

Android底层架构介绍与内幕分析

华清远见

版权



- } 华清远见嵌入式培训中心版权所有；
- } 未经华清远见明确许可，不能为任何目的以任何形式复制或传播此文档的任何部分；
- } 本文档包含的信息如有更改，恕不另行通知；
- } 保留所有权利。

内容提纲

- } 1、Google Android 软件架构
- } 2、Android 移植流程
- } 3、编写 / 移植Android内核驱动
- } 4、HAL层介绍
- } 5、Android移植实例

Google Android 软件架构



Google Android 软件架构 (cont.)

- } Android系统架构和其操作系统一样，采用了分层的架构。从架构图看，Android系统架构分为四个层，从高层到低层分别为
 - } 应用程序层、
 - } 应用程序框架层、
 - } 系统运行库层
 - } linux核心层。

应用程序层

} Android会同一系列核心应用程序包一起发布，该应用程序包包括email客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用JAVA语言编写的。

应用程序框架

- } 开发人员可以完全访问核心应用程序所使用的API框架。
- } 隐藏在每个应用后面的是一系列的服务和系统, 其中包括:
 - } 丰富而又可扩展的视图 (Views), 可以用来构建应用程序, 它包括列表 (lists), 网格 (grids), 文本框 (text boxes), 按钮 (buttons), 甚至可嵌入的web浏览器。
 - } 内容提供者 (Content Providers) 使得应用程序可以访问另一个应用程序的数据 (如联系人数据库), 或者共享它们自己的数据
 - } 资源管理器 (Resource Manager) 提供 非代码资源的访问, 如本地字符串, 图形, 和布局文件 (layout files) 。
 - } 通知管理器 (Notification Manager) 使得应用程序可以在状态栏中显示自定义的提示信息。
 - } 活动管理器 (Activity Manager) 用来管理应用程序生命周期并提供常用的导航回退功能。有关更多的细节和怎样从头写一个应用程序, 请参考如何编写一个 Android 应用程序.

系统运行库

- } **Android系统架构** 包含一些C/C++库，这些库能被Android系统中不同的组件使用。它们通过 **Android 应用程序框架** 为开发者提供服务。以下是一些核心库：
 - } **系统 C 库**：一个从 BSD 继承来的标准 C 系统函数库（libc），它是专门为基于 embedded linux 的设备定制的。
 - } **媒体库**：基于 PacketVideo OpenCORE；该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括MPEG4, H.264, MP3, AAC, AMR, JPG, PNG。
 - } **Surface Manager**：对显示子系统的管理，并且为多个应用程序提供了2D和3D图层的无缝融合。
 - } **LibWebCore**：一个最新的web浏览器引擎用，支持Android浏览器和一个可嵌入的web视图。
 - } **SGL**：底层的2D图形引擎
 - } **3D libraries**：基于OpenGL ES 1.0 APIs实现；该库可以使用硬件 3D加速（如果可用）或者使用高度优化的3D软加速。
 - } **FreeType** -位图（bitmap）和矢量（vector）字体显示。
 - } **SQLite** - 一个对于所有应用程序可用，功能强劲的轻型关系型数据库引擎。

Android 运行库

- } Android系统架构包括了一个核心库，该核心库提供了JAVA编程语言核心库的大多数功能。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。Dalvik被设计成一个设备可以同时高效地运行多个虚拟系统。Dalvik虚拟机执行（.dex）的Dalvik可执行文件，该格式文件针对小内存使用做了优化。
- } 同时虚拟机是基于寄存器的，所有的类都经由JAVA编译器编译，然后通过SDK中的"dx"工具转化成.dex格式由虚拟机执行。Dalvik虚拟机依赖于linux内核的一些功能，比如线程机制和底层内存管理机制。

Linux 内核

} Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性，内存管理，进程管理，网络协议栈和驱动模型。Linux 内核也同时作为硬件和软件栈之间的抽象层。

Android 系统源代码目录结构

Project	Description
bionic	C runtime: libc, libm, libdl, dynamic linker Bionic含义为仿生，这里面是一些基础的库的源代码
bootloader/legacy	Bootloader reference code 启动引导相关代码
build	Build system build目录中的内容不是目标所用的代码，而是编译和配置所需要的脚本和工具
cts	Android兼容性测试套件标准
dalvik	Dalvik virtual machine JAVA虚拟机
development	High-level development and debugging tools 程序开发所需要的模板和工具
frameworks/base	Core Android app framework libraries 目标机器使用的一些库
frameworks/policies/base	Framework configuration policies 应用程序的框架层
hardware/libhardware	Hardware abstraction library 与硬件相关的库
hardware/ril	Radio interface layer
out	编译完成后的代码输出与此目录

Android 系统源代码目录结构(cont.)

kernel	Linux kernel Linux2.6的源代码
prebuilt	Binaries to support Linux and Mac OS builds x86和arm架构下预编译的一些资源
packages	Android的各种应用程序
sdk	sdk及模拟器
recovery	System recovery environment 与目标的恢复功能相关
system	Android的底层的一些库
vendor	厂商定制代码

Bionic

- } bionic 目录
- } |-- libc (C库)
- } | |-- arch-arm (ARM架构, 包含系统调用汇编实现)
- } | |-- arch-x86 (x86架构, 包含系统调用汇编实现)
- } | |-- bionic (由C实现的功能, 架构无关)
- } | |-- docs (文档)
- } | |-- include (头文件)
- } | |-- inet (? inet相关, 具体作用不明)
- } | |-- kernel (Linux内核中的一些头文件)
- } | |-- netbsd (? netbsd系统相关, 具体作用不明)
- } | |-- private (? 一些私有的头文件)
- } | |-- stdio (stdio实现)
- } | |-- stdlib (stdlib实现)
- } | |-- string (string函数实现)
- } | |-- tools (几个工具)
- } | |-- tzcode (时区相关代码)
- } | |-- unistd (unistd实现)
- } | `-- zoneinfo (时区信息)

Bionic (cont.)

