac 00000000 S pdflush (u:0, s:0)
root      12    2     0      0     c006ecac 00000000 S pdflush (u:0, s:0)
root      13    2     0      0     c007349c 00000000 S kswapd0 (u:0, s:11)
root      14    2     0      0     c004aaf4 00000000 S aio/0 (u:0, s:0)
root      21    2     0      0     c017933c 00000000 S mtdblockd (u:0, s:0)
root      22    2     0      0     c004aaf4 00000000 S hid_compat (u:0, s:0)
root      23    2     0      0     c004aaf4 00000000 S rpciod/0 (u:0, s:0)
root      24    1     728    308   c01537e8 afe0c7dc S /system/bin/sh (u:0, s:2)

system    25    1     796    260   c019a854 afe0ca7c S /system/bin/servicemanage
r (u:6, s:33)
root      26    1     820    364   c009a694 afe0cba4 S /system/bin/vold (u:2, s:
4)
root      27    1     656    248   c01a65e8 afe0d40c S /system/bin/debuggerd (u:
0, s:2)
radio     28    1     5420   728   ffffffff afe0d0ec S /system/bin/rild (u:20, s
:123)
root      29    1     82016  26168 c009a694 afe0cba4 S zygote (u:1729, s:244)
media     30    1     20948  3332  ffffffff afe0ca7c S /system/bin/mediaserver
```

[그림 11-17] 커널 부팅 후 실행 중인 프로세스

11.2 안드로이드 커널

◉ 안드로이드 플랫폼 부팅

- 자이고트
- 달빅 -> 시스템 서버 -> 서피스 매니저 -> 오디오 매니저 시동
- 자이고트와 시스템 서버간의 IPC (127.0.0.x) 소켓 통신



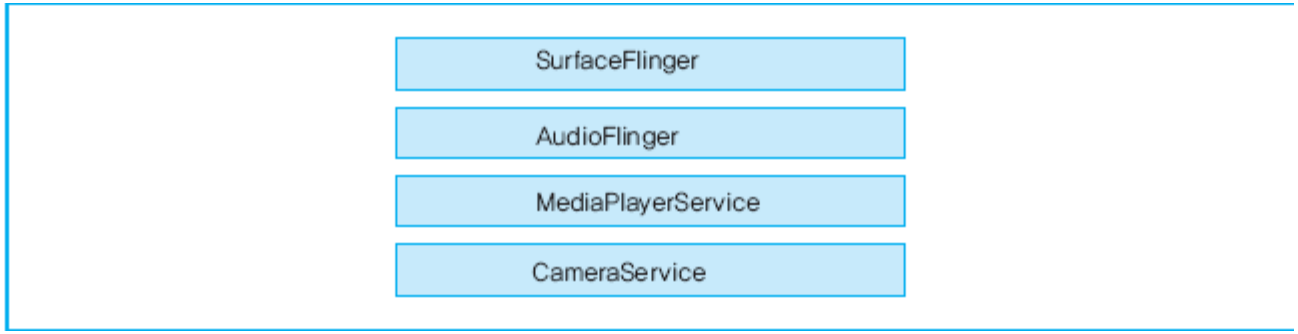
출처: http://yotteum.tistory.com/1

[그림 11-18] 안드로이드 부팅 과정

11.2 안드로이드 커널

안드로이드 플랫폼 부팅

- 시스템 서버(system server)
- Native services
- 미디어 관련 서비스 시작



[그림 11-19] 네이티브 서비스 초기화 순서

11.2 안드로이드 커널

안드로이드 플랫폼 부팅

- **Native service 초기화 코드 부분**

<코드 11-1> 네이티브 서비스 초기화 코드

```
extern "C" status_t system_init()
{
LOGI("Entered system_init()");
sp<ProcessState> proc(ProcessState::self());
sp<IServiceManager> sm = defaultServiceManager();
LOGI("ServiceManager: %p₩n", sm.get());
sp<GrimReaper> grim = new GrimReaper();
sm->asBinder()->linkToDeath(grim, grim.get(), 0);
char propBuf[PROPERTY_VALUE_MAX];
property_get("system_init.startsurfaceflinger", propBuf, "1");
if (strcmp(propBuf, "1") == 0) {
// Start the SurfaceFlinger
SurfaceFlinger::instantiate();
}
// On the simulator, audioflinger et al don't get started the
// same way as on the device, and we need to start them here
if (!proc->supportsProcesses()) {
// Start the AudioFlinger
AudioFlinger::instantiate();
// Start the media playback service
MediaPlayerService::instantiate();
// Start the camera service
CameraService::instantiate();
```

11.2 안드로이드 커널

◉ 안드로이드 플랫폼 부팅

- 자이고트 이후 각종 기본서비스 및 애플리케이션

Command Shell - adb shell

