Research on Android Architecture and Application Development

Yuanyi Chen1,*
1College of Internet of Things, Jiangxi Teachers College, JTC, Yingtan, China, 335000
*Corresponding author e-mail: chenyuanyi@jxsfgz.edu.com

Abstract. With the continuous development of the electronic industry, mobile phones are also developing towards convenient and intelligent direction. The system of mobile devices is constantly expanding, and Android has certain advantages over the system of wire transmission. In this paper, the architecture of The Android platform and the composition and working mechanism of Android applications are first described. Meanwhile, the architecture of the Android platform is illustrated with examples for readers’ reference.

Keywords: Android Architecture, Application Development, Linux Kernel Layer, The Application

1. Introduction
With the release of the Android Intelligent platform in November 2007, the integrated platform system for mobile devices has been improved. Android has the functions of an operating system, user interface middleware and applications, and can meet all the software requirements of users' mobile phones. Thus, the release of the Android intelligent platform guarantees the exclusives barrier to innovation in the mobile industry.

2. Overview of Android platform architecture

2.1. Android platform architecture
The Android platform adopts the architecture form of Software Stack, that is, Software Stack.

2.1.1. Linux kernel layer
As the bottom layer of the architecture as a whole, this layer is also the basic layer, in which the various drivers are the basic functions to realize the operation of the system [1-3]. Android is mainly based on Linux 26 kernel for system services, which covers security performance, memory management, process management, network protocol stack and model driver, etc. It also serves as the main abstraction layer between hardware and software stack [4-6].

2.1.2. The middle tier
The middle tier includes the libraries and the Android operating environment. Libraries are an
important way to implement native C++, providing a foundation for the use of various components in The Android system. The main function of the middle tier is to provide developers with corresponding services through the application framework. SQLite, Open GL, WebKit appear and the Core library contains the Android core API in addition to the Java API.

2.1.3. the application layer
At the heart of this layer is the API framework, the development platform that Android provides to developers, and the core mechanism for the android platform as a whole. Application layer: This layer is composed mainly of everyday applications such as information call records, calendars, etc., which are written using the Java language. These applications also come with a set of core application packages, including an email client, maps, browser, contacts, and more (figure 1. Calendar on the phone).

2.2. Main applications
The main applications on the Android platform are composed of four parts, namely, activity, broadcast receiver, content provider and service. The following is a simple analysis of these four parts, so as to lay a good foundation for the application development on the Android platform.

2.2.1. Activities
In Android, all apps are the primary manifestation of activity. As an important part of Android system, activity refers to the primary interface in android smartphones. In the activity can also use buttons to control other software, and the control to install, uninstall and other operations. In this way, in the process of opening various pages, the use trace will be saved in the operation history, if necessary, you can quickly find the corresponding record in the history interface and read. If you do not need to do this, you can enter other related programs to delete. Generally speaking, android system will save each interface you browse by default (Figure 2 Android activity).
2.2.2. Broadcast receiver
The broadcast Receiver is the Roadcast Receiver. In Android, instead of Broadcast, the Broadcast receiver filters the Broadcast, recycles it, and responds to it, eventually creating a component that transfers information between programs. Broadcast receivers can prompt applications to react and judge things in the outside world.

2.2.3. Content providers
A Content Provider is a Content Provider. In this application, relevant data can be saved to the appropriate folder, database, and also can be saved to various valid devices. The main role of the content provider is to bring the data to a final shared state with other applications, where it can be guaranteed that the relevant resources can be shared if needed.

3. The structure and working mechanism of Android applications
An Android application can be organized from four building blocks: Activity, BroadcastReceiver, Service, and Content Provider. Not every Android application must have these four building blocks, and in some cases only a few of these four are needed to build the application. In addition, the AndroidMainfest XML configuration file is used to define the components of the application, the functions and requirements of the components, etc.

3.1. Application description of the four building blocks
Activity: The most basic component in a building block, an Activity inherits from the Activity base class and is typically a separate screen that displays a GUI interface made up of Noconviews controls and responds to events. In addition, Android uses Intent classes to switch from screen to screen. Intent classes are used to describe what an application will do.

A Service is a program that has no user interface and is more responsible for the background work.

A BroadcastReceiver is a receiver that keeps the user informed of the state of the phone and of any pending or unusual behavior. When you need to respond to an external event, you can use a Broadcast Receiver. A BroadcastReceiver NotificationManager is used to notify the user when an external event occurs, but it does not generate a UI.

A ContentProvider implements a standard set of methods to use when an application needs to share data in a file, in an SQL database, or on a valid device with other applications.

Because of the importance of Intents, intents in Android can also be explained. Android application components can call and coordinate with each other, the communication between these components is
mainly assisted by Intents. Intent is responsible for describing the action of an operation in the application, the action involves data, additional data, etc. Based on the description of the Intent, Android finds the corresponding component and passes the Intent to the calling component to complete the component invocation. Intents therefore act as mediations, specifically providing components to call each other.