- } |-- libdl (libdl实现, dl是动态链接, 提供访问动态链接库的功能)
- } |-- libm (libm数学库的实现,)
- } | |-- alpha (alpha架构)
- } | |-- amd64 (amd64架构)
- } | |-- arm (arm架构)
- } | |-- bsdsrc (? bsd的源码)
- } | |-- i386 (i386架构)
- } | |-- i387 (i387架构?)
- } | |-- ia64 (ia64架构)
- } | |-- include (头文件)
- } | |-- man (数学函数, 后缀名为.3, 一些为freeBSD的库文件)
- } | |-- powerpc (powerpc架构)
- } | |-- sparc64 (sparc64架构)
- } | `-- src (源代码)
- } |-- libstdc++ (libstdc++ C++实现库)
- } | |-- include (头文件)
- } | `-- src (源码)
- } |-- libthread_db (多线程程序的调试器库)
- } | `-- include (头文件)
- } `-- linker (动态链接器)
- } `-- arch (支持arm和x86两种架构)

Bootloader

```
} bootable 目录
} .
} |-- bootloader      (适合各种bootloader的通用代码)
} | `-- legacy        (估计不能直接使用, 可以参考)
} |   |-- arch_armv6  (V6架构, 几个简单的汇编文件)
} |   |-- arch_msm7k  (高通7k处理器架构的几个基本驱动)
} |   |-- include     (通用头文件和高通7k架构头文件)
} |   |-- libboot     (启动库, 都写得很简单)
} |   |-- libc        (一些常用的c函数)
} |   |-- nandwrite   (nandwirte函数实现)
} |   `-- usbloader   (usbloader实现)
} |-- diskinstaller   (android镜像打包器, x86可生产iso)
} `-- recovery        (系统恢复相关)
}   |-- edify         (升级脚本使用的edify脚本语言)
}   |-- etc           (init.rc恢复脚本)
}   |-- minui         (一个简单的UI)
}   |-- minzip        (一个简单的压缩工具)
}   |-- mtdutils      (mtd工具)
}   |-- res           (资源)
}   | `-- images      (一些图片)
}   |-- tools         (工具)
}   | `-- ota         (OTA Over The Air Updates升级工具)
}   `-- updater       (升级器)
```

Dalvik

- } dalvik目录 dalvik虚拟机
 - } |-- dalvikvm (main.c的目录)
 - } |-- dexdump (dex反汇编)
 - } |-- dexlist (List all methods in all concrete classes in a DEX file.)
 - } |-- dexopt (预验证与优化)
 - } |-- docs (文档)
 - } |-- dvz (和zygote相关的一个命令)
 - } |-- dx (dx工具, 将多个java转换为dex)
 - } |-- hit (? java语言写成)
 - } |-- libcore (核心库)
 - } |-- libcore-disabled (? 禁用的库)
 - } |-- libdex (dex的库)
 - } |-- libnativehelper (Support functions for Android's class libraries)
 - } |-- tests (测试代码)
 - } |-- tools (工具)
 - } `-- vm (虚拟机实现)
- dalvik目录用于提供Android JAVA应用程序运行的基础———JAVA虚拟机。

Frameworks

} frameworks 目录（核心框架——java及C++语言）

-
- base （基本内容）
- |-- api （? 都是xml文件，定义了java的api? ）
- |-- awt （AWT库）
- |-- build （空的）
- |-- camera （摄像头服务程序库）
- |-- cmds （重要命令：am、app_proce等）
- |-- core （核心库）
- |-- data （字体和声音等数据文件）
- |-- docs （文档）
- |-- graphics （图形相关）
- |-- include （头文件）
- |-- keystore （和数据签名证书相关）
- |-- libs （库）
- |-- location （地区库）
- |-- media （媒体相关库）
- |-- obex （蓝牙传输库）
- |-- opengl （2D-3D加速库）
- |-- packages （设置、TTS、VPN程序）
- |-- sax （XML解析器）
- |-- **services** （各种服务程序）

Frameworks (cont.)

- } | |-- telephony (电话通讯管理)
- | |-- test-runner (测试工具相关)
- | |-- tests (各种测试)
- | |-- tools (一些叫不上名的工具)
- | |-- vpn (VPN)
- | `-- wifi (无线网络)
- | |-- opt (可选部分)
- | | |-- com.google.android (有个framework.jar)
- | | |-- com.google.android.googlelogin (有个client.jar)
- | | `-- emoji (standard message elements)
- | `-- policies (Product policies are operating system directions aimed at specific uses)
- | `-- base
 - | |-- mid (MID设备)
 - | `-- phone (手机类设备, 一般用这个)

Hardware

- } hardware 目录 (部分厂家开源的硬解适配层HAL代码)
 - |-- broadcom (博通公司)
 - `-- wlan (无线网卡)
 - |-- libhardware (硬件库)
 - |-- include (头文件)
 - `-- modules (Default (and possibly architecture dependents) HAL modules)
 - |-- gralloc (gralloc显示相关)
 - **overlay** (**Skeleton for the "overlay" HAL module.**)
 - |-- libhardware_legacy (旧的硬件库)
 - |-- **flashlight** (**背光**)
 - |-- **gps** (**GPS**)
 - |-- include (头文件)
 - |-- mount (旧的挂载器)
 - |-- **power** (**电源**)
 - |-- qemu (模拟器)
 - |-- qemu_tracing (模拟器跟踪)
 - |-- tests (测试)
 - |-- uevent (uevent)
 - |-- **vibrator** (**震动**)
 - `-- wifi (无线)

Hardware (cont.)

```
} |-- msm7k    (高通7k处理器开源抽象层)
    |-- boot   (启动)
    |-- libaudio (声音库)
    |-- libaudio-qsd8k (qsd8k的声音相关库)
    |-- libcamera (摄像头库)
    |-- libcopybit (copybit库)
    |-- libgralloc (gralloc库)
    |-- libgralloc-qsd8k (qsd8k的gralloc库)
    |-- liblights (背光库)
    |-- librpc   (RPC库)
    |-- ril     (无线电抽象层)
    |-- include (头文件)
    |-- libril  (库)
    |-- reference-cdma-sms (cdma短信参考)
    |-- reference-ril (ril参考)
    |-- rild    (ril后台服务程序)
    |-- ti     (ti公司开源HAL)
    |-- omap3  (omap3处理器)
    |-- dspbridge (DSP桥)
    |-- libopencorehw (opencore硬件库)
    |-- liboverlay (overlay硬件库)
    |-- libstagefrighthw (stagefright硬件库)
    |-- omx    (omx组件)
    |-- wlan  (无线网卡)
```