```
0, s:2)
radio     28    1     5420   728    ffffffff afe0d0ec S /system/bin/rild (u:20,
:123)
root      29    1     82016  26168  c009a694 afe0cba4 S zygote (u:1729, s:244)
media     30    1     20948  3332   ffffffff afe0ca7c S /system/bin/mediaserver
u:80, s:40)
root      31    1     784    316    c02094ac afe0c7dc S /system/bin/installd (u:
, s:4)
keystore  32    1     1616   404    c01a65e8 afe0d40c S /system/bin/keystore (u:
, s:2)
root      33    1     728    324    c003d444 afe0d6ac S /system/bin/sh (u:0, s:2

root      34    1     824    336    c00b7dd0 afe0d7fc S /system/bin/qemud (u:6,
:107)
root      36    1     4468   208    ffffffff 0000eca4 S /sbin/adbd (u:38, s:299)
root      43    33    780    308    c02094ac afe0c7dc S /system/bin/qemu-props (
:0, s:2)
system    51    29    148908 31100  ffffffff afe0ca7c S system_server (u:28146,
:17615)
app_7     99    29    108864 20216  ffffffff afe0da04 S com.android.inputmethod.
inyin (u:46, s:66)
radio     101   29    117948 22656  ffffffff afe0da04 S com.android.phone (u:709
, s:2735)
app_7     105   29    123452 24824  ffffffff afe0da04 S android.process.acore (u
385, s:206)
system    121   29    109724 19292  ffffffff afe0da04 S com.android.settings (u:
6, s:48)
app_17    145   29    102916 18896  ffffffff afe0da04 S com.android.alarmclock (
:44, s:57)
app_3     160   29    103576 19384  ffffffff afe0da04 S android.process.media (u
65, s:95)
app_14    179   29    112536 19480  ffffffff afe0da04 S com.android.mms (u:44, s
74)
app_23    204   29    106016 19892  ffffffff afe0da04 S com.android.email (u:56,
s:53)
app_12    214   29    101888 17524  ffffffff afe0da04 S com.svox.pico (u:17, s:4
)
app_24    223   29    103500 19356  ffffffff afe0da04 S com.corea.Android (u:39,
s:54)
root      240   36    736    336    c003d444 afe0d6ac S /system/bin/sh (u:2, s:1
)
root      266   240   868    332    00000000 afe0c7dc R ps (u:7, s:16)
# _
```

[그림 11-20] 안드로이드 데몬/애플리케이션 동작 상황

11.2 안드로이드 커널

안드로이드 플랫폼 부팅

- 안드로이드 부팅이 후 자바 서비스 시작 : 시스템 서버
- /frameworks/base/services/java/com/android/server/SystemServer.java
- 시스템 서버는 Native 서비스 외에도 다양한 서비스 지원

[그림 11-21] 시스템 서버 서비스 순서

11.2 안드로이드 커널

<코드 11-2> 시스템 서버에서 init2()와 스레드를 호출하는 코드

```
// And now start the Android runtime.
LOGI("System server: starting Android runtime.₩n");
AndroidRuntime* runtime = AndroidRuntime::getRuntime();
LOGI("System server: starting Android services.₩n");
runtime->callStatic("com/android/server/SystemServer",
"init2");
// If running in our own process, just go into the thread
// pool. Otherwise, call the initialization finished
// func to let this process continue its initilization.
if (proc->supportsProcesses()) {
LOGI("System server: entering thread pool.₩n");
ProcessState::self()->startThreadPool();
IPCThreadState::self()->joinThreadPool();
LOGI("System server: exiting thread pool.₩n");
}
return NO_ERROR;
}
```

11.2 안드로이드 커널

시스템 서버의 기타 서비스 목록

네트워크 상태 서비스NetStat Service,
연결 서비스Connectivity Service,
노티피케이션 매니저Notification Manager,
마운트 서비스Mount Service,
디바이스 스토리지 모니터 서비스
Device Storage Monitor Service,
위치 매니저Location Manager,
탐색 매니저 서비스Search Manager Service,
폴백 체크인 서비스Fallback Checkin Service,
배경화면 서비스Wallpaper Service,
오디오 서비스Audio Service,
핸드셋 옵저버Headset Observer 등

Command Shell - adb shell

```
# service list
service list
Found 47 services:
0       phone: [com.android.internal.telephony.ITelephony]
1       iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
2       simphonebook: [com.android.internal.telephony.IIccPhoneBook]
3       isms: [com.android.internal.telephony.ISms]
4       appwidget: [com.android.internal.appwidget.IAppWidgetService]
5       backup: [android.backup.IBackupManager]
6       audio: [android.media.IAudioService]
7       wallpaper: [android.app.IWallpaperManager]
8       checkin: [android.os.ICheckinService]
9       search: [android.app.ISearchManager]
10      location: [android.location.ILocationManager]
11      devicestoragemonitor: []
12      mount: [android.os.IMountService]
13      notification: [android.app.INotificationManager]
14      accessibility: [android.view.accessibility.IAccessibilityManager]
15      connectivity: [android.net.IConnectivityManager]
16      wifi: [android.net.wifi.IWifiManager]
17      netstat: [android.os.INetStatService]
18      input_method: [com.android.internal.view.IInputMethodManager]
19      clipboard: [android.text.IClipboard]
20      statusbar: [android.app.IStatusBar]
21      window: [android.view.IWindowManager]
22      sensor: [android.hardware.ISensorService]
23      alarm: [android.app.IAlarmManager]
24      hardware: [android.os.IHardwareService]
25      battery: []
26      content: [android.content.IContentService]
27      account: [android.accounts.IAccountManager]
28      permission: [android.os.IPermissionController]
29      activity.providers: []
30      activity.senders: []
31      activity.services: []
32      activity.broadcasts: []
33      cpuinfo: []
34      meminfo: []
35      activity: [android.app.IActivityManager]
36      package: [android.content.pm.IPackageManager]
37      telephony.registry: [com.android.internal.telephony.ITelephonyRegistry]
38      usagestats: [com.android.internal.app.IUsageStats]
39      batteryinfo: [com.android.internal.app.IBatteryStats]
40      power: [android.os.IPowerManager]
```

[그림 11-22] 서비스 목록