# 在Ubuntu上为Android系统内置Java应用程序测试Application Frameworks层的硬件服务

我们在Android系统增加硬件服务的目的是为了让应用层的APP能够通过Java接口来访问硬件服务。那么， APP如何通过Java接口来访问Application Frameworks层提供的硬件服务呢？在这一篇文章中，我们将在Android系统的应用层增加一个内置的应用程序，这个内置的应用程序通过ServiceManager接口获取指定的服务，然后通过这个服务来获得硬件服务。

**《Android系统源代码情景分析》一书正在进击的程序员网（**[http://0xcc0xcd.com](http://0xcc0xcd.com/" \t "_blank)**）中连载，点击进入！**

        一. 参照[在Ubuntu上为Android系统的Application Frameworks层增加硬件访问服务](http://blog.csdn.net/luoshengyang/article/details/6578352" \t "_blank)一文，在Application Frameworks层定义好自己的硬件服务HelloService，并提供IHelloService接口提供访问服务。

       二. 为了方便开发，我们可以在IDE环境下使用Android SDK来开发Android应用程序。开发完成后，再把程序源代码移植到Android源代码工程目录中。使用Eclipse的Android插件ADT创建Android工程很方便，这里不述，可以参考网上其它资料。工程名称为Hello，下面主例出主要文件：

    主程序是src/shy/luo/hello/Hello.java：

**[java]** [view plain](https://blog.csdn.net/luoshengyang/article/details/6580267) [copy](https://blog.csdn.net/luoshengyang/article/details/6580267)

1. **package** shy.luo.hello;
3. **import** shy.luo.hello.R;
4. **import** android.app.Activity;
5. **import** android.os.ServiceManager;
6. **import** android.os.Bundle;
7. **import** android.os.IHelloService;
8. **import** android.os.RemoteException;
9. **import** android.util.Log;
10. **import** android.view.View;
11. **import** android.view.View.OnClickListener;
12. **import** android.widget.Button;
13. **import** android.widget.EditText;
15. **public** **class** Hello **extends** Activity **implements** OnClickListener {
16. **private** **final** **static** String LOG\_TAG = "shy.luo.renju.Hello";
18. **private** IHelloService helloService = **null**;
20. **private** EditText valueText = **null**;
21. **private** Button readButton = **null**;
22. **private** Button writeButton = **null**;
23. **private** Button clearButton = **null**;
25. /\*\* Called when the activity is first created. \*/
26. @Override
27. **public** **void** onCreate(Bundle savedInstanceState) {
28. **super**.onCreate(savedInstanceState);
29. setContentView(R.layout.main);
31. helloService = IHelloService.Stub.asInterface(
32. ServiceManager.getService("hello"));
34. valueText = (EditText)findViewById(R.id.edit\_value);
35. readButton = (Button)findViewById(R.id.button\_read);
36. writeButton = (Button)findViewById(R.id.button\_write);
37. clearButton = (Button)findViewById(R.id.button\_clear);
39. readButton.setOnClickListener(**this**);
40. writeButton.setOnClickListener(**this**);
41. clearButton.setOnClickListener(**this**);
43. Log.i(LOG\_TAG, "Hello Activity Created");
44. }
46. @Override
47. **public** **void** onClick(View v) {
48. **if**(v.equals(readButton)) {
49. **try** {
50. **int** val = helloService.getVal();
51. String text = String.valueOf(val);
52. valueText.setText(text);
53. } **catch** (RemoteException e) {
54. Log.e(LOG\_TAG, "Remote Exception while reading value from device.");
55. }
56. }
57. **else** **if**(v.equals(writeButton)) {
58. **try** {
59. String text = valueText.getText().toString();
60. **int** val = Integer.parseInt(text);
61. helloService.setVal(val);
62. } **catch** (RemoteException e) {
63. Log.e(LOG\_TAG, "Remote Exception while writing value to device.");
64. }
65. }
66. **else** **if**(v.equals(clearButton)) {
67. String text = "";
68. valueText.setText(text);
69. }
70. }
71. }

    程序通过ServiceManager.getService("hello")来获得HelloService，接着通过IHelloService.Stub.asInterface函数转换为IHelloService接口。其中，服务名字“hello”是系统启动时加载HelloService时指定的，而IHelloService接口定义在android.os.IHelloService中，具体可以参考[在Ubuntu上为Android系统的Application Frameworks层增加硬件访问服务](http://blog.csdn.net/luoshengyang/article/details/6578352" \t "_blank)一文。这个程序提供了简单的读定自定义硬件有寄存器val的值的功能，通过IHelloService.getVal和IHelloService.setVal两个接口实现。