Prebuilt

} prebuilt目录展开的一个级别的目录如下所示：
prebuilt 目录 （x86和arm架构下预编译的一些资源）

```
.
|-- android-arm    (arm-android相关)
|  |-- gdbserver  (gdb调试器)
|  |-- kernel    (模拟的arm内核)
|-- android-x86   (x86-android相关)
|  |-- kernel    (空的)
|-- common       (通用编译好的代码, 应该是java的)
|-- darwin-x86   (darwin x86平台)
|  |-- toolchain (工具链)
|     |-- arm-eabi-4.2.1
|     |-- arm-eabi-4.3.1
|     |-- arm-eabi-4.4.0
|-- darwin-x86_64 (darwin x86 64bit平台)
|-- linux-x86    (linux x86平台)
|  |-- toolchain (工具链, 我们应该主要用这个)
|     |-- arm-eabi-4.2.1
|     |-- arm-eabi-4.3.1
|     |-- arm-eabi-4.4.0
|     |-- i686-unknown-linux-gnu-4.2.1 (x86版编译器)
|-- linux-x86_64 (linux x86 64bit平台)
|-- windows     (windows平台)
|-- windows-x86_64 (64bit windows平台)
```

System

```
} |-- Bluetooth (蓝牙相关)
|-- core (系统核心工具箱接口)
|-- adb (adb调试工具)
|-- cpio (cpio工具, 创建img)
|-- debuggerd (调试工具)
|-- fastboot (快速启动相关)
|-- include (系统接口头文件)
|-- init (init程序源代码)
|-- libacc (轻量级C编译器)
|-- libctest (libc测试相关)
|-- libcutils (libc工具)
|-- liblog (log库)
|-- libmncrypt (加密库)
|-- libnetutils (网络工具库)
|-- libpixelflinger (图形处理库)
|-- libsysutils (系统工具库)
|-- libzipfile (zip库)
|-- logcat (查看log工具)
|-- logwrapper (log封装工具)
|-- mkbootimg (制作启动boot.img的工具盒脚本)
|-- netcfg (网络配置netcfg源码)
|-- nexus (google最新手机的代码)
|-- rootdir (rootfs, 包含一些etc下的脚本和配置)
|-- sh (shell代码)
|-- toolbox (toolbox, 类似busybox的工具集)
|-- vold (SD卡管理器)
```

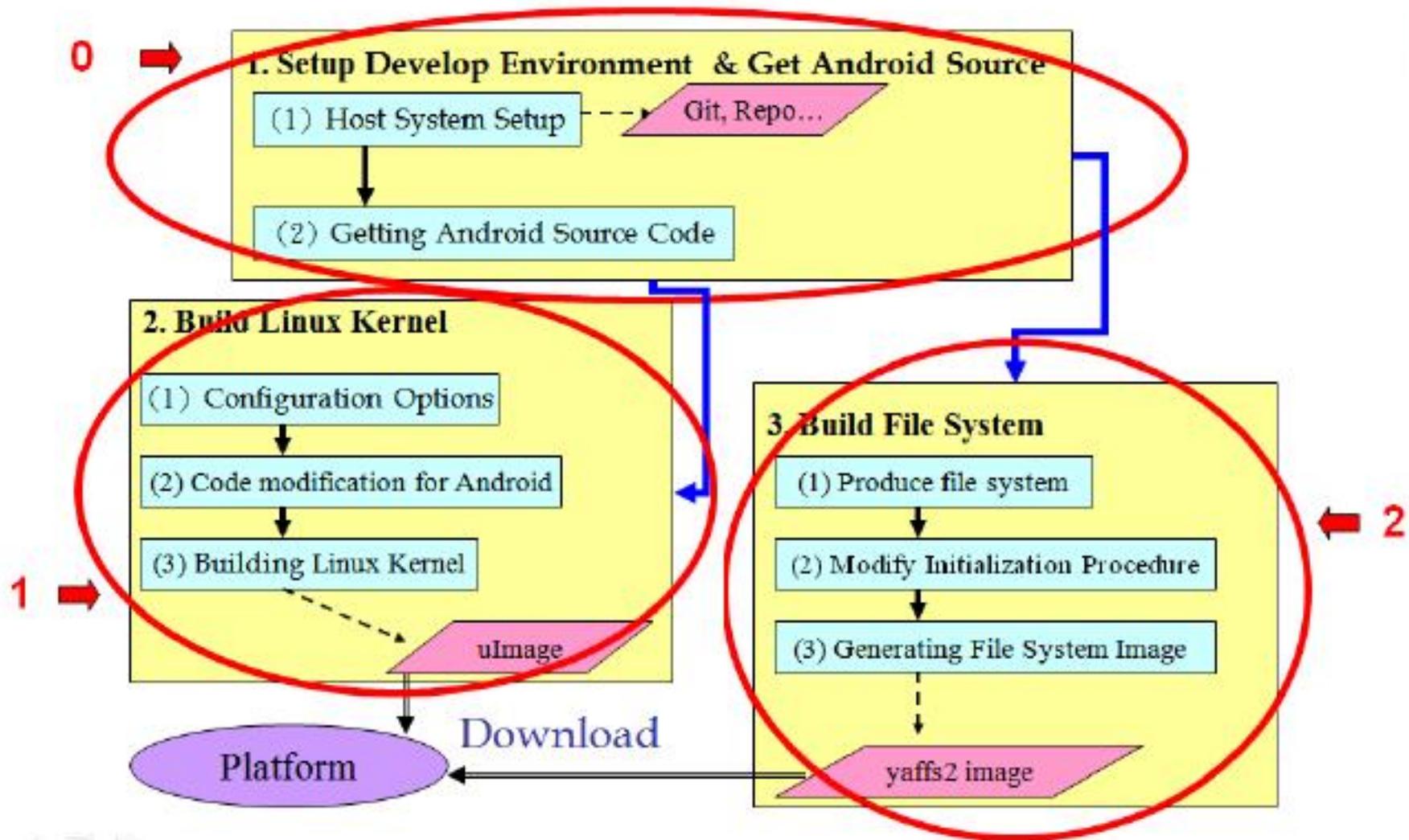
System (cont.)

