1:hal->jni->framework->app

Android vibrate上层主要是hal->jni->framework->app

hal层其实与linux上层一样,调用open close等接口,可以没有

jni层为固定模式

framework层其实就是service的服务端

app层其实就是service的客户端(获取服务端,调用服务端接口)

2:hal层

hal层文件为



主要定义并完善vibrator\_device\_t hw\_module\_t两个结构体

主要是vibrator\_device\_t 中vibrator\_on vibrator\_off两个接口

jni层会调用vibra\_open接口,vibra\_open会初始化vibrator\_device\_t结构,并返回给jni层

3:jni层

jni层文件为

主要通过一些接口获取hal层的vibrator\_device\_t结构,从而获取vibrator\_on vibrator\_off,等接口,进而封装完善成vibratorOn vibratorOff等接口,然后通过如下方式将接口名换掉,,framerwork层则使用左边的接口名

static JNINativeMethod method\_table[] = {

{ "vibratorExists", "()Z", (void\*)vibratorExists },

{ "vibratorOn", "(J)V", (void\*)vibratorOn },

{ "vibratorOff", "()V", (void\*)vibratorOff }

};

有时也可以不获取vibrator\_device\_t接口,直接调用操作vibrator的接口,本例中jni就是如此,如下,也就是说hal层并不一定是需要的



4:framework层

framework层主要是提供service的服务端

第一步是建立XXX.aidl接口文件



这个接口文件中定义的方法就是service服务端对外提供的服务

interface IVibratorService

{

boolean hasVibrator();

void vibrate(int uid, String opPkg, long milliseconds, int usageHint, IBinder token);

void vibratePattern(int uid, String opPkg, in long[] pattern, int repeat, int usageHint, IBinder token);

void cancelVibrate(IBinder token);

}

系统会根据aidl会自动生成XXX.Stub接口,这个接口实现了Ibinder接口及aidl中的服务接口IVibratorService

第二步就是实现service服务,其主要作用就是继承XXX.Stub接口并实现



实现服务接口时,会用到jni层提供的如下三个接口

native static boolean vibratorExists();

native static void vibratorOn(long milliseconds);

native static void vibratorOff();

5:app层

app层,app层通过获取服务方式,获取服务,也就是IvibratorService的实例,访问service服务

类似如下代码

1. vibrator = (Vibrator)getSystemService(Context.VIBRATOR\_SERVICE);
2. **long** [] pattern = {100,400,100,400};   // 停止 开启 停止 开启
3. vibrator.vibrate(pattern,2);           //重复两次上面的pattern 如果只想震动一次，index设

wangcq327

20160922