界面布局文件res/layout/main.xml：

**[html]** [view plain](https://blog.csdn.net/luoshengyang/article/details/6580267) [copy](https://blog.csdn.net/luoshengyang/article/details/6580267)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. android:orientation="vertical"
4. android:layout\_width="fill\_parent"
5. android:layout\_height="fill\_parent"**>**
6. **<LinearLayout**
7. android:layout\_width="fill\_parent"
8. android:layout\_height="wrap\_content"
9. android:orientation="vertical"
10. android:gravity="center"**>**
11. **<TextView**
12. android:layout\_width="wrap\_content"
13. android:layout\_height="wrap\_content"
14. android:text="@string/value"**>**
15. **</TextView>**
16. **<EditText**
17. android:layout\_width="fill\_parent"
18. android:layout\_height="wrap\_content"
19. android:id="@+id/edit\_value"
20. android:hint="@string/hint"**>**
21. **</EditText>**
22. **</LinearLayout>**
23. **<LinearLayout**
24. android:layout\_width="fill\_parent"
25. android:layout\_height="wrap\_content"
26. android:orientation="horizontal"
27. android:gravity="center"**>**
28. **<Button**
29. android:id="@+id/button\_read"
30. android:layout\_width="wrap\_content"
31. android:layout\_height="wrap\_content"
32. android:text="@string/read"**>**
33. **</Button>**
34. **<Button**
35. android:id="@+id/button\_write"
36. android:layout\_width="wrap\_content"
37. android:layout\_height="wrap\_content"
38. android:text="@string/write"**>**
39. **</Button>**
40. **<Button**
41. android:id="@+id/button\_clear"
42. android:layout\_width="wrap\_content"
43. android:layout\_height="wrap\_content"
44. android:text="@string/clear"**>**
45. **</Button>**
46. **</LinearLayout>**
47. **</LinearLayout>**

字符串文件res/values/strings.xml：

**[html]** [view plain](https://blog.csdn.net/luoshengyang/article/details/6580267) [copy](https://blog.csdn.net/luoshengyang/article/details/6580267)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<resources>**
3. **<string** name="app\_name"**>**Hello**</string>**
4. **<string** name="value"**>**Value**</string>**
5. **<string** name="hint"**>**Please input a value...**</string>**
6. **<string** name="read"**>**Read**</string>**
7. **<string** name="write"**>**Write**</string>**
8. **<string** name="clear"**>**Clear**</string>**
9. **</resources>**

程序描述文件AndroidManifest.xml：

**[html]** [view plain](https://blog.csdn.net/luoshengyang/article/details/6580267) [copy](https://blog.csdn.net/luoshengyang/article/details/6580267)

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<manifest** xmlns:android="http://schemas.android.com/apk/res/android"
3. package="shy.luo.hello"
4. android:versionCode="1"
5. android:versionName="1.0"**>**
6. **<application** android:icon="@drawable/icon" android:label="@string/app\_name"**>**
7. **<activity** android:name=".Hello"
8. android:label="@string/app\_name"**>**
9. **<intent-filter>**
10. **<action** android:name="android.intent.action.MAIN" **/>**
11. **<category** android:name="android.intent.category.LAUNCHER" **/>**
12. **</intent-filter>**
13. **</activity>**
14. **</application>**
15. **</manifest>**

三. 将Hello目录拷贝至packages/experimental目录，新增Android.mk文件：  
    **USER-NAME@MACHINE-NAME:~/Android/packages/experimental$ vi Android.mk**

Android.mk的文件内容如下：

**LOCAL\_PATH:= $(call my-dir)**

**include $(CLEAR\_VARS)**

**LOCAL\_MODULE\_TAGS := optional**

**LOCAL\_SRC\_FILES := $(call all-subdir-java-files)**

**LOCAL\_PACKAGE\_NAME := Hello**

**include $(BUILD\_PACKAGE)**

四. 编译：

**USER-NAME@MACHINE-NAME:~/Android$ mmm packages/experimental/Hello**

编译成功后，便可以在out/target/product/generic/system/app目录下看到Hello.apk文件了。  
    五. 重新打包系统镜像文件system.img：

**USER-NAME@MACHINE-NAME:~/Android$ make snod**  
    重新打包后的system.img文件就内置了Hello.apk文件了。

六. 运行Android模拟器：

**USER-NAME@MACHINE-NAME:~/Android$ emulator -kernel kernel/common/arch/arm/boot/zImage &**

在Home Screen中可以看到Hello应用程序：



打开Hello应用程序：



点击Read按钮，可以从HelloService中读取硬件寄存器val的值；点击Clear按钮，可以清空文本框的值；在文本框中输入一个数值，再点击Write按钮，便可以将这个值写入到硬件寄存器val中去，可以再次点击Read按钮来验证是否正确写入了值。

至此，我们就完整地学习了在Android的Linux内核空间添加硬件驱动程序、在Android的硬件抽象层添加硬件接口、在Android的Application Frameworks层提供硬件服务以及在Android的应用层调用硬件服务的整个过程了，希望能为读者进入Android系统提供入门帮助。重新学习整个过程，请参考[Android硬件抽象层（HAL）概要介绍和学习计划](http://blog.csdn.net/luoshengyang/article/details/6567257)。