- } |-- extras (额外工具)
 - | |-- latencytop (a tool for software developers , identifying system latency happen)
 - | |-- libpagemap (pagemap库)
 - | |-- librank (Java Library Ranking System库)
 - | |-- procmem (pagemap相关)
 - | |-- procrank (Java Library Ranking System相关)
 - | |-- showmap (showmap工具)
 - | |-- showslab (showslab工具)
 - | |-- sound (声音相关)
 - | |-- su (su命令源码)
 - | |-- tests (一些测试工具)
 - | `-- timeinfo (时区相关)
 - `-- wlan (无线相关)
 - `-- ti (ti网卡相关工具及库)

Vendor

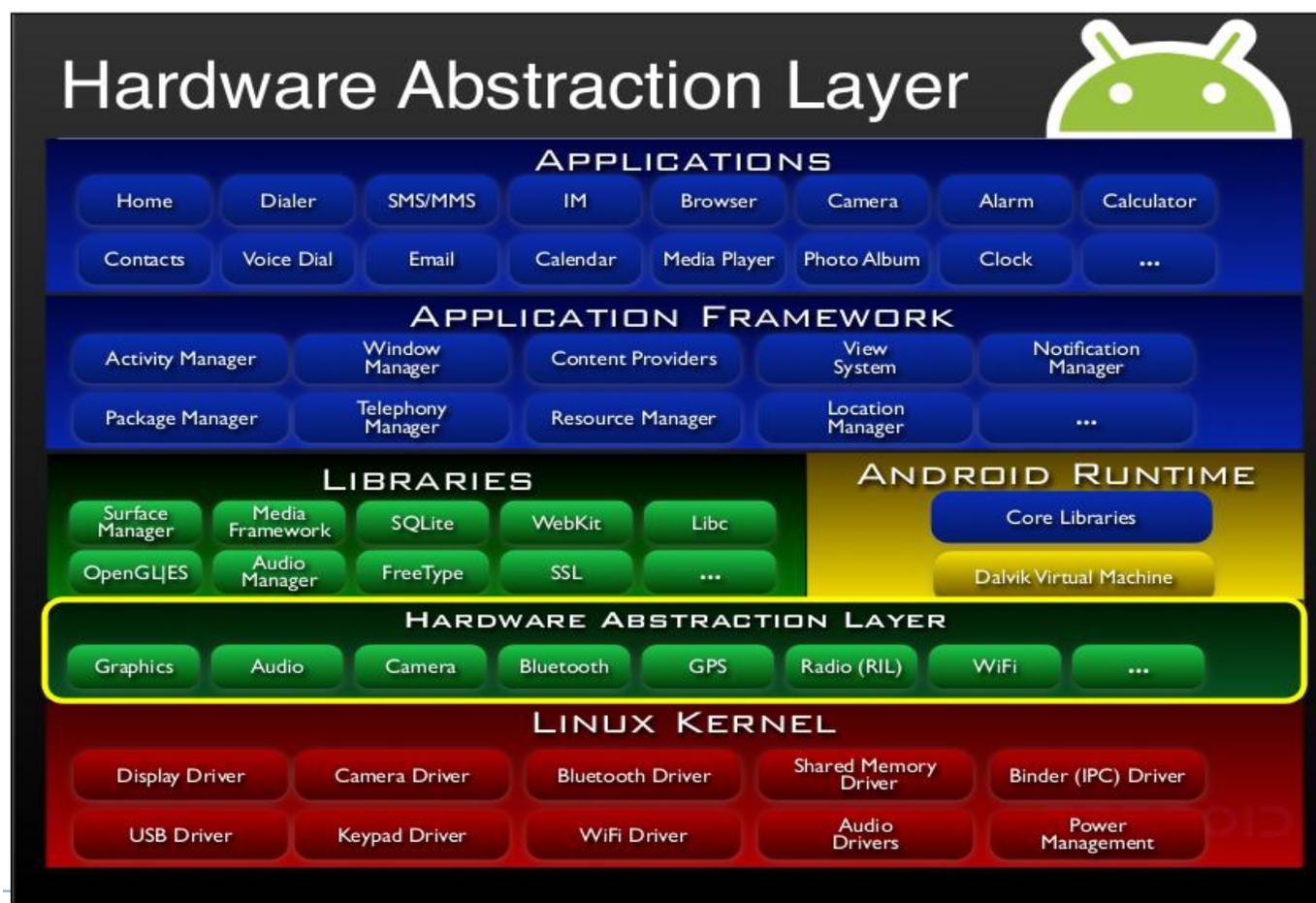
```
} |-- aosp    (android open source project)
   |-- products (一些板级规则)
   |-- htc    (HTC公司)
       |-- common-open (通用部分)
           |-- akmd    (解压img用的工具)
           |-- dream-open (G1开放部分)
           |-- prebuilt-open (预编译开放部分)
           |-- sapphire-open (sapphire这款型号开放内容)
   |-- pv-open (没东西)
   |-- qcom   (里面基本是空的)
   |-- sample (google提供的样例)
       |-- apps    (应用)
           |-- client (用户)
           |-- upgrade (升级)
       |-- frameworks (框架)
           |-- PlatformLibrary (平台库)
       |-- products (产品)
       |-- sdk_addon (sdk添加部分)
       |-- skins    (皮肤)
           |-- WVGA MedDpi (WVGA适用的图片)
```

Android 移植流程



HAL 概念

- } Android 的 HAL（Hardware Abstract Layer 硬件抽象层）是 Google 因应厂商「希望不公开源码」的要求下，所推出的新观念，其架构如下图。