To achieve decoupling between the caller and the called.

3.2. Dalvik virtual machine mechanism

In order to solve the operational efficiency problem, Google specially designed the Dalvik virtual machine for the platform. DVM is based on register architecture, which can effectively reduce the number of times to access the memory, thus achieving higher execution efficiency. DVM compiles the code before the program runs for better pre-optimization. Compiling before running does not take up the running time of the program, so you can optimize the code without considering the time cost. DVM executes faster than the JAVA virtual machine, but the larger code length can seriously affect the performance of Android applications, so instead of JAVA bytecode, DVM executes the more compact Dalvik bytecode -DEX class files. In general, a DEX file containing the same class is less than the space taken up by a JAR file. Due to security concerns with Android software, each Android application running has its own process and has its own instance of the Dalvik virtual machine. At the same time, due to the performance limitations of mobile devices, the Dalvik virtual machine is designed to be small in size and occupies small space. Therefore, a device can efficiently run multiple DVM instances at the same time (Figure 3 Android game interface).

Figure 3. Android game interface.

3.3. Android boot and operation mechanism

The Linux kernel is loaded and initialized by the Bootloader when the Android system with Linux at the bottom starts up. All daemons are started during initialization, including: USB daemons to manage USB connection, Android debugging bridge to manage the connection between ADB tool and simulator or real device in SDK, debugging daemons to manage debugging request of program, radio interface daemons to manage wireless communication, etc.

After the daemon is started, the Zygote process is started, which initializes a Dalvik vm instance, loads the required classes and listeners for Socket requests, and creates a process for managing the vm instance application.

After that, start the Runtime process, initialize the service manager, and register the service manager as the default Context manager for Binder services. After the Runtime starts, it sends a request to the Zygote process to start the system service. The Zygote process creates and starts a virtual machine instance of the user system service process. These Native system service processes register with the service Manager to provide process communication services, and the Android management service processes are started and registered with the service Manager by the Native system service process. After all system service processes are started, the system is started.
4. Implementation cases
According to the architecture of The Android platform, the procedures for sending short messages named My Message in combination with the actual situation are as follows.

4.1. Create a new project
Create a new project under Eclipse based on Android version 2.2, My Message.

4.2. Modify the user interface
Modify the content of the RES /layout/ main.xml file by adding text fields from top to bottom, an editable text box for entering numbers, text fields, an editable text box for entering SMS content, and a button box for sending SMS messages to complete the main interface for sending SMS messages.

4.3. Setting permissions
Add a declaration of sending SMS permissions to the Android manifest.xml with the code <uses-permission android:name="Android.permission.SEND_SMS">.

4.4. Realize SMS sending function
Modify the main program My Message Java to add a processing method for the click event of the send SMS button. The key code is white favorites

```java
btnSendSMS.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String phoneNo = txtPhoneNo.getText().toString();
        String message = txtMessage.getText().toString();
        if(phoneNo.length() > 0 && message.length() > 0) {
            Log.v("ROGER","will begin sendSMS");
            sendSMS(phoneNo, message);
        } else 
            Toast.makeText(TinySMS.This,"Please re-enter the phone number and the short Contents of the letter ",Toast.LENGTH_LONG).Show;
    }

    private void sendSMS(String phoneNumber, String message) {
        PendingIntent pi = PendingIntent.getActivity(this, 0, 
                new Intent(this,TinySMS.class), 0); 
        SmsManager sms = SmsManager.getDefault (); 
        sms.sendTextMessage(phoneNumber,null,message,pi,null) ;
    }
```

When the user presses the "Send Message" button, the user interface returns to the original interface of My Message.

4.5. Operation results
Running the program in Eclipse, the system will start one Android emulator, and then start another Android emulator through the Command line of Windows. In this way, the two emulators can realize the function of calling or texting between two phones.

5. Conclusion
To sum up, by explaining the Linux kernel layer, the middle layer and the application layer of the Android platform architecture, and introducing the main Android program teaching, and combining with the analysis of practical cases, I have a further understanding of the Android architecture and application development research, and understand the composition and working mechanism of Android applications.
References

[1] Zhang Lanchun. Research on application development based on Android platform architecture [J]. Computer programming skills and maintenance, 2017 (06): 52-53 + 85.

[2] Zhao Jing. Research on Android system architecture and application development [J]. Automation and instrumentation, 2017 (01): 86-87 + 90.

[3] Zhang Zhaoxin. Research on Android architecture and application development [J]. Inner Mongolia Science and technology and economy, 2016 (17): 72-73 + 78.

[4] Fan Feng. Research on Android architecture and application development [J]. Information and computer (theoretical Edition), 2012 (10): 55-56.

[5] Micro computer information, 2011,27 (09): 1-3. Zeng Jianping, Shao Yanjie. Research on Android system architecture and application development [J].

[6] Yao Yumin, Liu Weiguo. Research on Android architecture and application development [J]. Computer system application, 2008 (11): 110-112 + 24.