HAL存在的原因

- } 1、并不是所有的硬件设备都有标准的linux kernel的接口。
- } 2、Kernel driver涉及到GPL的版权。某些设备制造商并不原因公开硬件驱动，所以才去HAL方式绕过GPL。
- } 3、针对某些硬件，Android有一些特殊的需求。

HAL 简介及现状分析

- } 现有HAL架构由Patrick Brady (Google) 在2008 Google I/O演讲中提出，从上面架构图我们知道，HAL 的目的是为了把 Android framework 与 Linux kernel 完整「隔开」。让 Android 不至过度依赖 Linux kernel，有点「kernel independent」的意思。
- } 因为各厂商需要开发不开源代码的驱动程序模组要求下所规划的架构与概念要求下所规划的架构与观念
- } 但是目前的HAL架构抽象程度还不足
- } 需要变动框架来整合HAL模组

HAL内容

} HAL 主要的实作储存于以下目录:

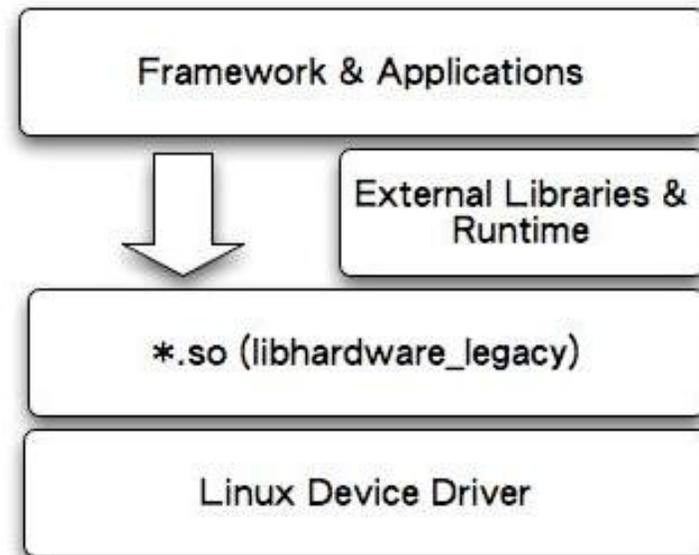
1. libhardware_legacy/ - 旧的架构、采取链接库模块的观念进行
2. libhardware/ - 新架构、调整为 HAL stub 的观念
3. ril/ - Radio Interface Layer
4. msm7k QUAL平台相关

} 主要包含一下一些模块:

Gps
Vibrator
Wifi
Uevent
Copybit
Audio
Camera
Lights
Ril
Overlay
.....

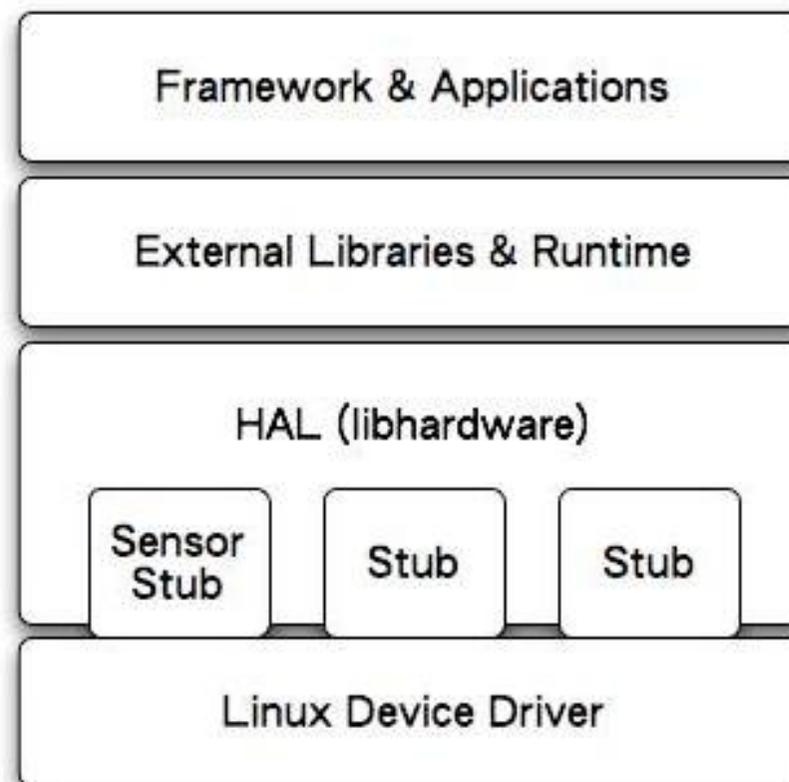
旧的HAL 架构(libhardware_legacy)

- } libhardware_legacy 作法，是传统的「module」方式，也就是将 *.so 文件当做「shared library」来使用，在runtime（JNI 部份）以 direct function call 使用 HAL module。通过直接函数调用的方式，来操作驱动程序。
- } 当然，应用程序也可以不需要通过 JNI 的方式进行，直接加载 *.so 檔（dlopen）的做法调用 *.so 里的符号（symbol）也是一种方式。
- } 总而言之是没有经过封装，上层可以直接操作硬件。

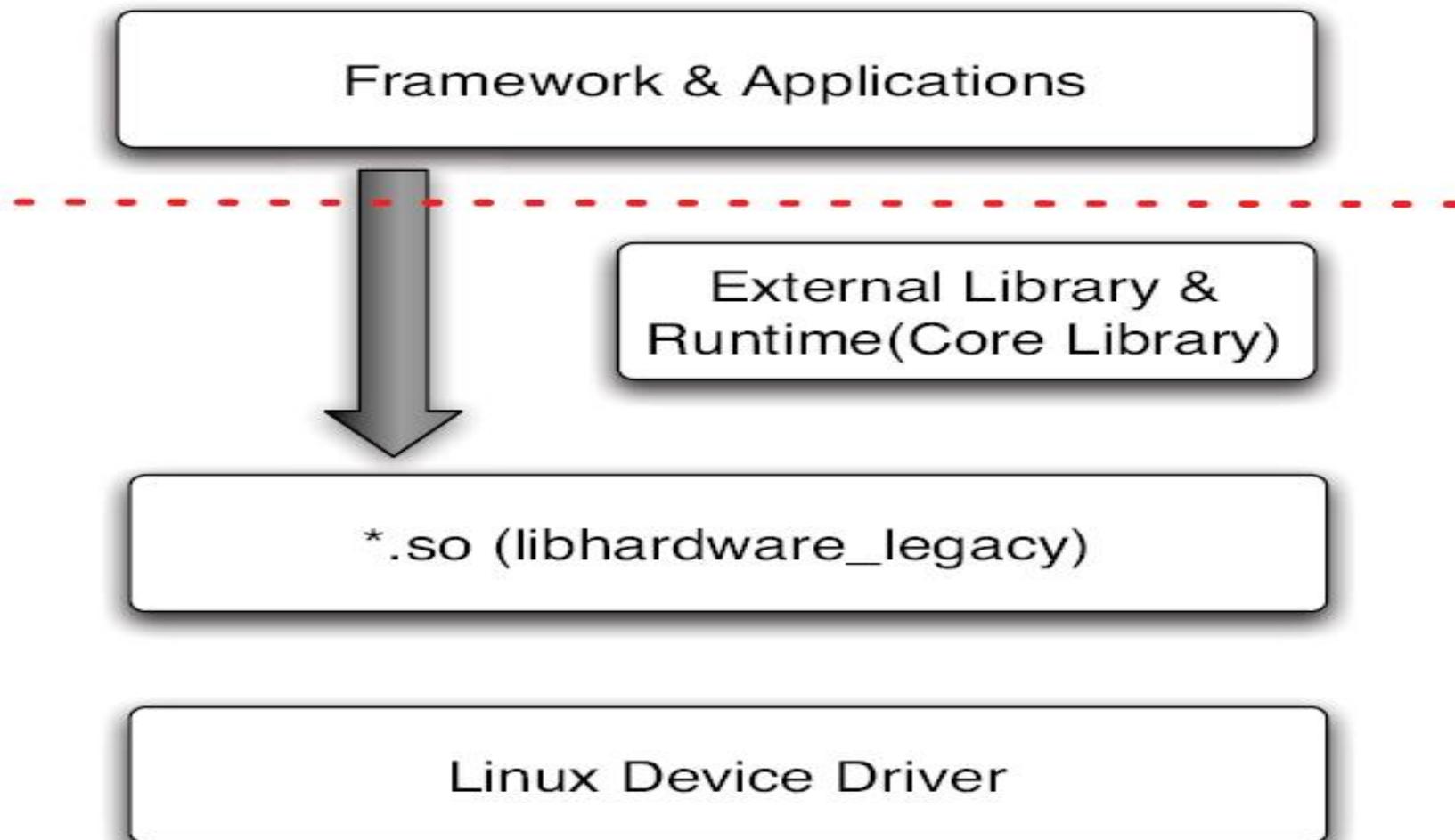


新的HAL 架构(libhardware)

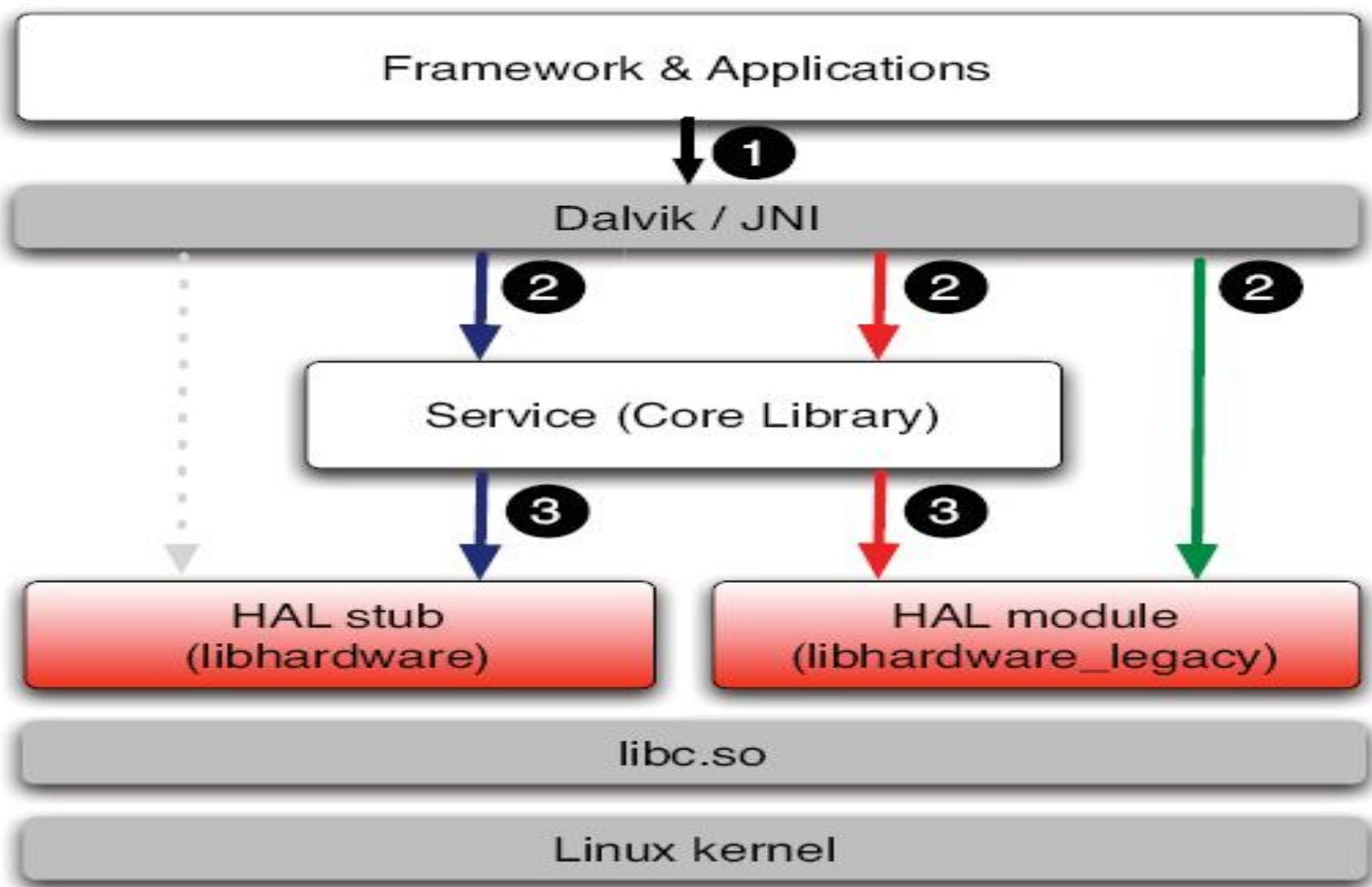
} 现在的 libhardware 架构，就有「stub」的味道了。HAL stub 是一种代理人（proxy）的概念，stub 虽然仍是以 *.so 档的形式存在，但 HAL 已经将 *.so 档隐藏起来了。Stub 向 HAL「提供」操作函数（operations），而 runtime 则是向 HAL 取得特定模块（stub）的 operations，再 callback 这些操作函数。这种以 indirect function call 的架构，让 HAL stub 变成是一种「包含」关系，即 HAL 里包含了许许多多的 stub（代理人）。Runtime 只要说明「类型」，即 module ID，就可以取得操作函数。对于目前的 HAL，可以认为 Android 定义了 HAL 层结构框架，通过几个接口访问硬件从而统一了调用方式。



JNI 在HAL架构中的位置



Android HAL 架构



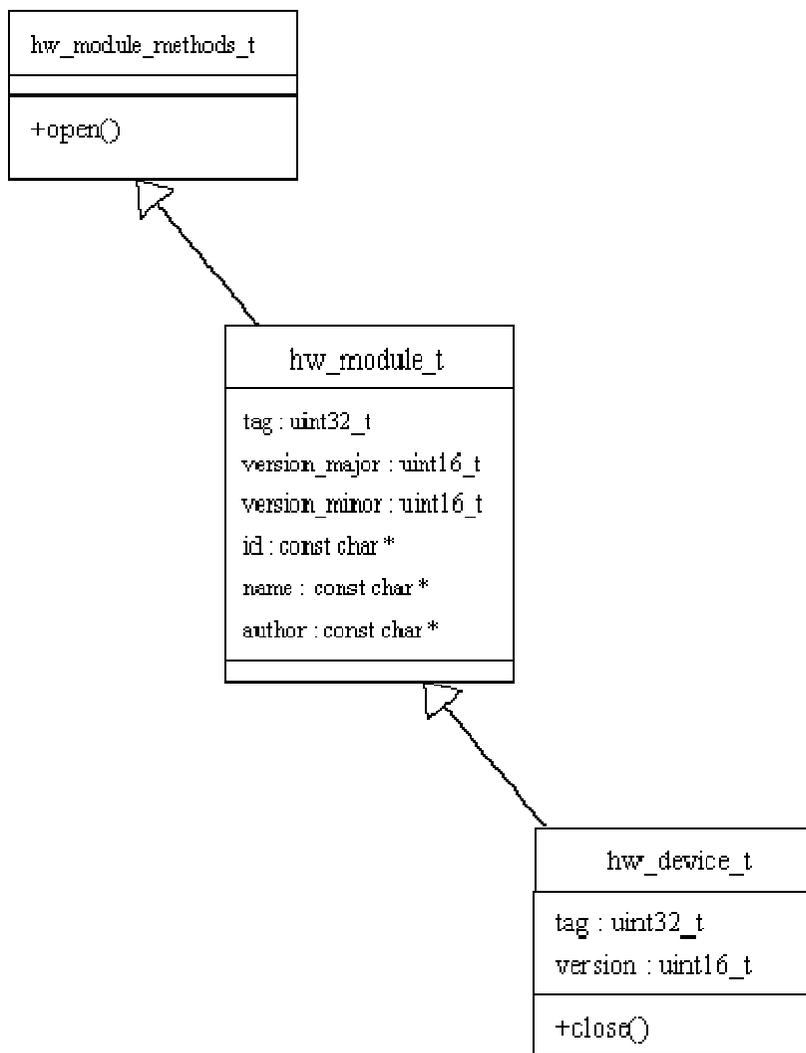
HAL module架构

} HAL module主要分为三个结构:

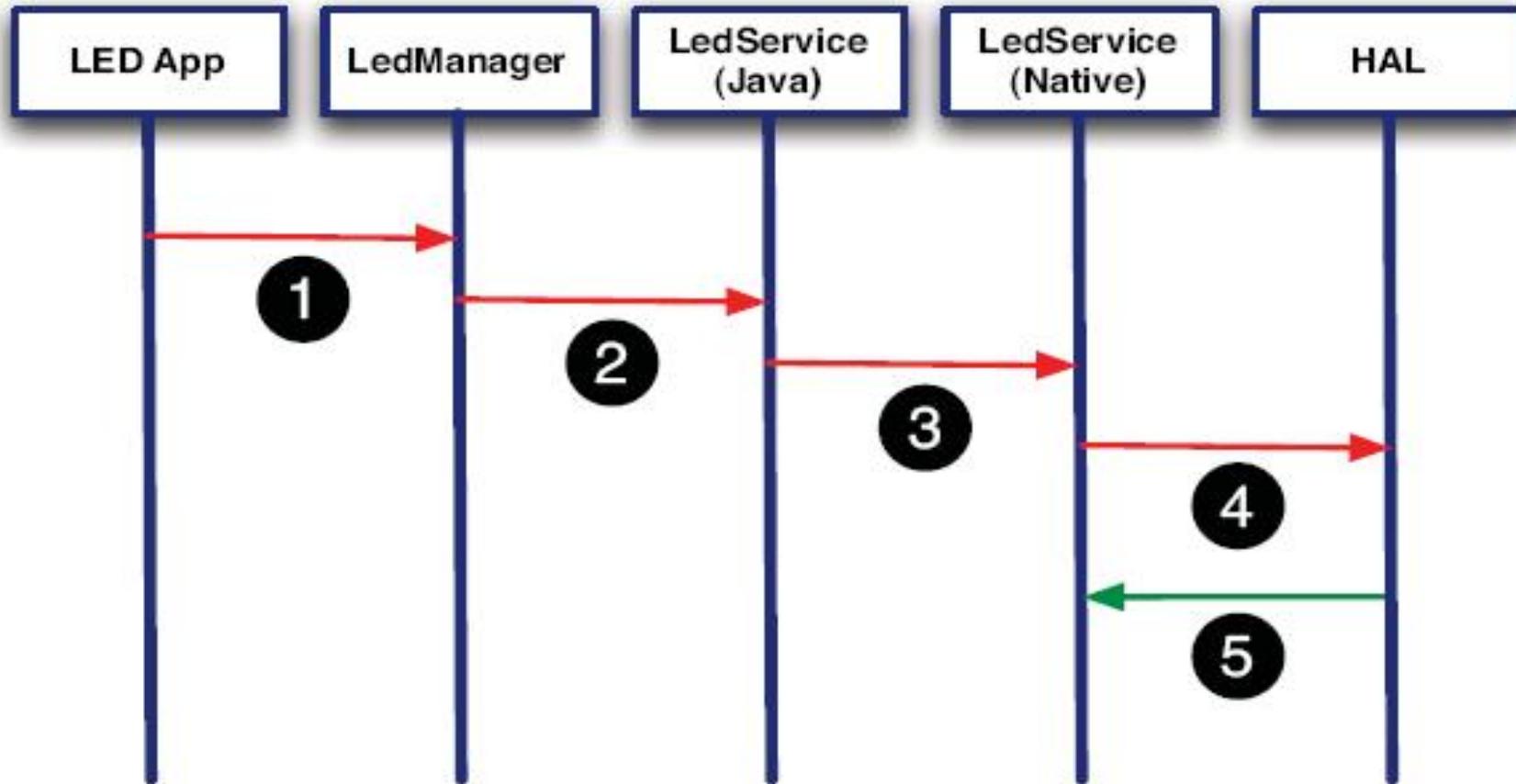
```
struct hw_module_t;
struct hw_module_methods_t;
struct hw_device_t;
```

} 定义在hardware.h文件里面

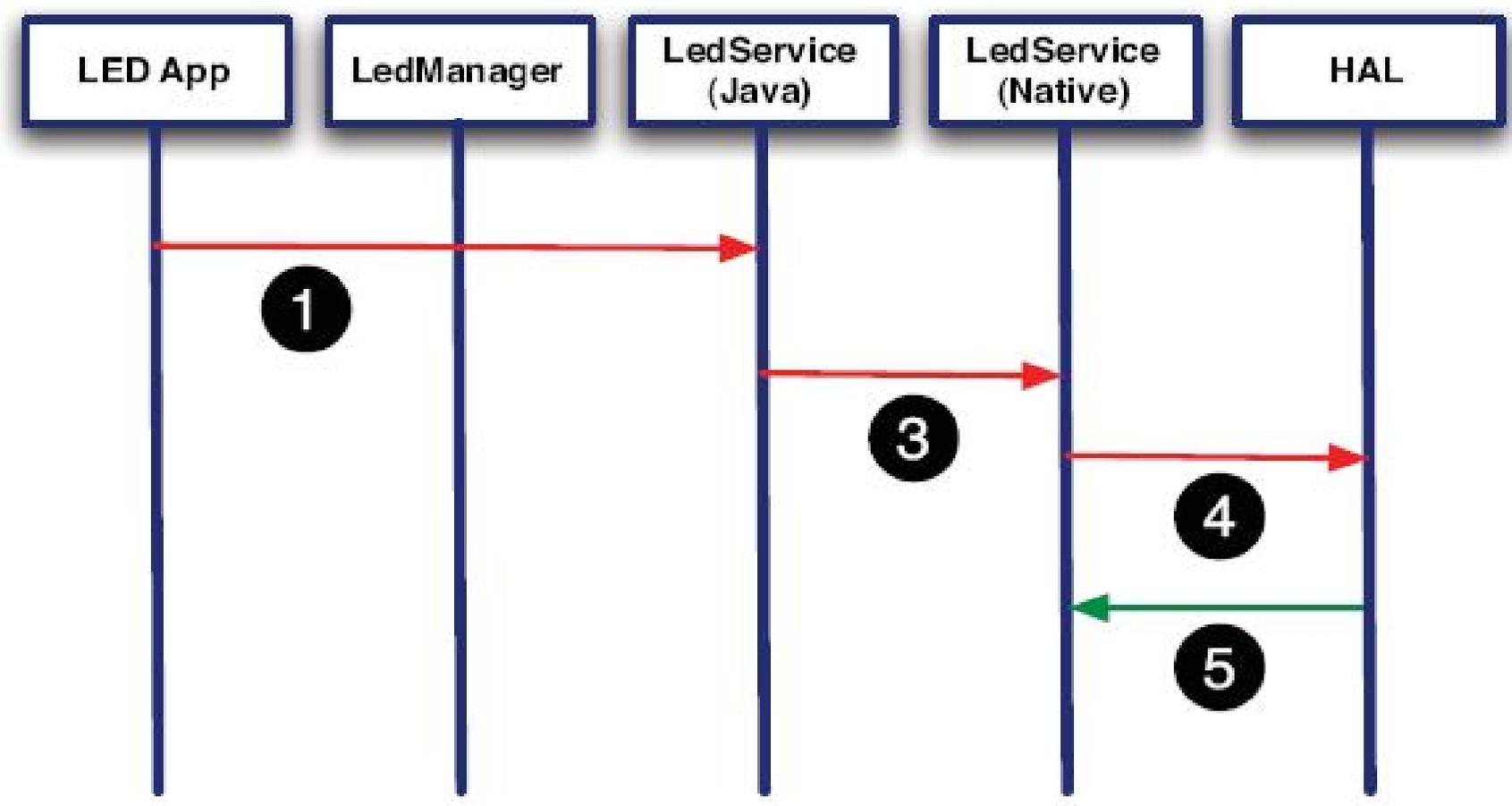
} 他们的继承关系如右图:



测试code整体框架



测试code整体框架(cont.)



结束

华清远见